ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΕΥΦΥΕΙΣ ΤΕΧΝΟΛΟΓΙΕΣ ΔΙΑΔΙΚΤΥΟΥ – WEB INTELLIGENCE

Εξόρυξη πληροφορίας και ανάλυση συναισθήματος με χρήση μεθόδων μηχανικής μάθησης και σύγχρονων μοντέλων επεξεργασίας φυσικής γλώσσας

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

των

Καμπατζή Αριστοτέλη του Δημητρίου
Σαρόγλου Στυλιανού του Ιωάννη

Επιβλέπων : Διαμαντάρας Κωνσταντίνος
Καθηγητής, ΔΙ.ΠΑ.Ε

Θεσσαλονίκη, Ιούνιος 2023

Η σελίδα αυτή είναι σκόπιμα λευκή.

ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ ΕΥΦΥΕΙΣ ΤΕΧΝΟΛΟΓΙΕΣ ΔΙΑΔΙΚΤΥΟΥ – WEB INTELLIGENCE

Εξόρυξη πληροφορίας και ανάλυση συναισθήματος με χρήση μεθόδων μηχανικής μάθησης και σύγχρονων μοντέλων επεξεργασίας φυσικής γλώσσας

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

των

Καμπατζή Αριστοτέλη του Δημητρίου
Σαρόγλου Στυλιανού του Ιωάννη

Επιβλέπων : Διαμαντάρας Κωνσταντίνος
Καθηγητής ΔΙ.ΠΑ.Ε.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή στις Choose a date.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Όνομα Επώνυμο	Όνομα Επώνυμο	Όνομα Επώνυμο
Choose an item.ΔΙ.ΠΑ.Ε.	Choose an item.ΔΙ.ΠΑ.Ε.	Choose an item.ΔΙ.ΠΑ.Ε.

Θεσσαλονίκη, Ιούνιος 2023

(Υπογραφή)

.....

Click here to enter text.

Click here to enter text.

«Στην σύζυγό μου Αγγελική και τον γιο μου Δημήτρη»

Καμπατζής Αριστοτέλης

«Στην οικογένεια και τους φίλους μου»

Σαρόγλου Στυλιανός

Η σελίδα αυτή είναι σκόπιμα λευκή.

Πρόλογος

Η διπλωματική μας εργασία ασχολείται με την εξόρυξη δεδομένων και την ανάλυση συναισθήματος, δύο αναπτυσσόμενα πεδία που έχουν μεγάλη επιρροή στη βιομηχανία και την κοινωνία. Αποσκοπεί να διερευνήσει την χρησιμότητα των αλγορίθμων μηχανικής μάθησης για την αναγνώριση συναισθήματος σε κείμενα και να δημιουργήσει ένα πλαίσιο εργασίας για την αξιολόγηση της απόδοσής τους. Οι λόγοι που επιλέξαμε αυτό το θέμα αφορούν τα εξής: Πρώτον, η εξόρυξη δεδομένων μας επιτρέπει να αντλήσουμε χρήσιμες πληροφορίες από μεγάλα σύνολα δεδομένων, κάτι που διαδραματίζει σημαντικό ρόλο στη λήψη αποφάσεων σε διάφορους τομείς, όπως ο οικονομικός, ο ιατρικός, ο επιχειρηματικός και πολλοί άλλοι. Δεύτερον, η ανάλυση συναισθημάτων μας βοηθά να κατανοήσουμε τις ανάγκες των ανθρώπων και των καταναλωτών, προκειμένου μελλοντικά να αναπτυχθούν προϊόντα και υπηρεσίες που ανταποκρίνονται στις εν λόγω ανάγκες. Τέλος, η συνδυασμένη χρήση εξόρυξης δεδομένων και ανάλυσης συναισθημάτων, μπορεί να παρέχει πολύτιμες προβλέψεις που βελτιώνουν την απόδοση μιας επιχείρησης και να οδηγήσει στην ανάπτυξη νέων ευκαιριών.

Περίληψη

Η εξόρυξη πληροφορίας και η ανάλυση συναισθημάτων σε κείμενα, είναι δύο σημαντικά πεδία στην επιστήμη της πληροφορικής και της τεχνητής νοημοσύνης. Αποτελούν ένα σημαντικό εργαλείο για την κατανόηση των στάσεων και των απόψεων που εκφράζονται σε κοινωνικά δίκτυα όπως το Twitter. Η χρήση μεθόδων μηχανικής μάθησης και σύγχρονων μοντέλων επεξεργασίας φυσικής γλώσσας επιτρέπει την αυτόματη ανάλυση του περιεχομένου κειμένων και την εξαγωγή σημαντικών πληροφοριών από αυτά, προσφέροντας παράλληλα ακρίβεια και ευκολία στην εξαγωγή συμπερασμάτων.

Στην παρούσα εργασία, αξιοποιούμε την χρήση του Twitter API για την συλλογή δεδομένων από το Twitter, σε συνδυασμό με την χρήση μεθόδων επεξεργασίας φυσικής γλώσσας (NLP). Ειδικότερα, χρησιμοποιούμε μοντέλα μηχανικής μάθησης της βιβλιοθήκης Scikit-learn, καθώς και πιο μοντέρνα μοντέλα όπως τα BERT, RoBERTa, DistilBERT και GPT-2 με σκοπό την αναγνώριση συναισθημάτων σε κείμενα (tweets) του κοινωνικού δικτύου Twitter, καθώς και σε κριτικές καταστημάτων που περιέχονται σε σύνολο δεδομένων της διαδικτυακής υπηρεσίας Skroutz.

Σύμφωνα με τα πειράματά μας, τα μοντέλα που σημειώνουν την καλύτερη απόδοση όσον αφορά την ακρίβεια (accuracy) πρόβλεψης σε νέα δεδομένα, είναι το BERT και το SVM σε συνδυασμό με την κωδικοποίηση TF-IDF.

Λέξεις Κλειδιά: Επεξεργασία Φυσικής Γλώσσας, Μηχανική Μάθηση, Βαθιά Μάθηση.

« Information extraction and sentiment analysis using machine learning methods and modern natural language processing models »

Saroglou Stylianos & Kampatzis Aristotelis

Abstract

Data Mining and Sentiment Analysis in texts are two important fields in Computer Science and Artificial Intelligence. They are a valuable tool for understanding attitudes and opinions expressed on social networks, such as Twitter. The use of Machine Learning methods and modern Natural Language Processing models allows for the automatic analysis of text content and the extraction of important information, while also offering, accuracy and convenience in drawing conclusions.

In this paper, we utilize the Twitter API for data collection from Twitter, in combination with Natural Language Processing (NLP) methods. Specifically, we use Machine Learning models from the Scikit-learn library, as well as more modern models, such as BERT, RoBERTa, DistilBERT, and GPT-2, with the aim of identifying sentiment in text from the Twitter social network, as well as in reviews of stores contained in a specific dataset from the Skroutz.gr online service.

According to our experiments, the models that show the best performance in terms of accuracy for predicting on new data, are BERT and SVM combined with the TF-IDF encoding.

Keywords: Twitter API, NLP, Machine Learning, Deep Learning, BERT, RoBERTa, DistilBERT, GPT-2, TF-IDF, Word2Vec, Transformers, TensorFlow, PyTorch, Keras, Scikit learn.

Ευχαριστίες

Α. Κ.

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή της διπλωματικής εργασίας κ. Διαμαντάρα Κωνσταντίνο για τις συμβουλές και την καθοδήγησή του σε όλη την διάρκεια εκπόνησης αυτής της εργασίας. Ακόμη, θα ήθελα να ευχαριστήσω την οικογένειά μου για την υποστήριξη που μου έδειξε σε όλη την διαδρομή και ολοκλήρωση των μεταπτυχιακών σπουδών μου.

Σ. Σ.

Μεγάλο ευχαριστώ χρωστάω στην οικογένειά μου, για την υποστήριξη και την υπομονή που έδειξε κατά την ολοκλήρωση των μεταπτυχιακών μου σπουδών. Αφιερώνω μεγάλο μέρος της εργασίας και στους φίλους μου, για να τους ευχαριστήσω για την ηθική τους στήριξη. Πολλές ευχαριστίες και στον κύριο Κωνσταντίνο Διαμαντάρα που μας καθοδήγησε άψογα στην εκπόνηση της εργασίας μας, δρώντας ως επιβλέπων καθηγητής.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κατάλογος Εικόνων

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
ΤΝ	Τεχνητή Νοημοσύνη
GPU	Graphics Processing Unit
TPU	Tensor Processing Unit
SVM	Support Vector Machine
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
BERT	Bidirectional Encoder Representations from Transformers
RoBERTa	Robustly Optimized BERT Approach
API	Application Programming Interface
JSON	JavaScript Object Notation
UTC	Coordinated Universal Time
WOEID	Where On Earth IDentifier
GPT-2	Generative Pre-trained Transformer 2
TF-IDF	Term Frequency–Inverse Document Frequency
CBOW	Continuous Bag Of Words
k-NN	k-Nearest Neighbors
CART	Classification And Regression Trees
ΤΝΔ	Τεχνητά Νευρωνικά Δίκτυα
MNB	Multinomial Naive Bayes
GNB	Gaussian Naive Bayes
NER	Named Entity Recognition

Κεφάλαιο 1ο: Εισαγωγή

1.1 Αντικείμενο και στόχοι διπλωματικής

Η παρούσα διπλωματική εργασία ερευνά την εξόρυξη δεδομένων αναζητώντας το συναίσθημα σε σύνολα κειμένου με την βοήθεια μεθόδων μηχανικής μάθησης. Στόχος της διπλωματικής εργασίας, είναι η μελέτη και η σύγκριση μοντέλων μηχανικής και βαθιάς μάθησης, τα οποία μαθαίνουν από τα δεδομένα προκειμένου να επιτύχουν υψηλή ακρίβεια ανίχνευσης συναισθήματος σε μελλοντικά δεδομένα. Ειδικότερα, για την διεργασία εξόρυξης επιλέξαμε το κοινωνικό δίκτυο Twitter διότι έχει αναδειχθεί ως μια από τις πιο δημοφιλείς πλατφόρμες για την έκφραση συναισθημάτων στο Διαδίκτυο. Έτσι, χρησιμοποιήσαμε το Twitter API προκειμένου να κάνουμε ανάκτηση tweets για τα πολιτικά κόμματα της Ελλάδας με σκοπό την δημιουργία ενός συνόλου δεδομένων (dataset) το οποίο θα χρησιμοποιηθεί από μοντέλα μηχανικής μάθησης για την πρόβλεψη συναισθήματος. Επιπλέον, ασχοληθήκαμε με ένα ακόμη σύνολο δεδομένων που περιλαμβάνει ελληνικά tweets σχετικά με αξιολογήσεις καταστημάτων στον ιστότοπο του Skroutz. Καθώς είναι δύσκολο να γνωρίζουμε εκ των προτέρων ποιο μοντέλο θα αποδώσει καλύτερα με τα σύνολα δεδομένων μας, θα δοκιμάσουμε διάφορα μοντέλα και θα παρατηρήσουμε για κάθε Dataset, ποιο μοντέλο σημειώνει μεγαλύτερη ακρίβεια (accuracy).

1.2 Συνεισφορά

Η συνεισφορά της διπλωματικής εργασίας συνοψίζεται ως εξής:

- Μελέτη έργων ανάλυσης συναισθήματος (Sentiment Analysis).
- Χρήση του Twitter API για την εξόρυξη tweets στην Ελληνική γλώσσα.
- Δημιουργία Dataset με tweets Ελληνικών πολιτικών κομμάτων.
- Χρήση Dataset με ελληνικές κριτικές καταστημάτων της υπηρεσίας Skroutz με σκοπό την κατηγοριοποίησή τους σε θετικά ή αρνητικά.
- Στατιστική ανάλυση και προεπεξεργασία των δεδομένων.
- Χρήση προ-εκπαιδευμένων ενσωματώσεων λέξεων (Word2Vec) σε συνδυασμό με μοντέλα μηχανικής μάθησης με σκοπό την ανάλυση συναισθήματος.
- Χρήση της τεχνικής TF-IDF για την αναπαράσταση κειμένων σε διανύσματα αριθμών σε συνδυασμό με μοντέλα μηχανικής μάθησης.
- Χρήση Transformers. Ένα είδος αλγορίθμων μηχανικής μάθησης που χρησιμοποιούνται για την επεξεργασία φυσικής γλώσσας, όπως το BERT της Google, το RoBERTa της εταιρείας Facebook, το DistilBERT της εταιρείας Hugging Face και το GPT-2 της OpenAI.
- Καταγραφή πειραμάτων και αποτελεσμάτων ακρίβειας πρόβλεψης.
- Σύγκριση μοντέλων σε διαφορετικά Datasets.
- Δημιουργία εφαρμογής με την χρήση της βιβλιοθήκης Tkinter, που παρέχει ένα πακέτο εργαλείων για την δημιουργία γραφικών διεπαφών χρήστη (GUI).

1.3 Δομή εργασίας

Η εργασία μας είναι οργανωμένη στα παρακάτω κεφάλαια:

Στο Κεφάλαιο 2, εξετάζεται η Επεξεργασία Φυσικής Γλώσσας (NLP) καθώς και οι βιβλιοθήκες που την απαρτίζουν όπως η TextBlob και η Spacy. Επίσης, γίνεται εκτενής ανάλυση σημαντικών διεργασιών προ-επεξεργασίας δεδομένων, όπως η διαίρεση ενός κειμένου σε tokens, η κανονικοποίηση, η απαλοιφή συνηθισμένων όρων (stop words), η λημματοποίηση κλπ.

Στο Κεφάλαιο 3, περιγράφονται έννοιες όπως το κοινωνικό δίκτυο Twitter, το Twitter API καθώς και η νεότερη έκδοση v2 του Twitter. Επίσης, δίνονται οδηγίες δημιουργίας εφαρμογής Twitter μέσω της οποίας μπορούμε να έχουμε πρόσβαση σε όλες τις μεθόδους ανάκτησης δεδομένων.

Στο Κεφάλαιο 4, συζητάμε για βασικές έννοιες και τεχνικές που χρησιμοποιούνται στην ανάπτυξη μοντέλων μηχανικής μάθησης, όπως κωδικοποίηση TF-IDF και Word2Vec, παρουσιάζουμε τα μοντέλα που χρησιμοποιήσαμε στην ανάλυση συναισθήματος και εξετάζουμε μετρικές ακρίβειας που χρησιμοποιούνται για να αξιολογηθεί η απόδοσή τους.

Στο Κεφάλαιο 5, εξετάζουμε τα μοντέλα βαθιάς μάθησης. Εστιάζουμε στα νευρωνικά δίκτυα πρόσθιας τροφοδότησης σήματος, στα αναδρομικά δίκτυα, καθώς και στα πιο σύγχρονα μοντέλα, τα λεγόμενα transformers που αναφέραμε και στη σύνοψη της συνεισφοράς.

Στο Κεφάλαιο 6, παρουσιάζουμε την μεθοδολογία που ακολουθήσαμε. Αρχικά, εξετάζουμε την διαδικασία εξόρυξης δεδομένων (tweets) που εφαρμόσαμε. Στην συνέχεια, αναλύουμε την προ-επεξεργασία των δεδομένων, περιγράφοντας τα βήματα που ακολουθήσαμε για τον καθαρισμό τους. Παρουσιάζουμε μια βιβλιοθήκη γεωκωδικοποίησης και χαρτογράφησης για απεικόνιση των τοποθεσιών των tweets που συλλέχθηκαν. Αναφέρουμε τις υπερ-παραμέτρους των μοντέλων και τέλος αξιολογούμε τα αποτελέσματα των πειραμάτων που εκτελέσαμε.

Στο Κεφάλαιο 7, καταγράψαμε τα συμπεράσματα που προέκυψαν από την μελέτη μας, καθώς και συστάσεις για πιθανές βελτιώσεις της εργασίας.

1.4 Τεχνολογία και εργαλεία ανάπτυξης

Για την ανάπτυξη του κώδικα της παρούσας εργασίας χρησιμοποιήθηκε η γλώσσα προγραμματισμού Python και τα περιβάλλοντα ανάπτυξης Kaggle, Colab, καθώς και το JupyterLab.

1.4.1 Η γλώσσα προγραμματισμού Python

Η Python είναι μια αντικειμενοστραφής γλώσσα σεναρίων που κυκλοφόρησε το 1991 και αναπτύχθηκε από τον Guido Van Rossum [1]. Έγινε πολύ γρήγορα μια από τις δημοφιλέστερες γλώσσες προγραμματισμού ιδιαίτερα στην εκπαιδευτική και επιστημονική πληροφορική [1].

Τα κύρια χαρακτηριστικά της είναι τα εξής:

- Είναι πιο εύκολη στην εκμάθηση από τις γλώσσες Java και C++.
- Είναι ανοικτού κώδικα και ευρέως διαθέσιμη.
- Τα τελευταία χρόνια άρχισε να αποκτά δημοφιλία στην ανάπτυξη ιστοτόπων όπως το Django.
- Χρησιμοποιείται για την δημιουργία απλών αλλά και πολύπλοκων σεναρίων (YouTube και Instagram).
- Χρησιμοποιείται ευρέως στην εκπαίδευση.
- Είναι δημοφιλής στην τεχνητή νοημοσύνη λόγω της σχέσης της με την επιστήμη δεδομένων.
- Χρησιμοποιείται σε εφαρμογές μηχανικής και βαθιάς μάθησης.

1.4.2 Kaggle

Το Kaggle είναι μια διαδικτυακή πλατφόρμα για επιστήμονες δεδομένων και λάτρεις της μηχανικής μάθησης [2]. Επιτρέπει στους χρήστες να συνεργάζονται μεταξύ τους, να δημοσιεύουν σύνολα δεδομένων και να αναπτύσσουν κώδικα Python σε ενσωματωμένα σημειωματάρια. Στόχος της πλατφόρμας, είναι να βοηθάει επαγγελματίες και ερευνητές προκειμένου να επιλύουν προβλήματα της επιστήμης δεδομένων με ισχυρά εργαλεία και πόρους [2]. Σημαντικό χαρακτηριστικό που ανέδειξε

την εν λόγω πλατφόρμα είναι οι διαγωνισμοί που λαμβάνουν χώρα, με σκοπό τον ανταγωνισμό επιστημόνων δεδομένων σε διάφορες προκλήσεις και προβλήματα μηχανικής μάθησης [2].

Το Kaggle παρέχει ισχυρούς πόρους GPU και TPU στο Cloud. Έτσι, δίνει την δυνατότητα σε προγραμματιστές και επιστήμονες δεδομένων να χρησιμοποιούν GPU έως και 30 ώρες την εβδομάδα, και TPU έως και 20 ώρες την εβδομάδα [2].

1.4.3 Colab

Το Colab [3] είναι και αυτό μια διαδικτυακή πλατφόρμα που επιτρέπει την ανάπτυξη κώδικα Python στο πρόγραμμα περιήγησης χωρίς ειδική διαμόρφωση και διαθέτει πρόσβαση σε GPU και TPU όπως και το Kaggle. Επίσης, είναι κατάλληλο για μηχανική μάθηση και ανάλυση δεδομένων.

1.4.4 JupyterLab

Το τρίτο περιβάλλον ανάπτυξης που χρησιμοποιήσαμε για την εργασία μας, είναι τα τετράδια JupyterLab. Σε ένα τετράδιο Jupyter γράφουμε τον πηγαίο κώδικα Python, τον αποθηκεύουμε στην μορφή "filename.ipynb" και τον εκτελούμε στον φυλλομετρητή μας. Τα τετράδια Jupyter είναι λογισμικά ανοικτού κώδικα και συνδυάζουν κείμενο, εικόνα, βίντεο και ήχο [1].

Η χρήση του JupyterLab, απαιτεί την εγκατάσταση της διανομής Anaconda η οποία περιλαμβάνει τις περισσότερες βιβλιοθήκες Python, οπτικοποίησης και επιστήμης δεδομένων που απαιτούνται. Πληροφορίες εγκατάστασης της διανομής Anaconda παραθέτουμε στο Παράρτημα Β.

Κεφάλαιο 2ο: Επεξεργασία Φυσικής Γλώσσας (NLP)

2.1 Εισαγωγή

Η επεξεργασία φυσικής γλώσσας (NLP), είναι ένας διεπιστημονικός κλάδος της επιστήμης της πληροφορικής, της υπολογιστικής γλωσσολογίας, και της τεχνητής νοημοσύνης. Ασχολείται με την αλληλεπίδραση υπολογιστών με φυσικές γλώσσες και εφαρμόζεται κυρίως σε συλλογές κειμένων (corpus), οι οποίες απαρτίζονται από τιτιβίσματα (tweets), αναρτήσεις του Facebook, κριτικές κινηματογράφου, ειδήσεις και πολλά άλλα [1].

2.2 Πώς λειτουργεί η επεξεργασία φυσικής γλώσσας

Συνίσταται στην εφαρμογή μιας σειράς τεχνικών που σκοπό έχουν να αντιστοιχίσουν το κείμενο σε έναν συμβολισμό κατανοητό από μια μηχανή. Αρχικά, η επεξεργασία φυσικής γλώσσας αρχίζει με την αναγνώριση και τον διαχωρισμό των λέξεων στο κείμενο. Στη συνέχεια, οι λέξεις αντιστοιχίζονται με τις αντίστοιχες μορφές τους (π.χ. τον ενικό ή τον πληθυντικό αριθμό), ενώ οι προτάσεις διαχωρίζονται και αναλύονται σε σημασιολογικές μονάδες (υποκείμενο, κατηγορούμενο, χρόνος). Στην πραγματικότητα για να μπορέσουν τα μοντέλα μηχανικής μάθησης να αναλύσουν τα δεδομένα κειμένου θα πρέπει να γίνει προεπεξεργασία με σκοπό την διατήρηση μόνο των σημαντικών λέξεων και την απαλοιφή σημείων στίξης ή και άλλου ενδεχόμενου θορύβου που υπάρχει στο κείμενο. Τέλος, το κείμενο θα μετατραπεί σε ένα διάνυσμα αριθμητικών χαρακτηριστικών και θα δοθεί στην είσοδο ενός μοντέλου. Αν το πρόβλημα είναι η ταξινόμηση του κειμένου σε μια κατηγορία (π.χ. Negative ή Positive), το μοντέλο θα προσπαθήσει να προβλέψει την ορθή κατηγορία στην οποία ανήκει το κείμενο που παρέλαβε.

2.3 Οφέλη NLP

Η επεξεργασία φυσικής γλώσσας (NLP), παρέχει πολλά οφέλη, μερικά από τα οποία είναι:

- Βελτιωμένη επικοινωνία ανθρώπου-υπολογιστή. Οι υπολογιστές μπορούν να κατανοήσουν την ανθρώπινη γλώσσα, κάνοντας την αλληλεπίδραση πιο διαισθητική για τους ανθρώπους.
- Βελτιωμένη ακρίβεια και αποτελεσματικότητα επεξεργασίας δεδομένων. Οι αλγόριθμοι NLP μπορούν να επεξεργαστούν μεγάλους όγκους πληροφοριών γρηγορότερα και αποτελεσματικότερα.
- Δυνατότητα αυτόματης δημιουργίας ευανάγνωστων περιλήψεων ενός μεγαλύτερου, πιο σύνθετου πρωτότυπου κειμένου.
- Χρήσιμο για προσωπικούς βοηθούς όπως η Alexa, επιτρέποντάς της να κατανοεί τον προφορικό λόγο.
- Ευκολότερη ανάλυση συναισθημάτων στο κείμενο.
- Βελτιώνει την ακρίβεια της αναζήτησης και της ταξινόμησης περιεχομένου.

2.4 Προκλήσεις NLP

Η επεξεργασία φυσικής γλώσσας αντιμετωπίζει πολλές προκλήσεις, με την κύρια να συνίσταται στο γεγονός ότι η φυσική γλώσσα είναι μια δυναμική και διαρκώς εξελισσόμενη μορφή επικοινωνίας.

Ορισμένες προκλήσεις της επεξεργασίας φυσικής γλώσσας είναι οι εξής:

- **Συντακτικά λάθη**. Η φυσική γλώσσα είναι πολύπλοκη και οι άνθρωποι συχνά κάνουν συντακτικά λάθη τα οποία μπορούν να δυσκολέψουν την αυτόματη επεξεργασία της γλώσσας.
- **Πολυσημία**. Οι λέξεις μπορούν να έχουν πολλαπλές σημασίες ανάλογα με το πλαίσιο στο οποίο χρησιμοποιούνται, και αυτό μπορεί να δημιουργήσει σύγχυση και λανθασμένες ερμηνείες.
- **Αναφορική ασάφεια**. Συχνά χρησιμοποιούμε αναφορές στο κείμενο για να αναφερθούμε σε κάτι που αναφέρθηκε προηγουμένως. Η κατανόηση αυτών των αναφορών απαιτεί κατανόηση του πλαισίου και της σημασίας του προηγούμενου κειμένου, κάτι που καθιστά ιδιαίτερα δύσκολη την ανάλυση και κατανόηση τέτοιων περιπτώσεων.
- **Σύνθετο λεξιλόγιο**. Η χρήση περίπλοκων λέξεων ή σύνθετου λεξιλογίου μπορεί να δυσκολέψει την κατανόηση της πληροφορίας.
- **Πολυγλωσσικότητα**. Κάθε γλώσσα έχει τις δικές της δομές και κανόνες, καθιστώντας την επεξεργασία πολυγλωσσικών κειμένων ιδιαίτερα δύσκολη.

2.5 Εργαλειοθήκη φυσικής γλώσσας (Natural Language Toolkit - NLTK)

Η εργαλειοθήκη φυσικής γλώσσας NLTK, είναι μια συλλογή βιβλιοθηκών για επεξεργασία φυσικής γλώσσας (NLP), γραμμένη στη γλώσσα προγραμματισμού Python [4]. Αναπτύχθηκε από τους Edward Loper, Ewan Klein και Steven Bird στο Τμήμα Επιστήμης Υπολογιστών και Πληροφορικής του Πανεπιστημίου της Πενσυλβάνια [4]. Από τότε και έπειτα, έχει επεκταθεί με την βοήθεια δεκάδων συντελεστών. Χρησιμοποιείται πλέον σε δεκάδες πανεπιστήμια και χρησιμεύει ως βάση πολλών ερευνητικών προγραμμάτων [4]. Το NLTK ορίζει μια υποδομή που μπορεί να χρησιμοποιηθεί για τη δημιουργία προγραμμάτων NLP στην Python. Παρέχει βασικές κλάσεις για την αναπαράσταση δεδομένων που σχετίζονται με την επεξεργασία φυσικής γλώσσας και διεπαφές για την εκτέλεση εργασιών όπως η ταξινόμηση κειμένου, η επισήμανση μερών του λόγου και η συντακτική ανάλυση [4]. Το NLTK συνοδεύεται από εκτενή τεκμηρίωση. Ο ιστότοπος στη διεύθυνση http://nltk.org/ παρέχει αναλυτική τεκμηρίωση API που καλύπτει κάθε ενότητα, κλάση και συνάρτηση στην εργαλειοθήκη, προσδιορίζοντας παραμέτρους και δίνοντας παραδείγματα χρήσης [4]. Στα μοντέλα της παρούσας εργασίας γίνεται χρήση του αρθρώματος nltk.corpus, του πακέτου stopwords και της μεθόδου words, για την αφαίρεση συνηθισμένων ελληνικών λέξεων από την συλλογή κειμένων.

2.6 Η βιβλιοθήκη TextBlob

Η TextBlob είναι μια αντικειμενοστραφής βιβλιοθήκη NLP για επεξεργασία κειμένου με πολλές δυνατότητες και βασίζεται στην βιβλιοθήκη NLTK [1]. Για πληροφορίες εγκατάστασης, δείτε το Παράρτημα Β. Κάποιες από τις εργασίες που μπορεί να εκτελέσει είναι οι παρακάτω:

- **Διαίρεση σε λεκτικές μονάδες – tokens (tokenization)**. Διαχωρισμός του κειμένου σε τμήματα που ονομάζονται tokens. Κάθε token αναπαριστά μία λέξη ή έναν αριθμό [1].
- **Περιστολή (stemming)**. Η εργασία της περιστολής αφορά μια "ωμή" ευρετική διαδικασία, η οποία αφαιρεί τις καταλήξεις των λέξεων και επιστρέφει την λέξη στη βάση ή ρίζα της. Π.χ. η βάση της αγγλικής λέξης varieties είναι varieti. Η βάση των ελληνικών λέξεων "παχύς" και "πηγαίνω" είναι "παχ" και "πηγαιν" αντίστοιχα [5].
- **Λημματοποίηση (lemmatization)**. Η λημματοποίηση είναι ένας ορθότερος τρόπος χειρισμού, διότι περιλαμβάνει την χρήση λεξιλογίου και την μορφολογική ανάλυση των λέξεων. Στοχεύει στην περικοπή μόνο των κλιτικών καταλήξεων και επιστρέφει την βασική μορφή της λέξης, δηλαδή το λήμμα. Π.χ. η λημματοποιημένη μορφή της αγγλικής λέξης varieties είναι variety. Η λημματοποιημένη μορφή των ελληνικών λέξεων "βλέπουμε" και "κατηγορεί" είναι "βλέπω" και "κατηγορώ" αντίστοιχα [5].

- **Απαλοιφή συνηθισμένων όρων – stop words**. Αφαίρεση συνηθισμένων λέξεων οι οποίες δεν συμβάλλουν σημαντικά στην επεξεργασία φυσικής γλώσσας ώστε να αναλυθούν μόνο σημαντικές λέξεις σε μια συλλογή κειμένων. Οι λέξεις αυτές ονομάζονται διακόπτουσες λέξεις (stop words). Λέξεις χωρία αξία στα Αγγλικά όπως "a", "the", "i", "you" κλπ. Λέξεις χωρίς αξία στα Ελληνικά όπως "που", "θα", "για" και άλλες [5].
- **Συχνότητα λέξεων (word frequency)**. Εύρεση του πόσο συχνά εμφανίζεται κάθε λέξη σε μια συλλογή κειμένων [1].
- **Μέρη του λόγου**. Αναγνώριση του μέρους του λόγου κάθε λέξης, όπως επίθετο, ρήμα κλπ.[1]
- **Έλεγχος ορθογραφίας** (spell check).
- **Διαγλωσσική μετάφραση** (inter-language translation) και **αναγνώριση γλώσσας** (language detection). Με την βοήθεια της Google Translate [1].
- **Κλίση (inflection)**. Σχηματισμός ενικού και πληθυντικού λέξεων [1].
- **N-γράμματα (N-grams)**. Δημιουργία διαδοχικών λέξεων σε μια συλλογή κειμένων προκειμένου να γίνεται αναγνώριση λέξεων που βρίσκονται η μια δίπλα στην άλλη [1].
- **Ανάλυση συναισθήματος (sentiment analysis)**. Κατηγοριοποίηση – ταξινόμηση ενός κειμένου για το εάν αυτό περιέχει θετικό ή αρνητικό συναίσθημα [1].

2.6.1 Χρήση της κλάσης TextBlob

Για την δημιουργία TextBlob χρησιμοποιείται η βασική κλάση TextBlob για NLP από το άρθρωμα textblob. Οι επόμενες γραμμές κώδικα Python στο Σενάριο 2.1 δημιουργούν ένα TextBlob.

```
[1]: from textblob import TextBlob

[2]: txt = "Hello world. Happy new year."

[3]: my_blob = TextBlob(txt)

[4]: my_blob

[4]: TextBlob("Hello world. Happy new year.")
```

Σενάριο 2.1: Δημιουργία TextBlob

Να σημειωθεί ότι η TextBlob κληρονομεί από την κλάση BaseBlob. Υποστηρίζει μεθόδους συμβολοσειρών και κάθε TextBlob μπορεί να συγκριθεί με άλλες συμβολοσειρές μέσω τελεστών σύγκρισης [1]. Στις επόμενες ενότητες θα αναφερθούμε στις κλάσεις Sentence και Word οι οποίες επίσης κληρονομούν από την BaseBlob. Η BaseBlob είναι μια αφηρημένη βασική κλάση από την οποία θα κληρονομήσουν όλες οι κλάσεις TextBlob [1].

2.6.2 Διαίρεση κειμένου σε προτάσεις

Πολλές φορές οι εργασίες επεξεργασίας φυσικής γλώσσας απαιτούν την διαίρεση του κειμένου σε προτάσεις. Η TextBlob περιλαμβάνει χρήσιμες ιδιότητες για την προσπέλαση προτάσεων επάνω σε αντικείμενα TextBlob. Μια βασική ιδιότητα είναι η sentences η οποία επιστρέφει μια λίστα

αντικειμένων Sentence [1]. Στον κώδικα Python του Σεναρίου 2.2, το αρχικό κείμενο "Hello world. Happy new year." χωρίζεται σε δύο προτάσεις.

```
[1]: from textblob import TextBlob

[2]: txt = "Hello world. Happy new year."

[3]: my_blob = TextBlob(txt)

[4]: my_blob

[4]: TextBlob("Hello world. Happy new year.")

[5]: my_blob.sentences

[5]: [Sentence("Hello world."), Sentence("Happy new year.")]
```

Σενάριο 2.2: Διαχωρισμός κειμένου σε δύο προτάσεις Sentence

2.6.3 Διαίρεση κειμένου σε λέξεις

Μια πιο σημαντική εργασία από τον χωρισμό κειμένου σε προτάσεις, είναι η διαίρεση κειμένου σε λέξεις πριν από την πραγματοποίηση άλλων ενεργειών NLP. Για την προσπέλαση λέξεων η TextBlob διαθέτει την ιδιότητα words η οποία επιστρέφει ένα αντικείμενο WordList [1]. Το εν λόγω αντικείμενο περιλαμβάνει μια λίστα αντικειμένων Word, τα οποία αντιπροσωπεύουν την κάθε λέξη μέσα στο TextBlob. Το Σενάριο 2.3 δείχνει την χρήση της ιδιότητας words.

```
[1]: from textblob import TextBlob

[2]: txt = "Hello world. Happy new year."

[3]: my_blob = TextBlob(txt)

[4]: my_blob

[4]: TextBlob("Hello world. Happy new year.")

[6]: my_blob.words

[6]: WordList(['Hello', 'world', 'Happy', 'new', 'year'])
```

Σενάριο 2.3: Διαίρεση κειμένου σε λέξεις

Αξιοσημείωτο εδώ είναι ότι η ιδιότητα words επιστρέφει μια λίστα λέξεων αφού πρώτα αφαιρέσει το σύμβολο της τελείας. Επίσης, αφαιρεί οποιοδήποτε σημείο στίξης εάν υπάρχει στο κείμενο.

2.6.4 Κανονικοποίηση (Normalization)

Η διαδικασία της κανονικοποίησης σχετίζεται με την προετοιμασία των λέξεων για ανάλυση. Για παράδειγμα, σε έναν υπολογισμό στατιστικών στοιχείων είναι σύνηθες να μετατρέπονται όλες οι λέξεις σε πεζά γράμματα προκειμένου να αντιμετωπίζονται με τον ίδιο τρόπο λέξεις με κεφαλαία και πεζά γράμματα. Σε άλλες περιπτώσεις θέλουμε να χρησιμοποιήσουμε την ρίζα μιας λέξης για να εκπροσωπήσουμε τις διάφορες μορφές της [1]. Δύο βασικές μέθοδοι κανονικοποίησης όπως αναφέρθηκε, είναι η **περιστολή (stemming)** και η **λημματοποίηση (lemmatization)**. Στο Dataset ελληνικών πολιτικών tweets "unbalanced_politics.csv", χρησιμοποιούμε λημματοποίηση κάνοντας χρήση του **Simplemma** [6]. Το Simplemma παρέχει μια απλή και πολύγλωσση προσέγγιση για την αναζήτηση βασικών μορφών ή λημμάτων. Είναι γενικά, εύκολο στην εγκατάσταση και απλό στη χρήση. Επί του παρόντος, υποστηρίζονται 48 γλώσσες ανάμεσά τους και τα Ελληνικά [6]. Η κλάση Word υποστηρίζει stemming και lemmatization μέσω των μεθόδων stem() και lemmatize() σε αγγλικές λέξεις αλλά όχι σε ελληνικές. Το Σενάριο 2.4 δείχνει το stemming και το lemmatization για την λέξη "studies".

```
[44]: from textblob import Word

[45]: word = Word("studies")

[46]: word.stem()

[46]: 'studi'

[47]: word.lemmatize()

[47]: 'study'
```

Σενάριο 2.4: Stemming & Lemmatization της λέξης studies

2.6.5 Απαλοιφή συνηθισμένων λέξεων (Stop Words)

Συνηθισμένες λέξεις ή stop words είναι λέξεις σε μια συλλογή κειμένων, οι οποίες είναι καλό να αφαιρούνται επειδή τυπικά δεν παρέχουν χρήσιμες πληροφορίες [7]. Συνεπώς, θα γίνεται ανάλυση μόνο των σημαντικών λέξεων του κειμένου. Η βιβλιοθήκη NLTK έχει λίστες συνηθισμένων λέξεων για διάφορες γλώσσες [7]. Ο παρακάτω κώδικας (Σενάριο 2.5) εισάγει το πακέτο nltk και κατεβάζει το πακέτο **stopwords**. Κατόπιν, από το **άρθρωμα nltk.corpus** κάνει import το πακέτο stopwords και χρησιμοποιεί την μέθοδο words() για να φορτώσει τη λίστα συνηθισμένων λέξεων 'greek'. Έτσι, η λίστα stop_words που προκύπτει περιλαμβάνει ελληνικές λέξεις χωρίς αξία. Επειδή η λίστα stop_words είναι αρκετά μεγάλη, φαίνεται μόνο ένα μέρος αυτής.

```
[50]: import nltk

[51]: nltk.download('stopwords')

      [nltk_data] Downloading package stopwords to
      [nltk_data]     C:\Users\ARISTOTELIS\AppData\Roaming\nltk_data...
      [nltk_data]   Package stopwords is already up-to-date!
[51]: True

[52]: # Για αφαίρεση συνηθισμένων λέξεων
      from nltk.corpus import stopwords
      stop_words = stopwords.words('greek')

[53]: stop_words

[53]: ['αλλα',
       'αν',
       'αντι',
       'απο',
       'αυτα',
       'αυτεσ',
       'αυτη',
       'αυτο',
       'αυτοι',
       'αυτοσ',
```

Σενάριο 2.5: Ελληνικές λέξεις Stop Words

2.6.6 Συχνότητα λέξεων (Word Frequency)

Η TextBlob διαθέτει το λεξικό **word_counts** για να έχει πρόσβαση στις συχνότητες των λέξεων. Για να δείξουμε τη χρήση του εν λόγω λεξικού, κατεβάσαμε το ηλεκτρονικό βιβλίο "Napoleon and his court by C. S. Forester.txt" από τον ιστότοπο https://www.gutenberg.org/ebooks/69585, και έπειτα το φορτώσαμε με την βοήθεια της κλάσης **Path**. Για την ανάγνωσή του χρησιμοποιούμε την μέθοδο **read_text()**. Στον επόμενο κώδικα, βρίσκουμε τις συχνότητες των λέξεων "napoleon", "bonaparte", "france" και των stop words "a" και "the" οι οποίες εμφανίζονται πάνω από 1000 φορές μέσα στο κείμενο. Συνεπώς οι λέξεις "a" και "the" θα πρέπει να αφαιρεθούν αν προχωρήσουμε σε περαιτέρω ανάλυση του κειμένου.

```
[54]: from pathlib import Path

[55]: from textblob import TextBlob

[56]: blob = TextBlob(Path('Napoleon_and_his_court.txt').read_text())

[57]: blob.word_counts['napoleon']

[57]: 866

[58]: blob.word_counts['bonaparte']

[58]: 84

[59]: blob.word_counts['france']

[59]: 101

[60]: blob.word_counts['a']

[60]: 1462

[61]: blob.word_counts['the']

[61]: 4653
```

Σενάριο 2.6: Χρήση λεξικού word_counts

2.6.7 Μέρη του λόγου

Η TextBlob διαθέτει την ιδιότητα **tags**, η οποία επιστρέφει μια λίστα πλειάδων. Κάθε πλειάδα περιέχει μια λέξη και μια συμβολοσειρά η οποία αναπαριστά το μέρος του λόγου της [1]. Κάποιες λέξεις έχουν πολλές έννοιες. Για παράδειγμα η λέξη "run", η οποία μπορεί να είναι επίθετο ή και ρήμα. Ο προσδιορισμός της έννοιας μιας λέξης είναι σημαντικός προκειμένου να εντοπίσουμε την ορθή έννοια με βάση τα συμφραζόμενα [1]. Επίσης βοηθά τους υπολογιστές να κατανοήσουν την φυσική γλώσσα. Στον παρακάτω κώδικα (Σενάριο 2.7), φαίνεται η χρήση της ιδιότητας tags. Για την πρόταση "Happy new year.", η ιδιότητα tags επιστρέφει μια λίστα με τρεις πλειάδες όπου η καθεμία περιλαμβάνει την κάθε λέξη και την σήμανση JJ και NN. Η λέξεις "Happy" και "new" έχουν την σήμανση JJ που σημαίνει ότι οι λέξεις είναι επίθετα. Η λέξη "year" έχει σήμανση NN που σημαίνει ότι πρόκειται για ουσιαστικό. Η TextBlob χρησιμοποιεί σημάνσεις μερών του λόγου της βιβλιοθήκης **pattern** (https://www.clips.uantwerpen.be/pattern).

```
[11]: from textblob import TextBlob

[12]: txt = "Happy new year."

[13]: my_blob = TextBlob(txt)

[14]: my_blob

[14]: TextBlob("Happy new year.")

[15]: my_blob.tags

[15]: [('Happy', 'JJ'), ('new', 'JJ'), ('year', 'NN')]
```

Σενάριο 2.7: Προσδιορισμός μέρους του λόγου με την ιδιότητα tags

2.6.8 Έλεγχος ορθογραφίας (Spell Check)

Στις εργασίες επεξεργασίας φυσικής γλώσσας πολλές φορές είναι απαραίτητο η συλλογή κειμένων να μην έχει ορθογραφικά λάθη. Λογισμικά επεξεργασίας κειμένου όπως τα **Microsoft Word** και **Google Docs**, ελέγχουν αυτόματα την ορθογραφία του κειμένου κατά την πληκτρολόγηση και εντοπίζουν ανορθόγραφες λέξεις [1]. Η κλάση Word ελέγχει την ορθογραφία λέξεων με την βοήθεια της μεθόδου **spellcheck()**. Η εν λόγω μέθοδος επιστρέφει μια λίστα πλειάδων οι οποίες περιέχουν πιθανές ορθές λέξεις και μια τιμή εμπιστοσύνης [1]. Στο παρακάτω πρόγραμμα, υποθέτουμε ότι θέλουμε να γράψουμε την λέξη "they", αλλά την γράψαμε λανθασμένα ως "theyr". Αφού γίνει ο έλεγχος της λέξης με την μέθοδο spellcheck(), θα επιστραφούν δυο πιθανές διορθώσεις όπου η λέξη "they" θα έχει την υψηλότερη τιμή εμπιστοσύνης [1]. Προφανώς και η λέξη με την υψηλότερη τιμή εμπιστοσύνης δεν αποτελεί απαραίτητα και την ορθή λέξη. Οι κλάσεις Sentence, Word και TextBlob διαθέτουν μια μέθοδο correct() η οποία διορθώνει την ορθογραφία [1]. Η κλήση της μεθόδου correct() για ένα αντικείμενο Word, επιστρέφει ως ορθή λέξη αυτή με την υψηλότερη τιμή εμπιστοσύνης που δίνει η spellcheck() [1].

```
[1]: from textblob import Word

[2]: word = Word('theyr')

[3]: word.spellcheck()

[3]: [('they', 0.5713042216741622), ('their', 0.42869577832583783)]

[4]: word.correct()

[4]: 'they'
```

Σενάριο 2.8: Μέθοδοι spellcheck() και correct()

Η μέθοδος correct() για ένα αντικείμενο TextBlob ή Sentence, ελέγχει την ορθογραφία σε μία ολόκληρη πρόταση και όχι μόνο σε μια λέξη. Η correct() αντικαθιστά κάθε ανορθόγραφη λέξη, μέσα στην πρόταση, με την ορθογραφημένη η οποία έχει την υψηλότερη τιμή εμπιστοσύνης [1].

```
[19]: from textblob import TextBlob

[20]: sentence = TextBlob('Happo nep yiar')

[21]: sentence.correct()

[21]: TextBlob("Happy new year")
```

Σενάριο 2.9: Μέθοδος correct() σε TextBlob

2.6.9 Διαγλωσσική μετάφραση (Inter-Language Translation) και αναγνώριση γλώσσας (Language Detection)

Η διαγλωσσική μετάφραση αποτελεί μια μεγάλη πρόκληση στην **επεξεργασία φυσικής γλώσσας** και την **τεχνητή νοημοσύνη**. Υπηρεσίες όπως η **Google Translate** και η **Microsoft Bing Translator** εκτελούν απευθείας μετάφραση στο κείμενο εισόδου [7]. Η διαγλωσσική μετάφραση είναι επίσης σημαντική για ανθρώπους που ταξιδεύουν σε χώρες του εξωτερικού. Έχουν γίνει πολλές προσπάθειες για μετάφραση ζωντανής ομιλίας με στόχο την συνομιλία μεταξύ ανθρώπων που δεν μιλούν την ίδια γλώσσα [7]. Είναι διαθέσιμη η βιβλιοθήκη βαθιάς μετάφρασης **deep-translator Library**, η οποία υποστηρίζει πολλές μεταφραστικές υπηρεσίες για να μεταφράσουμε ξενόγλωσσα κείμενα στην γλώσσα που επιθυμούμε [7]. Για την εγκατάσταση πρέπει να γράψουμε την οδηγία:

```
pip install -U deep_translator
```

Ένα παράδειγμα χρήσης της βιβλιοθήκης deep-translator φαίνεται στο Σενάριο 2.10. Το πρώτο βήμα είναι να φορτώσουμε την κλάση GoogleTranslator από την βιβλιοθήκη. Έπειτα δημιουργούμε ένα αντικείμενο GoogleTranslator με όνομα translatorEN. Το όρισμα source='auto', αναγνωρίζει την γλώσσα του κειμένου εισόδου, ενώ το target='en' καθορίζει την γλώσσα στόχου του κειμένου. Χρησιμοποιούμε την μέθοδο translate() δίνοντάς της το όρισμα text το οποίο είναι γραμμένο στα Ελληνικά, προκειμένου να γίνει η μετάφραση του κειμένου text στην Αγγλική γλώσσα. Τέλος, η μεταβλητή textEN περιέχει το μεταφρασμένο κείμενο, το οποίο το μετατρέπουμε σε TextBlob. Το εν λόγω TextBlob μπορούμε να το χρησιμοποιήσουμε αργότερα για άλλες εργασίες NLP, όπως για παράδειγμα την ανάλυση συναισθήματος με τον προεπιλεγμένο αναλυτή συναισθήματος της TextBlob.

```
[34]: from textblob import TextBlob

[35]: from deep_translator import GoogleTranslator

[36]: text = "Είναι καλό παιδί"

[37]: translatorEN = GoogleTranslator(source='auto', target='en')
      textEN = translatorEN.translate(text)

[38]: blob = TextBlob(textEN)

[39]: blob

[39]: TextBlob("He is a good guy")
```

Σενάριο 2.10: Χρήση της βιβλιοθήκης deep_translator

2.6.10 Κλίση (Inflection)

Η κλίση σχετίζεται με τις διαφορετικές μορφές της ίδιας λέξης όπως είναι ο ενικός και ο πληθυντικός. Σε εργασίες NLP που θέλουμε να υπολογίσουμε συχνότητες λέξεων συχνά θέλουμε να μετατρέψουμε όλες τις κλιτές λέξεις στην ίδια μορφή. Οι TextBlob, Word και WordList υποστηρίζουν την μετατροπή λέξεων στον ενικό και πληθυντικό αριθμό [7]. Το παρακάτω πρόγραμμα παρουσιάζει την χρήση των μεθόδων **pluralize()** και **singularize()** οι οποίες μετατρέπουν λέξεις σε πληθυντικό και ενικό αριθμό αντίστοιχα.

```
[50]: from textblob import Word

[51]: mouse = Word('mouse')

[52]: mouse.pluralize()

[52]: 'mice'

[53]: dogs = Word('dogs')

[54]: dogs.singularize()

[54]: 'dog'

 [ ]: #########

[58]: from textblob import TextBlob

[59]: animals = TextBlob('bird cat fish dog elephant').words

[60]: animals.pluralize()

[60]: WordList(['birds', 'cats', 'fish', 'dogs', 'elephants'])
```

Σενάριο 2.11: Χρήση μεθόδων pluralize() και singularize()

2.6.11 Ν-γράμματα (N-grams)

Ένα **Ν-γράμμα (N-gram)** είναι μια ακολουθία από Ν στοιχεία κειμένου εντός μιας πρότασης. Στην επεξεργασία φυσικής γλώσσας τα Ν-γράμματα χρησιμοποιούνται για την αναγνώριση λέξεων που εμφανίζονται η μία δίπλα στην άλλη [7]. Η TextBlob διαθέτει την μέθοδο **ngrams()** η οποία παράγει μια λίστα από WordList Ν-γράμματα με προεπιλεγμένο μήκος την τιμή τρία (τριγράμματα). Η μέθοδος προαιρετικά μπορεί να δεχτεί την παράμετρο n για να δημιουργήσει Ν-γράμματα οποιουδήποτε μήκους n [7]. Όπως φαίνεται και παρακάτω στο Σενάριο 2.12, αρχικά δημιουργείται το πρώτο τρίγραμμα που περιέχει τις τρεις πρώτες ημέρες της εβδομάδας, μετά δημιουργείται το δεύτερο τρίγραμμα που αρχίζει με την δεύτερη ημέρα κ.ο.κ., μέχρι τελικά να δημιουργηθεί το τελευταίο τρίγραμμα με τις τρεις τελευταίες ημέρες της εβδομάδας. Για n=5 δημιουργούνται 5-γράμματα.

```
[7]: from textblob import TextBlob

[8]: text = 'Monday Tuesday Wednesday Thursday Friday Saturday Sunday'

[9]: blob = TextBlob(text)

[10]: blob.ngrams()

[10]: [WordList(['Monday', 'Tuesday', 'Wednesday']),
       WordList(['Tuesday', 'Wednesday', 'Thursday']),
       WordList(['Wednesday', 'Thursday', 'Friday']),
       WordList(['Thursday', 'Friday', 'Saturday']),
       WordList(['Friday', 'Saturday', 'Sunday'])]

[11]: blob.ngrams(n=5)

[11]: [WordList(['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']),
       WordList(['Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday']),
       WordList(['Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'])]
```

Σενάριο 2.12: Δημιουργία Ν-γραμμάτων με την μέθοδο ngrams()

2.6.12 Ο αναλυτής της TextBlob

Μια από τις πιο βασικές και πολύτιμες εργασίες NLP, είναι η **ανάλυση συναισθήματος (Sentiment Analysis)** η οποία προσδιορίζει εάν ένα κείμενο είναι θετικό, αρνητικό ή ουδέτερο [1]. Το Sentiment Analysis είναι χρήσιμο σε επιχειρήσεις προκειμένου αυτές να μπορούν να διαπιστώσουν αν οι άνθρωποι έχουν θετική ή αρνητική άποψη για τα προϊόντα ή τις υπηρεσίες τους. Γενικά, η ανάλυση συναισθήματος είναι ένα πολύπλοκο πρόβλημα μηχανικής μάθησης.

Η TextBlob διαθέτει την ιδιότητα **sentiment** η οποία προσδιορίζει αν ένα κείμενο είναι θετικό ή αρνητικό. Επίσης, η εν λόγω ιδιότητα επιστρέφει και ένα βαθμό υποκειμενικότητας που δείχνει κατά πόσο ένα κείμενο είναι υποκειμενικό ή αντικειμενικό [1]. Πιο συγκεκριμένα η ιδιότητα sentiment επιστρέφει την μεταβλητή **polarity (σημασιολογία)**, η οποία υποδηλώνει συναίσθημα αρνητικό με

τιμή -1, θετικό με τιμή 1 και ουδέτερο με τιμή 0. Για τον βαθμό υποκειμενικότητας επιστρέφεται η μεταβλητή **subjectivity (υποκειμενικότητα)**, η οποία δείχνει ότι το κείμενο είναι αντικειμενικό με την τιμή 0, και υποκειμενικό με την τιμή 1. Στο Σενάριο 2.13, παρουσιάζεται η ιδιότητα sentiment για δύο προτάσεις sentence1 και sentence2. Η sentence1 τείνει να δίνει θετικό συναίσθημα λόγω polarity = 0,85 με βαθμό υποκειμενικότητας 1 και συνεπώς είναι υποκειμενική. Αντίθετα, η πρόταση sentence2, δίνει αρνητικό συναίσθημα και είναι κατά κύριο λόγο υποκειμενική λόγω subjectivity = 0,66.

```
[27]: from textblob import TextBlob

[28]: sentence1 = TextBlob('Today is a beautiful day')

[29]: sentence1

[29]: TextBlob("Today is a beautiful day")

[30]: sentence1.sentiment

[30]: Sentiment(polarity=0.85, subjectivity=1.0)

[31]: sentence2 = TextBlob('Tomorrow looks like bad weather.')

[32]: sentence2

[32]: TextBlob("Tomorrow looks like bad weather.")

[33]: sentence2.sentiment

[33]: Sentiment(polarity=-0.6999999999999998, subjectivity=0.6666666666666666)
```

Σενάριο 2.13: Χρήση της ιδιότητας sentiment

2.6.13 Ο αναλυτής NaiveBayesAnalyzer

Η βιβλιοθήκη TextBlob διαθέτει τον **Analyzer NaiveBayesAnalyzer** μέσα στο πακέτο **textblob.sentiments**. Ο εν λόγω Analyzer έχει εκπαιδευτεί σε μια βάση δεδομένων για κριτικές ταινιών [1]. Ο NaiveBayes γενικά είναι ένας από τους πιο γνωστούς αλγόριθμους μηχανικής μάθησης ο οποίος χρησιμοποιείται στην ταξινόμηση κειμένου [1]. Για να τον χρησιμοποιήσουμε σε ένα TextBlob αρκεί να τον δηλώσουμε στην παράμετρο analyzer. Όταν κληθεί η ιδιότητα sentiment ενός TextBlob, ο NaiveBayesAnalyzer θα επιστρέψει τις μεταβλητές classification, p_pos και p_neg. Η classification ορίζει το γενικό συναίσθημα στο οποίο κατηγοριοποιείται το κείμενο. Οι p_pos και p_neg υποδεικνύουν την πιθανότητα το κείμενο να είναι θετικό ή αρνητικό αντίστοιχα. Στο Σενάριο 2.14, γίνεται import ο NaiveBayesAnalyzer και δηλώνεται η παράμετρος analyzer= NaiveBayesAnalyzer(). Η πρόταση 'The movie is excellent' κατηγοριοποιείται ως θετική (classification='pos'), εφόσον η μεταβλητή p_pos > p_neg.

```
[6]: from textblob.sentiments import NaiveBayesAnalyzer

[7]: from textblob import TextBlob

[8]: text = 'The movie is excellent'

[9]: my_blob = TextBlob(text, analyzer=NaiveBayesAnalyzer())

[10]: my_blob.sentiment

[10]: Sentiment(classification='pos', p_pos=0.7385905362564933, p_neg=0.26140946374350665)
```

Σενάριο 2.14: Ο αναλυτής NaiveBayesAnalyzer()

2.7 Βιβλιοθήκη Spacy

Η Spacy [1], είναι μία από τις δημοφιλέστερες ανοικτού-κώδικα βιβλιοθήκες NLP, που τα τελευταία χρόνια έχει κερδίσει την ερευνητική κοινότητα. Ένα από τα κύρια πλεονεκτήματα της βιβλιοθήκης Spacy είναι ο τρόπος που έχει κατασκευαστεί προγραμματιστικά, προσφέροντας έναν αποδοτικό και ευέλικτο τρόπο εργασίας χάρη στην εύκολη διεπαφή που παρέχει η γλώσσα προγραμματισμού Python. Επίσης, υποστηρίζει πολλές γλώσσες, μεταξύ αυτών και τα Ελληνικά. Η αντικειμενοστραφής λογική της σε συνδυασμό με την καθιέρωση μιας ένθερμης και υποστηρικτικής κοινότητας, καθιστά την Spacy ένα εξαιρετικό εργαλείο.

Όπως και η TextBlob, προσφέρει μία μεγάλη ποικιλία δυνατοτήτων προσαρμοσμένων στις ανάγκες του NLP, όπως ο διαμερισμός λέξεων σε διακριτά τμήματα (Tokenization), η σήμανση μερών του λόγου (Part-of-SpeechTagging), η αναγνώριση ονομαστικών οντοτήτων (NER – Named Entity Recognition), η λημματοποίηση (Lemmatization), η κατηγοριοποίηση κειμένου κ.α.

Οδηγίες εγκατάστασης της βιβλιοθήκης δίνονται στο Παράρτημα Β, στην ενότητα "Εγκατάσταση της Spacy". Στο Σενάριο 2.15, φαίνεται ένα παράδειγμα tokenization. Στα Σενάρια 2.16 και 2.17, φαίνονται το lemmatization και το NER αντίστοιχα.

```
[1]: import spacy
     nlp = spacy.load("el_core_news_sm")

[3]: # Κείμενο για tokenization
     text = "Ας χωρίσουμε την πρόταση σε λέξεις."

     # Εφαρμόζουμε το tokenization
     doc = nlp(text)

     # Εκτύπωση των λέξεων
     for token in doc:
         print(token.text)

     Ας
     χωρίσουμε
     την
     πρόταση
     σε
     λέξεις
     .
```

Σενάριο 2.15: Tokenization με Spacy

```
[7]: # Κείμενο για lemmatization
     txt = "Μοντέλο μηχανικής μάθησης που ξέρει να βρίσκει λήμματα"

     # Εφαρμόζουμε το lemmatization
     doc = nlp(txt)

     # Εκτύπωση των βασικών μορφών των λέξεων
     for token in doc:
         print(token.text, token.lemma_)

     Μοντέλο μοντέλο
     μηχανικής μηχανικός
     μάθησης μάθηση
     που που
     ξέρει ξέρω
     να να
     βρίσκει βρίσκω
     λήμματα λήμμα
```

Σενάριο 2.16: Lemmatization με Spacy

```
[11]: # Κείμενο για NER
      txt = "Ο Τάσος ζει στην Θεσσαλονίκη και εργάζεται στην Microsoft."

      # Εφαρμόζουμε το NER
      doc = nlp(txt)

      # Εκτύπωση των ονοματισμένων οντοτήτων
      for ent in doc.ents:
          print(ent.text, ent.label_)

      Τάσος PERSON
      Θεσσαλονίκη GPE
      Microsoft ORG
```

Σενάριο 2.17: NER με Spacy

2.8 Οπτικοποίηση συχνοτήτων λέξεων

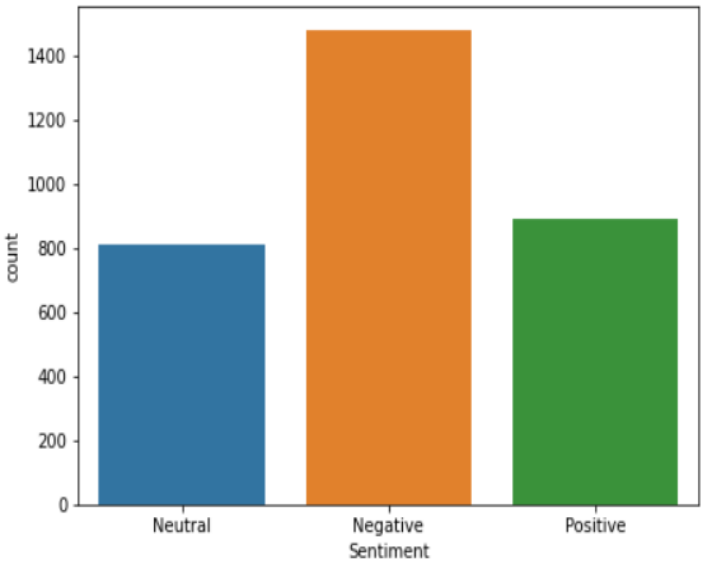
Σημαντική εργασία στην επεξεργασία φυσικής γλώσσας είναι και η οπτικοποίηση των συχνοτήτων των λέξεων σε ένα σώμα κειμένου. Δεν υπάρχει μόνο ένας τρόπος οπτικοποίησης. Στην παρούσα ενότητα θα δείξουμε τους δύο σημαντικότερους τρόπους οπτικοποίησης των συχνοτήτων.

- **Ραβδογράμματα**. Οπτικοποιούν ποσοτικά τις συχνότερες λέξεις ως ράβδους οι οποίες αναπαριστούν την κάθε λέξη και την συχνότητά της.
- **Σύννεφα λέξεων** (Word Cloud). Ένα σύννεφο λέξεων οπτικοποιεί ποιοτικά τις συχνότερες λέξεις με μεγαλύτερου μεγέθους γραμματοσειρές και λέξεις με μικρότερη συχνότητα με μικρότερου μεγέθους γραμματοσειρές.

2.8.1 Οπτικοποίηση με ραβδογράμματα

Στην παρούσα εργασία χρησιμοποιούμε ραβδογράμματα για την οπτικοποίηση των Neutral, Positive και Negative του Dataset με τα πολιτικά κόμματα. Χρησιμοποιούμε την βιβλιοθήκη **Matplotlib**, η οποία είναι μια ολοκληρωμένη βιβλιοθήκη για την δημιουργία στατικών και διαδραστικών απεικονίσεων στην γλώσσα Python [8]. Χρησιμοποιούμε επίσης την βιβλιοθήκη **Seaborn**, για οπτικοποίηση δεδομένων Python η οποία βασίζεται στην Matplotlib. Παρέχει μια διεπαφή υψηλού επιπέδου και μπορεί να σχεδιάσει ελκυστικά στατιστικά γραφήματα [9].

Η Εικόνα 2.1 παρουσιάζει το ραβδόγραμμα για τα Positive, Neutral και Negative για το σύνολο δεδομένων των πολιτικών κομμάτων. Ο κώδικας σε Python παρατίθεται στο αρχείο "Data_Analysis_Twitter.ipynb" του Παραρτήματος Α.



Εικόνα 2.1: Ραβδόγραμμα για Neutral, Negative, Positive

Τέλος, στο ίδιο αρχείο, παρατίθενται και επιπλέον ραβδογράμματα, όπως για οπτικοποίηση του Sentiment για κάθε κόμμα και για κάθε περιφέρεια της Ελλάδος, καθώς και ραβδογράμματα για παρουσίαση των tweets που συλλέξαμε ανά εβδομάδα.

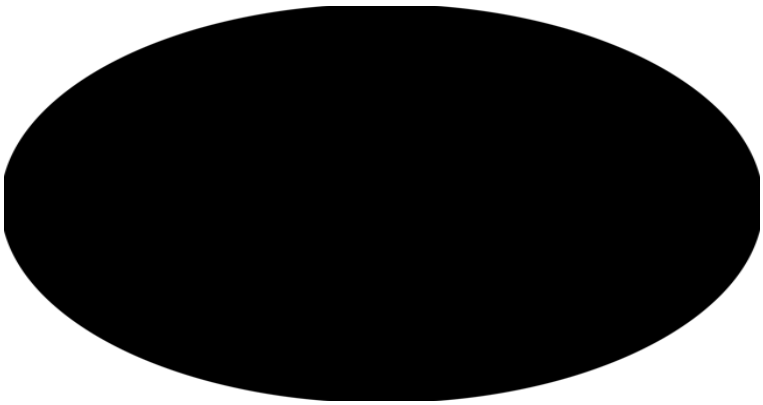
2.8.2 Οπτικοποίηση με σύννεφα λέξεων

Ένας εναλλακτικός τρόπος οπτικοποίησης συχνοτήτων λέξεων είναι τα γνωστά σύννεφα λέξεων. Για την υλοποίηση χρησιμοποιείται η κλάση **WordCloud** του αρθρώματος ανοιχτού κώδικα **wordcloud** (https://github.com/amueller/word_cloud) [7].

Από προεπιλογή το wordcloud δημιουργεί σύννεφα λέξεων με ορθογώνιο σχήμα, αλλά η βιβλιοθήκη μπορεί με κατάλληλες ρυθμίσεις να δημιουργήσει σύννεφα λέξεων με οποιοδήποτε σχήμα θέλουμε [7].

Για την δημιουργία ενός σύννεφου λέξεων, αρχικοποιούμε ένα αντικείμενο WordCloud με μια εικόνα της επιλογής μας η οποία θα χρησιμοποιηθεί ως μάσκα. Το WordCloud γεμίζει τις περιοχές της μάσκας με λέξεις. Στην Εικόνα 2.2 φαίνεται μάσκα με σχήμα οβάλ η οποία περιλαμβάνει λέξεις όπως Twitter, NLP, BERT, RoBERTa, DeepLearning κλπ. Για μάσκα χρησιμοποιούμε μια εικόνα με όνομα "mask_oval.png" (Εικόνα 2.3). Για να φορτώσουμε την εικόνα της μάσκας χρησιμοποιούμε την συνάρτηση **imread()** από το πακέτο **imageio**. Ο κώδικας δημιουργίας του σύννεφου λέξεων της Εικόνα 2.2, παρατίθεται στο αρχείο "cover.ipynb" του Παραρτήματος Α. Οδηγίες εγκατάστασης του wordcloud δίνονται στο Παράρτημα Β στην ενότητα "Εγκατάσταση του wordcloud".

Εικόνα 2.2: Οπτικοποίηση με σύννεφο λέξεων



Εικόνα 2.3: χρήση "mask_oval.png" για μάσκα

2.9 Επιπλέον εργαλεία και βιβλιοθήκες NLP

Στα προηγούμενα παρουσιάστηκαν βιβλιοθήκες NLP όπως NLTK, TextBlob και Spacy. Ωστόσο, στην επεξεργασία φυσικής γλώσσας υπάρχουν πολλά περισσότερα εργαλεία τα οποία μπορούμε να εκμεταλλευτούμε για την υλοποίηση εφαρμογών. Παρακάτω παραθέτουμε μια λίστα με API και βιβλιοθήκες NLP.

- **Google Cloud Natural Language API**. Βασίζεται στο Cloud για εργασίες NLP.
- **TextRazor API**. Βοηθάει στην εξαγωγή νοήματος σε σώμα κειμένου [1].
- **KoNLPy**. Πρόκειται για πακέτο Python για την επεξεργασία της Κορεατικής γλώσσας [1].
- **SnowNLP**. Βιβλιοθήκη Python για την επεξεργασία της Κινεζικής γλώσσας [1].
- **PyTorch NLP**. Το PyTorch είναι ένα Framework μηχανικής μάθησης ανοιχτού κώδικα που βασίζεται στη γλώσσα προγραμματισμού Python και στη βιβλιοθήκη **Torch**, η οποία χρησιμοποιείται για τη δημιουργία νευρωνικών δικτύων [7].
- **Apache OpenNLP**. Είναι μια εργαλειοθήκη για επεξεργασία φυσικής γλώσσας βασισμένη στην γλώσσα προγραμματισμού **JAVA** [7].
- **Stanford CoreNLP**. Το Stanford CoreNLP είναι επίσης μια εργαλειοθήκη JAVA που παρέχει μια μεγάλη ποικιλία εργαλείων NLP [7].
- **Transformers**. Είναι βιβλιοθήκη Python που περιέχει υλοποιήσεις ανοιχτού κώδικα μοντέλων μετασχηματιστών για εργασίες κειμένου, εικόνας και ήχου. Είναι συμβατό με τις

βιβλιοθήκες βαθιάς μάθησης **TensorFlow**, **PyTorch**, και περιλαμβάνει υλοποιήσεις αξιόλογων μοντέλων όπως το **BERT** και το **GPT** [10].

- **Gensim**. Το Gensim είναι μια βιβλιοθήκη για μοντελοποίηση θεμάτων χωρίς επίβλεψη, για **ευρετηρίαση εγγράφων** και άλλες εργασίες φυσικής γλώσσας, χρησιμοποιώντας σύγχρονη στατιστική μηχανική μάθηση [11]. Έχει σχεδιαστεί για να εξάγει σημασιολογικά θέματα από έγγραφα και μπορεί να χειριστεί μεγάλες συλλογές κειμένου.

2.10 Εφαρμογές φυσικής γλώσσας

Υπάρχουν πολλές εφαρμογές φυσικής γλώσσας οι οποίες υλοποιούνται με αλγόριθμους μηχανικής και βαθιάς μάθησης. Παρακάτω παρατίθενται ορισμένες από αυτές.

- **Ταξινόμηση – Κατηγοριοποίηση κειμένου**. Συλλογή κριτικών-αξιολογήσεων που αφορούν καταστήματα του Skroutz, προκειμένου να κατηγοριοποιηθούν ως αρνητικά ή θετικά.
- **Απλοποίηση κειμένου**. Ένα κείμενο γίνεται πιο συνοπτικό και ευανάγνωστο.
- **Υποτιτλισμός**. Προσθήκη υπότιτλων σε βίντεο.
- **Περίληψη κειμένων**. Ανάλυση εγγράφων και δημιουργία περίληψης.
- **Παραγωγή συστάσεων**. Συστήματα που προτείνουν σε πελάτη να αγοράσει κάτι που πιθανόν θα του αρέσει επειδή αγόρασε κάτι παρόμοιο.
- **Σύνθεση και αναγνώριση φωνής**.
- **Μοντελοποίηση θεμάτων**. Εύρεση θεμάτων που υπάρχουν σε έγγραφα.
- **Απαντήσεις σε ερωτήσεις**. Όπου κάποιος χρήστης υποβάλλει ερωτήσεις σε ένα σύστημα και λαμβάνει αυτόματες απαντήσεις.
- **Μετατροπή ομιλίας σε νοηματική γλώσσα**. Κατανόηση ομιλίας σε ανθρώπους με προβλήματα ακοής.
- **Συστήματα ανάγνωσης χειλιών** που μετατρέπουν την κίνηση των χειλιών σε ομιλία ή κείμενο προκειμένου να μπορούν να κάνουν διάλογο άνθρωποι που δεν μπορούν να μιλήσουν.

2.11 Επίλογος

Στο κεφάλαιο αυτό παραθέσαμε ορισμένες βιβλιοθήκες που διαθέτει η επεξεργασία φυσικής γλώσσας NLP. Δόθηκε έμφαση στις εργασίες NLP της βιβλιοθήκης TextBlob, στις οποίες περιλαμβάνονται η διαίρεση κειμένου σε λεκτικές μονάδες (tokens), η περιστολή (stemming) η οποία αφαιρεί τις καταλήξεις των λέξεων, η λημματοποίηση η οποία στοχεύει στην περικοπή καταλήξεων και την επιστροφή του λήμματος, η απαλοιφή των συνηθισμένων λέξεων (stop words), η εύρεση συχνοτήτων των λέξεων σε μια συλλογή κειμένων, η αναγνώριση του μέρους του λόγου μιας λέξης, ο έλεγχος ορθογραφίας, η διαγλωσσική μετάφραση και η αναγνώριση γλώσσας, η κλίση ενικού και πληθυντικού, η δημιουργία διαδοχικών λέξεων σε μια συλλογή κειμένων, καθώς και η ανάλυση συναισθήματος (sentiment analysis). Επίσης, αναφερθήκαμε στους τρόπους οπτικοποίησης των συχνοτήτων είτε πρόκειται για λέξεις είτε για μεταβλητές όπως η Sentiment. Πιο συγκεκριμένα, παραθέσαμε τα ραβδογράμματα τα οποία οπτικοποιούν ποσοτικά τις συχνότερες λέξεις ως ράβδους, καθώς και τα σύννεφα λέξεων τα οποία οπτικοποιούν ποιοτικά τις συχνότερες λέξεις ανάλογα με το μέγεθός τους. Τέλος, αναφέραμε επιπλέον βιβλιοθήκες και εργαλεία NLP που χρησιμοποιούνται ευρέως στις εφαρμογές φυσικής γλώσσας όπως είναι η κατηγοριοποίηση κειμένου, η περίληψη κειμένων, η παραγωγή συστάσεων, κλπ. Στο επόμενο κεφάλαιο, θα εισάγουμε την έννοια της εξόρυξης δεδομένων από το Twitter μέσω της διεπαφής προγραμματισμού εφαρμογών Twitter API.

Κεφάλαιο 3ο: Twitter και Εξόρυξη Πληροφορίας

3.1 Εισαγωγή

Τα τελευταία χρόνια, τα μέσα κοινωνικής δικτύωσης διαδραματίζουν σημαντικό ρόλο στη σύγχρονη ζωή. Ο αριθμός των χρηστών των μέσων κοινωνικής δικτύωσης αυξάνεται συνεχώς κάθε μέρα [12]. Οι χρήστες δημοσιεύουν την άποψή τους και τις σκέψεις τους χωρίς ιδιαίτερους περιορισμούς ακολουθώντας ένα σχετικά ελεύθερο επίπεδο απορρήτου. Έτσι λοιπόν, λόγω αυτών των "απλών" πολιτικών απορρήτου και της εύκολης πρόσβασης σε ορισμένα μέσα κοινωνικής δικτύωσης, οι χρήστες εγκαταλείπουν τα παραδοσιακά μέσα επικοινωνίας, όπως τα ιστολόγια-blogs, και προτιμούν πλέον ιστότοπους όπως Twitter, Facebook κλπ. Επίσης, λόγω του μεγάλου όγκου δεδομένων κειμένου που υπάρχει σε αυτά τα δίκτυα, θεωρούνται συναρπαστικά μέσα για την ανάλυση δεδομένων από τους ερευνητές [12].

3.2 Κοινωνικό δίκτυο Twitter

Το Twitter είναι ένας ιστότοπος κοινωνικής δικτύωσης που επιτρέπει στους χρήστες να δημοσιεύουν μηνύματα 140 χαρακτήρων (σε ορισμένες γλώσσες 280 χαρακτήρες) που ονομάζονται tweets. Ιδρύθηκε το 2006 και είναι ένας από τους δημοφιλέστερους ιστότοπους στο διαδίκτυο. Σύμφωνα με την έρευνα Statista, στο Twitter υπάρχουν 304 εκατομμύρια ενεργοί χρήστες μηνιαίως και αναρτώνται καθημερινά περίπου 500 εκατομμύρια tweets [12]. Πρόκειται για τεράστιο όγκο πολύτιμων δεδομένων που δίνει την ευκαιρία σε πολλούς ερευνητές να ψάξουν σε tweets και να κάνουν προβλέψεις για προϊόντα, γεγονότα, χρηματιστήρια, ανάλυση συναισθήματος κλπ. χρησιμοποιώντας διάφορες μεθόδους ταξινόμησης [12]. Από τις 23 Σεπτεμβρίου 2016 είχαν ξεκινήσει διαδικασίες εξαγοράς του Twitter, όπου τελικά στις 25 Απριλίου 2022 ανακοινώθηκε η εξαγορά του από τον Ίλον Μασκ.

3.3 Εξόρυξη δεδομένων (Data Mining)

Εξόρυξη δεδομένων ονομάζεται η αναζήτηση μέσα σε μεγάλες συλλογές δεδομένων, προκειμένου οι ερευνητές να εντοπίσουν πληροφορίες που είναι πολύτιμες σε ιδιώτες και οργανισμούς [1]. Η εξόρυξη δεδομένων είναι ένας διεπιστημονικός τομέας της επιστήμης των υπολογιστών και της στατιστικής με γενικό σκοπό την εξαγωγή πληροφοριών από ένα σύνολο δεδομένων και τη μετατροπή των πληροφοριών αυτών σε μια κατανοητή δομή για περαιτέρω χρήση και επεξεργασία [1]. Όπως έχουμε αναφέρει ήδη, σε μια από τις υλοποιήσεις μας, εφαρμόζουμε εξόρυξη tweets για πολιτικά κόμματα. Για τον σκοπό αυτό έχουμε δημιουργήσει το αρχείο "Data_Mining_Twitter.ipynb" το οποίο δίνεται στο Παράρτημα Α.

3.4 Twitter API

Το API του Twitter (Twitter API v1.1) είναι μια υπηρεσία διαδικτύου που βασίζεται στο Cloud, οπότε απαιτείται προφανώς σύνδεση στο διαδίκτυο για την εκτέλεση όλων των μεθόδων του API. Κάθε μέθοδος έχει ένα τερματικό (endpoint) το οποίο παρέχει διαδικτυακές υπηρεσίες και εκπροσωπείται από μια διεύθυνση URL που χρησιμοποιείται για την κλήση της μεθόδου μέσω του διαδικτύου [13].

Το Twitter API περιλαμβάνει πολλές κατηγορίες λειτουργικότητας. Κάποιες από αυτές είναι δωρεάν και κάποιες με πληρωμή. Όλες οι κατηγορίες υπόκεινται σε όρια χρήσης τα οποία καθορίζουν τον χρόνο που δύναται να χρησιμοποιηθούν [13].

Στο κεφάλαιο αυτό θα δείξουμε την βιβλιοθήκη **Tweepy** (https://www.tweepy.org/), μέσω της οποίας κάνουμε εξόρυξη δεδομένων από το Twitter.

3.5 Twitter API v2

Τα τελευταία χρόνια το Twitter API, βελτιώθηκε με την προσθήκη νέων επιπέδων πρόσβασης για προγραμματιστές και ερευνητές, προκειμένου να μπορούν να έχουν πρόσβαση σε δημόσια tweets [14]. Έτσι, το Νοέμβριο του 2021 κυκλοφόρησε το **Twitter API v2** με ακόμα προηγμένες δυνατότητες.

Υπάρχουν τρεις κατηγορίες πρόσβασης στην έκδοση v2 οι οποίες παρουσιάζονται παρακάτω:

- **Essential**. Δωρεάν και άμεση πρόσβαση στο Twitter API. Επιτρέπει 500.000 tweets ανά μήνα και είναι δωρεάν [15].
- **Elevated**. Παρέχει υψηλότερα επίπεδα πρόσβασης. Επιτρέπει 2 εκατομμύρια tweets ανά μήνα και είναι δωρεάν [15].
- **Academic Research**. Παρέχει πρόσβαση σε δημόσια δεδομένα για οποιοδήποτε θέμα με σκοπό την προώθηση των ερευνητικών στόχων ακαδημαϊκών και ερευνητών. Επιτρέπει 10 εκατομμύρια tweets ανά μήνα, είναι δωρεάν και μόνο για μη εμπορική χρήση [15].

Για τις ανάγκες της παρούσας εργασίας κάναμε αίτηση και έχουμε πρόσβαση στις κατηγορίες Elevated και Academic Research.

3.6 Όρια χρήσης

Όπως αναφέρθηκε και στα προηγούμενα, κάθε μέθοδος του Twitter API έχει ένα όριο χρήσης, το οποίο καθορίζει τον μέγιστο αριθμό κλήσεων που μπορεί να κληθεί σε ένα διάστημα 15 λεπτών [7]. Το Twitter μπορεί να μπλοκάρει τη χρήση του API του, αν μια μέθοδος υπερβεί το όριο χρήσης της. Επομένως, πριν την χρήση οποιασδήποτε μεθόδου είναι καλό να γνωρίζουμε την τεκμηρίωσή της [7].

Το σύνολο δεδομένων "unbalanced_politics.csv" το οποίο περιέχει tweets για πολιτικά κόμματα έχει δημιουργηθεί με την βοήθεια της βιβλιοθήκης Tweepy, την οποία έχουμε διαμορφώσει έτσι, ώστε να περιμένει κάποιο χρονικό διάστημα όταν μια μέθοδος εξόρυξης φτάσει στα όρια χρήσης της.

3.7 Δημιουργία Twitter Account

Για να δώσει πρόσβαση στις υπηρεσίες του, το Twitter απαιτεί ο κάθε προγραμματιστής, ακαδημαϊκός ή ερευνητής να κάνει αίτηση για λογαριασμό (https://developer.twitter.com/en/portal/dashboard). Αφού γίνει αίτηση από ένα χρήστη, το Twitter προχωρά σε έλεγχο προκειμένου να την αποδεχθεί ή να την απορρίψει.

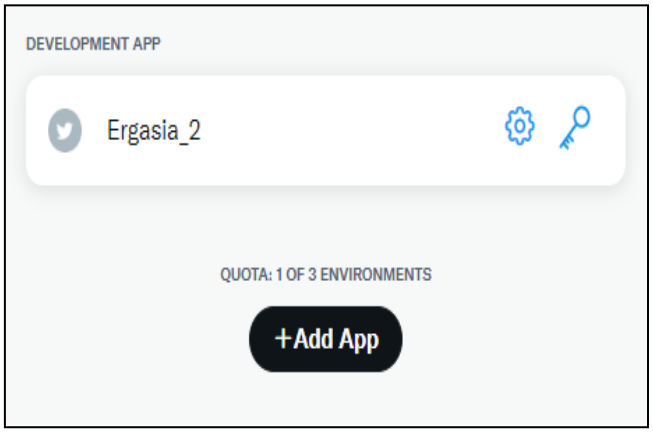
Όταν εγκριθεί μια αίτηση, ο χρήστης θα πρέπει να λάβει **διαπιστευτήρια (credentials)** για την αλληλεπίδρασή του με το **API**. Αυτό ισοδυναμεί με την δημιουργία μιας **εφαρμογής (app)**.

Για την δημιουργία μιας εφαρμογής Twitter απαιτούνται τα παρακάτω βήματα:

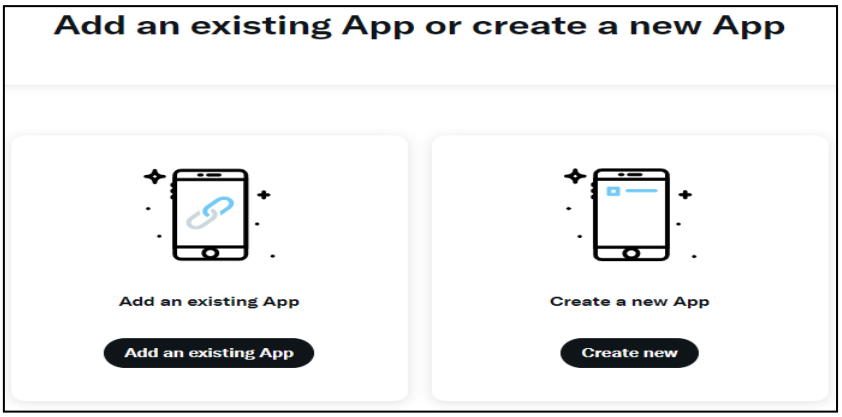
- Πηγαίνουμε στη σελίδα https://developer.twitter.com/en/portal/projects-and-apps και επιλέγουμε **Add App** όπως φαίνεται στην Εικόνα 3.1

- Στην οθόνη με τίτλο "Add an existing App or create a new App", επιλέγουμε το κουμπί **Create new** (Εικόνα 3.2)
- Στο πεδίο **App name** καθορίζουμε το όνομα της εφαρμογής μας (Εικόνα 3.3)
- Στο πεδίο **Description** εισάγουμε μια περιγραφή της εφαρμογής μας. (Εικόνα 3.3)
- Στο πεδίο **Environment** εισάγουμε την τιμή Development από την αναδυόμενη λίστα (Εικόνα 3.3)
- Τέλος, επιλέγουμε το κουμπί **Create** για να ολοκληρώσουμε την δημιουργία της εφαρμογής.

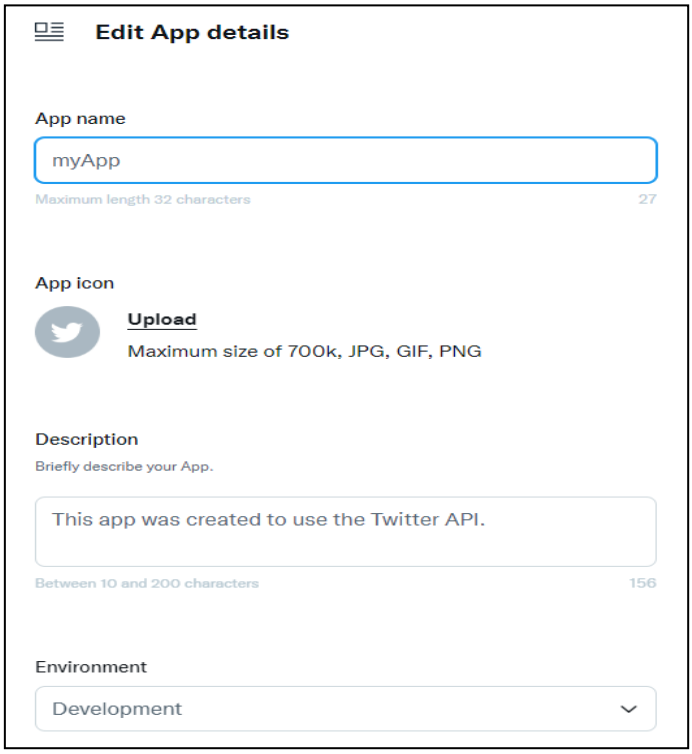
Μόλις ολοκληρωθεί η δημιουργία της εφαρμογής, το Twitter εμφανίζει μια σελίδα διαχείρισης. Μέσα από την σελίδα αυτή μπορούμε να δούμε την καρτέλα **Keys and tokens** η οποία περιλαμβάνει τα διαπιστευτήρια της εφαρμογής μας **Consumer Keys** και **Authentication Tokens**. Στο σημείο αυτό για λόγους ασφαλείας καλό είναι να αποθηκεύσουμε σε ένα αρχείο "my_key.py" όλα τα διαπιστευτήρια API consumer Key and Secret, Bearer Token και Access Token and Secret. Ο κώδικας Python με τον οποίο κάνουμε εξόρυξη από το Twitter, χρησιμοποιεί το αρχείο "my_key.py" μέσω του οποίου ουσιαστικά μας ταυτοποιεί το Twitter και μας επιτρέπει την πρόσβαση στο API του. Αυτή η διαδικασία πιστοποίησης ονομάζεται **OAuth 2.0**. Η μορφή του αρχείου "my_key.py" φαίνεται στην Εικόνα 3.4. Τέλος, είναι σημαντικό να αναφέρουμε ότι το Twitter API v2 απαιτεί για πιστοποίηση εφαρμογών μόνο το **bearer_token**, ενώ η αρχική έκδοση Twitter API απαιτεί τα **consumer_key**, **consumer_secret**, **access_token** και **access_token_secret**.



Εικόνα 3.1: Δημιουργία εφαρμογής Twitter (Βήμα 1)



Εικόνα 3.2: Δημιουργία εφαρμογής Twitter (Βήμα 2)



Εικόνα 3.3: Δημιουργία εφαρμογής Twitter (App name, Description, Environment)

```
consumer_key = 'YourConsumerKey'
consumer_secret = 'YourConsumerSecret'
access_token = 'YourAccessToken'
access_token_secret = 'YourAccessTokenSecret'

bearer_token ='YourBearerToken'
```

Εικόνα 3.4: Αρχείο my_key.py

3.8 Μέθοδοι Twitter API και JSON

Οι μέθοδοι του Twitter API επιστρέφουν αντικείμενα μορφής **JSON**. Το JSON είναι μια ελαφριά μορφή κειμένου για αποθήκευση και μεταφορά δεδομένων [16]. Είναι παρόμοιο με τα λεξικά της Python και αναπαριστά αντικείμενα ως συλλογές ζευγών χαρακτηριστικών/τιμών. Είναι απλό στην κατανόηση τόσο από τους ανθρώπους όσο και από τους υπολογιστές και καθιστά εύκολη την ανταλλαγή δεδομένων στο διαδίκτυο [16]. Ένα αντικείμενο JSON έχει γενική μορφή:

```
{ property_1: value_1, property_2: value_2 }
```

3.9 Δομή ενός αντικειμένου tweet

Ένα αντικείμενο tweet, όπως έχουμε αναφέρει μπορεί να περιέχει μέχρι 280 χαρακτήρες και συνιστά ένα σύντομο μήνυμα που δημοσιεύει ένας χρήστης στην πλατφόρμα του Twitter [1]. Αλλά δεν είναι μόνο αυτό, επιπλέον περιλαμβάνει και πολλά γνωρίσματα μεταδεδομένων (metadata) τα οποία περιγράφουν διάφορες παραμέτρους του tweet όπως για παράδειγμα ποιος χρήστης το δημιούργησε, πότε δημιουργήθηκε ακόμα και την τοποθεσία που έλαβε χώρα το tweet [1]. Επίσης, στο Twitter γίνεται η χρήση των συμβόλων @ και #. Το σύμβολο @ ακολουθούμενο από κάποιο όνομα χρήστη, ονομάζεται mention και προσδιορίζει ένα χρήστη Twitter π.χ. @AristotelisKamp. Ενώ, το σύμβολο # σε συνδυασμό με μια λέξη, ονομάζεται hashtag και χρησιμοποιείται για την υπόδειξη δημοφιλών θεμάτων, όπως για παράδειγμα #volleyball. Τέλος, υπάρχει το λεκτικό σύμβολο RT, το οποίο βρίσκεται στην αρχή ενός tweet και σημαίνει ότι το εν λόγω tweet είναι ένα Retweet, δηλαδή μια επανάληψη του tweet ενός άλλου χρήστη.

Στην παρακάτω λίστα παραθέτουμε τα πιο βασικά γνωρίσματα ενός tweet.

- **User**. Το αντικείμενο User που δηλώνει τον χρήστη που δημοσίευσε το tweet.
- **Text**. Το κείμενο του tweet.
- **Id**. Ένα μοναδικό αναγνωριστικό του tweet.
- **Created_at**. Ημερομηνία και ώρα δημιουργίας του tweet σε μορφή UTC.
- **Coordinates**. Οι συντεταγμένες από τις οποίες εστάλη το tweet. Πολλοί χρήστες απενεργοποιούν τα δεδομένα τοποθεσίας, με αποτέλεσμα η τιμή των συντεταγμένων να είναι None.
- **Lang**. Η γλώσσα του tweet. Συντομογραφία 'el' για Ελληνικά και 'en' για Αγγλικά.
- **Retweet_count**. Μετρητής που δείχνει πόσες φορές άλλοι χρήστες αναδημοσίευσαν το tweet.
- **Favorite_count**. Μετρητής που δείχνει πόσες φορές άλλοι χρήστες όρισαν το tweet ως αγαπημένο.

3.10 Βιβλιοθήκη Tweepy

Η **Tweepy** είναι μια από τις δημοφιλέστερες βιβλιοθήκες Python για την αλληλεπίδραση με το Twitter API η οποία επιστρέφει αντικείμενα JSON [1], [17]. Πληροφορίες εγκατάστασης της βιβλιοθήκης παρατίθενται στο Παράρτημα Β στην ενότητα "Εγκατάσταση της Tweepy".

3.10.1 Έλεγχος ταυτότητας - Tweepy

Για την **πιστοποίηση** στο Twitter πρέπει να δημιουργηθεί ένα αντικείμενο Tweepy. Το αντικείμενο αυτό ουσιαστικά είναι η πύλη μας για την χρήση του Twitter API [1]. Όλα τα API του Twitter διαχειρίζονται τεράστιες ποσότητες δεδομένων. Ο τρόπος με τον οποίο διασφαλίζουμε ότι τα

δεδομένα αυτά είναι ασφαλή τόσο για τους προγραμματιστές όσο και για τους χρήστες είναι μέσω του ελέγχου ταυτότητας [18]. Για να μπορέσουμε να καλούμε μεθόδους του Twitter API πρέπει να χρησιμοποιούμε τα **διαπιστευτήρια** consumer Key and Secret, και Access Token and Secret που αναφέραμε νωρίτερα στο παρόν κεφάλαιο. Έτσι, σε κάθε εφαρμογή Python, η εισαγωγή της βιβλιοθήκης Tweepy γίνεται με την εντολή:

```
import tweepy
```

Έπειτα, ακολουθεί η πιστοποίηση στο Twitter, με την εισαγωγή του αρχείου "my_key.py" το οποίο περιλαμβάνει τα διαπιστευτήρια της εφαρμογής μας. Η εισαγωγή του αρχείου "my_key.py" γίνεται με την εντολή:

```
import my_key
```

Εδώ, η εισαγωγή του αρχείου γίνεται χωρίς την κατάληξη ".py" και έτσι μπορούμε να προσπελάσουμε κάθε μια από τις μεταβλητές που περιλαμβάνει το αρχείο με μια γραμμή της μορφής:

```
my_key.variable_name
```

όπου variable_name μπορεί να είναι ένα από τα **consumer_key**, **consumer_secret**, **access_token** και **access_token_secret**.

Αφού έχουμε εισάγει την βιβλιοθήκη tweepy και το αρχείο "my_key" που περιέχει τα διαπιστευτήρια της εφαρμογής μας, πρέπει τώρα να δημιουργήσουμε και να διαμορφώσουμε την λειτουργία του ελέγχου ταυτότητας. Αυτό επιτυγχάνεται με την κλάση **tweepy.OAuthHandler**. Έτσι, δημιουργούμε ένα αντικείμενο της κλάσης OAuthHandler μεταβιβάζοντας τα ορίσματα my_key.consumer_key και my_key.consumer_secret [19]. Επόμενο βήμα είναι να καθορίσουμε τα επόμενα δυο ορίσματα, my_key.access_token και my_key.access_token_secret, μέσω της μεθόδου **set_access_token()** του αντικειμένου OAuthHandler [19]. Η δημιουργία του αντικειμένου OAuthHandler καθώς και η χρήση της συνάρτησης set_access_token(), δίνονται στον παρακάτω κώδικα Python:

```
[2]: auth=tweepy.OAuthHandler(my_key.consumer_key,
                              my_key.consumer_secret)

     auth.set_access_token(my_key.access_token,
                           my_key.access_token_secret)
```

Σενάριο 3.1: Κλάση OAuthHandler

Τελευταίο βήμα είναι να δημιουργήσουμε το αντικείμενο API, μέσω του οποίου θα αλληλεπιδρούμε με το Twitter [19]. Αυτό γίνεται με τον παρακάτω κώδικα Python:

```
api=tweepy.API(auth)
```

Εκτός του αντικειμένου auth που περιέχει τα διαπιστευτήριά μας, μπορούμε να περάσουμε στον κατασκευαστή API και 2 ακόμη προαιρετικά ορίσματα:

- **wait_on_rate_limit=True**. Το όρισμα αυτό επιβάλλει στην Tweepy να περιμένει 15 λεπτά κάθε φορά που μια μέθοδος φτάνει σε κάποιο συγκεκριμένο χρονικό όριο χρήσης [1].
- **wait_on_rate_limit_notify=True**. Το όρισμα αυτό επιβάλλει στην Tweepy να μας ενημερώνει με ένα μήνυμα κάθε φορά που περιμένει [1].

Στο σημείο αυτό έχουμε ολοκληρώσει τις ρυθμίσεις που απαιτούνται και είμαστε έτοιμοι να αλληλεπιδράσουμε με το Twitter μέσω της Tweepy. Οι ρυθμίσεις και τα βήματα που παρουσιάζονται σε αυτή την ενότητα αφορούν το Twitter API v1.1. Επιπλέον, για την αλληλεπίδραση με το Twitter API v1.1, η Tweepy υποστηρίζει τις μεθόδους ελέγχου ταυτότητας **OAuth 1.0a** και **OAuth 2.0** [20].

Για την πιστοποίηση **OAuth 1.0a** χρησιμοποιεί την κλάση **OAuth1UserHandler** [20]:

```
[2]: auth = tweepy.OAuth1UserHandler(
        my_key.consumer_key,
        my_key.consumer_secret,
        my_key.access_token,
        my_key.access_token_secret
     )
     api = tweepy.API(auth)
```

Σενάριο 3.2: Κλάση OAuth1UserHandler

Για την πιστοποίηση **OAuth 2.0** μπορεί να χρησιμοποιήσει την κλάση **OAuth2BearerHandler**, η οποία απαιτεί ως όρισμα το **bearer_token**, ή την κλάση **OAuth2AppHandler** η οποία απαιτεί τα ορίσματα **consumer_key** και **consumer_secret** [20].
Ο κώδικας Python για τις κλάσεις **OAuth2BearerHandler** και **OAuth2AppHandler** δίνεται παρακάτω:

```
[2]: auth = tweepy.OAuth2BearerHandler(bearer_token=my_key.bearer_token)
     api = tweepy.API(auth)

     auth = tweepy.OAuth2AppHandler(
        my_key.consumer_key,
        my_key.consumer_secret
     )
     api = tweepy.API(auth)
```

Σενάριο 3.3: Κλάσεις OAuth2BearerHandler και OAuth2AppHandler

3.10.2 Έλεγχος ταυτότητας - Tweepy (Twitter API v2)

Το **Twitter API v2** είναι πλέον το βασικό API του Twitter και είναι αυτό όπου επικεντρώνεται η επένδυση και η καινοτομία προϊόντων. Τα τελευταία χρόνια, το Twitter συνεργάστηκε με προγραμματιστές προκειμένου να δημιουργηθεί εκ νέου το API για να εξυπηρετεί καλύτερα μια ευρύτερη συλλογή αναγκών και να βελτιώσει την εμπειρία των προγραμματιστών [21].

Το Twitter API v2 είναι κατασκευασμένο με μοντέρνα θεμέλια, περιλαμβάνει βελτιωμένα τελικά σημεία, καθώς και νέες λειτουργίες. Ενώ τα περισσότερα από τα τελικά σημεία έχουν αντικαταστήσει τα παλαιότερα της έκδοσης v1.1, έχουν εισαχθεί πολλά νέα στην API v2 [21].

Το Twitter API v2 περιλαμβάνει επίσης νέες δυνατότητες που βοηθούν τους προγραμματιστές να επιστρέψουν πολύ εύκολα tweets που ταιριάζουν με ένα ερώτημα. Επιτρέπει τον εντοπισμό και το φιλτράρισμα συνομιλιών που ανήκουν σε ένα νήμα απαντήσεων [21]. Αν κοιτάξουμε πως εξελίσσονται οι συνομιλίες στο Twitter, θα παρατηρήσουμε ότι ένα tweet μπορεί να πυροδοτήσει πολλά νήματα συνομιλίας, καθένα από τα οποία μπορεί να αυξηθεί σε μήκος καθώς περισσότεροι άνθρωποι εισέρχονται [22]. Όταν τα tweets δημοσιεύονται ως απάντηση σε ένα tweet ή ως απάντηση σε μια απάντηση, υπάρχει πλέον ένα αναγνωριστικό συνομιλίας (conversation_id) σε κάθε απάντηση, το οποίο ταυτίζεται με το αναγνωριστικό tweet του αρχικού tweet που ξεκίνησε τη συνομιλία [22]. Έτσι, όλα τα νήματα απαντήσεων όσο πολύπλοκα και αν είναι θα μοιραστούν το μοναδικό **conversation_id**, του αρχικού tweet που πυροδότησε την συνομιλία [22]. Επομένως, οι προγραμματιστές που αλληλεπιδρούν με το Twitter API v2, έχουν τη δυνατότητα να ανακτούν ένα ολόκληρο νήμα συνομιλίας, προκειμένου να κατανοούν τι λέγεται για κάποιο θέμα, και πώς εξελίσσονται οι συζητήσεις και οι απόψεις [22]. Επιπλέον δυνατότητες στο Twitter API v2 περιλαμβάνουν:

- Προηγμένη λειτουργικότητα και πρόσβαση σε δεδομένα για ακαδημαϊκούς προγραμματιστές και ερευνητές.
- Υποστήριξη επεξεργασίας σε tweets.
- Απλοποιημένα αντικείμενα απόκρισης JSON.
- Επιστροφή 100% των αντίστοιχων δημόσιων και διαθέσιμων tweets σε ερωτήματα αναζήτησης.

Για την αλληλεπίδραση με το Twitter API v2, η Tweepy υποστηρίζει την μέθοδο ελέγχου ταυτότητας **OAuth 2.0** [20]. Χρησιμοποιεί την κλάση **Client** η οποία απαιτεί το όρισμα **Bearer Token** [20]. Ο κώδικας της κλάσης Client είναι ο εξής:

```
client = tweepy.Client(bearer_token=my_key.bearer_token)
```

3.11 Ανάκτηση πληροφοριών λογαριασμού Twitter

Αφού η εφαρμογή μας πιστοποιηθεί στο Twitter, μπορούμε να καλέσουμε μεθόδους του API προκειμένου να ανακτήσουμε διάφορες πληροφορίες. Τέτοιες πληροφορίες μπορεί να αφορούν γνωρίσματα όπως το όνομα ενός λογαριασμού χρήστη, το ψευδώνυμο ενός χρήστη, τον αριθμό id ενός λογαριασμού, την περιγραφή για ένα προφίλ, τους φίλους που έχει κάποιος χρήστης καθώς και τους φίλους που ακολουθούν ένα χρήστη. Τα δύο Twitter API v1.1 και v2, διαθέτουν την ίδια μέθοδο **get_user()** για λήψη όλων των παραπάνω χαρακτηριστικών.

Υλοποίηση μεθόδου get_user στο API v1.1

Στην έκδοση Twitter v1.1, η μέθοδος **get_user()** καλεί το τελικό σημείο (Endpoint) **GET users/show** [23]. Η μέθοδος επιστρέφει ένα σύνολο πληροφοριών σχετικά με τον χρήστη που καθορίζεται από την παράμετρο user_id ή την screen_name [24]. Στο Σενάριο 3.4, δίνεται η χρήση της μεθόδου get_user(). Χρησιμοποιείται η μέθοδος ελέγχου ταυτότητας **OAuth 1.0a** μέσω της κλάσης **OAuth1UserHandler**. Έπειτα, καλείται η get_user() προκειμένου να ανακτήσουμε μια σειρά γνωρισμάτων.

```
[7]: # Twitter API v1.1
     # OAuth 1.0a
     import tweepy
     import my_key

[8]: auth = tweepy.OAuth1UserHandler(
        my_key.consumer_key,
        my_key.consumer_secret,
        my_key.access_token,
        my_key.access_token_secret
     )
     api = tweepy.API(auth)

[9]: user = api.get_user(screen_name='AristotelisKamp')
     print(user.id)
     print(user.name)
     print(user.screen_name)
     print(user.description)
     # φίλοι
     print(user.friends_count)
     # ακόλουθοι
     print(user.followers_count)

     1448409710458294275
     Αριστοτέλης
     AristotelisKamp
     MSc in Web Intelligence
     7
     0
```

Σενάριο 3.4: Μέθοδος get_user() – OAuth 1.0a

Στο Σενάριο 3.5, χρησιμοποιείται η μέθοδος ελέγχου ταυτότητας **OAuth 2.0** μέσω της κλάσης **OAuth2BearerHandler** και η μέθοδος get_user(). Τέλος, στο Σενάριο 3.6 χρησιμοποιείται η κλάση **OAuth2AppHandler**.

```
[77]: # Twitter API v1.1
      # OAuth 2.0

[1]:  import tweepy
      import my_key
      auth = tweepy.OAuth2BearerHandler(bearer_token=my_key.bearer_token)
      api = tweepy.API(auth)

[2]:  user = api.get_user(screen_name='AristotelisKamp')
      print(user.id)
      print(user.name)
      print(user.screen_name)
      print(user.description)
      # φίλοι
      print(user.friends_count)
      # ακόλουθοι
      print(user.followers_count)

      1448409710458294275
      Αριστοτέλης
      AristotelisKamp
      MSc in Web Intelligence
      7
      0
```

Σενάριο 3.5: Μέθοδος get_user() – OAuth2BearerHandler

```
[83]: # Twitter API v1.1
      # OAuth 2.0

[12]: import tweepy
      import my_key

      auth = tweepy.OAuth2AppHandler(
         my_key.consumer_key,
         my_key.consumer_secret
      )
      api = tweepy.API(auth)

[13]: user = api.get_user(screen_name='AristotelisKamp')

      print(user.id)
      print(user.name)
      print(user.screen_name)
      print(user.description)
      # φίλοι
      print(user.friends_count)
      # ακόλουθοι
      print(user.followers_count)

      1448409710458294275
      Αριστοτέλης
      AristotelisKamp
      MSc in Web Intelligence
      7
      0
```

Σενάριο 3.6: Μέθοδος get_user() – OAuth2AppHandler

Όπως έγινε προφανές, είναι στην κρίση του κάθε προγραμματιστή ποια υλοποίηση της μεθόδου get_user() θα χρησιμοποιήσει στο Twitter API v1.1.

Υλοποίηση μεθόδου get_user στο API v2

Στην έκδοση Twitter v2, η μέθοδος **get_user()** καλεί το τελικό σημείο (Endpoint) **GET /2/users/by/username/:username** [25]. Η μέθοδος επιστρέφει ένα σύνολο πληροφοριών σχετικά με τον χρήστη που καθορίζεται από την παράμετρο username [26]. Στο Σενάριο 3.7, χρησιμοποιείται η μέθοδος get_user() και η κλάση Client για την πιστοποίηση της εφαρμογής μέσω του bearer_token το οποίο περιλαμβάνεται μέσα στο αρχείο "my_key.py". Επίσης, για την ορθή ανάκτηση των γνωρισμάτων, χρησιμοποιείται η λίστα **user_fields** η οποία μας υποδεικνύει τα πεδία που θέλουμε να προσπελάσουμε. Το πεδίο **public_metrics** είναι ένα λεξικό Python, που περιέχει το κλειδί (key) **followers_count**. Έτσι, η ανάκτηση του πλήθους των φίλων ενός χρήστη γίνεται με την γραμμή:

```
public_metrics['followers_count']
```

```
[4]: # Twitter API v2
     # OAuth 2.0
     import tweepy
     import my_key

[5]: client = tweepy.Client(bearer_token=my_key.bearer_token,
                            wait_on_rate_limit=True)

[6]: user = client.get_user(username='AristotelisKamp',
                            user_fields=['description', 'public_metrics'])
     print(user.data.id)
     print(user.data.name)
     print(user.data.username)
     print(user.data.description)
     print('Τον χρήστη', user.data.name, 'ακολουθούν')
     print(user.data.public_metrics['followers_count'], 'χρήστες')
     print('Ο χρήστης', user.data.name, 'ακολουθεί')
     print(user.data.public_metrics['following_count'], 'χρήστες')

     1448409710458294275
     Αριστοτέλης
     AristotelisKamp
     MSc in Web Intelligence
     Τον χρήστη Αριστοτέλης ακολουθούν
     0 χρήστες
     Ο χρήστης Αριστοτέλης ακολουθεί
     7 χρήστες
```

Σενάριο 3.7: Μέθοδος get_user() – OAuth 2.0

3.12 Μετεγκατάσταση σε ενημερωμένα Endpoints

Όπως έχουμε ήδη αναφέρει, το Twitter API v2 αποτελεί πλέον το κύριο API για ανάπτυξη εφαρμογών. Η πλατφόρμα επί του παρόντος υποστηρίζει κάποιες μεθόδους της έκδοσης API v1.1 και κάποιες άλλες τις έχει καταργήσει. Το ίδιο το Twitter συστήνει σε όλους τους χρήστες να ξεκινήσουν με το API v2, καθώς σε αυτό θα ενσωματώνονται όλες οι νέες λειτουργίες.

Σύμφωνα με την τεκμηρίωση του Twitter API v1.1 [23], και του API v2 [25] είναι διαθέσιμες πολλές μέθοδοι για διαφορετικές λειτουργίες. Ωστόσο, στους πίνακες που ακολουθούν Πίνακας 3.1 έως Πίνακας 3.3, γίνεται αναφορά σε μεθόδους του API v1.1 και του v2 για συγκεκριμένες κατηγορίες.

Πίνακας 3.1: Κατηγορία Timelines

Α/Α	API	Τελικό Σημείο (Endpoint)	Μέθοδος	Περιγραφή
1	**v1.1**	GET statuses/home_timeline	API.home_timeline	Επιστρέφει tweets του χρήστη που καλεί την εν λόγω μέθοδο καθώς και tweets των ατόμων που ακολουθεί.
	v2	GET /2/users/:id/timelines/reverse_chronological	Client.get_home_timeline	
2	**v1.1**	GET statuses/user_timeline	API. user_timeline	Επιστρέφει τα τελευταία 20 tweets από το χρονολόγιο ενός συγκεκριμένου λογαριασμού.
	v2	GET /2/users/:id/tweets	Client.get_users_tweets	Επιστρέφει τα τελευταία 10 tweets από το χρονολόγιο ενός συγκεκριμένου λογαριασμού. Με σελιδοποίηση, μπορούν να ανακτηθούν τα πιο πρόσφατα 3.200 tweet.

Πίνακας 3.2: Κατηγορία Search tweets

Α/Α	API	Τελικό Σημείο (Endpoint)	Μέθοδος	Περιγραφή
1	**v1.1**	GET search/tweets	API.search_tweets	Επιστρέφει μια συλλογή σχετικών tweets που ταιριάζουν με ένα συγκεκριμένο ερώτημα, από τις τελευταίες 7 ημέρες.
	v2	GET /2/tweets/search/recent	Client.search_recent_tweets	
2	**v2**	GET /2/tweets/search/all	Client.search_all_tweets	Η μέθοδος είναι διαθέσιμη μόνο σε όσους χρήστες έχουν εγκριθεί για Ακαδημαϊκή Έρευνα. Επιστρέφει tweets έως και 30 ημέρες πριν.

Πίνακας 3.3: Κατηγορία Follows

Α/Α	API	Τελικό Σημείο (Endpoint)	Μέθοδος	Περιγραφή
1	**v1.1**	GET followers/list	API.get_followers	Επιστρέφει μια λίστα χρηστών που είναι ακόλουθοι του καθορισμένου αναγνωριστικού χρήστη.
	v2	GET /2/users/:id/followers	Client.get_users_followers	
2	**v1.1**	GET friends/list	API.get_friends	Επιστρέφει μια λίστα χρηστών που ακολουθεί ένα καθορισμένο αναγνωριστικό χρήστη. Δηλ. οι φίλοι κάποιου συγκεκριμένου χρήστη.
	v2	GET /2/users/:id/following	Client.get_users_following	

Ο Πίνακας 3.2, περιέχει μεθόδους που πραγματοποιούν αναζήτηση με βάση κάποιο συγκεκριμένο ερώτημα το οποίο καθορίζεται από το όρισμα (**q** για search_tweets και **query** για search_all_tweets και search_recent_tweets). Το όρισμα **count** καθορίζει το πλήθος tweets που θα επιστραφούν για την μέθοδο search_tweets. Για τις άλλες δύο μεθόδους χρησιμοποιείται το όρισμα **max_results**. Επίσης, σε κάθε αναζήτηση μπορούμε να βελτιώσουμε τα αποτελέσματα καθορίζοντας συμβολοσειρές ερωτημάτων με την χρήση τελεστών του Twitter. Μπορούμε να συνδυάσουμε πολλούς τελεστές για πιο πολύπλοκες αναζητήσεις [1]. Στον Πίνακα 3.4, παρουσιάζονται οι τελεστές αναζήτησης του Twitter.

Πίνακας 3.4: Τελεστές αναζήτησης

Α/Α	Τελεστής	Σημασία
1	Suzuki OR Nissan	Λογικός τελεστής OR. Εντοπίζει tweets που περιέχουν την λέξη Suzuki ή Nissan.
2	Suzuki Nissan	Λέξεις με ένα κενό μεταξύ τους. Έμμεσος τελεστής ΚΑΙ. Εντοπίζει tweets που περιέχουν την λέξη Suzuki και την Nissan.
3	Suzuki – Nissan	Σύμβολο παύλα. Εντοπίζει tweets που περιέχουν την λέξη Suzuki αλλά όχι την λέξη Nissan.
4	Suzuki ?	Σύμβολο ερωτηματικό. Εντοπίζει tweets που κάνουν ερωτήματα για την λέξη Suzuki.

5	Suzuki :(	Σύμβολο λυπημένη φατσούλα. Εντοπίζει tweets αρνητικού συναισθήματος που περιέχουν την λέξη Suzuki.
6	Suzuki :)	Σύμβολο χαρούμενη φατσούλα. Εντοπίζει tweets θετικού συναισθήματος που έχουν την λέξη Suzuki.
7	near:"Athens"	Εντοπίζει tweets που εστάλησαν από μέρη κοντά στην Αθήνα.
8	to:user	Όπου user ένας λογαριασμός χρήστη. Εντοπίζει tweets που έγιναν για τον λογαριασμό @user.
9	from:user	Βρίσκει tweets που έγιναν από τον λογαριασμό @user.
10	since:2022-12-20	Εντοπίζει tweets με ημερομηνία ίδια η μεταγενέστερη της 2022-12-20.

Τα tweets πολύ συχνά περιέχουν ετικέτες (hashtags), οι οποίες αρχίζουν με το σύμβολο #. Τα hashtags υποδηλώνουν ότι ένα θέμα είναι δημοφιλές. Για την αναζήτηση ενός σημαντικού θέματος βάζουμε το σύμβολο # μαζί με το θέμα στο ερώτημα αναζήτησης. Για παράδειγμα `query='#topic'`. Όπου `topic` ένα γνωστό δημοφιλές θέμα.

Στο Παράρτημα Α, στην ενότητα "Twitter API v1.1 και ενημερωμένα Endpoints v2" δίνονται παραδείγματα σε κώδικα Python για μεθόδους που αφορούν τις κατηγορίες Timelines, Search tweets και Follows.

3.13 Twitter Trends API

Όπως είπαμε και νωρίτερα, τα hashtags αναφέρονται σε δημοφιλή θέματα για τα οποία μπορεί να υπάρχουν εκατομμύρια άνθρωποι που να δημοσιεύουν για αυτά ταυτόχρονα. Τα θέματα αυτά το Twitter τα ονομάζει **Trending Topics**. Για την πρόσβαση σε καταλόγους τοποθεσιών χρησιμοποιείται το **Twitter Trends API**. Έτσι, μπορούμε να ανακτήσουμε τοποθεσίες με δημοφιλή θέματα [1].

Η Tweepy διαθέτει την μέθοδο **available_trends()** η οποία καλεί το τελικό σημείο (Endpoint) **GET trends/available** [23]. Η εν λόγω μέθοδος επιστρέφει μια **λίστα λεξικών** όπου το καθένα αντιπροσωπεύει μια τοποθεσία με δημοφιλή θέματα [27]. Το κάθε λεξικό (τοποθεσία) της λίστας, περιέχει διάφορες πληροφορίες όπως μεταξύ άλλων country και woeid. Τα WOEIDs είναι κωδικοί τοποθεσιών με τους οποίους γίνεται αναζήτηση δημοφιλών θεμάτων. Κάθε τιμή woeid μπορεί να χρησιμοποιηθεί από την μέθοδο **get_place_trends()**, η οποία καλεί το τελικό σημείο **GET trends/place** και επιστρέφει τα 50 δημοφιλέστερα θέματα για την τοποθεσία με το καθοριζόμενο WOEID. Για τον καθορισμό του WOEID χρησιμοποιείται το όρισμα id [28]. Συγκεκριμένα, επιστρέφεται μια λίστα που περιέχει ένα λεξικό. Το κλειδί "trends" του λεξικού παραπέμπει σε μια λίστα λεξικών όπου το καθένα εκπροσωπεί ένα δημοφιλές θέμα. Ο Πίνακας 3.5, δείχνει τιμές WOEID για μερικές τοποθεσίες. Στο Παράρτημα Α, ενότητα "Twitter Trends API" παρατίθεται κώδικας σε Python για τις μεθόδους **available_trends()** και **get_place_trends()**.

Πίνακας 3.5: WOEIDs Τοποθεσιών

Α/Α	Τοποθεσία	WOEID
1	London	44418
2	New York	2459115
3	Paris	615702
4	Los Angeles	2442047
5	Toronto	4118
6	Sydney	1105779
7	San Francisco	2487956

3.14 Καθαρισμός και προ-επεξεργασία tweets

Πολύ σημαντική εργασία στην ανάλυση κειμένου, αποτελεί ο καθαρισμός των δεδομένων. Με τον όρο καθαρισμό εννοούμε τις εργασίες προ-επεξεργασίας του κειμένου ώστε αυτό να περιέχει μόνο ουσιώδεις όρους. Στο κεφάλαιο 2, είδαμε κάποιες εργασίες προ-επεξεργασίας όπως είναι η κανονικοποίηση. Γενικά τα tweets, λόγω της φύσης τους περιέχουν θόρυβο όπως είναι οι υπερσύνδεσμοι, τα email, διάφορα σημεία στίξης, η ακόμα και οι ανορθόγραφες λέξεις. Συνεπώς, επιβάλλεται ο καθαρισμός των tweets προκειμένου να προκύψει ένα "καθαρό" σύνολο δεδομένων έτσι ώστε να αυξηθεί και η απόδοση των μοντέλων.

Εργασίες καθαρισμού tweets περιλαμβάνουν τα εξής:

- Διαγραφή @-αναφορών
- Διαγραφή ετικετών (hashtags)
- Διαγραφή σημείων στίξης
- Διαγραφή RT (retweet) και FAV (favorite)
- Διαγραφή διευθύνσεων URL και e-mail.
- Διαγραφή πλεονάζοντος λευκού χώρου
- Διαγραφή διπλότυπων κειμένων
- Διαγραφή αριθμών
- Μετατροπή κειμένου σε πεζά γράμματα
- Διαγραφή emojis

Για ορισμένες από τις παραπάνω εργασίες καθαρισμού, το Twitter διαθέτει την βιβλιοθήκη **tweet-preprocessor**. Ο Πίνακας 3.6 δείχνει τις επιλογές καθαρισμού για κάθε είδος θορύβου.

Πίνακας 3.6: Καθαρισμός tweets

Α/Α	Επιλογή καθαρισμού	Είδος θορύβου
1	OPT.MENTION	@-αναφορά
2	OPT.HASHTAG	Ετικέτα (hashtag)
3	OPT.RESERVED	RT (retweet) και FAV (favorite)
4	OPT.URL	URL
5	OPT.NUMBER	Αριθμός
6	OPT.EMOJI	Φατσούλα (emoji)
7	OPT.SMILEY	Χαμογελαστή φατσούλα (smiley)

Το Σενάριο 3.8 δείχνει έναν απλό καθαρισμό ενός tweet. Για να καθορίσουμε τις επιλογές καθαρισμού χρησιμοποιούμε τη μέθοδο **set_options()**. Για την πραγματοποίηση του καθαρισμού καλείται η μέθοδος **clean()** η οποία δέχεται ως όρισμα το tweet. Πληροφορίες εγκατάστασης της βιβλιοθήκης tweet-preprocessor δίνονται στο Παράρτημα Β.

```
[63]: import preprocessor as p

[64]: p.set_options(p.OPT.URL, p.OPT.HASHTAG, p.OPT.MENTION)

[65]: tweet_text = """tweet with URL https://https://twitter.com/
                     a hashtag #topic and mention @user"""

[66]: p.clean(tweet_text)

[66]: 'tweet with URL a hashtag and mention'
```

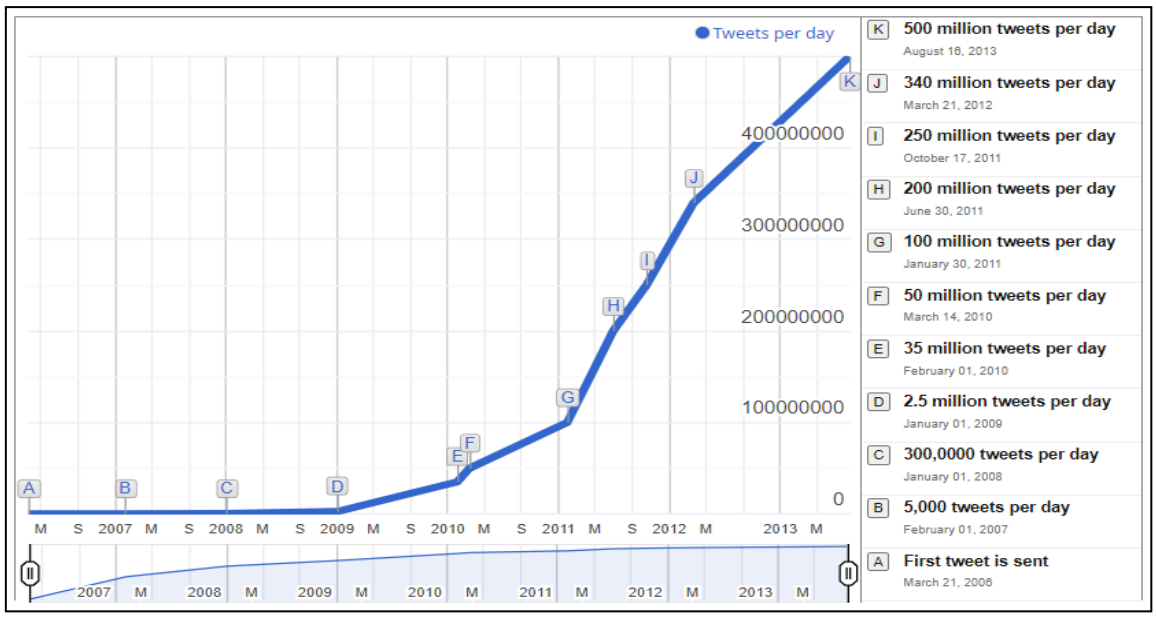
Σενάριο 3.8: Καθαρισμός URL, Hashtag και Mention

3.15 Twitter Streaming API

Το Twitter Streaming API, έχει την δυνατότητα να στέλνει δυναμικά στην εφαρμογή μας τυχαία επιλεγμένα tweets την ώρα που αυτά συμβαίνουν. Επιτρέπει την πρόσβαση στο 1% των συνολικών tweets ημερησίως [1]. Σύμφωνα με την [29], υπολογίζεται ότι δημοσιεύονται πάνω από 6000 tweets ανά δευτερόλεπτο και κατά συνέπεια περισσότερα από 500 εκατομμύρια tweets την ημέρα και περίπου 200 δισεκατομμύρια tweets ετησίως. Η Εικόνα 3.5, δείχνει στατιστικά χρήσης του Twitter. Η μέτρηση που μας δίνει ξεκινά στις 21 Μαρτίου 2006 όπου στάλθηκε το πρώτο tweet και στις 16 Αυγούστου 2013 σημειώνονται 500 εκατομμύρια tweets την ημέρα.

Συνεπώς, το Twitter Streaming API δίνει πρόσβαση σε περίπου 5 εκατομμύρια tweets ανά ημέρα. Παλαιότερα επέτρεπε την πρόσβαση στο 10% της ροής των tweets ημερησίως, όμως πλέον η υπηρεσία αυτή διατίθεται μόνο με πληρωμή [1]. Επιστρέφει tweets που ικανοποιούν κριτήρια αναζήτησης και χρησιμοποιεί μια μόνιμη σύνδεση προκειμένου να προωθεί τα tweets στην εφαρμογή

μας. Ο ρυθμός με τον οποίο φτάνουν τα tweets ποικίλει ανάλογα με τα κριτήρια αναζήτησης τα οποία καθορίζονται με τα αντικείμενα **StreamRule**. Όσο πιο δημοφιλές είναι ένα θέμα, τόσο πιο γρήγορα θα γίνεται η λήψη των tweets [1].



Εικόνα 3.5: Twitter Usage Statistics [28]

Για την επεξεργασία μιας ροής από tweets, πρέπει να δημιουργηθεί μια **υποκλάση** (έστω TweetListener) της κλάσης **StreamingClient** της βιβλιοθήκης Tweepy. Ένα αντικείμενο της εν λόγω κλάσης είναι ο **ακροατής** (tweet_listener), ο οποίος ενημερώνεται όταν φτάνει ένα νέο tweet. Κάθε μήνυμα που στέλνει το Twitter έχει ως συνέπεια την κλήση μιας μεθόδου της StreamingClient [1].

Οι μέθοδοι της υποκλάσης TweetListener φαίνονται στον παρακάτω Πίνακα 3.7.

Πίνακας 3.7: Μέθοδοι υποκλάσης TweetListener

Α/Α	Μέθοδος	Περιγραφή
1	__init__	Καλείται όταν δημιουργείται ένα νέο TweetListener αντικείμενο. Η παράμετρος bearer_token χρησιμοποιείται για έλεγχο ταυτότητας με το Twitter. Το όρισμα limit καθορίζει το πλήθος tweets που θα επιστραφούν.
2	on_connect	Καλείται όταν η εφαρμογή μας συνδεθεί με επιτυχία στη ροή Twitter.
3	on_response	Καλείται όταν φτάνει ένα tweet.

Στο Παράρτημα Α, στην ενότητα "Twitter Streaming API", δίνεται το αρχείο "tweetlistener.py" το οποίο περιλαμβάνει την κλάση TweetListener καθώς και επιπλέον κώδικας για την ανάκτηση tweets σε πραγματικό χρόνο.

3.16 Επίλογος

Στο παρόν κεφάλαιο ερευνήσαμε την εξόρυξη δεδομένων του Twitter, το οποίο είναι ένα από τα πιο δημοφιλή κοινωνικά δίκτυα. Είδαμε πως μπορούμε να δημιουργήσουμε λογαριασμό προγραμματιστή και αναλύσαμε την βιβλιοθήκη Tweepy η οποία υποστηρίζει μεθόδους ελέγχου ταυτότητας OAuth 1.0a και OAuth 2.0 για την πιστοποίηση μιας εφαρμογής στο Twitter API. Επιπλέον, είδαμε πως μπορούμε να καλέσουμε άλλες μεθόδους του API προκειμένου να ανακτήσουμε γνωρίσματα όπως το όνομα ενός λογαριασμού χρήστη, τους φίλους που τον ακολουθούν ή και τους φίλους που ο ίδιος ακολουθεί. Συζητήσαμε τις δύο εκδόσεις του Twitter API (v1.1 και v2), και παραθέσαμε τις διαφορετικές υλοποιήσεις μεθόδων για κάθε έκδοση. Λόγω του μεγάλου όγκου μεθόδων, παραθέσαμε τρεις κατηγορίες που σχετίζονται με εξόρυξη tweets και users (Timelines, Search, Follows). Αναφέραμε τους τελεστές αναζήτησης του Twitter με τους οποίους μπορούμε να εφαρμόσουμε πιο περίπλοκες αναζητήσεις. Χρησιμοποιήσαμε τις μεθόδους του Twitter Trends API, προκειμένου να έχουμε πρόσβαση σε καταλόγους τοποθεσιών με δημοφιλή θέματα. Τονίσαμε πόσο σημαντική είναι η διεργασία του καθαρισμού των δεδομένων ώστε να προκύπτουν σύνολα δεδομένων με ουσιώδεις όρους και δείξαμε πόσο εύκολα μπορούμε να καθαρίσουμε tweets με την βοήθεια της βιβλιοθήκης tweet-preprocessor. Τέλος, αναφερθήκαμε στο Twitter Streaming API το οποίο επιστρέφει tweets την στιγμή που συμβαίνουν. Στο επόμενο κεφάλαιο, θα παρουσιάσουμε την μηχανική μάθηση, έναν από τους πιο συναρπαστικούς και υποσχόμενους τομείς της τεχνητής νοημοσύνης.

Κεφάλαιο 4ο: Μηχανική Μάθηση

4.1 Εισαγωγή

Ξεκινώντας με τον **ορισμό**, μηχανική μάθηση [30], [31] είναι η μελέτη αλγορίθμων και στατιστικών μοντέλων τα οποία μαθαίνουν από δεδομένα και δεν ακολουθούν συγκεκριμένες εντολές για να εκτελέσουν μια εργασία. Βασική προϋπόθεση σε όλους τους αλγορίθμους είναι η δημιουργία μιας βάσης δεδομένων εκπαίδευσης (training dataset). Κλασσικό παράδειγμα αποτελεί η αναγνώριση ψηφίων με το MNIST dataset [32]. Στο κεφάλαιο αυτό θα εξεταστούν αλγόριθμοι εκτός των Νευρωνικών Δικτύων, καθώς αποτελούν αντικείμενο του κεφαλαίου 5: Βαθιά Μάθηση.

Αναλόγως με το πρόβλημα που αντιμετωπίζεται, υπάρχουν διαφορετικές κατηγορίες αλγορίθμων Μηχανικής Μάθησης. Στην εργασία του ο Batta Mahesh [30] παραθέτει μία αναλυτική περιγραφή των περισσότερων αλγορίθμων που θα συναντήσει κανείς στη βιβλιογραφία. Συνοψίζοντας, οι αλγόριθμοι μηχανικής μάθησης μπορούν να διαχωριστούν σε τρεις βασικές κατηγορίες:

- **Μηχανική Μάθηση με επίβλεψη (Supervised Machine Learning).** Η επιτηρούμενη ή επιβλεπόμενη μηχανική μάθηση εμπίπτει σε δύο κατηγορίες, την ταξινόμηση και την παλινδρόμηση. Τα μοντέλα μηχανικής μάθησης εκπαιδεύονται σε σύνολα δεδομένων που αποτελούνται από γραμμές και στήλες όπου κάθε γραμμή αντιπροσωπεύει ένα δείγμα δεδομένων και κάθε στήλη ένα χαρακτηριστικό του δείγματος [1]. Τα δεδομένα εκπαίδευσης είναι συνήθως μεγάλα στον αριθμό και κάθε καταχώρηση στη βάση αυτή χαρακτηρίζεται από μία ετικέτα (label) που αποκαλείται στόχος (target). Ο στόχος είναι η τιμή που προβλέπουμε για νέα δεδομένα τα οποία δεν γνωρίζουν τα μοντέλα. Βασική ιδιότητα των αλγορίθμων με επίβλεψη είναι η κατηγοριοποίηση των δεδομένων σε μία από τις ετικέτες.
- **Μηχανική Μάθηση χωρίς επίβλεψη (Unsupervised Machine Learning).** Τα δεδομένα εκπαίδευσης είναι μεγάλα στον αριθμό, όμως δεν υπάρχουν ετικέτες για τις καταχωρήσεις. Γίνεται η λεγόμενη συσταδοποίηση των καταχωρήσεων σε ομάδες με βάση το διάνυσμα χαρακτηριστικών τους.
- **Μηχανική Μάθηση με ημι-επίβλεψη (Semi-Supervised Machine Learning).** Πρόκειται για συγχώνευση των παραπάνω κατηγοριών. Σε αυτό τον τύπο μάθησης, ένα μικρό μέγεθος των δειγμάτων στο σύνολο δεδομένων έχει ετικέτες, ενώ κάποιο άλλο όχι. Στόχος είναι να εκπαιδευτεί ένα μοντέλο που θα αξιοποιεί και τις επιβλεπόμενες και τις μη επιβλεπόμενες πληροφορίες για να βελτιώσει την απόδοσή του.

4.2 Προβλέψεις

Θα μπορούσαμε να προβλέψουμε την πρόγνωση του καιρού προκειμένου να βοηθήσουμε τους αγρότες να πάρουν αποφάσεις σχετικά με το πότισμα των καλλιεργειών τους. Θα μπορούσαμε να βελτιώσουμε τις διαγνώσεις καρκίνου ή άλλων ασθενειών για να σώσουμε ζωές. Ίσως να βελτιώναμε τις επιχειρηματικές προβλέψεις για μεγιστοποίηση των κερδών μιας επιχείρησης και την διασφάλιση θέσεων εργασίας. Θα μπορούσαμε ακόμα, να προβλέψουμε τις καταλληλότερες στρατηγικές που θα χρησιμοποιούσαν προπονητές ώστε να κερδίσουν περισσότερους αγώνες ή πρωταθλήματα. Όλες οι παραπάνω προβλέψεις γίνονται πλέον με την βοήθεια της μηχανικής μάθησης [1].

4.3 Σύνολα δεδομένων (Datasets)

Στον κόσμο των μαζικών δεδομένων, είναι συνηθισμένο να βρίσκονται σύνολα δεδομένων με εκατομμύρια δείγματα ή περισσότερα. Υπάρχουν πολλά ελεύθερα σύνολα δεδομένων διαθέσιμα για την επιστήμη των δεδομένων, ενώ πολλές βιβλιοθήκες, όπως η Scikit-learn, παρέχουν συσκευασμένα σύνολα δεδομένων για εξερεύνηση και πειραματισμό. Επιπλέον, κυβερνήσεις, επιχειρήσεις και άλλοι οργανισμοί παγκοσμίως παρέχουν σύνολα δεδομένων για πολλούς τομείς και θέματα [1].

Το πρόβλημα της παρούσας εργασίας όπως έχει αναφερθεί, είναι η κατηγοριοποίηση κειμένου και συγκεκριμένα η ανάλυση συναισθήματος σε δύο διαφορετικές βάσεις δεδομένων, το Skroutz και το Politics Dataset. Καθώς τα δεδομένα εκπαίδευσης διαθέτουν διακριτές ετικέτες και είναι ικανοποιητικά μεγάλα γίνεται χρήση αλγορίθμων με επίβλεψη με σκοπό την κατηγοριοποίηση σχολίων σε θετικά ή αρνητικά (και ουδέτερα στην περίπτωση του Politics Dataset).

Το Dataset των πολιτικών κομμάτων έχει δημιουργηθεί από εμάς, αφορά ένα σύνολο δεδομένων το οποίο περιλαμβάνει πάνω από 4000 ελληνικά tweets για τα πολιτικά κόμματα της Ελλάδας. Δημιουργήθηκε μετά από εξόρυξη δεδομένων με την βοήθεια του Twitter API. Το Skroutz Dataset [33] αφορά τις αξιολογήσεις καταστημάτων στην ελληνική γλώσσα που έχουν γίνει στην ιστοσελίδα της υπηρεσίας Skroutz. Περισσότερα για τα datasets αναφέρονται στην ενότητα 6.6.

4.4 Ταξινόμηση

Όπως έγινε σαφές και από τα προηγούμενα, η ταξινόμηση αποτελεί ένα από τα βασικά προβλήματα της μηχανικής μάθησης, και αναφέρεται στη διαδικασία ταξινόμησης ενός αντικειμένου σε μία από διακριτές κατηγορίες, βάσει των χαρακτηριστικών του. Αυτή η διαδικασία μπορεί να γίνει χρησιμοποιώντας μια πληθώρα αλγορίθμων μηχανικής μάθησης, συμπεριλαμβανομένων των k-NN (k-Nearest Neighbors), SVM (Support Vector Machines) και Decision Trees. Η ταξινόμηση έχει εφαρμογές σε πολλά πεδία, όπως η αναγνώριση προτύπων, η ανίχνευση ανεπιθύμητων μηνυμάτων στα emails, η κατηγοριοποίηση εικόνων και η πρόβλεψη των χρηματιστηριακών τάσεων. Επίσης, μπορεί να χρησιμοποιηθεί στον κλάδο της ασφάλειας για την ανίχνευση απάτης και στην αυτοματοποίηση και ρομποτική για την αυτόματη αναγνώριση και διάκριση αντικειμένων και ανθρώπων. Το πρόβλημα της ανάλυσης συναισθήματος προϋποθέτει την κατηγοριοποίηση κειμένων σε διακριτές κατηγορίες (θετικό, αρνητικό ή ουδέτερο) και συνεπώς εκπίπτει σαν πρόβλημα ταξινόμησης. Επομένως, η ταξινόμηση είναι μία σημαντική τεχνική της μηχανικής μάθησης που βοηθά στην κατανόηση και επεξεργασία των δεδομένων [1], [31].

4.5 Παλινδρόμηση

Η παλινδρόμηση είναι μια μέθοδος στη μηχανική μάθηση που χρησιμοποιείται για την εκμάθηση μιας συνάρτησης πρόβλεψης ή εκτίμησης, που συνδέει μια εξαρτημένη μεταβλητή με ένα ή περισσότερα ανεξάρτητα χαρακτηριστικά. Στόχος είναι να βρεθεί η **βέλτιστη** συνάρτηση πρόβλεψης η οποία προσαρμόζεται σε ένα σύνολο δεδομένων εκπαίδευσης, έτσι ώστε να μπορεί να κάνει προβλέψεις για νέα δεδομένα. Σε αντίθεση με την ταξινόμηση όπου οι στόχοι είναι διακριτές τιμές, τα μοντέλα παλινδρόμησης προβλέπουν μια συνεχή έξοδο, δηλαδή οι στόχοι αντιστοιχούν σε αξίες κάποιων ποσοτήτων [1], [31]. Επομένως, σύμφωνα με τα παραπάνω, η παλινδρόμηση μπορεί να χρησιμοποιηθεί σε πολλά προβλήματα μηχανικής μάθησης, όπως η πρόβλεψη τιμών ακινήτων, η πρόβλεψη της ζήτησης ή η πρόβλεψη του κόστους παραγωγής ενός προϊόντος.

4.6 Scikit Learn

Το Scikit-learn [34] είναι μια δημοφιλής ανοιχτού κώδικα βιβλιοθήκη μηχανικής μάθησης για τη γλώσσα προγραμματισμού Python. Παρέχει μια σειρά από εργαλεία για την προεπεξεργασία δεδομένων, την επιλογή χαρακτηριστικών, καθώς και μια ποικιλία αλγορίθμων μηχανικής μάθησης. Όλοι οι αλγόριθμοι είναι ενσωματωμένοι, έτσι ώστε να μην φαίνονται τα "βαριά" μαθηματικά και οι λεπτομέρειες του τρόπου λειτουργίας τους [1].

Η βιβλιοθήκη έχει μια απλή και συνεπή διεπαφή προγραμματισμού που την καθιστά εύκολη στη χρήση και συνδυάζεται καλά με άλλες βιβλιοθήκες Python όπως η NumPy [35], η Pandas [36] και η Matplotlib [8]. Έχει επίσης μια ισχυρή κοινότητα και εκτενή τεκμηρίωση με μια μεγάλη ποικιλία παραδειγμάτων και εκπαιδευτικών υλικών. Γενικά, το Scikit-learn είναι μια ισχυρή και ευέλικτη βιβλιοθήκη για την κατασκευή αλγορίθμων μηχανικής μάθησης στην Python.

Επιπλέον, η Scikit-learn περιλαμβάνει και συσκευασμένα σύνολα δεδομένων για πειραματισμό [1]. Παρακάτω παρουσιάζονται ορισμένα απο αυτά:

- Είδη φυτού ίριδας (Iris plants)
- Οπτική αναγνώριση χειρόγραφων ψηφίων (Optical recognition of handwritten digits)
- Αναγνώριση κρασιών (Wine recognition)
- Τιμές σπιτιών Βοστώνης (Boston house prices)
- Ανίχνευση εισβολών σε δίκτυα (Knowledge Discovery and Data Mining - Kdd Cup 1999)

4.7 Δεδομένα και υπολογιστική ισχύς

Σήμερα, η ποσότητα των δεδομένων που είναι διαθέσιμη είναι τεράστια και αυξάνεται εκθετικά. Ο όρος "μεγάλα δεδομένα" μπορεί να μην είναι αρκετά ισχυρός για να περιγράψει το πραγματικό μέγεθός τους. Παλιότερα, οι άνθρωποι έλεγαν ότι πνίγονται στα δεδομένα και δεν ξέρουν τι να κάνουν με αυτά. Τώρα όμως, με την επαναστατική ανάπτυξη της μηχανικής μάθησης, οι άνθρωποι πλέον επιδιώκουν να συλλέξουν όσο περισσότερα δεδομένα μπορούν, έτσι ώστε να χρησιμοποιήσουν την τεχνολογία μηχανικής μάθησης για να εξάγουν χρήσιμες πληροφορίες και να προβλέψουν τάσεις και πρότυπα βάσει αυτών των δεδομένων. Αυτό συμβαίνει σε μια εποχή όπου η ισχύς των υπολογιστών αυξάνει εκρηκτικά, μαζί με τη μνήμη και την αποθηκευτική χωρητικότητά τους, ενώ τα κόστη τους μειώνονται δραστικά. Όλα αυτά μας επιτρέπουν να σκεφτόμαστε διαφορετικά και να προγραμματίζουμε τους υπολογιστές να μαθαίνουν απο δεδομένα. Τα πάντα πλέον είναι ζήτημα πρόβλεψης βάσει δεδομένων [1].

4.8 Βήματα περίπτωσης χρήσης μηχανικής μάθησης

Γενικά, σε κάθε πείραμα μηχανικής μάθησης, εφαρμόζονται τα παρακάτω βήματα:

- Φόρτωση των συνόλων δεδομένων.
- Προεπεξεργασία / καθαρισμός των δεδομένων.
- Μετατροπή των μη αριθμητικών δεδομένων σε αριθμητικά απο τη στιγμή που το απαιτούν τα μοντέλα.
- Διαίρεση των δεδομένων σε δείγματα εκπαίδευσης και ελέγχου.
- Δημιουργία και εκπαίδευση του μοντέλου.
- Εκτέλεση προβλέψεων για νέα δεδομένα τα οποία το μοντέλο δεν έχει δει στο παρελθόν.
- Προβολή πίνακα που παρουσιάζει μετρικές αξιολόγησης (recall, precision και F1-score).

4.9 Διανυσματοποιητές (Tokenizers)

Στην περίπτωση της επεξεργασίας φυσικής γλώσσας χρειάζεται ένας τρόπος να αναπαραστήσουμε τις λέξεις σε αριθμούς, ώστε να είναι κατανοητές από τους υπολογιστές. Αυτό επιτυγχάνεται με πολλούς τρόπους. Εμείς θα εφαρμόσουμε τις τεχνικές TF-IDF για αναπαράσταση των λέξεων ως αριθμούς μέσα σε ένα διάνυσμα, και το Word2Vec για την αναπαράσταση των λέξεων ως αριθμητικά διανύσματα, προκειμένου τα κείμενα να μπορούν να αναλυθούν μαθηματικά και να εφαρμοστούν σε αλγορίθμους μηχανικής μάθησης.

4.9.1 Κωδικοποίηση TF-IDF

Η μέθοδος TF-IDF (Term Frequency-Inverse Document Frequency) είναι μια τεχνική που χρησιμοποιείται στην επεξεργασία φυσικής γλώσσας και αποσκοπεί στον υπολογισμό της σημασιολογικής σημασίας ενός όρου σε ένα σύνολο κειμένων. Αξιολογεί μέσω δύο μετρικών: του αριθμού εμφανίσεών του όρου σε ένα κείμενο (TF) και της συχνότητας εμφάνισής του σε όλα τα κείμενα (IDF). Η μέθοδος TF-IDF συνδυάζει αυτές τις δύο μετρικές για να προσδιορίσει ποιοι όροι είναι πιο σημαντικοί στα δεδομένα [5].

Πιο συγκεκριμένα, η μέθοδος TF-IDF υπολογίζει το TF για κάθε όρο στο κάθε κείμενο ως τον αριθμό των εμφανίσεών του όρου στο κείμενο. Έπειτα, υπολογίζει το IDF για κάθε όρο ως το λογάριθμο του αριθμού των κειμένων στο σύνολο δεδομένων διαιρεμένο με τον αριθμό των κειμένων που περιέχουν τον συγκεκριμένο όρο. Ο συνολικός βαθμός TF-IDF για έναν όρο υπολογίζεται ως το γινόμενο των μετρικών TF και IDF.

Στην εργασία μας κάνουμε χρήση του αλγορίθμου TF-IDF ώστε να αναθέσουμε ένα σκορ σε κάθε λέξη και να δημιουργήσουμε διανύσματα με τιμές το tf-idf score. Για παράδειγμα, ας θεωρήσουμε την πρόταση ‘Γρήγορη παραλαβή και φιλική εξυπηρέτηση’ η οποία αποτελείται από πέντε λέξεις και ανήκει στο train set του Dataset Skroutz. Το διάνυσμα της πρότασης θα περιλαμβάνει πέντε αριθμούς της μορφής:

X=[0.3583028082181012, 0.6847026225285427, 0.15015921969190346, 0.3821833261093528, 0.4839313280942549].

Κάθε αριθμός του διανύσματος αντιστοιχεί στην αντίστοιχη λέξη στην πρόταση. Για χάριν απλούστευσης παρουσιάσαμε την πρόταση X σαν ένα διάνυσμα 5 διαστάσεων. Για την ακρίβεια, το διάνυσμα της παραπάνω πρότασης θα έχει τόσες διαστάσεις όσες οι λέξεις που απαρτίζουν το λεξικό των δεδομένων εκπαίδευσης. Οι λέξεις που δεν εμφανίζονται στην πρόταση λαμβάνουν την τιμή 0.

Το tf-idf score της λέξης t μιας πρότασης d υπολογίζεται από τον τύπο:

$$tfidf(t,d) = tf(t,d) * idf(t) \tag{4.1}$$

όπου,

$$tf(t,d)\ o\ αριθμός\ των\ φορών\ που\ εμφανίζεται\ ο\ όρος\ t\ στο\ corpus\ d.$$

$$idf(t) = log\frac{1+n}{1+df(t)} + 1 \tag{4.2}$$

Όπου n ο συνολικός αριθμός σχολίων και df(t) είναι ο αριθμός των σχολίων που περιέχουν τη λέξη t.

Στη συνέχεια το διάνυσμα κανονικοποιείται με χρήση της Ευκλείδειας νόρμας.

$$v_{norm} = \frac{v}{\|v\|_2} = \frac{v}{\sqrt{{v_1}^2 + {v_2}^2 + \cdots + {v_k}^2}} \tag{4.3}$$

Όπου k, η διάσταση του διανύσματος, δηλαδή το πλήθος των λέξεων του λεξικού.

4.9.2 Μοντέλο Word2Vec

Στο μοντέλο Word2Vec, όπως υποδεικνύει και το όνομά του, κάθε λέξη αντιστοιχίζεται σε ένα διάνυσμα στον n-διάστατο χώρο, όπου n είναι ένας προκαθορισμένος αριθμός. Τα διανύσματα αυτά αντιπροσωπεύουν τη σημασία των λέξεων και είναι εκτιμώμενα από ένα νευρωνικό δίκτυο. Με αυτό τον τρόπο, λέξεις που εμφανίζονται συχνά μαζί σε μια πρόταση ή σε ένα κείμενο, έχουν παρόμοια διανύσματα.

Το Word2Vec βασίζεται σε δύο αλγορίθμους [37], [38]: τον Continuous Bag of Words (CBoW) και τον Skip-Gram. Στη μέθοδο CBoW, ο στόχος είναι να προβλέψουμε μια λέξη βάσει των γειτονικών της λέξεων στην πρόταση ή στο κείμενο, ενώ η μέθοδος Skip-Gram προσπαθεί να προβλέψει τις γειτονικές λέξεις για μια δεδομένη λέξη. Με αυτό τον τρόπο κατασκευάζει έναν πίνακα διανυσμάτων για ένα corpus το οποίο έχει και σημασιολογικές δυνατότητες. Για παράδειγμα, η λέξη ‘βασιλιάς’ είναι κοντά σημασιολογικά στη λέξη ‘βασίλισσα’, συνεπώς θα αναπαρασταθούν με αριθμούς πολύ κοντινούς. Η προσέγγιση αυτή δημιουργεί αριθμούς που επιτρέπουν και πράξεις του τύπου: ‘γυναίκα’ + ‘βασιλιάς’ – ‘άνδρας’ = ‘βασίλισσα’, όπου οι αριθμητικές αναπαραστάσεις προσεγγίσουν την σημασιολογία των λέξεων.

Οι δύο παραπάνω μέθοδοι συνιστούν νευρωνικά δίκτυα οπότε περισσότερα θα αναφερθούν στο κεφάλαιο 5 Βαθιά Μάθηση.

4.10 Μετρικές ακρίβειας μοντέλων

Για κάθε μοντέλο που εκπαιδεύουμε, πρέπει να μετράμε την ακρίβειά του σε νέα δεδομένα. Παρακάτω θα δείξουμε μετρικές όπως ο πίνακας σύγχυσης (confusion matrix) και η αναφορά ταξινόμησης (classification report) η οποία προβάλλει έναν πίνακα με μετρικές ταξινόμησης (classification metrics).

4.10.1 Πίνακας Σύγχυσης (Confusion Matrix)

Ένας τρόπος να ελέγξουμε την ακρίβεια ενός ταξινομητή είναι μέσω ενός **πίνακα σύγχυσης** (confusion matrix), ο οποίος παρουσιάζει τις σωστές και λανθασμένες προβλέψεις για κάθε κατηγορία. Για τον σκοπό αυτό χρησιμοποιείται η συνάρτηση **confusion_matrix()** της βιβλιοθήκης Scikit-learn. Η εν λόγω συνάρτηση δέχεται δύο ορίσματα, τις πραγματικές και τις προβλεφθείσες κλάσεις και παράγει τον πίνακα σύγχυσης [1]. Οι σωστές προβλέψεις εμφανίζονται στην κύρια διαγώνιο του πίνακα ενώ τιμές που εμφανίζονται εκτός της διαγωνίου επισημαίνουν λανθασμένες προβλέψεις [1]. Επίσης, υπάρχει η δυνατότητα χρήσης ενός **χάρτη θερμότητας** (heat map) ο οποίος προβάλλει τις τιμές του πίνακα σύγχυσης με διάφορα χρώματα. Με έντονο χρωματικό background

εμφανίζονται συνήθως οι μεγαλύτερες τιμές. Η δημιουργία του χάρτη θερμότητας γίνεται με την συνάρτηση **heatmap()** της βιβλιοθήκης Seaborn [9].

Η Εικόνα 4.1, παρουσιάζει έναν χάρτη θερμότητας (heatmap) για το μοντέλο **BERT** το οποίο αναλύεται στο κεφάλαιο 5. Ο εν λόγω χάρτης θερμότητας παρουσιάζει τις ορθές και λανθασμένες προβλέψεις για τρεις κλάσεις (Negative, Neutral και Positive). Παρατηρώντας την κύρια διαγώνιο με τιμές 200, 164 και 149, διαπιστώνουμε ότι 200 δείγματα της κλάσης Negative έχουν ταξινομηθεί ορθά ως Negative. Ομοίως, 164 δείγματα της κλάσης Neutral έχουν ταξινομηθεί ορθά ως Neutral, και 149 δείγματα της κλάσης Positive έχουν ταξινομηθεί σαν Positive. Κάθε άλλη τιμή εκτός διαγωνίου συνιστά λανθασμένη πρόβλεψη όπως για παράδειγμα η τιμή 45, η οποία μας αναφέρει ότι 45 δείγματα της κλάσης Neutral, έχουν ταξινομηθεί λανθασμένα ως Positive. Η ιδανικότερη περίπτωση προφανώς θα ήταν να έχουμε μηδενικές τιμές σε όλα τα τετράγωνα εκτός της κύρια διαγωνίου.

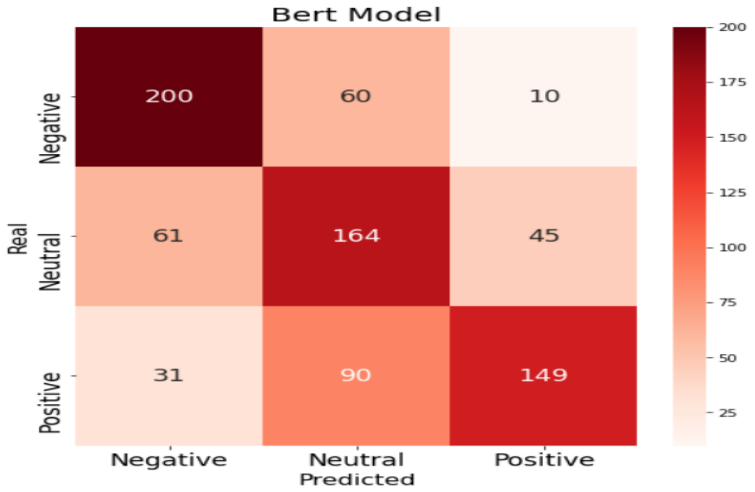
Συνεπώς, το πρώτο μέτρο εκτίμησης της επίδοσης είναι ο λόγος των ορθών προβλέψεων προς το σύνολο όλων των προβλέψεων. Ο λόγος αυτός ονομάζεται ακρίβεια (accuracy) και δίνεται απο τον τύπο:

$$Accuracy = \frac{Ορθές\ προβλέψεις\ διαγωνίου}{Σύνολο\ προβλέψεων} \tag{4.4}$$

Και ισχύει: $0 \leq Accuracy \leq 1$

Επομένως, για το παράδειγμα της εικόνας 4.1, η ακρίβεια που παίρνουμε είναι:

$$Accuracy = \frac{200 + 164 + 149}{200 + 164 + 149 + 60 + 10 + 61 + 45 + 31 + 90} = \frac{513}{810} = 0.63$$



Εικόνα 4.1: Χάρτης θερμότητας τριών κλάσεων

4.10.2 Αναφορά Ταξινόμησης (Classification Report)

Η βιβλιοθήκη Scikit-learn επίσης διαθέτει την συνάρτηση **classification_report()**, η οποία παράγει ένα πίνακα με μετρικές ταξινόμησης [1]. Όπως στην ενότητα 4.10.1 δημιουργήθηκε ο χάρτης θερμότητας για μια αρχιτεκτονική μοντέλου BERT, έτσι και εδώ θα γίνει η παρουσίαση της αντίστοιχης αναφοράς ταξινόμησης για την ίδια αρχιτεκτονική του μοντέλου BERT. Πιο συγκεκριμένα, η εν λόγω αναφορά (Εικόνα 4.2), εμφανίζει μετρικές όπως η ευστοχία (precision), η ανάκληση (recall), η ακρίβεια (accuracy) που πολλές φορές εμφανίζεται και ως micro avg, καθώς και το F1-score ή F-measure.

	precision	recall	f1-score	support
Negative	0.68	0.74	0.71	270
Neutral	0.52	0.61	0.56	270
Positive	0.73	0.55	0.63	270
micro avg	0.63	0.63	0.63	810
macro avg	0.65	0.63	0.63	810
weighted avg	0.65	0.63	0.63	810
samples avg	0.63	0.63	0.63	810

Εικόνα 4.2: Αναφορά ταξινόμησης τριών κλάσεων

Ευστοχία (Precision)

Είναι ο συνολικός αριθμός σωστών προβλέψεων για μια συγκεκριμένη κλάση διαιρεμένος με τον συνολικό αριθμό προβλέψεων για αυτή την κλάση [1], [38], [39]. Παρατηρώντας την στήλη Negative του πίνακα σύγχυσης της εικόνας 4.1, βλέπουμε τους αριθμούς 61 και 31, που σημαίνει ότι 61 δείγματα που ανήκουν στην κλάση Neutral ταξινομήθηκαν λανθασμένα στην κλάση Negative. Επίσης, 31 δείγματα της κλάσης Positive ταξινομήθηκαν ως Negative. Ο αριθμός 200 με έντονο κόκκινο, σημαίνει ότι 200 δείγματα της κλάσης Negative, ταξινομήθηκαν σωστά ως Negative. Συνεπώς, ο τύπος που υπολογίζει την ευστοχία (precision), είναι ο εξής:

$$Precision = \frac{TP}{TP + FP} \tag{4.5}$$

Όπου TP ο αριθμός των σωστών προβλέψεων για μια συγκεκριμένη κλάση, και FP ο αριθμός των λανθασμένων προβλέψεων για μια συγκεκριμένη κλάση.

Έτσι, στο παράδειγμά μας το precision της κλάσης Negative ισούται με:

$$Precision = \frac{TP}{TP + FP} = \frac{200}{200 + (61 + 31)} = \frac{200}{292} = 0.68$$

Το precision της κλάσης Positive είναι:

$$Precision = \frac{TP}{TP + FP} = \frac{149}{149 + (45 + 10)} = \frac{149}{204} = 0.73$$

Το precision της κλάσης Neutral είναι:

$$Precision = \frac{TP}{TP + FP} = \frac{164}{164 + (60 + 90)} = \frac{164}{314} = 0.52$$

Ανάκληση (Recall)

Είναι ο συνολικός αριθμός σωστών προβλέψεων για μια συγκεκριμένη κλάση, διαιρεμένος με τον συνολικό αριθμό δειγμάτων που θα έπρεπε να έχουν προβλεφθεί σωστά ως αυτή η κλάση [1], [38], [39]. Παρατηρώντας την γραμμή της κλάσης Negative του πίνακα σύγχυσης της εικόνας 4.1, βλέπουμε τους αριθμούς 60 και 10, οι οποίοι υποδηλώνουν ότι 60 δείγματα της κλάσης Negative ταξινομήθηκαν λανθασμένα ως Neutral και 10 δείγματα της κλάσης Negative ταξινομήθηκαν και αυτά λανθασμένα ως Positive. Επίσης, φαίνεται ο αριθμός 200 που υποδηλώνει ότι 200 δείγματα της κλάσης Negative ταξινομήθηκαν σωστά στην κλάση Negative. Επομένως, η ανάκληση (recall) δίνεται απο τον τύπο:

$$Recall = \frac{TP}{TP + FN} \tag{4.6}$$

Όπου TP ο συνολικός αριθμός σωστών προβλέψεων για μια συγκεκριμένη κλάση, και FN οι προβλέψεις των δειγμάτων της συγκεκριμένης κλάσης, που όμως τελικά ταξινομήθηκαν ως άλλες κλάσεις. Έτσι, στο παράδειγμά μας το recall της κλάσης Negative ισούται με:

$$Recall = \frac{TP}{TP + FN} = \frac{200}{200 + (60 + 10)} = \frac{200}{270} = 0.74$$

Το recall της κλάσης Positive είναι:

$$Recall = \frac{TP}{TP + FN} = \frac{149}{149 + (31 + 90)} = \frac{149}{270} = 0.55$$

Το recall της κλάσης Neutral είναι:

$$Recall = \frac{TP}{TP + FN} = \frac{164}{164 + (61 + 45)} = \frac{164}{270} = 0.61$$

Μετρική F1 (F1-score ή F1-measure)

Στην ταξινόμηση, η ευστοχία (precision) και η ανάκληση (recall) είναι δύο σημαντικές μετρικές που χρησιμοποιούνται για να μετρήσουν την απόδοση ενός μοντέλου. Ωστόσο, αυτές οι μετρικές δεν είναι αρκετές για να δώσουν μια πλήρη εικόνα της συνολικής απόδοσης. Για τον λόγο αυτό, συνήθως συνδυάζονται στο κριτήριο **F1-score** [38], [39].

Πιο συγκεκριμένα, το F1-score υπολογίζεται ως το πηλίκο του γεωμετρικού μέσου προς τον αλγεβρικό μέσο όρο των μετρικών precision και recall [38]:

$$F1.score = \frac{Precision \cdot Recall}{(Precision + Recall)/2} \tag{4.7}$$

Εύκολα αποδεικνύεται από τον τύπο ότι κυμαίνεται μεταξύ 0 και 1. Επιτυγχάνει την μέγιστη τιμή 1 αν ισχύει $Precision = Recall = 1$.

Έτσι, το F1-score είναι ένας καλός τρόπος για να αξιολογηθεί η συνολική απόδοση ενός μοντέλου στην ταξινόμηση.

Άρα, με εφαρμογή του τύπου, το F1-score της κλάσης Negative, είναι:

$$F1.score = \frac{Precision \cdot Recall}{(Precision + Recall)/2} = \frac{0.68 \cdot 0.74}{(0.68 + 0.74)/2} = \frac{0.5032}{1.42/2} = \frac{0.5032}{0.71} = 0.70873$$

F1-score της κλάσης Neutral:

$$F1.score = \frac{0.52 \cdot 0.61}{(0.52 + 0.61)/2} = \frac{0.3172}{0.565} = 0.56141$$

F1-score της κλάσης Positive:

$$F1.score = \frac{0.73 \cdot 0.55}{(0.73 + 0.55)/2} = \frac{0.4015}{0.64} = 0.62734$$

Υποστήριξη (Support)

Πρόκειται για τον αριθμό των δειγμάτων σε μια συγκεκριμένη κατηγορία - κλάση. Στην αναφορά της εικόνας 4.2, το support μας πληροφορεί ότι έχουμε 270 δείγματα σε κάθε κατηγορία. Επομένως, τα δεδομένα έχουν **ισορροπημένες κατηγορίες** (balanced classes), δηλαδή τα δείγματα είναι ομαλά κατανεμημένα μεταξύ των κλάσεων. Αυτό ισχύει για όλα τα σύνολα δεδομένων που είναι

συσκευασμένα στη Scikit-learn. Μη ισορροπημένες κλάσεις μπορεί να οδηγήσουν σε λανθασμένα αποτελέσματα.

Στη συνέχεια παρουσιάζουμε τους μέσους όρους που εμφανίζονται στο τέλος της αναφοράς ταξινόμησης των τριών κλάσεων της εικόνας 4.2. Επίσης, οι υπολογισμοί γίνονται με βάση τον πίνακα σύγχυσης της ενότητας 4.10.1.

Μικρομέσος όρος (Micro Average F1-score)

Ο μικρομέσος όρος [39], υπολογίζει έναν καθολικό μέσο όρο βαθμολογίας F1 μετρώντας τα αθροίσματα των αληθινών θετικών (TP), των ψευδών αρνητικών (FN) και των ψευδών θετικών (FP). Για το λόγο αυτό, για τον υπολογισμό του δεν θα χρησιμοποιηθεί ο τύπος (4.7) ο οποίος υπολογίζει το F1-score για κάθε κλάση με βάση τις τιμές precision και recall. Επομένως, θα πρέπει να εκφράσουμε το F1-score με βάση τις μεταβλητές TP, FN, και FP, οπότε προκύπτει ο τύπος:

$$F1.score = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{4.8}$$

Αρχικά, κατασκευάζουμε τον Πίνακα 4.1 ο οποίος θα περιλαμβάνει τα αθροίσματα των μεταβλητών TP, FP, FN, και στη συνέχεια θα εφαρμόσουμε τον τύπο (4.8) για να πάρουμε το Micro Average F1-score.

Πίνακας 4.1: Αθροίσματα TP, FP, FN

CLASS	TP	FP	FN
Negative	200	92	70
Neutral	164	150	106
Positive	149	55	121
Total	**513**	**297**	**297**

Σύμφωνα με τον τύπο (4.8) έχουμε:

$$F1.score = \frac{TP}{TP + \frac{1}{2}(FP + FN)} = \frac{513}{513 + \frac{1}{2}(297 + 297)} = \frac{513}{810} = 0.63$$

Συνεπώς, έχουμε υπολογίσει το Micro Average F1-score το οποίο ισούται με **0.63**

Σε πολλές αναφορές ταξινόμησης, ο μικρομέσος όρος εμφανίζεται με την σήμανση **accuracy** και όχι ως **micro avg**. Αυτό συμβαίνει επειδή ο μικρομέσος όρος ουσιαστικά υπολογίζει το ποσοστό των σωστά ταξινομημένων δειγμάτων από όλα τα δείγματα [39]. Να σημειωθεί ότι, όπως ο τύπος (4.7)

χρησιμοποιείται για να υπολογίσουμε το F1-score της κάθε κλάσης, έτσι και ο τύπος (4.8) μπορεί να χρησιμοποιηθεί για τον ίδιο σκοπό, αφού πρώτα βέβαια υπολογιστούν οι μεταβλητές TP, FP, FN.

Επιπλέον, αν υπολογίσουμε τον μικρομέσο όρο για την ευστοχία (precision) και για την ανάκληση (recall), θα πάρουμε την ίδια τιμή **0.63**, όπως θα δούμε παρακάτω.

Μικρομέσος όρος για Ευστοχία / Ανάκληση (Micro Average precision / recall)

Για τον υπολογισμό του Micro Average για το precision παίρνουμε τα αθροίσματα του Πίνακα 4.1 και τα βάζουμε στον τύπο υπολογισμού του precision, οπότε έχουμε:

$$Precision = \frac{TP}{TP + FP} = \frac{513}{513 + 297} = \frac{513}{810} = 0.63$$

$$Άρα: Micro\ Average\ precision = 0.63$$

Ομοίως, για τον υπολογισμό του Micro Average για το recall, θα έχουμε:

$$Recall = \frac{TP}{TP + FN} = \frac{513}{513 + 297} = \frac{513}{810} = 0.63$$

$$Άρα: Micro\ Average\ recall = 0.63$$

Όπως γίνεται φανερό, τα παραπάνω αποτελέσματα υποδηλώνουν ότι σε περιπτώσεις ταξινόμησης πολλών κατηγοριών όπου κάθε δείγμα ανήκει σε μία μόνο κλάση, το micro average F1-score, micro average precision, micro average recall, και η ακρίβεια (accuracy) μοιράζονται την ίδια τιμή. Για τον λόγο αυτό σε πολλές αναφορές ταξινόμησης εμφανίζεται μόνο η ακρίβεια (accuracy), καθώς η micro average precision και η micro average recall έχουν την ίδια τιμή [39]. Οπότε γενικά ισχύει:

$$accuracy = micro\ avg\ F1.score = micro\ avg\ precision = micro\ avg\ recall$$

Μακρομέσος όρος (Macro Average)

Ο μακρομέσος όρος (Macro Average), είναι ίσως η πιο απλή μεταξύ των πολυάριθμων μεθόδων υπολογισμού μέσου όρου. Ο μακρομέσος όρος F1-score, υπολογίζεται χρησιμοποιώντας τον αριθμητικό μέσο όρο (μη σταθμισμένος μέσος όρος) όλων των F1-score ανά κλάση. Έτσι, όλες οι κλάσεις αντιμετωπίζονται εξίσου, ανεξάρτητα από την υποστήριξη (support) που δίνουν [39].

Έτσι, για την αναφορά ταξινόμησης του παραδείγματός μας, έχουμε:

$$Macro\ avg\ F1.score = \frac{0.71 + 0.56 + 0.63}{3} = \frac{1.9}{3} = 0.63$$

Για την περίπτωση του Macro avg precision θα έχουμε:

$$Macro\ avg\ precision = \frac{0.68 + 0.52 + 0.73}{3} = \frac{1.93}{3} \approx 0.65$$

Και για την περίπτωση του Macro avg recall θα έχουμε:

$$Macro\ avg\ recall = \frac{0.74 + 0.61 + 0.55}{3} = \frac{1.9}{3} = 0.63$$

Σταθμισμένος μέσος όρος (Weighted Average)

Ο σταθμισμένος μέσος όρος (Weighted Average) F1-score υπολογίζεται παίρνοντας τον μέσο όρο όλων των F1-score ανά κλάση, λαμβάνοντας υπόψη την υποστήριξη κάθε κατηγορίας [39].

Η υποστήριξη (support) αναφέρεται στον αριθμό των πραγματικών εμφανίσεων της κλάσης στο σύνολο δεδομένων. Έτσι, η τιμή υποστήριξης 270 στη κλάση Positive, σημαίνει ότι υπάρχουν 270 δείγματα που ανήκουν στην κλάση Positive. Το βάρος (weight) ουσιαστικά αναφέρεται στην αναλογία της υποστήριξης κάθε κλάσης σε σχέση με το άθροισμα όλων των τιμών υποστήριξης.

Προς διευκόλυνσή μας, πριν τον υπολογισμό κατασκευάζουμε τον παρακάτω πίνακα.

Πίνακας 4.2: Βοηθητικός πίνακας υπολογισμού Weighted Average F1-score

CLASS	PRECISION	RECALL	F1-SCORE	SUPPORT	SUPPORT PROPORTION
Negative	0.68	0.74	0.71	270	0.33333
Neutral	0.52	0.61	0.56	270	0.33333
Positive	0.73	0.55	0.63	270	0.33333
Total	-	-	-	**810**	**1.0**

Επομένως, για Weighted Average F1-score θα έχουμε:

$$Weighted\ avg\ F1.score = 0.71 \cdot 0.33333 + 0.56 \cdot 0.33333 + 0.63 \cdot 0.33333 \approx 0.63$$

Ομοίως για Weighted Average precision:

$$Weighted\ avg\ precision = 0.68 \cdot 0.33333 + 0.52 \cdot 0.33333 + 0.73 \cdot 0.33333 \approx 0.65$$

Ομοίως για Weighted Average recall:

$$Weighted\ avg\ recall = 0.74\ \cdot 0.33333 + 0.61 \cdot 0.33333 + 0.55 \cdot 0.33333 \approx 0.63$$

Με τον σταθμισμένο μέσο όρο, ο μέσος όρος αντιπροσωπεύει τη συνεισφορά κάθε κλάσης όπως σταθμίζεται με τον αριθμό των δειγμάτων αυτής.

4.11 Η έννοια της Γενίκευσης

Όπως αναφέραμε στην ενότητα της παλινδρόμησης, στόχος είναι να βρεθεί η **βέλτιστη** συνάρτηση πρόβλεψης f η οποία προσαρμόζεται σε ένα σύνολο δεδομένων εκπαίδευσης, προκειμένου να μπορεί να κάνει προβλέψεις για νέα δεδομένα. Επειδή η εύρεση της συνάρτησης f είναι ιδιαίτερα δύσκολη υπόθεση, κάνουμε την υπόθεση ότι η f ανήκει σε μια μεγάλη οικογένεια συναρτήσεων F η οποία παραμετροποιείται από το διάνυσμα **w** [38].

Έτσι, για παράδειγμα το σύνολο F_{Linear} των γραμμικών συναρτήσεων της μορφής:

$$f(\boldsymbol{x}; \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{x} + w_0 = w_1 x_1 + \ldots + w_n x_n + w_0$$

παραμετροποιείται από το διάνυσμα:

$$\boldsymbol{w} = [w_0, w_1, \ldots, w_n]^T$$

Γενικά ισχύει:

$$t = f(\boldsymbol{x}; \boldsymbol{w}) + s \tag{4.9}$$

Όπου:

t η επιθυμητή έξοδος, $\boldsymbol{x}$ το διάνυσμα εισόδου και s το σφάλμα παρατήρησης ή θόρυβος.

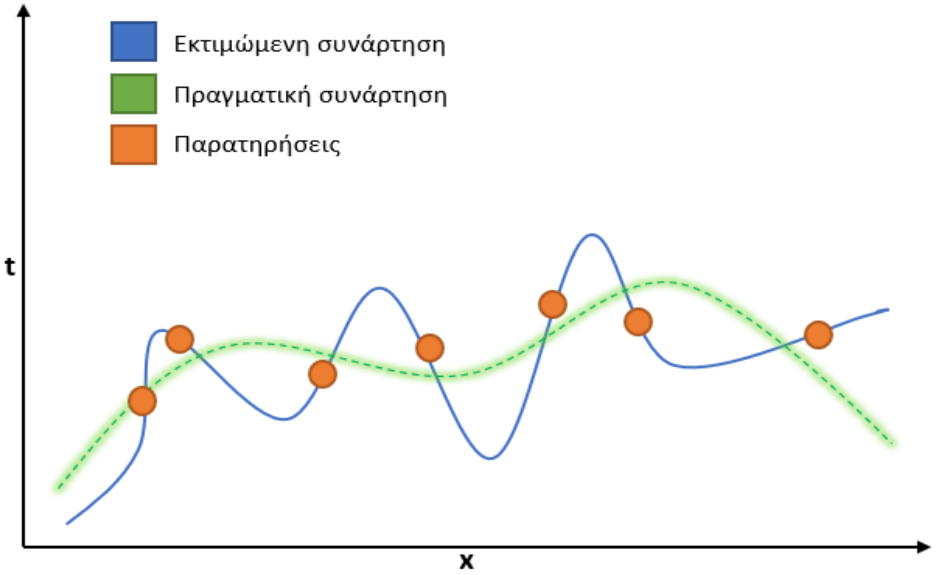
Αν μπορέσουμε να βρούμε τις κατάλληλες παραμέτρους $\boldsymbol{w}$ έτσι ώστε η συνάρτηση να προσεγγίζει την επιθυμητή έξοδο με όσο το δυνατόν μικρότερο σφάλμα s για κάθε δυνατή είσοδο $\boldsymbol{x}$, τότε λέμε ότι το μοντέλο μας γενικεύει [38].

Σύμφωνα λοιπόν με τα παραπάνω, προκύπτει ο **ορισμός** της **γενίκευσης**:

Η ικανότητα ενός μοντέλου να προβλέπει σωστά την έξοδο για οποιαδήποτε δεδομένα εισόδου ακόμα και αν αυτά τα δεδομένα δεν τα έχει ξανα δει κατά την εκπαίδευση [38].

Αυτό είναι ιδιαίτερα σημαντικό, καθώς δείχνει ότι το μοντέλο μπορεί να χρησιμοποιηθεί αποτελεσματικά σε πραγματικά προβλήματα, όπου οι είσοδοι μπορεί να είναι ποικίλες και διαφορετικές από αυτές που χρησιμοποιήθηκαν κατά την εκπαίδευσή του.

Στο Σχήμα 4.1, φαίνεται ο στόχος της μάθησης.



Σχήμα 4.1: Στόχος είναι να βρεθεί η συνάρτηση που δημιούργησε τα δεδομένα [38]

Εφαρμογή γενίκευσης για την αναγνώριση ανεπιθύμητων emails (spam)

Η γενίκευση είναι μια κρίσιμη ικανότητα για ένα μοντέλο ταξινόμησης ή αναγνώρισης ανεπιθύμητων (spam) και κανονικών (ham) email, καθώς ο στόχος είναι να μπορεί να αναγνωρίσει και να ταξινομήσει σωστά τα μηνύματα ανεξάρτητα από το περιεχόμενό τους. Ένα καλά εκπαιδευμένο μοντέλο θα πρέπει να μπορεί να αναγνωρίζει μηνύματα από άγνωστους αποστολείς, καθώς και μηνύματα με νέους τύπους ανεπιθύμητης αλληλογραφίας. Αυτό απαιτείται για να εξασφαλιστεί ότι το μοντέλο μπορεί να εφαρμοστεί με αποτελεσματικότητα σε νέες καταστάσεις, όπου μπορεί να εμφανιστούν απειλές από ανεπιθύμητα μηνύματα. Συνεπώς, η γενίκευση είναι ένας βασικός παράγοντας για την αξιοπιστία και απόδοση των μοντέλων ταξινόμησης και αναγνώρισης ανεπιθύμητων email.

Έτσι, για την αναγνώριση και ταξινόμηση των email, θα πρέπει να εφαρμοστεί μια προσέγγιση μηχανικής μάθησης κατά την οποία θα γίνεται εξαγωγή των χαρακτηριστικών των email και τοποθέτησή τους σε ένα διάνυσμα [38]. Στη συνέχεια γίνεται εκτίμηση της συνάρτησης f η οποία παράγει την ορθή ετικέτα spam ή ham για κάθε διάνυσμα εισόδου $\boldsymbol{x}$, δηλαδή [38]:

$$y = f(\boldsymbol{x}; \boldsymbol{w}) = \begin{cases} 1 \rightarrow ham \\ 0 \rightarrow spam \end{cases}$$

Γίνεται εκπαίδευση του συστήματος-μοντέλου, χρησιμοποιώντας ένα σύνολο προτύπων-δειγμάτων και στόχων-ετικετών:

$$X_{train} = \{(\boldsymbol{x}_1, t_1), (\boldsymbol{x}_2, t_2), (\boldsymbol{x}_3, t_3), \ldots, (\boldsymbol{x}_n, t_n)\}$$

Το σύνολο X_{train} ονομάζεται σύνολο εκπαίδευσης (train set). Κατά την διαδικασία της εκπαίδευσης, ουσιαστικά γίνεται ρύθμιση των παραμέτρων $\boldsymbol{w}$ προκειμένου είτε να μεγιστοποιείται το κριτήριο επίδοσης (accuracy), είτε να ελαχιστοποιείται το κριτήριο σφάλματος (loss) [38].

Τέλος, πρέπει να γίνει έλεγχος του μοντέλου με ένα διαφορετικό σύνολο προτύπων-δειγμάτων και στόχων-ετικετών, το οποίο ονομάζεται σύνολο ελέγχου (test set) [38]:

$$X_{test} = \{(\boldsymbol{x}'_1, t'_1), (\boldsymbol{x}'_2, t'_2), (\boldsymbol{x}'_3, t'_3), \ldots, (\boldsymbol{x}'_n, t'_n)\}$$

4.11.1 Υπερπροσαρμογή (Overfitting) και Υποπροσαρμογή (Underfitting)

Όπως έγινε ήδη προφανές, βασικός στόχος κατά την δημιουργία ενός μοντέλου μηχανικής μάθησης είναι να διασφαλιστεί ότι θα είναι ικανό να κάνει ακριβείς προβλέψεις για δεδομένα που δεν έχει δει ακόμη. Δύο συνηθισμένα προβλήματα που εμποδίζουν τις ακριβείς προβλέψεις είναι η υπερπροσαρμογή ή υπερμοντελοποίηση (overfitting) και η υποπροσαρμογή ή υπομοντελοποίηση (underfitting) [1], [38].

Υπερπροσαρμογή (Overfitting)

Συμβαίνει όταν ένα μοντέλο παραμετροποιείται υπερβολικά, με αποτέλεσμα να υπερπροσαρμοστεί στα δεδομένα εκπαίδευσης και να αποτυγχάνει στο να γενικεύει σε νέα δεδομένα. Αυτό συνήθως συμβαίνει, όταν ένα μοντέλο έχει πολλές παραμέτρους σε σχέση με τον αριθμό των δεδομένων εκπαίδευσης και έτσι μπορεί να μάθει ακόμα και τα θορυβώδη χαρακτηριστικά των δεδομένων εκπαίδευσης [1], [38].

Το αποτέλεσμα είναι ένα μοντέλο που λειτουργεί πολύ καλά στα δεδομένα εκπαίδευσης, αλλά έχει κακή προσέγγιση σε νέα δεδομένα.

Από μαθηματικής απόψεως, όπως έχει αναφερθεί και στα προηγούμενα, η εκτιμώμενη συνάρτηση ανήκει σε μια οικογένεια συναρτήσεων F η οποία επιλέγεται είτε με βάση την εμπειρία είτε με βάση κάποια γνώση για το πρόβλημα προς επίλυση. Ωστόσο, η οικογένεια αυτή μπορεί να είναι τελείως διαφορετική απο την οικογένεια της πραγματικής συνάρτησης. Έτσι, κατά την επιλογή της οικογένειας F, ενδέχεται να προκύψει σφάλμα υπερπροσαρμογής εάν για παράδειγμα τα δεδομένα δημιουργούνται από μια συνάρτηση 2^{ου} βαθμού, ενώ η εκτιμώμενη συνάρτηση είναι 5^{ου} βαθμού [38].

Υποπροσαρμογή (Underfitting)

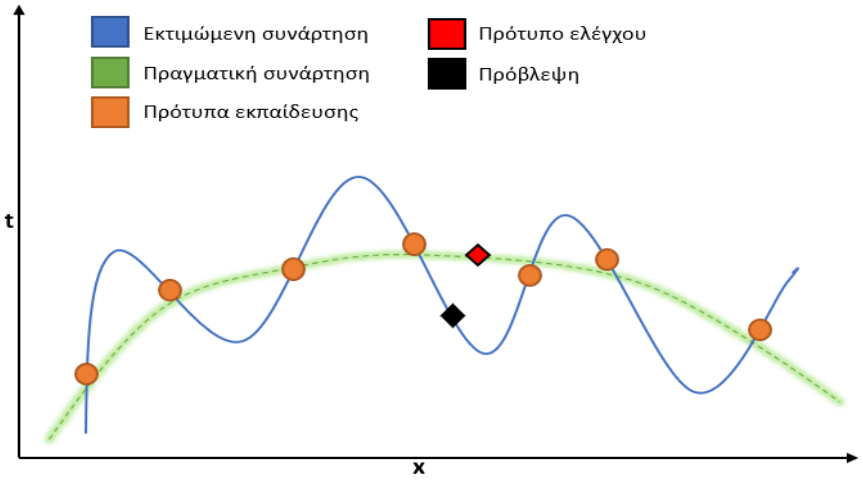
Η υποπροσαρμογή συμβαίνει όταν έχουμε ένα πολύ απλό μοντέλο ή όταν έχουμε πολύ λίγες εποχές κατά τη διαδικασία εκπαίδευσης [1]. Η έννοια των εποχών σχετίζεται με το πλήθος επαναλήψεων που πραγματοποιείται σε μια διαδικασία εκπαίδευσης.

Για παράδειγμα, ίσως χρησιμοποιούμε ένα γραμμικό μοντέλο όπως η απλή γραμμική παλινδρόμηση, ενώ το πρόβλημα να απαιτεί την χρήση ενός μη γραμμικού μοντέλου [1]. Αποτέλεσμα αυτού είναι ότι το μοντέλο δεν θα μπορεί να προβλέψει σωστά τα δεδομένα ελέγχου.

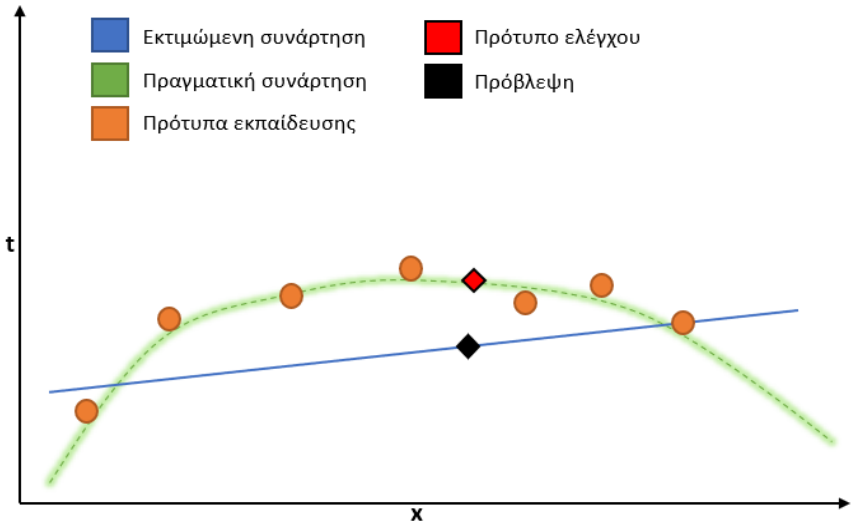
Επομένως, για να αποφύγουμε την υποπροσαρμογή, μπορούμε να χρησιμοποιήσουμε πιο πολύπλοκα μοντέλα ή να αυξήσουμε τον αριθμό των επαναλήψεων κατά τη διαδικασία εκπαίδευσης, ώστε το μοντέλο να μπορεί να προσαρμοστεί καλύτερα στα δεδομένα εκπαίδευσης και να γενικεύσει στα νέα δεδομένα.

Από μαθηματικής απόψεως, θα προκύψει σφάλμα υπομοντελοποίησης, εάν τα δεδομένα δημιουργούνται από μια συνάρτηση 2^{ου} βαθμού, αλλά η εκτιμώμενη συνάρτηση είναι 1^{ου} βαθμού [38].

Στο Σχήμα 4.2 και 4.3, παρουσιάζεται το σφάλμα υπερμοντελοποίησης και υπομοντελοποίησης αντίστοιχα.



Σχήμα 4.2: Σφάλμα Overfitting [38]



Σχήμα 4.3: Σφάλμα Underfitting [38]

Στις εικόνες επίσης παρατηρούμε ότι, για την περίπτωση της υπερμοντελοποίησης η εκτιμώμενη συνάρτηση επιτυγχάνει πολύ καλή προσέγγιση των προτύπων εκπαίδευσης και κακή προσέγγιση στα πρότυπα ελέγχου. Αντίστοιχα, στο σφάλμα υπομοντελοποίησης, το μοντέλο επιτυγχάνει μέτρια προσέγγιση στα πρότυπα εκπαίδευσης και κακή προσέγγιση στα πρότυπα ελέγχου [38].

4.11.2 Διασταυρωμένη επικύρωση k-πτυχών (k-fold cross-validation)

Γενικά, για να ελέγξουμε την ικανότητα γενίκευσης ενός μοντέλου μηχανικής μάθησης, πρέπει να χωρίσουμε το σύνολο δεδομένων (dataset), σε δύο διαφορετικά σύνολα, όπου το ένα θα χρησιμοποιηθεί για εκπαίδευση και το άλλο για έλεγχο. Συνήθως, κρατάμε ένα ποσοστό 80% του συνόλου δεδομένων για δεδομένα εκπαίδευσης και ένα ποσοστό 20% για δεδομένα ελέγχου [1], [7], [38].

Για τον υπολογισμό της επίδοσης ενός μοντέλου, χρησιμοποιούμε δύο ελέγχους επίδοσης, έναν έλεγχο για το σύνολο εκπαίδευσης και έναν για το σύνολο ελέγχου. Σε προβλήματα ταξινόμησης χρησιμοποιείται ως κριτήριο επίδοσης η ακρίβεια (accuracy), οπότε θα έχουμε μια επίδοση για το σύνολο εκπαίδευσης και μια επίδοση για το σύνολο ελέγχου [38].

Εφαρμογή επίδοσης μοντέλου

Έστω ότι για κριτήριο επίδοσης χρησιμοποιούμε τον παρακάτω τύπο που αφορά το μέσο τετραγωνικό σφάλμα (Mean Squared Error - MSE) [38]:

$$J_{MSE} = \sum_{p=1}^{N} \left\| \boldsymbol{t}_p - \boldsymbol{y}_p \right\|^2 \tag{4.10}$$

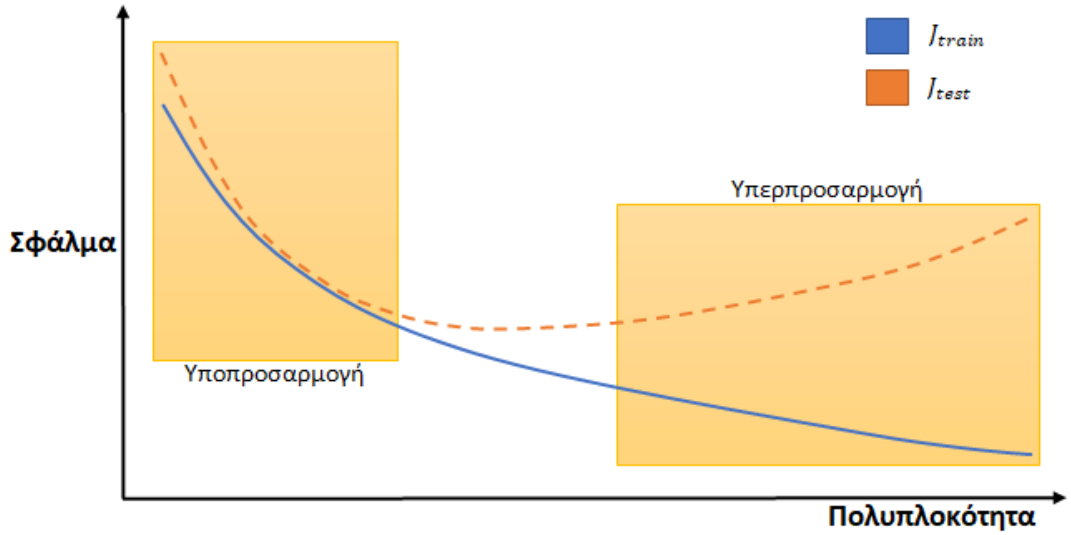
Τότε, η επίδοση για το σύνολο εκπαίδευσης θα είναι:

$$J_{train} = \sum_{p \in Train\_Set} \left\| \boldsymbol{t}_p - \boldsymbol{y}_p \right\|^2$$

Για το σύνολο ελέγχου θα είναι:

$$J_{test} = \sum_{p \in Test\_Set} \left\| \boldsymbol{t}_p - \boldsymbol{y}_p \right\|^2$$

Στο παρακάτω Σχήμα 4.4, φαίνεται πως όσο η πολυπλοκότητα του μοντέλου αυξάνεται, το σφάλμα ελέγχου αυξάνεται και το σφάλμα εκπαίδευσης μειώνεται με συνέπεια την εμφάνιση της υπερπροσαρμογής [38]. Η υποπροσαρμογή εντοπίζεται όταν η πολυπλοκότητα είναι μικρή και τα δύο σφάλματα είναι μεγάλα, με συνέπεια η προσέγγιση του μοντέλου στα πρότυπα ελέγχου να μην είναι καλή όπως και η προσέγγιση στα πρότυπα εκπαίδευσης.



Σχήμα 4.4: Πολυπλοκότητα – Jtrain και Jtest [38]

Η μέθοδος της διασταυρωμένης επικύρωσης

Η διασταυρωμένη επικύρωση (k-fold cross-validation), προσφέρει την δυνατότητα να χρησιμοποιούμε τα δεδομένα μας, τόσο για εκπαίδευση όσο και για έλεγχο. Με αυτό τον τρόπο μπορούμε να διαπιστώσουμε την δυνατότητα γενίκευσης ενός μοντέλου. Η μέθοδος διαχωρίζει το σύνολο δεδομένων σε k πτυχές (folds). Στη συνέχεια γίνεται εκπαίδευση του μοντέλου με k-1 πτυχές και έλεγχος με την πτυχή που έχει απομείνει [1], [38].

Για παράδειγμα, αν έχουμε k=7 πτυχές, θα γίνουν 7 επαναλήψεις εκπαίδευσης και ελέγχου [1], [7]:

- Εκπαίδευση με τις πτυχές 1-6 και έλεγχος με την πτυχή 7
- Εκπαίδευση με τις πτυχές 1-5 και 7, έλεγχος με την πτυχή 6
- Εκπαίδευση με τις πτυχές 1-4 και 6-7, έλεγχος με την 5
- Εκπαίδευση με τις πτυχές 1-3 και 5-7, έλεγχος με την 4
- Εκπαίδευση με τις πτυχές 1-2 και 4-7, έλεγχος με την 3
- Εκπαίδευση με τις πτυχές 1 και 3-7, έλεγχος με την 2
- Εκπαίδευση με τις πτυχές 2-7, έλεγχος με την 1

Η βιβλιοθήκη Scikit-learn διαθέτει την κλάση **KFold** και την συνάρτηση **cross_val_score()**, με τις οποίες μπορούμε να εκτελέσουμε επαναλήψεις εκπαίδευσης και ελέγχου.

4.12 Αλγόριθμοι μηχανικής μάθησης

Με την πολύτιμη βοήθεια της βιβλιοθήκης Scikit learn, για την ανάλυση συναισθήματος θα χρησιμοποιηθούν οι εξής αλγόριθμοι:

- Δέντρο Αποφάσεων (Decision Tree Classifier)
- Αλγόριθμος k-πλησιέστερων γειτόνων (KNearest Neighbors Classifier)
- Ταξινομητής Τυχαίων Δασών (Random Forest Classifier)
- Ταξινομητής Naive Bayes για πολυωνυμικά μοντέλα (MultinomialNB)

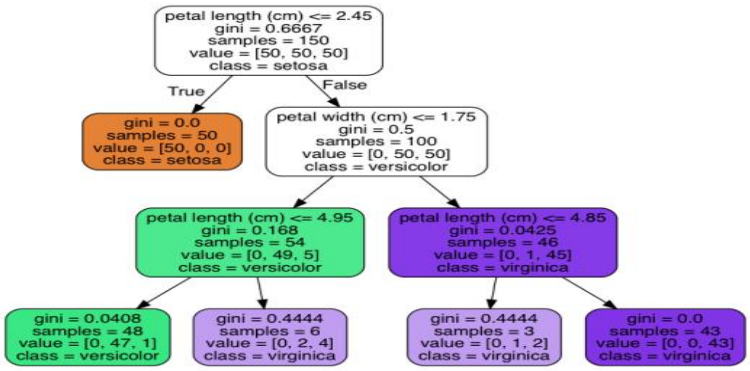
- Ταξινομητής Naive Bayes (GaussianNB)
- Μηχανές Διανυσμάτων Υποστήριξης (SVMs)
- Ταξινομητής Λογιστικής Παλινδρόμησης (Logistic Regression)

4.12.1 Δέντρο Αποφάσεων (Decision Tree Classifier)

Έστω μία βάση δεδομένων (database) που περιέχει διανύσματα το καθένα με κάποια χαρακτηριστικά. Δέντρα αποφάσεων είναι μία τεχνική "διαίρει και βασίλευε" [40] αυτών των χαρακτηριστικών με σκοπό την κατηγοριοποίηση ή παλινδρόμησή τους. Στο Σχήμα 4.5 φαίνεται ένα παράδειγμα ενός δέντρου απόφασης για την κατηγοριοποίηση φυτών με βάση το iris dataset [41].

Ένα δέντρο απόφασης είναι μια δομή δεδομένων για μάθηση με επίβλεψη. Προβλέπει κλάσεις για δείγματα εισόδου εκτελώντας συνεχώς ελέγχους χαρακτηριστικών στα εισαγόμενα δεδομένα, μειώνοντας έτσι το σύνολο των δυνατών κλάσεων ή τιμών στόχου. Αποτελείται από ενδιάμεσους κόμβους με τους τελευταίους να είναι γνωστοί ως φύλλα του δέντρου [42].

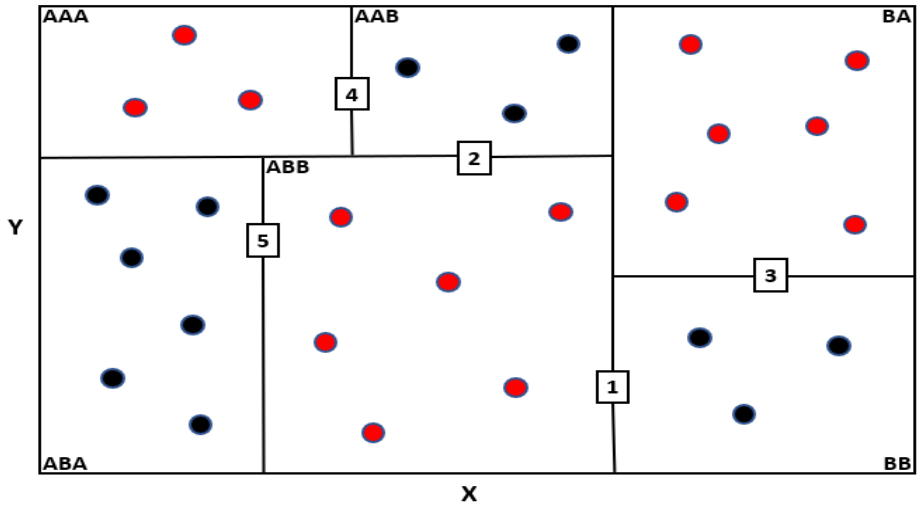
Είναι από τα πιο δημοφιλή εργαλεία στη μηχανική μάθηση και έχει επηρεάσει σημαντικά την περιοχή της ταξινόμησης και παλινδρόμησης. Αναπαριστά ένα μοντέλο αποφάσεων σε μορφή δέντρου, όπου κάθε κόμβος αντιπροσωπεύει μια συνθήκη ή μια ιδιότητα και οι αποφάσεις λαμβάνονται με βάση τα χαρακτηριστικά των δεδομένων εισόδου. Μια περίπτωση ταξινομείται αρχίζοντας από τη ρίζα και ακολουθώντας τα κλαδιά του δέντρου προς κάποιο φύλλο. Σε κάθε κόμβο ελέγχεται η τιμή της περίπτωσης για το χαρακτηριστικό του κόμβου και ακολουθείται το αντίστοιχο κλαδί [31].



Σχήμα 4.5: Παράδειγμα δέντρου απόφασης [41]

Σύμφωνα με τους Κ. Διαμαντάρα και Δ. Μπότση [38], η ιδέα πίσω από τα δέντρα απόφασης είναι να κατατμηθεί ο χώρος των δεδομένων σε ορθογώνιες περιοχές, ώστε σε κάθε περιοχή να υπάρχουν πρότυπα που προέρχονται αποκλειστικά από μία μόνο κλάση. Αυτές οι περιοχές ονομάζονται **ομοιογενείς**. Τα χαρακτηριστικά των προτύπων μπορεί να είναι είτε πραγματικοί αριθμοί είτε κατηγορικές μεταβλητές όπως τα χρώματα κόκκινο, πράσινο, κίτρινο κλπ, όπου δεν ορίζεται συγκεκριμένη σειρά ταξινόμησης των τιμών.

Στο Σχήμα 4.6, ο χώρος των δεδομένων χωρίζεται σε υποπεριοχές όπου η καθεμία περιέχει πρότυπα μιας κλάσης (μαύροι ή κόκκινοι κύκλοι). Έτσι οι νέες περιοχές παρουσιάζουν ομοιογένεια [38].

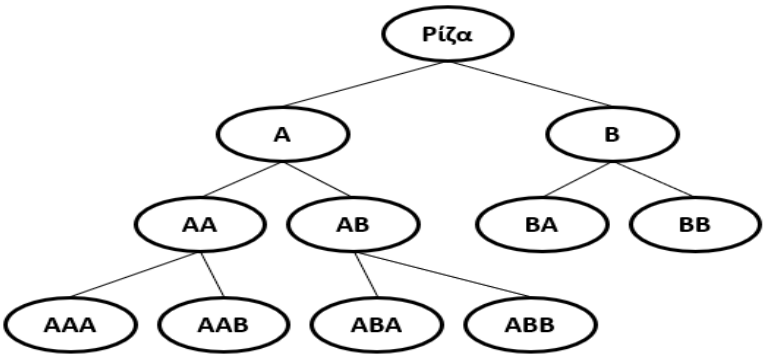


Σχήμα 4.6: Κατάτμηση του δισδιάστατου χώρου ώστε οι νέες περιοχές να είναι ομοιογενείς [38]

Η κατάτμηση του χώρου σε μικρότερα τμήματα γίνεται με αναδρομικό τρόπο, χρησιμοποιώντας μια δομή δέντρου για την οργάνωση αυτών των τμημάτων. Με βάση το Σχήμα 4.6, βρίσκουμε τα εξής τμήματα [38]:

- Ο αρχικός χώρος αποτελεί την ρίζα του δέντρου.
- Η γραμμή 1, χωρίζει τον χώρο στις περιοχές Α αριστερά και Β δεξιά οι οποίες αποτελούν τα παιδιά της ρίζας του δέντρου.
- Η γραμμή 2, χωρίζει το τμήμα Α στις περιοχές ΑΑ και ΒΒ, πάνω και κάτω αντίστοιχα. Οι περιοχές αυτές είναι τα παιδιά του κόμβου Α.
- Η γραμμή 3, χωρίζει το τμήμα Β στις περιοχές ΒΑ και ΒΒ, πάνω και κάτω αντίστοιχα. Είναι τα παιδιά του κόμβου Β.
- Η γραμμή 4, χωρίζει το τμήμα ΑΑ στις περιοχές ΑΑΑ και ΑΑΒ, αριστερά και δεξιά αντίστοιχα. Είναι τα παιδιά του κόμβου ΑΑ.
- Η γραμμή 5, χωρίζει το τμήμα ΑΒ στις περιοχές ΑΒΑ και ΑΒΒ, αριστερά και δεξιά αντίστοιχα. Είναι τα παιδιά του κόμβου ΑΒ.

Έτσι, το δέντρο απόφασης που δημιουργείται είναι το εξής (Σχήμα 4.7):



Σχήμα 4.7: Δέντρο απόφασης της προηγούμενης κατάτμησης [38]

Για να επιτευχθεί μια ομοιογενή κατάτμηση του χώρου δεδομένων, πρέπει να λάβουμε μερικές αποφάσεις. Προφανώς αν μια περιοχή είναι ήδη ομοιογενής, τότε δεν χρειάζεται να τη διαιρέσουμε περαιτέρω. Αντίθετα, αν μια περιοχή παρουσιάζει ανομοιογένεια, τότε επιλέγουμε μια διάσταση **j** του χώρου των χαρακτηριστικών και χωρίζουμε την περιοχή σε δύο υποπεριοχές $\mathbf{x_j<s}$ και $\mathbf{x_j>s}$, με βάση ένα κατώφλι **s** [38]. Επιλέγουμε το χαρακτηριστικό $\mathbf{x_j}$ και το κατώφλι **s** τα οποία δίνουν την μικρότερη ανομοιογένεια για να διασφαλίσουμε ότι οι περιοχές που δημιουργούμε είναι όσο το δυνατόν πιο ομοιογενείς. Επομένως, απαιτείται ο ορισμός ενός κριτηρίου ανομοιογένειας. Τα πιο συνηθισμένα κριτήρια μέτρησης της ανομοιογένειας μιας περιοχής $\mathcal{R}$ η οποία περιέχει N διανύσματα - πρότυπα από K κλάσεις είναι τα παρακάτω [38]:

- Εντροπία:

$$E_{Ent}(\mathcal{R}) = -\sum_{k=1}^{K} P(k)\ log_2 P(k) \tag{4.11}$$

Όπου $P(k)$ είναι το ποσοστό των δειγμάτων της περιοχής $\mathcal{R}$ που ανήκουν στην κλάση k.

- Κόστος Gini:

$$E_{Gini}(\mathcal{R}) = \sum_{k\neq m} P(k)\ P(m) \tag{4.12}$$

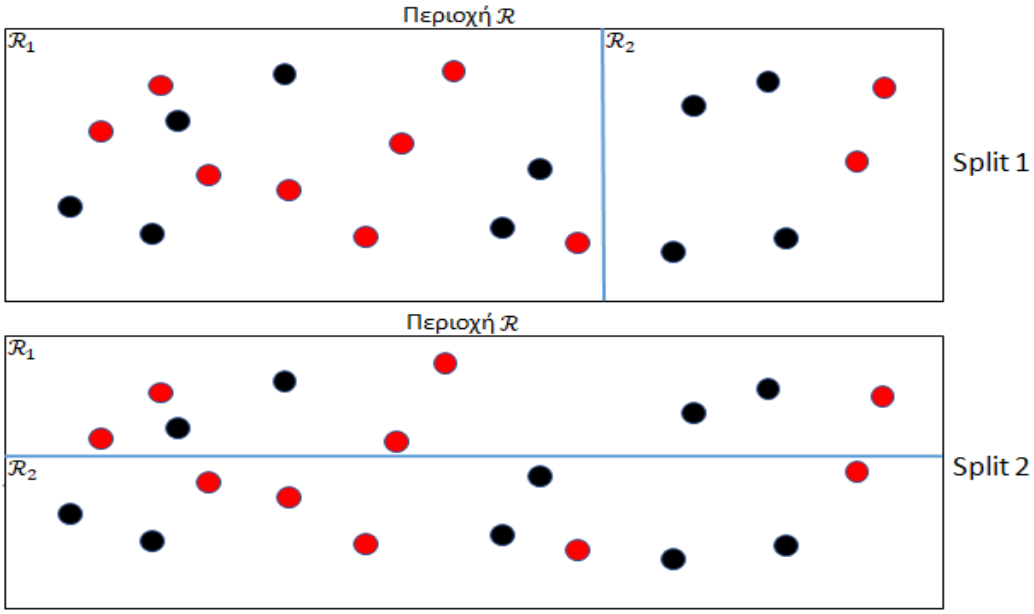
Όπου k και m διαφορετικές κλάσεις.

Τέλος, χωρίζουμε την περιοχή $\mathcal{R}$ σε δύο υποπεριοχές $\mathcal{R}_1$, $\mathcal{R}_2$ με N_1 και N_2 δείγματα αντίστοιχα και υπολογίζουμε την ανομοιογένεια σύμφωνα με τον τύπο:

$$E_{split} = \frac{N_1}{N}\ E(\mathcal{R}_1) + \frac{N_2}{N}\ E(\mathcal{R}_2) \tag{4.13}$$

Εφαρμογή υπολογισμού ανομοιογένειας

Βασιζόμενοι στο Σχήμα 4.8 [38], ανεξαρτήτως κριτηρίου ανομοιογένειας (Εντροπία ή κόστος Gini), θα υπολογίσουμε την μικρότερη ανομοιογένεια.



Σχήμα 4.8: Δύο κατατμήσεις (splits) της περιοχής $\mathcal{R}$ [38]

Και στις δύο κατατμήσεις (Split 1 και Split 2), υπάρχουν N = 20 δείγματα. Κλάση_1 = μαύροι κύκλοι, κλάση_2 = κόκκινοι κύκλοι.

Όσον αφορά το **Split 1**, έχουμε δύο υποπεριοχές $\mathcal{R}_1$ και $\mathcal{R}_2$.

Περιοχή $\mathcal{R}_1$:

Υπάρχουν $\mathrm{N}_{11} = 6$ δείγματα (μαύροι κύκλοι). Άρα, ποσοστό $P(1) = \frac{6}{14}$

Υπάρχουν $\mathrm{N}_{12} = 8$ δείγματα (κόκκινοι κύκλοι). Άρα, ποσοστό $P(2) = \frac{8}{14}$

Υπολογισμός ανομοιογένειας

Για Εντροπία:

$$\mathrm{E}_{Ent}(\mathcal{R}_1) = -\left[\frac{6}{14} \cdot \log_2\left(\frac{6}{14}\right) + \frac{8}{14} \cdot \log_2\left(\frac{8}{14}\right)\right] = 0.98522$$

Για κόστος Gini:

$$\mathrm{E}_{Gini}(\mathcal{R}_1) = \frac{6}{14} \cdot \frac{8}{14} = 0.24489$$

Περιοχή $\mathcal{R}_2$:

Υπάρχουν $\mathrm{N}_{21} = 4$ δείγματα (μαύροι κύκλοι). Άρα, ποσοστό $P(1) = \frac{4}{6}$

Υπάρχουν $\mathrm{N}_{22} = 2$ δείγματα (κόκκινοι κύκλοι). Άρα, ποσοστό $P(2) = \frac{2}{6}$

Υπολογισμός ανομοιογένειας

Για Εντροπία:

$$\mathrm{E}_{Ent}(\mathcal{R}_2) = -\left[\frac{4}{6}\cdot \log_2\left(\frac{4}{6}\right) + \frac{2}{6}\cdot \log_2\left(\frac{2}{6}\right)\right] = 0.91829$$

Για κόστος Gini:

$$\mathrm{E}_{Gini}(\mathcal{R}_2) = \frac{4}{6}\cdot\frac{2}{6} = \frac{8}{36} = 0.22222$$

Επομένως, επειδή το συνολικό πλήθος των δειγμάτων είναι N = 20, η περιοχή $\mathcal{R}_1$ έχει $\mathrm{N}_1 = 14$ δείγματα και η περιοχή $\mathcal{R}_2$ έχει $\mathrm{N}_2 = 6$ δείγματα, η συνολική ανομοιογένεια μετά το Split θα είναι:

$$\mathrm{E}_{split} = \frac{14}{20}\,\mathrm{E}(\mathcal{R}_1) + \frac{6}{20}\,\mathrm{E}(\mathcal{R}_2)$$

Τελικά, για το κριτήριο της εντροπίας είναι:

$$\mathrm{E}_{split}^{Ent} = \frac{14}{20}\cdot \mathrm{E}_{Ent}(\mathcal{R}_1) + \frac{6}{20}\cdot \mathrm{E}_{Ent}(\mathcal{R}_2) = 0.7\cdot 0.98522 + 0.3\cdot 0.91829 = 0.96514$$

Για το κόστος Gini θα είναι:

$$\mathrm{E}_{split}^{Gini} = 0.7\cdot \mathrm{E}_{Gini}(\mathcal{R}_1) + 0.3\cdot \mathrm{E}_{Gini}(\mathcal{R}_2) = 0.7\cdot 0.24489 + 0.3\cdot 0.22222 = 0.23808$$

Όσον αφορά το **Split 2**, έχουμε δύο υποπεριοχές $\mathcal{R}_1$και $\mathcal{R}_2$.

Περιοχή $\mathcal{R}_1$:

Υπάρχουν $\mathrm{N}_{11} = 4$ δείγματα (μαύροι κύκλοι). Άρα, ποσοστό $P(1) = \frac{4}{9}$

Υπάρχουν $\mathrm{N}_{12} = 5$ δείγματα (κόκκινοι κύκλοι). Άρα, ποσοστό $P(2) = \frac{5}{9}$

Υπολογισμός ανομοιογένειας

Για Εντροπία:

$$\mathrm{E}_{Ent}(\mathcal{R}_1) = -\left[\frac{4}{9}\cdot \log_2\left(\frac{4}{9}\right) + \frac{5}{9}\cdot \log_2\left(\frac{5}{9}\right)\right] = 0.99107$$

Για κόστος Gini:

$$\mathrm{E}_{Gini}(\mathcal{R}_1) = \frac{4}{9}\cdot\frac{5}{9} = 0.24691$$

Περιοχή $\mathcal{R}_2$:

Υπάρχουν $\mathrm{N}_{21} = 6$ δείγματα (μαύροι κύκλοι). Άρα, ποσοστό $P(1) = \frac{6}{11}$

Υπάρχουν $\mathrm{N}_{22} = 5$ δείγματα (κόκκινοι κύκλοι). Άρα, ποσοστό $P(2) = \frac{5}{11}$

Υπολογισμός ανομοιογένειας

Για Εντροπία:

$$\mathrm{E}_{Ent}(\mathcal{R}_2) = -\left[\frac{6}{11} \cdot \log_2\left(\frac{6}{11}\right) + \frac{5}{11} \cdot \log_2\left(\frac{5}{11}\right)\right] = 0.99403$$

Για κόστος Gini:

$$\mathrm{E}_{Gini}(\mathcal{R}_2) = \frac{6}{11} \cdot \frac{5}{11} = \frac{30}{121} = 0.24793$$

Επειδή το συνολικό πλήθος των δειγμάτων είναι N = 20, η περιοχή $\mathcal{R}_1$ έχει $\mathrm{N}_1 = 9$ δείγματα και η περιοχή $\mathcal{R}_2$ έχει $\mathrm{N}_2 = 11$ δείγματα, η συνολική ανομοιογένεια μετά το Split θα είναι:

$$\mathrm{E}_{split} = \frac{9}{20}\,\mathrm{E}(\mathcal{R}_1) + \frac{11}{20}\,\mathrm{E}(\mathcal{R}_2)$$

Τελικά, για το κριτήριο της εντροπίας θα έχουμε:

$$\mathrm{E}_{split}^{Ent} = 0.99269$$

Και για κόστος Gini θα έχουμε:

$$\mathrm{E}_{split}^{Gini} = 0.24747$$

Στον Πίνακας 4.3, παρουσιάζονται τα τελικά αποτελέσματα, από τα οποία προκύπτει πως προτιμάται το Split 1 επειδή έχει μικρότερη ανομοιογένεια, για οποιοδήποτε κριτήριο.

Πίνακας 4.3: Το Split 1 έχει την μικρότερη ανομοιογένεια [38]

Κριτήριο Ανομοιογένειας	Split 1	Split 2
Εντροπία	0.96514	0.99269
Κόστος Gini	0.23808	0.24747

Πλεονεκτήματα και μειονεκτήματα δέντρων απόφασης

Τα κύρια **πλεονεκτήματα** των δέντρων απόφασης είναι τα εξής [38]:

- Τα δέντρα απόφασης μπορούν να αναπαραστήσουν απλά και κατανοητά τη λογική πίσω από τη λήψη αποφάσεων. Είναι ευκολότερο να κατανοήσουμε τον τρόπο που λαμβάνεται μια απόφαση από ένα δέντρο απόφασης σε σχέση με άλλους αλγορίθμους μηχανικής μάθησης.
- Δεν είναι απαραίτητη συνήθως κάποια μέθοδος προεπεξεργασίας των δεδομένων.
- Τα δέντρα απόφασης μπορούν να παράγουν ακριβείς προβλέψεις σε σχετικά σύνθετα προβλήματα. Οι προβλέψεις αυτές είναι συχνά ακριβείς σε σημαντικό βαθμό.
- Μπορούν να αντιμετωπίσουν προβλήματα με πολλαπλά κριτήρια που πρέπει να εξεταστούν κατά τη λήψη αποφάσεων. Έτσι, αναζητά τα πιο κρίσιμα χαρακτηριστικά των δεδομένων που οδηγούν στις σωστές αποφάσεις.
- Σε γενικές γραμμές, μπορούμε να πούμε ότι όσο αυξάνεται το πλήθος των δεδομένων, τόσο πιθανότερο είναι να αυξηθεί και το βάθος του δέντρου απόφασης, καθώς θα χρειαστεί περισσότερα επίπεδα για να αποφασίσει σωστά για κάθε δείγμα.
- Μπορούμε με διάφορα εργαλεία όπως το WEKA, να τα οπτικοποιήσουμε.

Τα βασικότερα **μειονεκτήματα** των δέντρων απόφασης είναι τα εξής [38]:

- Παρουσιάζουν μεγάλη ευαισθησία, καθώς μικρές αλλαγές στα δεδομένα εκπαίδευσης προκαλούν μεγάλες διαφορές στο δέντρο.
- Μπορεί να δημιουργηθούν δέντρα με πολύ μεγάλο βάθος που είναι υπερ-προσαρμοσμένα στα δεδομένα εκπαίδευσης και δεν γενικεύουν καλά σε νέα δεδομένα.
- Τα δέντρα απόφασης δεν είναι καλά στην αντιμετώπιση δεδομένων με ανισορροπία κλάσεων, δηλαδή όταν ο αριθμός των δειγμάτων - προτύπων σε κάθε κλάση διαφέρει σημαντικά. Σε τέτοιες περιπτώσεις συνήθως δημιουργούνται δέντρα τα οποία μεροληπτούν στην κλάση με τα περισσότερα πρότυπα.
- Δυσκολία στην αντιμετώπιση δεδομένων με ελλιπείς τιμές (missing values).
- Προβλήματα όπως το XOR, είναι δύσκολα για τα δέντρα απόφασης.

Τέλος, να σημειωθεί πως η ανάλυση αποφάσεων με την βοήθεια δέντρων απόφασης, μπορεί να χρησιμοποιηθεί για την εξαγωγή στρατηγικής σε διάφορους τομείς, όπως για παράδειγμα η διάγνωση ιατρικών περιστατικών. Σε γενικές γραμμές, το δέντρο αποφάσεων αποτελεί ένα ισχυρό εργαλείο για την αναπαράσταση και τη λήψη αποφάσεων, καθώς προσφέρει μια οπτική αναπαράσταση των δεδομένων.

4.12.2 Αλγόριθμος k-πλησιέστερων γειτόνων (K-Nearest Neighbors Classifier)

Ο αλγόριθμος k-πλησιέστερων γειτόνων (KNN) [43], βασίζεται στην ιδέα πως ένα διάνυσμα **x** "κοντινό" σε ένα άλλο πάνω σε έναν μετρικό χώρο θα μοιράζεται τις ιδιότητές του. Βασική υπόθεση είναι ο μετρικός χώρος. Συνήθως, χρησιμοποιείται η μετρική του Minkowski.

$$\|x' - x_j\|^p = \left(\sum_{i=1}^{q} |(x_i)' - (x_i)_j|^p\right)^{\frac{1}{p}} \tag{4.14}$$

Όπου p ένας φυσικός αριθμός (για p=2 προκύπτει η ευκλείδεια μετρική) και q η διάσταση των διανυσμάτων.

Ο αλγόριθμος για να κατηγοριοποιήσει ένα διάνυσμα, εξετάζει με βάση την μετρική, την ομοιότητά του με τους k κοντινότερους γείτονές του, και του αναθέτει την εκάστοτε ετικέτα [1], [38].

Έτσι, αν έχουμε δύο κλάσεις 0, 1 και θέλουμε να προβλέψουμε την κατηγορία (κλάση) ενός δείγματος ελέγχου, εξετάζονται τα k δείγματα εκπαίδευσης που είναι πιο κοντά ως προς την απόσταση, στο δείγμα ελέγχου, και [1], [38]:

- Αν η πλειοψηφία των γειτόνων ανήκουν στην κλάση 0, τότε το διάνυσμα – πρότυπο ταξινομείται στην κλάση 0.
- Αν η πλειοψηφία των γειτόνων ανήκουν στην κλάση 1, τότε το διάνυσμα – πρότυπο ταξινομείται στην κλάση 1.

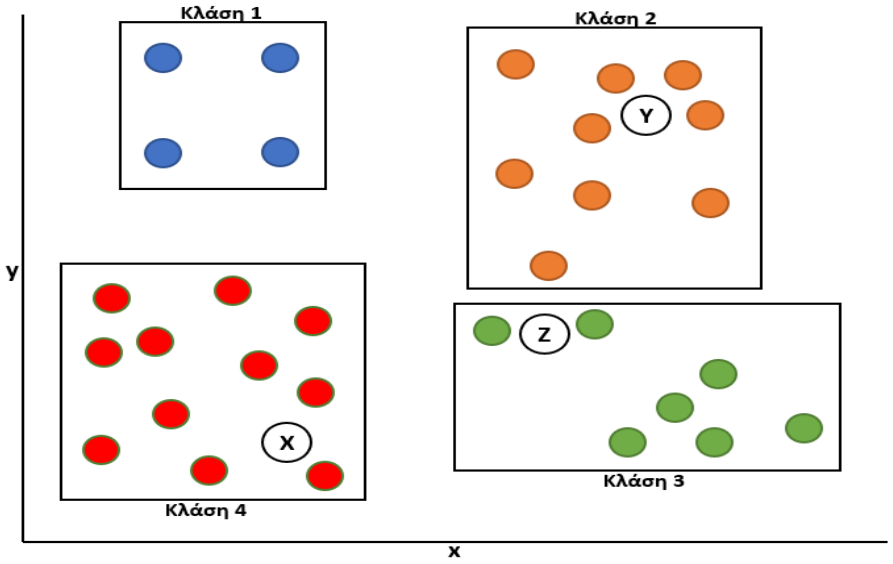
Για να αποφεύγεται η πιθανότητα να υπάρξει ισοψηφία στη διαδικασία της ταξινόμησης, συνήθως επιλέγεται ένας περιττός αριθμός για τον αριθμό των γειτόνων k [38]. Αυτή η μέθοδος θεωρείται ως μη παραμετρική (non - parametric), καθώς δεν χρησιμοποιεί μάθηση και δεν απαιτεί αυτορύθμιση κάποιας παραμέτρου. Αντίθετα, υπάρχει μόνο η υπερπαράμετρος k, η οποία καθορίζεται εκ των προτέρων από τον χρήστη [38].

Στην πράξη, η μέθοδος k-NN μπορεί να παράγει αξιόλογα αποτελέσματα, αλλά συχνά δεν επιτυγχάνει καλύτερη απόδοση σε σχέση με άλλες μεθόδους όπως για παράδειγμα τα νευρωνικά δίκτυα (neural networks). Επιπλέον, δεν είναι κατάλληλη για απαιτητικά προβλήματα με μεγάλα σύνολα δεδομένων, καθώς ο υπολογιστικός φόρτος αυξάνεται δραματικά με το μέγεθος του συνόλου δεδομένων. Το βασικό μειονέκτημά της είναι το υπολογιστικό κόστος, καθώς για κάθε νέο πρότυπο που πρέπει να ταξινομηθεί απαιτείται ο υπολογισμός της απόστασής του από όλα τα πρότυπα του συνόλου δεδομένων, προκειμένου να βρεθούν τα k πλησιέστερα πρότυπα, δηλαδή οι γείτονες. Έτσι, αν έχουμε για παράδειγμα ένα dataset με P = 2000000 διανύσματα – πρότυπα, τότε για την ταξινόμηση ενός προτύπου **x** απαιτείται ο υπολογισμός της Ευκλείδειας απόστασης $\|x - x_i\|^2$ για 2000000 πρότυπα [38].

Εφαρμογή k-πλησιέστερων γειτόνων

Έστω το Σχήμα 4.9, το οποίο δείχνει 4 κλάσεις (κύκλοι διαφορετικού χρώματος). Οι κύκλοι X, Y, Z, είναι πρότυπα - δείγματα τα οποία θέλουμε να ταξινομήσουμε. Έστω επίσης ότι θα κάνουμε τις προβλέψεις χρησιμοποιώντας τους 3 πλησιέστερους γείτονες κάθε δείγματος, δηλαδή k=3 [1]. Παρατηρούμε ότι [1]:

- Οι 3 πλησιέστεροι γείτονες του δείγματος Y, είναι όλοι κύκλοι της κλάσης 2. Άρα, το μοντέλο θα προβλέψει ότι το Y ανήκει στην κλάση 2.
- Οι 3 πλησιέστεροι γείτονες του δείγματος X, είναι όλοι κύκλοι της κλάσης 4. Άρα, το μοντέλο θα προβλέψει ότι το δείγμα X, ανήκει στην κλάση 4.
- Το δείγμα Z, βρίσκεται στα όρια της κλάσης 2 και 3. Από τους 3 πλησιέστερους γείτονες, ένας ανήκει στην κλάση 2 (πορτοκαλί κύκλος), και οι άλλοι δύο ανήκουν στην κλάση 3 (πράσινοι κύκλοι). Επειδή, στον αλγόριθμο k-πλησιέστερων γειτόνων κερδίζει πάντα η κλάση που έχει την πλειοψηφία, το μοντέλο τελικά θα προβλέψει ότι το δείγμα Z ανήκει στην κλάση 3. Η επιλογή ενός περιττού αριθμού για την υπερπαράμετρο k, εξασφαλίζει ότι δεν θα υπάρξει ποτέ ισοψηφία.



Σχήμα 4.9: Ταξινόμηση των δειγμάτων X, Y, Z [1]

4.12.3 Ταξινομητής Τυχαίων Δασών (Random Forest Classifier)

Τα τυχαία δάση [44] είναι μια τεχνική μηχανικής μάθησης που χρησιμοποιείται για την κατηγοριοποίηση παρατηρήσεων σε δύο ή περισσότερες κατηγορίες. Αποτελούν μια επέκταση των δέντρων απόφασης, όπου πολλά δέντρα απόφασης συνδυάζονται για να δώσουν μια πιο ακριβή κατηγοριοποίηση. Ένας αλγόριθμος τυχαίου δάσους δημιουργεί έναν προκαθορισμένο αριθμό m δέντρων αποφάσεων εκπαιδεύοντάς τα σε διαφορετικά κομμάτια-χαρακτηριστικά της βάσης δεδομένων. Αυτά τα κομμάτια προκύπτουν από τη μεγιστοποίηση του κριτηρίου CART. Πιο συγκεκριμένα, το CART (Classification and Regression Trees) είναι ένα κριτήριο απόφασης που χρησιμοποιείται στα δέντρα απόφασης των Random Forests για την επιλογή των καλύτερων χαρακτηριστικών (features) και την κατηγοριοποίηση των παρατηρήσεων. Χρησιμοποιεί την Gini impurity ή την Entropy (cross-entropy) ως μέτρο ποιότητας της κατηγοριοποίησης στα επιμέρους δέντρα απόφασης (decision trees). Κατά τη διάρκεια της εκπαίδευσης, τα δέντρα απόφασης δημιουργούνται τυχαία, χρησιμοποιώντας διαφορετικά υποσύνολα του συνόλου δεδομένων και των χαρακτηριστικών. Αυτό βοηθά στη μείωση του φαινομένου της υπερπροσαρμογής (overfitting) και στη βελτίωση της γενίκευσης του μοντέλου. Τέλος, για να ταξινομηθεί μια νέα παρατήρηση στο Random Forest, τα αποτελέσματα των διαφορετικών δέντρων απόφασης συνδυάζονται μέσω μιας διαδικασίας ψηφοφορίας, για να προκύψει η τελική απόφαση. Η τεχνική του Random Forest χρησιμοποιείται σε πολλές εφαρμογές της μηχανικής μάθησης, όπως η αναγνώριση προτύπων σε εικόνες και ήχο, η κατηγοριοποίηση εισερχομένων email σε κατηγορίες spam και μη spam, η αναγνώριση ανωμαλιών σε δεδομένα κλπ.

4.12.4 Naïve Bayes

Το μοντέλο Naive Bayes είναι ένα από τα πιο απλά και αποτελεσματικά μοντέλα ταξινόμησης κειμένου. Πρόκειται για στατιστικό μοντέλο που βασίζεται στο διάσημο θεώρημα δεσμευμένης πιθανότητας του Bayes [45] για A και B τυχαία γεγονότα σε ένα δειγματοχώρο:

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)} \tag{4.15}$$

Στο πλαίσιο της επεξεργασίας φυσικής γλώσσας ο ταξινομητής Naive Bayes [45] υποθέτει ότι κάθε χαρακτηριστικό (δηλαδή λέξη) είναι ανεξάρτητο από όλα τα άλλα χαρακτηριστικά, δεδομένης της ετικέτας της κλάσης. Αυτή η υπόθεση απλοποιεί τον υπολογισμό της πιθανότητας ενός κειμένου να ανήκει σε μια συγκεκριμένη κλάση, καθώς απαιτεί μόνο τον υπολογισμό της πιθανότητας κάθε χαρακτηριστικού δεδομένης της ετικέτας της κλάσης.

Η διαδικασία γίνεται ως εξής: Έστω x = (x_1, x_2,…,x_n) ένα διάνυσμα του χώρου των κειμένων και K το σύνολο των πιθανών ετικετών. Ο τύπος που δίνει την πιθανότητα ένα διάνυσμα x να ανήκει στην κλάση K είναι:

$$P(K|x_1, \dots, x_n) = \frac{1}{Z}\, P(K) \prod\nolimits_{i=1}^{n} P(x_i|K) \tag{4.16}$$

Όπου, $Z = P(x) = \sum_k P(K)P(x|K)$

Σε αυτή την εργασία έγινε χρήση δύο παραλλαγών του Naïve Bayes, του Gaussian Naïve Bayes και του Multinomial Naïve Bayes. Η βασική διαφορά τους έγκειται στον υπολογισμό των δεσμευμένων πιθανοτήτων $P(x_i|K)$, i=1,2,...,n

Ο **Gaussian Naïve Bayes** υποθέτει κανονική κατανομή των δεδομένων και άρα:

$$P(x = v|K) = \frac{1}{\sqrt{2\pi\sigma_\kappa^2}} e^{-\frac{(v-\mu_k)^2}{2\sigma_k^2}} \tag{4.17}$$

Όπου μ_κ, σ_κ η μέση τιμή και η τυπική απόκλιση αντίστοιχα της κλάσης K.

Στον **Multinomial Naïve Bayes**, η κατανομή πιθανοτήτων κάθε λέξης στο έγγραφο μοντελοποιείται ως μια πολυωνυμική κατανομή, και η πιθανότητα ενός διανύσματος να ανήκει σε μια συγκεκριμένη κατηγορία υπολογίζεται χρησιμοποιώντας πάλι το θεώρημα Bayes. Κάθε λέξη στο διάνυσμα θεωρείται ως ξεχωριστό χαρακτηριστικό και η συχνότητα κάθε λέξης χρησιμοποιείται ως η τιμή του χαρακτηριστικού. Σε αυτή την περίπτωση,

$$P(x|K) = \frac{(\sum_{i=1}^{n} x_i)!}{\prod_{i=1}^{n} x_i\,!} \prod\nolimits_{i=1}^{n} p_{ki}^{x_i} \tag{4.18}$$

Όπου $p_{ki} \coloneqq P(x_i|\, K)$

4.12.5 Support Vector Machines (SVMs)

Το Support Vector Machine (SVM) είναι ίσως ένας από τους πιο πολυσυζητημένους αλγόριθμους μηχανικής μάθησης που χρησιμοποιείται για προβλήματα ταξινόμησης και παλινδρόμησης [46]. Ήταν εξαιρετικά δημοφιλής την εποχή που αναπτύχθηκε, στη δεκαετία του 1990 και συνεχίζει να είναι η πιο διαδεδομένη μέθοδος υψηλής απόδοσης με μικρή ρύθμιση [46].

Τα SVMs [46] λειτουργούν βρίσκοντας ένα υπερεπίπεδο (γραμμή) που διαχωρίζει τα δεδομένα σε διαφορετικές κλάσεις. Στην περίπτωση του NLP, το υπερεπίπεδο είναι ένα όριο απόφασης που διαχωρίζει τα έγγραφα σε διαφορετικές κατηγορίες.

Η απόσταση μεταξύ της γραμμής και των πλησιέστερων σημείων δεδομένων αναφέρεται ως περιθώριο. Η καλύτερη ή βέλτιστη γραμμή που μπορεί να χωρίσει τις δύο κατηγορίες είναι η γραμμή με το μεγαλύτερο περιθώριο. Αυτό ονομάζεται υπερεπίπεδο Maximal-Margin [46]. Το περιθώριο υπολογίζεται ως η κάθετη απόσταση από τη γραμμή μόνο στα πλησιέστερα σημεία. Μόνο αυτά τα σημεία είναι σχετικά με τον καθορισμό της γραμμής και την κατασκευή του ταξινομητή. Αυτά τα σημεία ονομάζονται **διανύσματα υποστήριξης** και ουσιαστικά είναι αυτά που υποστηρίζουν ή ορίζουν το υπερεπίπεδο [46].

Στην πράξη, τα πραγματικά δεδομένα είναι σύνθετα και δεν μπορούν να διαχωριστούν τέλεια με ένα υπερεπίπεδο. Ο περιορισμός της μεγιστοποίησης του περιθωρίου της γραμμής που χωρίζει τις κλάσεις πρέπει να χαλαρώσει. Αυτό ονομάζεται συχνά ταξινομητής soft margin. Αυτή η αλλαγή επιτρέπει σε ορισμένα σημεία στα δεδομένα εκπαίδευσης να παραβιάζουν τη διαχωριστική γραμμή [46].

Εισάγεται μια παράμετρος συντονισμού που ονομάζεται C και ορίζει το όριο παραβίασης του επιτρεπόμενου περιθωρίου. Το C=0 δεν αποτελεί παραβίαση και επιστρέφουμε στον άκαμπτο ταξινομητή Maximal-Margin που περιγράφεται παραπάνω. Όσο μεγαλύτερη είναι η τιμή του C τόσο λιγότερες παραβιάσεις του υπερεπιπέδου επιτρέπονται [46].

Βασικά χαρακτηριστικά ενός SVM αλγόριθμου εκτός από το soft margin που αναφέρθηκε παραπάνω, είναι και η **συνάρτηση πυρήνα**. Η συνάρτηση πυρήνα ουσιαστικά ανάγει το πρόβλημα σε μεγαλύτερη διάσταση ώστε να είναι ευκολότερη η κατηγοριοποίηση των δεδομένων. Στα προβλήματα της εργασίας μας χρησιμοποιήθηκε ο προεπιλεγμένος πυρήνας των SVMs, ο πυρήνας RBF (Radial Basis Function).

Η Radial Basis Function (RBF) είναι μια συνάρτηση πυρήνα που χρησιμοποιείται συχνά στα Support Vector Machines (SVMs) για την ανίχνευση μη γραμμικών σχέσεων μεταξύ των δεδομένων. Η συνάρτηση πυρήνα RBF υπολογίζεται ως εξής [46]:

$$K(x, x') = exp(-\gamma \cdot \|x - x'\|^2) \tag{4.19}$$

όπου $\boldsymbol{x}$ και $\boldsymbol{x'}$ είναι δύο διανύσματα χαρακτηριστικών, το σύμβολο $\|.\|$ αναπαριστά το μέτρο της Ευκλείδειας απόστασης μεταξύ τους και το γ (gamma) είναι μια θετική καθορισμένη από τον χρήστη παράμετρος που καθορίζει την πολυπλοκότητα του μοντέλου. Πιο συγκεκριμένα, η παράμετρος γ (gamma) είναι μια σταθερά που ελέγχει το βαθμό πτώσης της συνάρτησης πυρήνα στην εκθετική μείωση της απόστασης $\|x - x'\|$ μεταξύ των δύο διανυσμάτων $\boldsymbol{x}$ και $\boldsymbol{x'}$. Όταν το γ είναι μεγάλο, η

πτώση της τιμής του πυρήνα γίνεται πιο απότομη, επιτρέποντας στον αλγόριθμο SVM να δημιουργήσει έναν σκληρό (hard) ή πιο αυστηρό διαχωριστικό υπερεπίπεδο μεταξύ των κλάσεων.

Αντίθετα, όταν το γ είναι μικρό, η πτώση της τιμής του πυρήνα γίνεται πιο ασθενής, επιτρέποντας στον αλγόριθμο SVM να δημιουργήσει ένα πιο μαλακό (soft) ή πιο επιεικές διαχωριστικό υπερεπίπεδο μεταξύ των κλάσεων. Συνεπώς, η τιμή της παραμέτρου γ πρέπει να επιλέγεται προσεκτικά και να προσαρμόζεται ανάλογα με τα χαρακτηριστικά των δεδομένων και του προβλήματος που επιλύεται.

Τέλος, η τιμή της RBF συνάρτησης πυρήνα είναι μια μέτρηση της ομοιότητας μεταξύ των δύο διανυσμάτων χαρακτηριστικών, όπου ένα κοντινό ζεύγος διανυσμάτων έχει μια υψηλή τιμή RBF, ενώ ένα μακρινό ζεύγος έχει μια χαμηλή τιμή.

4.12.6 Λογιστική Παλινδρόμηση (Logistic Regression)

Η λογιστική παλινδρόμηση (Logistic Regression) είναι μια τεχνική μηχανικής μάθησης που χρησιμοποιείται για προβλέψεις πιθανοτήτων κυρίως για δυαδικές κλάσεις (binary classification) με βάση τις εισόδους (features) ενός συνόλου δεδομένων. Η λογιστική παλινδρόμηση είναι ένας αλγόριθμος εποπτευόμενης μάθησης, καθώς απαιτεί ένα σύνολο δεδομένων εκπαίδευσης που περιλαμβάνει δείγματα με γνωστή κλάση - ετικέτα.

Ανάλογα με τον τρόπο που γίνεται η ταξινόμηση, διακρίνονται τρεις βασικοί τύποι λογιστικής παλινδρόμησης:

- **Δυαδική λογιστική παλινδρόμηση** (Binary Logistic Regression). Χρησιμοποιείται για προβλήματα κατηγοριοποίησης με δύο κλάσεις, δηλαδή ένα πρόβλημα δυαδικής ταξινόμησης.
- **Πολυκατηγορική λογιστική παλινδρόμηση** (Multinomial Logistic Regression). Χρησιμοποιείται για προβλήματα κατηγοριοποίησης με τρεις ή περισσότερες κλάσεις.
- **Λογιστική παλινδρόμηση One-vs-Rest** (One-vs-Rest Logistic Regression). Χρησιμοποιείται για προβλήματα πολυκατηγορικής κατηγοριοποίησης, όπου δημιουργούνται k μοντέλα λογιστικής παλινδρόμησης, ένα για κάθε κατηγορία, και για κάθε δείγμα προβλέπεται η πιθανότητα να ανήκει σε κάθε μία από τις κατηγορίες [47]. Περιλαμβάνει τον διαχωρισμό του συνόλου δεδομένων πολλών κλάσεων σε πολλαπλά προβλήματα δυαδικής ταξινόμησης. Στη συνέχεια εκπαιδεύεται ένας δυαδικός ταξινομητής σε κάθε πρόβλημα δυαδικής ταξινόμησης και γίνονται προβλέψεις [47]. Για παράδειγμα, δίνεται ένα πρόβλημα ταξινόμησης πολλών κλάσεων με δείγματα που ανήκουν στις κατηγορίες “κόκκινο”, “μπλε” και “πράσινο”. Αυτό θα μπορούσε να χωριστεί σε τρία δυαδικά σύνολα δεδομένων ταξινόμησης ως εξής:

  Πρόβλημα δυαδικής ταξινόμησης 1: κόκκινο vs [μπλε, πράσινο]
  Πρόβλημα δυαδικής ταξινόμησης 2: μπλε vs [κόκκινο, πράσινο]
  Πρόβλημα δυαδικής ταξινόμησης 3: πράσινο vs [κόκκινο, μπλε]

4.13 Υπερ-παράμετροι μοντέλων

Στην μηχανική μάθηση υπάρχουν διάφοροι τύποι παραμέτρων που μπορούν να επηρεάσουν την εκπαίδευση και την απόδοση των μοντέλων. Οι βασικοί τύποι παραμέτρων που συναντώνται συχνά περιλαμβάνουν τους παρακάτω [1], [7]:

- **Βάρη** (Weights). Τα βάρη αντιπροσωπεύουν τις παραμέτρους που προσαρμόζονται κατά τη διάρκεια της εκπαίδευσης του μοντέλου. Αυτές οι παράμετροι εκτιμώνται ή προσαρμόζονται από το μοντέλο μηχανικής μάθησης για να επιτύχει το καλύτερο δυνατό αποτέλεσμα.
- **Υπερ-παράμετροι** (Hyperparameters). Οι υπερ-παράμετροι είναι παράμετροι που καθορίζουν τη δομή ή τη συμπεριφορά του μοντέλου μηχανικής μάθησης. Δεν μαθαίνονται αυτόματα από τα δεδομένα εκπαίδευσης, αλλά πρέπει να οριστούν από τον χρήστη πριν την εκπαίδευση του μοντέλου. Παραδείγματα υπερ-παραμέτρων είναι ο αριθμός των γειτόνων στον k-Nearest Neighbors (k-NN), το μέγιστο βάθος του δέντρου απόφασης, ο συντελεστής μάθησης σε ένα νευρωνικό δίκτυο κλπ.

Όπως θα δούμε και στα πειράματά μας, ο αλγόριθμος της Scikit-learn, k-πλησιέστερων γειτόνων (K-Nearest Neighbors k-NN) χρησιμοποιεί ως αριθμό γειτόνων την υπερ-παράμετρο **n_neighbors**, η οποία καθορίζει τον αριθμό των κοντινότερων γειτόνων που θα χρησιμοποιηθούν για να καθορίσουν την κατηγορία ενός νέου δείγματος. Η επιλογή της κατάλληλης τιμής για το n_neighbors είναι σημαντική και επηρεάζει την απόδοση του αλγορίθμου. Επίσης, ο αλγόριθμος δέντρων απόφασης (Decision Tree), χρησιμοποιεί ως μέγιστο βάθος του δέντρου την υπερ-παράμετρο **max_depth**. η οποία περιορίζει τον αριθμό των επιπέδων στο δέντρο. Αν τεθεί μια συγκεκριμένη τιμή για το max_depth, τότε το δέντρο θα δημιουργηθεί με το συγκεκριμένο μέγιστο βάθος.

4.14 Επίλογος

Σε αυτό το κεφάλαιο ασχοληθήκαμε με διάφορα θέματα στον τομέα της μηχανικής μάθησης. Αρχικά, εξετάστηκε η επιτηρούμενη μάθηση, όπου ένα μοντέλο εκπαιδεύεται σε ένα σύνολο δεδομένων που περιέχει ετικέτες. Στη συνέχεια, αναφέρθηκε η μάθηση χωρίς επίβλεψη, όπου το μοντέλο ανακαλύπτει αμέτρητα πρότυπα και δομές από μη επισημασμένα δεδομένα.

Δόθηκε επίσης ο ορισμός της ταξινόμησης και της παλινδρόμησης. Η ταξινόμηση αναφέρεται στον τομέα όπου τα δεδομένα ανήκουν σε πεπερασμένο αριθμό κατηγοριών ή κλάσεων και το μοντέλο προσπαθεί να προβλέψει την κατηγορία ενός νέου δείγματος. Από την άλλη πλευρά, η παλινδρόμηση ασχολείται με την πρόβλεψη μιας συνεχούς τιμής ή μεγέθους.

Για την υλοποίηση κλασσικών μοντέλων μηχανικής μάθησης, χρησιμοποιείται η βιβλιοθήκη Scikit-learn, η οποία προσφέρει πολλούς αλγορίθμους και εργαλεία για την εκπαίδευση και την αξιολόγηση μοντέλων.

Στη συνέχεια, αναφερθήκαμε σε διάφορα κρίσιμα θέματα της μηχανικής μάθησης, όπως το TF-IDF και το Word2Vec.

Το TF-IDF είναι μια μέθοδος που αξιολογεί τη σημασία μιας λέξης σε ένα έγγραφο μέσω του συνδυασμού της συχνότητας εμφάνισής της (term frequency - tf) και της αντίστοιχης συχνότητας αναφοράς της σε όλα τα έγγραφα (inverse document frequency - idf). Αυτή η μέθοδος είναι χρήσιμη για την αναπαράσταση κειμένων και την αναζήτηση πληροφορίας.

Το Word2Vec αποτελεί μια τεχνική για την αναπαράσταση λέξεων σε μια διανυσματική μορφή. Χρησιμοποιείται για να ανακαλύψει σημασιολογικές συσχετίσεις μεταξύ των λέξεων μέσω της εκπαίδευσης ενός νευρωνικού δικτύου πάνω σε μεγάλα κείμενα.

Συνεχίζοντας, προσεγγίσαμε τις μετρικές ακρίβειας μοντέλων όπως ο πίνακας σύγχυσης (Confusion Matrix), η αναφορά ταξινόμησης (Classification Report), οι μετρικές Precision, Recall και F1-score.

Ο πίνακας σύγχυσης παρέχει μια αναλυτική εικόνα των αποτελεσμάτων της ταξινόμησης, καθορίζοντας τον αριθμό των σωστών και λανθασμένων προβλέψεων για κάθε κατηγορία.

Το Classification Report, παρέχει τις μετρικές: ευστοχία (precision), ανάκληση (recall) και το F1-score. Η ευστοχία αποτελεί το ποσοστό των σωστών προβλέψεων μιας κλάσης σε σχέση με τον συνολικό αριθμό προβλέψεων για αυτή την κλάση, ενώ η ανάκληση αποτελεί το ποσοστό των σωστών προβλέψεων μιας κλάσης σε σχέση με τον συνολικό αριθμό δειγμάτων της κλάσης.

Το F1-score συνδυάζει την ευστοχία και την ανάκληση, παρέχοντας μια μετρική που αξιολογεί τη συνολική απόδοση του μοντέλου.

Επιπλέον, εξετάστηκε το θεωρητικό και μαθηματικό υπόβαθρο διάφορων αλγορίθμων μηχανικής μάθησης. Κάθε αλγόριθμος έχει τα δικά του χαρακτηριστικά, πλεονεκτήματα και περιορισμούς, και επιλέγεται ανάλογα με τα χαρακτηριστικά των δεδομένων και το επιθυμητό πρόβλημα μάθησης.

Τέλος, τονίστηκε η σημασία των υπερ-παραμέτρων στην εκπαίδευση των μοντέλων. Οι υπερ-παράμετροι είναι παράμετροι που πρέπει να οριστούν πριν την εκπαίδευση του μοντέλου. Μερικές από αυτές, περιλαμβάνουν τον αριθμό των δέντρων στο Random Forest, την παράμετρο C στο SVM και τον αριθμό των γειτόνων στο k-NN. Η βέλτιστη επιλογή των υπερ-παραμέτρων είναι κρίσιμη για την απόδοση και τη γενίκευση του μοντέλου.

Κεφάλαιο 5ο: Βαθιά Μάθηση

5.1 Εισαγωγή

Ένας από τους πιο σπουδαίους κλάδους της τεχνητής νοημοσύνης είναι η βαθιά μάθηση (deep learning), η οποία αποτελεί υποσύνολο της μηχανικής μάθησης [1], [7]. Αναφέρεται σε αλγορίθμους μάθησης που χρησιμοποιούνται για την αναγνώριση πολύπλοκων προτύπων και χαρακτηριστικών σε δεδομένα υψηλής διάστασης, όπως εικόνες, ήχος, βίντεο, κείμενο και χρονοσειρές. Οι αλγόριθμοι deep learning αξιοποιούν τη δομή των δεδομένων και τη χρήση πολλαπλών στρωμάτων νευρωνικών δικτύων προκειμένου να αντλήσουν πιο σύνθετα χαρακτηριστικά από τα δεδομένα εισόδου.

Οι Ajay Shrestha και Ausif Mahmood [48], μας παρουσιάζουν μια σύντομη ιστορία των νευρωνικών δικτύων, ξεκινώντας το 1957 με τη δημιουργία του μοντέλου Perceptron, το πρώτο πρωτότυπο αυτού που σήμερα γνωρίζουμε ως νευρωνικό δίκτυο. Αναφέρουν επίσης τα προβλήματα που αντιμετώπισε το Perceptron στα τέλη της δεκαετίας του 1960, καθώς αντιμετώπιζε δυσκολίες στη μάθηση βασικών συναρτήσεων, καθώς και στην εύρεση του καθολικού θεωρήματος προσέγγισης. Αυτά τα γεγονότα συνέβαλαν στον λεγόμενο "χειμώνα της τεχνητής νοημοσύνης", μια περίοδο μειωμένης χρηματοδότησης και ενδιαφέροντος για τον τομέα. Η μέθοδος εκμάθησης back-propagation [48], [49], η οποία προτάθηκε για πρώτη φορά τη δεκαετία του 1970 και έγινε πλήρως κατανοητή στα μέσα της δεκαετίας του 1980, ήταν το κλειδί για την πρόοδο των νευρωνικών δικτύων [50]. Ως αποτέλεσμα, οι εξαγωγείς χαρακτηριστικών θα μπορούσαν να αυτοματοποιηθούν και θα μπορούσαν να δημιουργηθούν βαθιά νευρωνικά δίκτυα. Η χαρτογράφηση πιο περίπλοκων και μη γραμμικών χαρακτηριστικών έχει καταστεί δυνατή με τη μετάβαση από την επιφανειακή στη βαθιά εκμάθηση. Σε συνδυασμό με τη διαθεσιμότητα φθηνότερων μονάδων επεξεργασίας και πληθώρας δεδομένων, αυτό οδήγησε στην εξάπλωση της βαθιάς μάθησης σε διάφορους κλάδους.

Έτσι, η βαθιά μάθηση έχει εφαρμογές σε πολλούς τομείς, όπως η επεξεργασία φυσικής γλώσσας, η αναγνώριση φωνής, η ρομποτική, η ικανότητα αναγνώρισης προσώπων, η ανάλυση συναισθήματος, η αναγνώριση λέξεων σε ένα ηχητικό αρχείο, η αυτόνομη οδήγηση, η ιατρική, η πρόβλεψη αποτελεσμάτων εκλογών, η επιστήμη των υλικών, η πρόγνωση σεισμών κλπ. Η διαθεσιμότητα μαζικών δεδομένων, οι μεγαλύτερες ταχύτητες Internet, και οι εξελίξεις στο υλικολογισμικό ενθαρρύνουν περισσότερους οργανισμούς και ιδιώτες να ασχοληθούν με λύσεις βαθιάς μάθησης που απαιτούν σημαντικούς πόρους [1], [7].

5.2 Μαζικά δεδομένα

Τα μαζικά δεδομένα (big data) αναφέρονται σε μεγάλους όγκους δεδομένων που παράγονται από διάφορες πηγές, όπως αισθητήρες, κινητά τηλέφωνα, κοινωνικά δίκτυα, διαδίκτυο των πραγμάτων, αρχεία βίντεο, εικόνες, κείμενα και ήχοι. Τα μαζικά δεδομένα εκτός από τον όγκο τους, χαρακτηρίζονται και από την ταχύτητα με την οποία παράγονται και μεταβάλλονται, καθώς και την ποικιλία και την πολυπλοκότητα των δεδομένων.

Η μεθοδολογία που ακολουθείται αφορά τη διαδικασία της εξόρυξης των δεδομένων από μαζικά δεδομένα και κατ' επέκταση την ανάλυση αυτών μέσω προηγμένων τεχνολογιών όπως είναι η μηχανική και η βαθιά μάθηση. Οι εφαρμογές των μαζικών δεδομένων είναι πολλές και ποικίλες και περιλαμβάνουν την βελτιστοποίηση διαδικασιών παραγωγής, την πρόβλεψη τάσεων και

συμπεριφορών, τη βελτίωση της ποιότητας και της ασφάλειας των προϊόντων όπως επίσης και την ανάπτυξη νέων προϊόντων και υπηρεσιών.

Η βαθιά μάθηση αποδίδει καλύτερα όταν διαθέτουμε πολλά δεδομένα. Για μικρότερα σύνολα δεδομένων συνιστάται η χρήση της αύξησης των δεδομένων (data augmentation) ή η μεταφορά μάθησης (transfer learning), προκειμένου να επιτευχθεί υψηλή απόδοση. Η μεταφορά μάθησης χρησιμοποιεί την υπάρχουσα γνώση από ένα ήδη εκπαιδευμένο μοντέλο ως βάση για ένα νέο μοντέλο, ενώ η αύξηση δεδομένων είναι μια τεχνική της μηχανικής μάθησης και της επεξεργασίας εικόνας, που χρησιμοποιείται για την δημιουργία νέων δεδομένων εκπαίδευσης από τα υπάρχοντα δεδομένα εκπαίδευσης. Γενικά, όσο περισσότερα δεδομένα διαθέτουμε, τόσο καλύτερη εκπαίδευση γίνεται σε ένα μοντέλο βαθιάς μάθησης [1], [7].

5.3 Υπολογιστική ισχύς

Η βαθιά μάθηση απαιτεί συνήθως σημαντική υπολογιστική ισχύ για την εκπαίδευση των μοντέλων και την επεξεργασία των δεδομένων. Τα μοντέλα βαθιάς μάθησης έχουν πολλά επίπεδα (layers) και μεγάλο αριθμό παραμέτρων, οπότε για να εκπαιδεύσουμε αποτελεσματικά αυτά τα μοντέλα, απαιτούνται συνήθως ειδικές μονάδες υλικού υψηλής απόδοσης όπως μονάδες επεξεργασίας γραφικών (Graphics Processing Units - GPU) και μονάδες επεξεργασίας τανυστών (Tensor Processing Units - TPU) ώστε να ικανοποιήσουν τις υψηλές απαιτήσεις ισχύος των εφαρμογών βαθιάς μάθησης [1], [7].

5.4 TensorFlow και Keras

Το TensorFlow είναι μια πλατφόρμα ανοιχτού κώδικα αναφορικά με τη μηχανική (machine learning) και την βαθιά μάθηση (deep learning) που αναπτύχθηκε από την Google. Παρέχει ένα πλούσιο σετ από εργαλεία και βιβλιοθήκες για τη δημιουργία και εκτέλεση αλγορίθμων μηχανικής μάθησης [51].

Το Keras αποτελεί μια υψηλού επιπέδου βιβλιοθήκη μηχανικής μάθησης που βασίζεται στο TensorFlow και επιτρέπει την εύκολη και γρήγορη δημιουργία, εκπαίδευση και αξιολόγηση νευρωνικών δικτύων. Αναπτύχθηκε από τον François Chollet και κυκλοφόρησε για πρώτη φορά το 2015. Το Keras δίνει τη δυνατότητα στους μηχανικούς και τους ερευνητές να εκμεταλλευτούν πλήρως την επεκτασιμότητα και τις δυνατότητες πολλαπλών πλατφορμών του TensorFlow [51].

5.5 Μοντέλα βαθιάς μάθησης και πειραματισμός

Ενώ τα κλασικά μοντέλα μηχανικής μάθησης της βιβλιοθήκης Scikit-learn ορίζονται εύκολα με μια εντολή, τα μοντέλα βαθιάς μάθησης απαιτούν πιο σύνθετες δομές, οι οποίες διασυνδέουν πολλά αντικείμενα που ονομάζονται στρώματα (layers). Τα εν λόγω μοντέλα μάθησης απαιτούν εκτεταμένο μαθηματικό υπόβαθρο για την κατανόηση της εσωτερικής λειτουργίας τους. Η Keras αναλαμβάνει την ενσωμάτωση των σύνθετων μαθηματικών, έτσι ώστε οι προγραμματιστές να περιορίζονται μόνο στον ορισμό, την παραμετροποίηση και των χειρισμό αντικειμένων [1], [7].

Η μηχανική και η βαθιά μάθηση είναι κλάδοι που βασίζονται πολύ στον πειραματισμό. Για το λόγο αυτό, στην παρούσα εργασία πειραματιστήκαμε με αρκετά μοντέλα, τροποποιώντας τα με διάφορους τρόπους, προκειμένου να εντοπίσουμε αυτά που αποδίδουν καλύτερα σε κάθε σύνολο δεδομένων.

5.6 Datasets της Keras

Η βιβλιοθήκη Keras παρέχει μερικά σύνολα δεδομένων που είναι ενσωματωμένα και μπορούν να χρησιμοποιηθούν για εκπαίδευση και δοκιμή μοντέλων νευρωνικών δικτύων. Ορισμένα από αυτά τα σύνολα δεδομένων είναι τα εξής:

- **IMDB Movie Reviews**. Το σύνολο δεδομένων IMDB περιέχει κριτικές ταινιών που έχουν ταξινομηθεί ως θετικές ή αρνητικές (25000 κριτικές εκπαίδευσης και 25000 για έλεγχο). Χρησιμοποιείται συχνά για την εκπαίδευση μοντέλων που ασχολούνται με την αναγνώριση συναισθημάτων ή την ανάλυση κειμένου. [52].
- **MNIST**. Το MNIST είναι ένα σύνολο δεδομένων που περιέχει εικόνες χειρόγραφων ψηφίων (επισημασμένες από το 0 έως το 9) σε κλίμακα του γκρι, από τις οποίες οι 60000 είναι για εκπαίδευση και οι 10000 για έλεγχο. Χρησιμοποιείται συχνά ως ένα απλό παράδειγμα για την εκπαίδευση και την αξιολόγηση μοντέλων επεξεργασίας εικόνας [32], [53].
- **Fashion-MNIST**. Το Fashion-MNIST, είναι ένα σύνολο δεδομένων που περιέχει 60000 εικόνες εκπαίδευσης και 10000 εικόνες ελέγχου, όλες σε αποχρώσεις του γκρι. Οι εικόνες έχουν ανάλυση 28x28 pixel και αντιστοιχούν σε 10 διαφορετικές κατηγορίες ρούχων, όπως φούστες, μπουφάν, μπλουζάκια κλπ. Χρησιμοποιείται για την αξιολόγηση της ικανότητας των μοντέλων να αναγνωρίζουν και να κατηγοριοποιούν διάφορα είδη μόδας με βάση τις εικόνες που τους δίνονται ως είσοδο [54].
- **Boston Housing**. Το Boston Housing dataset, είναι ένα δημοφιλές σύνολο δεδομένων που περιλαμβάνει πληροφορίες σχετικά με τις κατοικίες σε διάφορες περιοχές της πόλης της Βοστόνης. Για κάθε κατοικία, διαθέτει χαρακτηριστικά όπως, ο μέσος αριθμός δωματίων ανά κατοικία, η απόσταση από την κοντινότερη εμπορική περιοχή, ο δείκτης προσβασιμότητας σε αυτοκινητόδρομους, η μέση απόσταση σε μίλια από πέντε εμπορικά κέντρα της Βοστόνης, το ποσοστό εμπορικής μη κατοικημένης έκτασης ανά πόλη, κλπ. Το σύνολο δεδομένων χρησιμοποιείται συχνά για προβλήματα πρόβλεψης της μέσης τιμής των κατοικιών σε διάφορες περιοχές της Βοστόνης βασιζόμενοι στα διαθέσιμα χαρακτηριστικά. Περιλαμβάνει συνολικά 506 δείγματα, με τις τιμές να κυμαίνονται από 5 έως 50 χιλιάδες δολάρια [55].
- **CIFAR-10 και CIFAR-100**. Είναι δύο σύνολα δεδομένων που περιέχουν εικόνες που ανήκουν σε διάφορες κατηγορίες, όπως πουλιά, γάτες, αυτοκίνητα, κ.λπ. Το CIFAR-10 περιέχει 10 κατηγορίες, ενώ το CIFAR-100 περιέχει 100 κατηγορίες. Και τα δύο σύνολα διαθέτουν 50000 εικόνες για εκπαίδευση και 10000 για έλεγχο [56-58].

5.7 Νευρωνικά δίκτυα

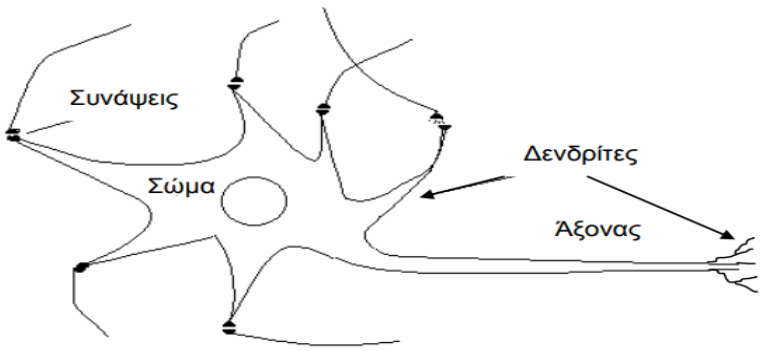
Τα Νευρωνικά Δίκτυα (Neural Networks), είναι υπολογιστικά μοντέλα που αντλούν έμπνευση από τον τρόπο λειτουργίας του ανθρώπινου εγκεφάλου, αντί να επιχειρούν να αποτυπώσουν ρητά την γνώση και να χρησιμοποιήσουν αλγόριθμους αναζήτησης. Στα βιολογικά μας συστήματα, ο έλεγχος γίνεται μέσω νευρώνων που αλληλεπιδρούν μεταξύ τους. Τα νευρωνικά δίκτυα επιδιώκουν να αναπαραστήσουν αυτήν την ιδέα με μια οργάνωση σε επίπεδα νευρώνων. Κάθε νευρώνας λαμβάνει εισόδους, επεξεργάζεται τις πληροφορίες και παράγει μια έξοδο, αντικατοπτρίζοντας έτσι την αλληλεπίδραση των νευρώνων στον εγκέφαλο [31], [59].

5.7.1 Βιολογικά νευρωνικά δίκτυα (Biological Neural Networks)

Στη βιολογία, ο νευρώνας (neuron) αναγνωρίζεται ως η βασική μονάδα του εγκεφάλου. Οι νευρώνες αλληλεπιδρούν μεταξύ τους με τη βοήθεια συνάψεων, δημιουργώντας ένα πυκνό δίκτυο συνδέσεων. Μέσω αυτών των συνδέσεων, μπορούν να μεταδίδουν πληροφορίες και να επικοινωνούν μεταξύ τους. Αυτή η οργάνωση επιτρέπει στον εγκέφαλο να λειτουργεί ως ένα πολύπλοκο δίκτυο νευρωνικών συνδέσεων, που επεξεργάζεται πληροφορίες και παράγει αποτελέσματα. Αυτή η φυσική αρχιτεκτονική των νευρώνων έχει εμπνεύσει την δημιουργία των **τεχνητών νευρωνικών δικτύων**, που αποσκοπούν να αναπαράγουν και να μοντελοποιήσουν αυτήν την ιδιότητα του ανθρώπινου εγκεφάλου για την επίλυση προβλημάτων [31].

Ένας βιολογικός νευρώνας αποτελείται από τρία βασικά στοιχεία: το σώμα, τους δενδρίτες και τον άξονα. Το σώμα του νευρώνα αποτελεί τον πυρήνα του και περιλαμβάνει τον κυτταρικό πυρήνα και άλλα κυτταρικά συστατικά. Οι δενδρίτες εκτείνονται από το σώμα και λαμβάνουν σήματα από άλλους νευρώνες μέσω συνάψεων. Ο άξονας αποτελεί το μέσο σύνδεσης του νευρώνα με άλλους νευρώνες και μεταδίδει τα σήματα που παράγονται από το σώμα. Κάθε δενδρίτης διαθέτει συνάψεις, σημεία στα οποία μπορεί να επιτείνεται ή να επιβραδύνεται η ροή ηλεκτρικών φορτίων προς το σώμα του νευρώνα. Οι νευρώνες λαμβάνουν ηλεκτρικά σήματα μέσω των δενδριτών τους, τα οποία συνδυάζονται και αν το αποτέλεσμα ξεπερνά μια καθορισμένη τιμή κατωφλίου, το σήμα αυτό μεταδίδεται στους άλλους νευρώνες μέσω του άξονα του νευρώνα [31].

Η εκτίμηση για τον αριθμό των νευρώνων στον εγκέφαλο ενός ανθρώπου είναι περίπου 100 δισεκατομμύρια. Ωστόσο, αυτός ο αριθμός μπορεί να διαφέρει από άτομο σε άτομο. Είναι σημαντικό να σημειωθεί ότι η απόπειρα αντιγραφής της πλήρους δομής και λειτουργίας του εγκεφάλου σε μια τέτοια κλίμακα είναι αδύνατη. Οι υπάρχουσες προσπάθειες κατασκευής μοντέλων συνήθως περιορίζονται σε εκατοντάδες ή ακόμα και χιλιάδες νευρώνες. Παρόλα αυτά, αυτές οι προσπάθειες μπορούν να μας βοηθήσουν να κατανοήσουμε κάποιες από τις βασικές αρχές και λειτουργίες του εγκεφάλου [31]. Τέλος, να σημειωθεί ότι, ο εγκέφαλος είναι ικανός να παίρνει πολύπλοκες αποφάσεις με μεγάλη ταχύτητα. Αυτό οφείλεται σε μεγάλο βαθμό στο γεγονός ότι, η ικανότητα και η πληροφορία του εγκεφάλου είναι κατανεμημένες σε όλο τον όγκο του. Ο εγκέφαλος λειτουργεί ως ένα παράλληλο και κατανεμημένο σύστημα, όπου πολλοί νευρώνες εργάζονται ταυτόχρονα για την επεξεργασία διαφορετικών πτυχών της πληροφορίας. Αυτά τα χαρακτηριστικά αποτελούν το κύριο κίνητρο πίσω από την επιθυμία να μοντελοποιηθεί ο ανθρώπινος εγκέφαλος χρησιμοποιώντας τεχνητά νευρωνικά δίκτυα [31]. Η Εικόνα 5.1 παρουσιάζει έναν βιολογικό νευρώνα.



Εικόνα 5.1: Βιολογικός νευρώνας [31]

5.7.2 Τεχνητά νευρωνικά δίκτυα (Artificial Neural Networks)

Πριν προχωρήσουμε στα τεχνητά νευρωνικά δίκτυα, είναι απαραίτητο να γίνει μια περιγραφή της βασικής μονάδας τους, που είναι ο **τεχνητός νευρώνας**.

Ένας τεχνητός νευρώνας, είναι μια υπολογιστική μονάδα που μοντελοποιεί τη λειτουργία ενός βιολογικού νευρώνα. Δέχεται σήματα εισόδου σε μορφή συνεχών μεταβλητών, αντίθετα με τους ηλεκτρικούς παλμούς που παράγει ο εγκέφαλος. Κάθε εισερχόμενο σήμα τροποποιείται από ένα βάρος (weight), το οποίο αντιπροσωπεύει την ισχύ της σύνδεσης, αντίστοιχα με τη σύναψη μεταξύ βιολογικών νευρώνων. Το βάρος μπορεί να είναι θετικό ή αρνητικό, επηρεάζοντας την ταχύτητα της μετάδοσης των σημάτων [31], [59].

Κάθε τεχνητός νευρώνας χωρίζεται συνήθως σε τρία βασικά μέρη: τον **αθροιστή** (sum), τη **συνάρτηση ενεργοποίησης** (activation function) και την **έξοδο** (output). Η λειτουργία του αθροιστή, είναι να αθροίζει τις εισόδους οι οποίες έχουν επηρεαστεί από τα βάρη και να παράγει μια ποσότητα S [59]. Η συνάρτηση ενεργοποίησης λαμβάνει υπόψη τις εισόδους και τις συνολικές επιρροές που προκύπτουν από τα βάρη των εισόδων, εφαρμόζει μια **μη γραμμική** μετασχηματιστική λειτουργία στο συνολικό εισερχόμενο σήμα και παράγει μια έξοδο, η οποία αντιπροσωπεύει το τελικό αποτέλεσμα της επεξεργασίας. Είναι σημαντικό να σημειωθεί ότι η μοναδικότητα της εξόδου του νευρώνα σχετίζεται με την τιμή της εξόδου και όχι με τον αριθμό των εξόδων που διαθέτει. Με άλλα λόγια, ένας τεχνητός νευρώνας μπορεί να έχει πολλαπλές εξόδους, αλλά όλες θα παράγουν την ίδια τιμή [31].

Επομένως, σύμφωνα με τα παραπάνω, κάθε νευρώνας υπολογίζει ένα σταθμισμένο άθροισμα των εισόδων του [59]:

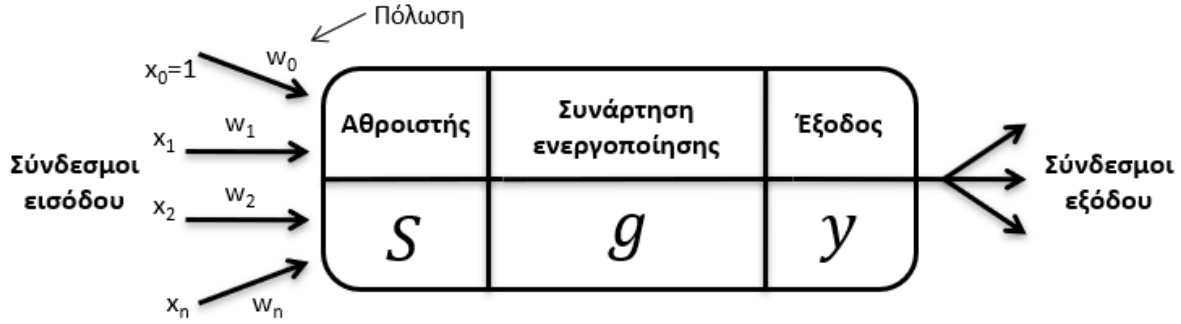
$$S = \sum_{i=0}^{n} w_i x_i \tag{5.1}$$

Στη συνέχεια, εφαρμόζει μια συνάρτηση ενεργοποίησης, έστω g, σε αυτό το άθροισμα για να παραγάγει την έξοδο [59]:

$$y = g(S) = g\left(\sum_{i=0}^{n} w_i x_i\right) \tag{5.2}$$

Όπου w_i είναι τα βάρη, $(i = 0,1,2,\dots,n)$ και x_i είναι οι είσοδοι.

Επιπλέον των εισόδων και των βαρών, ο νευρώνας έχει και ένα βάρος w_0 που ονομάζεται **πόλωση** (bias). Το βάρος πόλωσης είναι συνδεδεμένο με μια σταθερή είσοδο $x_0 = 1$ [31]. Έτσι, το βάρος πόλωσης είναι ένας παράγοντας που επηρεάζει την απόφαση του νευρώνα για την παραγωγή της εξόδου. Με άλλα λόγια, το βάρος πόλωσης επηρεάζει το σημείο έναρξης της ενεργοποίησης του νευρώνα και τον τρόπο που ανταποκρίνεται στις εισόδους [31]. Στο Σχήμα 5.1, φαίνεται το μοντέλο του τεχνητού νευρώνα, το οποίο επινοήθηκε από τους McCulloch και Pitts [59].



Σχήμα 5.1: Απλό μοντέλο νευρώνα [59]

Τα **Τεχνητά Νευρωνικά Δίκτυα** – **ΤΝΔ** (Artificial Neural Networks), είναι ένα είδος μηχανικής μάθησης που αναπτύχθηκε για να μιμηθεί τη λειτουργία του ανθρώπινου εγκεφάλου. Σε αντίθεση με τα παραδοσιακά προγράμματα υπολογιστών που ακολουθούν αυστηρά οδηγίες, τα νευρωνικά δίκτυα είναι ικανά να αναγνωρίζουν και να μαθαίνουν από πολύπλοκα μοτίβα στα δεδομένα. Στην ουσία, ένα νευρωνικό δίκτυο αποτελείται από ένα σύνολο τεχνητών νευρώνων που συνδέονται μεταξύ τους με κατευθυνόμενους συνδέσμους (links) [31], [59], [60].

Η βασική ιδέα πίσω από τα νευρωνικά δίκτυα είναι η επεξεργασία των εισόδων με τη χρήση ενός συνόλου από **επίπεδα** ή **στρώματα** νευρώνων που λειτουργούν ως φίλτρα και εξάγουν χρήσιμα χαρακτηριστικά από τα δεδομένα. Κάθε επίπεδο του δικτύου επεξεργάζεται την είσοδο με βάση τη συνάρτηση ενεργοποίησής του και εξάγει νέα χαρακτηριστικά που θα χρησιμοποιηθούν στο επόμενο επίπεδο [31], [59], [60].

Γενικά, υπάρχουν τρεις διακριτές κατηγορίες στρωμάτων νευρώνων [31], [59]:

- Το στρώμα των **εισόδων** (input layer) ή αλλιώς διάνυσμα χαρακτηριστικών.
- Ένα ή περισσότερα στρώματα **κρυφών νευρώνων** (hidden layers).
- Το στρώμα **εξόδων** (output layer).

Αξίζει να επισημάνουμε πως ένα νευρωνικό δίκτυο χαρακτηρίζεται ως **βαθύ**, εάν διαθέτει πολλά κρυφά στρώματα, ενώ **ρηχό** είναι ένα δίκτυο με μικρό βάθος π.χ. 2 – 4 στρώματα [38].

5.7.3 Συναρτήσεις ενεργοποίησης

Υπάρχουν πολλές εναλλακτικές μορφές της συνάρτησης ενεργοποίησης του νευρώνα του σχήματος 5.1. Παρακάτω παρουσιάζονται 6 περιπτώσεις [31], [38], [59]. Το κοινό γνώρισμα των 5 πρώτων είναι η **μη γραμμικότητά** τους. Αυτό είναι σημαντικό για την μοντελοποίηση μη γραμμικών προβλημάτων [38].

α. Βηματική συνάρτηση (step function)

Αν η τιμή του αθροιστή είναι μεγαλύτερη απο ένα κατώφλι T, τότε δίνει στην έξοδο την τιμή 1.

$$g(S) = \begin{cases} -1 & \alpha\nu\ S \leq 0 \\ 1 & \alpha\nu\ S > 0 \end{cases}$$

β. Λογιστική συνάρτηση (logistic function)

Ανήκει σε μια οικογένεια συναρτήσεων που λέγεται **σιγμοειδής** (sigmoid).

$$g(S) = \frac{1}{1 + e^{-S}}$$

γ. Υπερβολική εφαπτομένη (hyperbolic tangent)

$$g(S) = tanh(S) = \frac{1 - e^{-S}}{1 + e^{-S}}$$

δ. Συνάρτηση ράμπας (ramp function)

Ονομάζεται και ανορθωμένη γραμμική μονάδα (Rectified Linear Unit - **ReLU**).

$$g(S) = \begin{cases} 0 & \alpha\nu\ S \leq 0 \\ S & \alpha\nu\ S > 0 \end{cases}$$

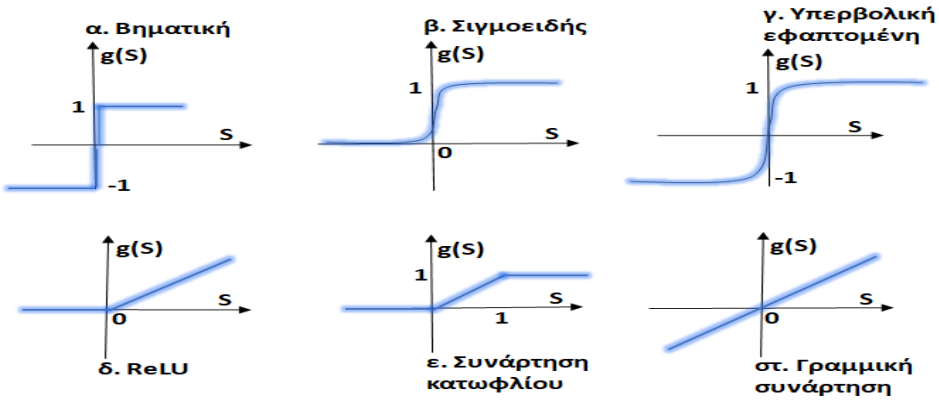
ε. Συνάρτηση κατωφλίου (threshold function)

$$g(S) = \begin{cases} 0 & \alpha\nu\ S \leq 0 \\ S & \alpha\nu\ 0 < S < 1 \\ 1 & \alpha\nu\ S \geq 1 \end{cases}$$

στ. Γραμμική συνάρτηση (linear function)

$$g(S) = S$$

Στο Σχήμα 5.2, φαίνονται οι γραφικές παραστάσεις των παραπάνω συναρτήσεων ενεργοποίησης:



Σχήμα 5.2: Συναρτήσεις ενεργοποίησης [31], [38]

Εφαρμογή νευρώνων για την υλοποίηση αλγεβρικών συναρτήσεων

Με κατάλληλη χρήση τιμών στις εισόδους και στα βάρη, οι νευρώνες μπορούν να υλοποιήσουν λογικές συναρτήσεις όπως AND, OR, NOT. Για παράδειγμα, αν θέλουμε να υλοποιήσουμε την πύλη AND και χρησιμοποιήσουμε την βηματική συνάρτηση (0/1) με κατώφλι $T = 1.5$ και βάρη $w_1 = 1$ και $w_2 = 1$, τότε για εισόδους $x_1 = 1$ και $x_2 = 1$ θα έχουμε:

$$S = x_1 \cdot w_1 + x_2 \cdot w_2 = 1 + 1 = 2$$

$$S = 2 > T$$

Εφόσον $S > T$, τότε η έξοδος είναι 1.

Για εισόδους $x_1 = 0$ και $x_2 = 1$, θα έχουμε:

$$S = x_1 \cdot w_1 + x_2 \cdot w_2 = 0 + 1 = 1$$

$$S = 1 < T$$

Εφόσον $S < T$, τότε η έξοδος είναι 0.

Αντίστοιχα για την υλοποίηση της πύλης NOT, με κατώφλι $T = -0.5$, βάρος $w_1 = -1$, και είσοδο $x_1 = 1$ θα έχουμε:

$$S = x_1 \cdot w_1 = -1$$

$$S = -1 < T$$

Άρα, η έξοδος θα είναι 0.

Ενώ για είσοδο $x_1 = 0$, θα έχουμε:

$$S = 0 > T$$

Άρα, η έξοδος θα είναι 1.

Με την ίδια λογική μπορεί να υλοποιηθεί και η πύλη OR.

5.7.4 Κατηγορίες δομών νευρωνικών δικτύων

Υπάρχουν τέσσερις βασικές κατηγορίες δομών νευρωνικών δικτύων, τα δίκτυα **πρόσθιας τροφοδότησης** (feed-forward networks), τα κυκλικά ή **αναδρομικά δίκτυα** (recurrent neural networks - RNN), τα **συνελικτικά νευρωνικά δίκτυα** (convolutional neural networks - CNN), καθώς και μια καινοτόμο αρχιτεκτονική νευρωνικών δικτύων, που ονομάζονται **μετασχηματιστές** (transformers).

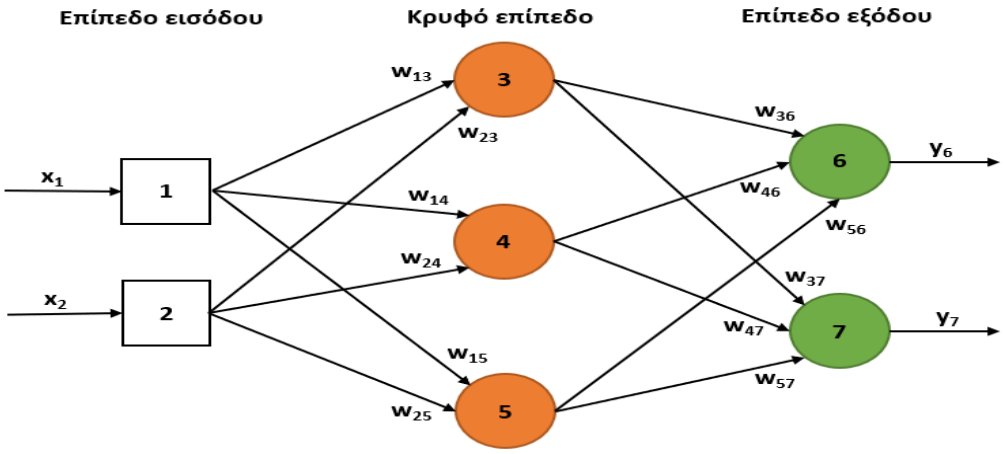
Ένα **δίκτυο πρόσθιας τροφοδότησης** αναφέρεται στην απλούστερη μορφή νευρωνικών δικτύων. Σε αυτό τον τύπο δικτύου, η ροή της πληροφορίας είναι μονόδρομη, κινείται δηλαδή μόνο προς μία κατεύθυνση. Το δίκτυο αποτελείται από ένα επίπεδο εισόδου, ένα επίπεδο εξόδου, και μπορεί να περιλαμβάνει προαιρετικά ένα ή περισσότερα κρυφά επίπεδα. Κάθε επίπεδο αποτελείται από μια συλλογή νευρώνων που λαμβάνουν εισόδους από το προηγούμενο επίπεδο και παράγουν εξόδους που

διαβιβάζονται στο επόμενο επίπεδο. Οι κρυφοί νευρώνες αντιπροσωπεύουν ενδιάμεσες αναπαραστάσεις των δεδομένων, ενώ οι είσοδοι και οι έξοδοι του δικτύου αντιπροσωπεύουν τις αρχικές εισόδους και τις τελικές προβλέψεις αντίστοιχα [31]. Μια απλή δομή δικτύου πρόσθιας τροφοδότησης χωρίς κρυφά στρώματα το οποίο αποτελεί μια πρώτη προσέγγιση τεχνητών νευρωνικών δικτύων είναι το **Perceptron** [31] που αναφέρθηκε και στην εισαγωγή αυτού του κεφαλαίου. Το εν λόγω μοντέλο προτάθηκε απο τον Rosenblatt ως ένας μηχανισμός για ταξινόμηση προτύπων και εξακολουθεί να υφίσταται ακόμα και σήμερα [48].

Κατά την υλοποίηση δικτύων πρόσθιας τροφοδότησης, υπάρχουν δύο σημαντικά θέματα που προκύπτουν. Το πρώτο θέμα αφορά τη **μάθηση**, δηλαδή τον τρόπο με τον οποίο το δίκτυο εκπαιδεύεται για να επιδείξει την επιθυμητή συμπεριφορά. Συνήθως, χρησιμοποιούνται μέθοδοι μάθησης με επίβλεψη, όπου το δίκτυο εκπαιδεύεται με βάση ένα σύνολο δεδομένων που περιέχει εισόδους και τις αντίστοιχες επιθυμητές εξόδους. Ο στόχος είναι να βρεθούν τα βάρη των νευρώνων που ελαχιστοποιούν το σφάλμα μεταξύ των προβλέψεων του δικτύου και των πραγματικών εξόδων. Το δεύτερο θέμα αφορά την **τοπολογία** του δικτύου, δηλαδή τον αριθμό των κρυφών επιπέδων, τον αριθμό των νευρώνων σε κάθε επίπεδο και τον τρόπο με τον οποίο οι νευρώνες συνδέονται μεταξύ τους. Προφανώς, σύμφωνα με τα παραπάνω, η τοπολογία επηρεάζει την απόδοση και την ικανότητα γενίκευσης του δικτύου [31].

Για το θέμα της τοπολογίας, δεν υπάρχει ένας συγκεκριμένος κανόνας που πρέπει να ακολουθηθεί. Η επιλογή της αρχιτεκτονικής του δικτύου εξαρτάται συνήθως από το είδος του προβλήματος που επιχειρείται να επιλυθεί. Κάθε πρόβλημα μπορεί να απαιτεί διαφορετική δομή δικτύου για να επιτευχθούν οι καλύτερες αποδόσεις. Γενικά, η διαδικασία απαιτεί πολλούς πειραματισμούς και δοκιμές προτού βρεθεί μια κατάλληλη δομή για ένα συγκεκριμένο πρόβλημα [31]. Οι επιστήμονες και οι μηχανικοί μηχανικής μάθησης πρέπει να πραγματοποιούν διάφορες δοκιμές, να προσαρμόζουν τον αριθμό των κρυφών επιπέδων, τον αριθμό των νευρώνων και τον τρόπο σύνδεσης μεταξύ τους, προκειμένου να βρουν μια καλή αρχιτεκτονική που θα επιτυγχάνει την επιθυμητή απόδοση για το συγκεκριμένο πρόβλημα.

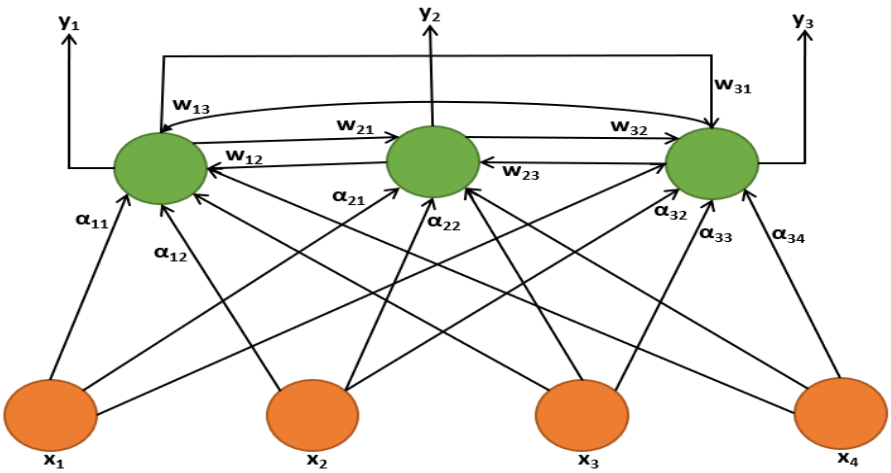
Στο Σχήμα 5.3 φαίνεται ένα νευρωνικό δίκτυο πρόσθιας τροφοδότησης. Ο νευρώνας 3 δέχεται δύο σήματα εισόδου απο τους νευρώνες 1 και 2 του στρώματος εισόδου. Έπειτα, τα σήματα αυτά τροποποιούνται από τα βάρη w_{13} και w_{23} και αθροίζονται απο τον αθροιστή του νευρώνα 3. Στη συνέχεια αναλαμβάνει δράση η συνάρτηση ενεργοποίησης του νευρώνα 3, και το αποτέλεσμα μεταφέρεται στους νευρώνες 6 και 7. Παρόμοια είναι η λειτουργία και των υπόλοιπων νευρώνων, εκτός απο αυτούς του επιπέδου εισόδου, όπου απλά αναλαμβάνουν να στείλουν το σήμα στο αμέσως επόμενο επίπεδο. Επομένως, το εν λόγω νευρωνικό δίκτυο, δέχεται τις εισόδους x_1 και x_2 που είναι το διάνυσμα εισόδου, και μετά απο επεξεργασία παράγει το διάνυσμα εξόδου που είναι τα σήματα εξόδου y_6 και y_7.



Σχήμα 5.3: Νευρωνικό δίκτυο πρόσθιας τροφοδότησης [31]

Ένα **αναδρομικό νευρωνικό δίκτυο** (RNN) [61], αποτελείται από νευρώνες που συνδέονται μεταξύ τους, δημιουργώντας κύκλους. Αυτό επιτρέπει στο δίκτυο να μεταφέρει τις εξόδους του πίσω στις εισόδους του. Ως εκ τούτου, η τιμή του νευρώνα i εξαρτάται από τις τιμές των υπόλοιπων νευρώνων, και οι τιμές των υπόλοιπων νευρώνων εξαρτώνται από την τιμή του νευρώνα i [38], [59].

Η αναδρομική δομή του RNN επιτρέπει την αναλογική εξέταση του παρελθόντος κατά την επεξεργασία εισόδου. Αυτό το καθιστά κατάλληλο για προβλήματα που απαιτούν αναγνώριση μοτίβων και ακολουθιών, καθώς οι προηγούμενες καταστάσεις επηρεάζουν την τρέχουσα κατάσταση και την παραγωγή εξόδου. Άρα, τα αναδρομικά δίκτυα σε αντίθεση με τα πρόσθιας τροφοδότησης, διαθέτουν τη δυνατότητα να υποστηρίξουν βραχυπρόθεσμη μνήμη [59]. Αυτό τα καθιστά πιο ενδιαφέροντα ως μοντέλα, καθώς μπορούν να αναγνωρίσουν προηγούμενες πληροφορίες κατά την επεξεργασία εισόδου. Ωστόσο, η παρουσία της αναδρομής καθιστά την αρχιτεκτονική τους πιο πολύπλοκη και δυσκολότερη στην κατανόηση και την ανάπτυξη. Η εκπαίδευσή τους επίσης, απαιτεί περαιτέρω μελέτη και εξοικείωση με τον τομέα της μηχανικής μάθησης. Στο Σχήμα 5.4 παρουσιάζεται ένα αναδρομικό νευρωνικό δίκτυο.



Σχήμα 5.4: Αναδρομικό νευρωνικό δίκτυο - RNN [38]

Ουσιαστικά υλοποιείται το παρακάτω σύστημα εξισώσεων [38]:

$$y_i = g\left(\sum_{j=1}^{4} a_{i,j}x_j + \sum_{j=1}^{3} w_{i,j}y_j + b_i\right), i = 1,2,3 \quad (5.3)$$

Όπου:

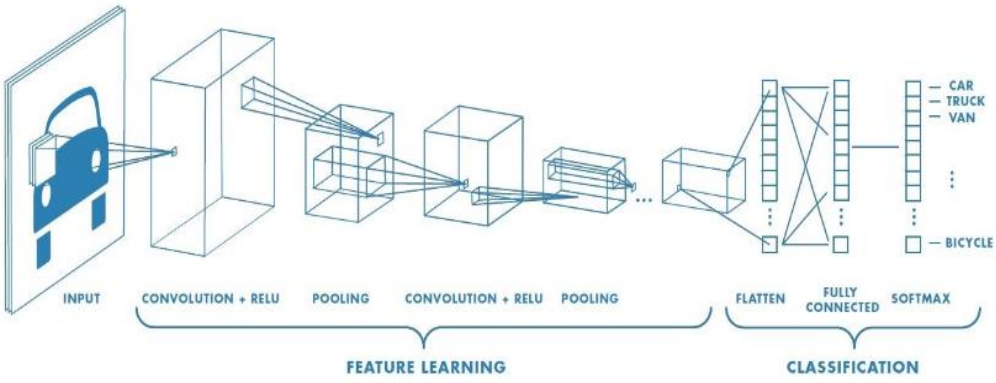
$a_{i,j}$ τα βάρη μεταξύ των εισόδων και των νευρώνων εξόδου, $w_{i,j}$ τα βάρη μεταξύ των νευρώνων εξόδου και b_i οι πολώσεις.

Ένα από τα πρώτα αναδρομικά νευρωνικά δίκτυα πολλών επιπέδων είναι το **μοντέλο του Jordan** το οποίο προτάθηκε το 1986 [38], [62]. Το μοντέλο αυτό περιλαμβάνει τέσσερα επίπεδα [38], [62]:

- Ένα επίπεδο εισόδου
- Ένα κρυφό επίπεδο
- Ένα επίπεδο εξόδου
- Ένα επίπεδο καθυστέρησης

Ένα ακόμη αναδρομικό μοντέλο είναι το λεγόμενο **απλό αναδρομικό μοντέλο** (Simple Recurrent Network - SRN), το οποίο προτάθηκε από τον Elman [38], [63]. Ουσιαστικά είναι μια παραλλαγή του Jordan με την ίδια δομή. Περισσότερα για το μοντέλο Perceptron, καθώς και για τα αναδρομικά μοντέλα Jordan και Elman αναφέρονται σε επόμενη ενότητα.

Μια άλλη κατηγορία νευρωνικών δικτύων είναι τα **συνελικτικά νευρωνικά δίκτυα** (convolutional neural networks - CNN), τα οποία είναι δίκτυα πολλών επιπέδων και προτάθηκαν το 1980 από τους Yann LeCun, Yoshua Bengio, Leon Bottou και Patrick Haffner [38], [64]. Τα εν λόγω νευρωνικά δίκτυα χρησιμοποιούνται συνήθως για εφαρμογές αναγνώρισης εικόνας, αναγνώρισης προσώπου, αυτόνομα οχήματα, σούπερ μάρκετ αυτοεξυπηρέτησης και έξυπνες ιατρικές θεραπείες [65]. Όσον αφορά την αρχιτεκτονική τους, αποτελούνται από εναλλασσόμενα επίπεδα συνέλιξης (convolution layers) και υποδειγματοληψίας (pooling layers). Μετά το τελευταίο επίπεδο (συνέλιξης ή υποδειγματοληψίας), ακολουθεί ένα πλήρως συνδεδεμένο επίπεδο το οποίο λειτουργεί ως ταξινομητής (π.χ. ένα επίπεδο Softmax) [38], [66]. Στην Εικόνα 5.2, φαίνεται η αρχιτεκτονική ενός συνελικτικού νευρωνικού δικτύου.



Εικόνα 5.2: Αρχιτεκτονική συνελικτικού νευρωνικού δικτύου – CNN [66]

Τέλος, μια ιδιαίτερη κατηγορία μοντέλων, είναι οι **μετασχηματιστές** (transformers). Αναφέρονται σε μια κατηγορία προηγμένων αρχιτεκτονικών νευρωνικών δικτύων που αναδείχθηκαν ως ένας σημαντικός στόχος στον τομέα της επεξεργασίας φυσικής γλώσσας (NLP) καθώς και άλλων κατηγοριών εργασιών μηχανικής μάθησης. Τα transformers κατέχουν εξαιρετικά αποτελέσματα σε πολλές εργασίες NLP και έχουν επιτύχει σημαντικές προόδους σε αυτόν τον τομέα.

Αποτελούν μια αρχιτεκτονική δικτύου στηριζόμενη στο μοντέλο κωδικοποιητή - αποκωδικοποιητή και βασίζεται στην ιδέα του **μηχανισμού προσοχής** (attention mechanism), ο οποίος επιτρέπει στο μοντέλο να εστιάζει σε διάφορα μέρη της εισόδου κατά την επεξεργασία [67], [68]. Αυτό επιτυγχάνεται μέσω της αλληλεπίδρασης μεταξύ διαφορετικών επιπέδων προσοχής (attention layers) στο μοντέλο.

Στην συνέχεια της έρευνάς μας, θα επικεντρωθούμε σε συγκεκριμένα μοντέλα πρόσθιας τροφοδότησης, αναδρομικής αρχιτεκτονικής καθώς και μοντέλα μετασχηματιστών. Δεν θα αναλύσουμε τα συνελικτικά νευρωνικά δίκτυα, καθώς αυτά χρησιμοποιούνται κυρίως για την ταξινόμηση εικόνων.

5.7.4.1 Μοντέλο πρόσθιας τροφοδότησης Perceptron

Όπως έχουμε αναφέρει, το μοντέλο **Perceptron** είναι μια απλή τοπολογία νευρωνικού δικτύου πρόσθιας τροφοδότησης του σήματος για ταξινόμηση δειγμάτων (πρότυπα), χωρίς κρυφά επίπεδα, το οποίο διαθέτει μόνο ένα τεχνητό νευρώνα [31], [38]. Σύμφωνα με τον Rosenblatt [69], το Perceptron υλοποιεί την σχέση (5.2) που αναφέραμε στην ενότητα 5.7.2, και χρησιμοποιεί ως συνάρτηση ενεργοποίησης την βηματική συνάρτηση της **δυαδικής** μορφής (0/1) ή της **διπολικής** μορφής (-1/1) [38].

Διπολική μορφή (-1/1):

$$g(S) = \begin{cases} -1 & \alpha\nu\ S \leq 0 \\ 1 & \alpha\nu\ S > 0 \end{cases}$$

Δυαδική μορφή (0/1):

$$g(S) = \begin{cases} 0 & \alpha\nu\ S \leq 0 \\ 1 & \alpha\nu\ S > 0 \end{cases}$$

Έτσι, αφού το Perceptron χρησιμοποιεί την βηματική συνάρτηση, μπορεί να αναπαραστήσει τις Boolean συναρτήσεις AND, OR και NOT αλλά δυστυχώς όχι την συνάρτηση XOR. Σύμφωνα με την σχέση (5.2), αν χρησιμοποιηθεί η βηματική συνάρτηση, τότε ο νευρώνας δίνει στην έξοδο τιμή 1, μόνο αν το σταθμισμένο άθροισμα των εισόδων του, μαζί με την πόλωση, είναι θετικό. Δηλαδή ισχύει:

$$\sum_{i=0}^{n} w_i x_i > 0 \tag{5.4}$$

Ή ισοδύναμα:

$$\boldsymbol{w} \cdot \boldsymbol{x} > 0 \tag{5.5}$$

Όπου $\boldsymbol{w}$ και $\boldsymbol{x}$, τα διανύσματα των βαρών και των εισόδων αντίστοιχα.

Η εξίσωση:

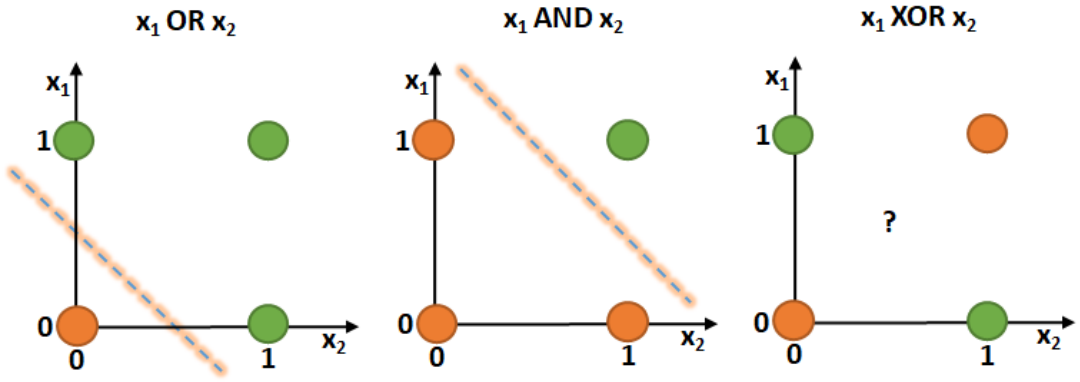
$$\boldsymbol{w} \cdot \boldsymbol{x} = 0 \tag{5.6}$$

ορίζει ένα **υπερεπίπεδο** (γραμμή) στο χώρο εισόδου. Έτσι, το Perceptron επιστρέφει τιμή 1, όταν η είσοδος βρίσκεται στη μια πλευρά του υπερεπιπέδου [59].

Αν ένα σύνολο δεδομένων μπορεί να διαχωριστεί με μια γραμμή, τότε το Perceptron μπορεί να εκπαιδευτεί να προβλέπει την κατηγορία των δεδομένων με βάση τη θέση τους σε σχέση με την γραμμή [31], [59]. Για τον λόγο αυτό το Perceptron θεωρείται ένας **γραμμικός διαχωριστής**.

Στο Σχήμα 5.5, φαίνονται οι αναπαραστάσεις του Perceptron για τις συναρτήσεις OR και AND με δύο εισόδους. Οι πορτοκαλί και οι πράσινοι κύκλοι αναπαριστούν σημεία στον χώρο, όπου η τιμή μιας συνάρτησης είναι 0 (ψευδής) και 1 (αληθής) αντίστοιχα. Το Perceptron μπορεί να αντιστοιχίσει αυτές τις συναρτήσεις επειδή υπάρχει μια ευθεία γραμμή που μπορεί να διαχωρίσει τους πορτοκαλί κύκλους από τους πράσινους. Επομένως, οι λογικές συναρτήσεις OR και AND είναι γραμμικά διαχωρίσιμες, καθώς μπορούν να αναπαρασταθούν από ένα Perceptron. Αντίθετα, η συνάρτηση XOR δεν είναι γραμμικά διαχωρίσιμη. Δεν υπάρχει μια ευθεία γραμμή που να μπορεί να διαχωρίσει σωστά τους πορτοκαλί από τους πράσινους κύκλους [59]. Αυτό σημαίνει ότι ένα απλό μοντέλο Perceptron δεν μπορεί να επιτύχει την ακριβή αναπαράσταση της λογικής συνάρτησης XOR. Για να αντιμετωπίσουμε την XOR, απαιτείται η χρήση πιο πολύπλοκων νευρωνικών δικτύων που περιλαμβάνουν κρυφά επίπεδα, δηλαδή νευρωνικά δίκτυα πολλών επιπέδων.

Σύμφωνα με τα παραπάνω, όπως έγινε φανερό, το μοντέλο Perceptron μπορεί να αναπαραστήσει μόνο **γραμμικά διαχωρίσιμες συναρτήσεις**.



Σχήμα 5.5: Γραμμικά διαχωρίσιμες συναρτήσεις OR και AND. Η XOR δεν είναι γραμμικά διαχωρίσιμη [59]

Εκπαίδευση Perceptron

Η εκπαίδευση στο μοντέλο είναι μια διαδικασία που βασίζεται στο **σφάλμα** (error driven). Αυτό σημαίνει ότι το μοντέλο προσπαθεί να υπολογίσει τα κατάλληλα βάρη έτσι ώστε, με δεδομένο ένα δυαδικό ή διπολικό διάνυσμα εισόδου, να παράγει την επιθυμητή έξοδο. Καθώς η επιθυμητή έξοδος χρησιμοποιείται ως καθοδήγηση για τον υπολογισμό των βαρών, πρόκειται για εκπαίδευση με επιτήρηση [31]. Στην αρχή της εκπαίδευσης, τα βάρη του μοντέλου ορίζονται τυχαία σε ένα διάστημα από το 0 έως το 1. Καθώς το μοντέλο εκπαιδεύεται, προσαρμόζει αυτά τα βάρη βάσει του σφάλματος που παράγεται μεταξύ της εξόδου πρόβλεψης και της επιθυμητής εξόδου. Δηλαδή, με κάθε επανάληψη της διαδικασίας εκπαίδευσης, τα βάρη αναπροσαρμόζονται για να **μειώσουν** το σφάλμα και να προσεγγίσουν την επιθυμητή έξοδο. Στην Εικόνα 5.3, φαίνεται ο αλγόριθμος εκπαίδευσης του Perceptron. Η μεταβλητή L, γνωστή και ως **ρυθμός μάθησης** (learning rate), είναι υπεύθυνη για τον **ρυθμό μεταβολής των βαρών** σε ένα μοντέλο μηχανικής μάθησης. Ο ρυθμός μάθησης καθορίζει πόσο γρήγορα ή αργά θα προσαρμοστούν τα βάρη κατά την εκπαίδευση. Η τιμή της μεταβλητής L κυμαίνεται από το 0 έως το 1, όπου το 0 σημαίνει απουσία μεταβολής και το 1 μέγιστος ρυθμός μεταβολής [31].

Ο ρυθμός μεταβολής εφαρμόζεται μόνο όταν η έξοδος y του μοντέλου είναι διαφορετική από την επιθυμητή έξοδο t, δηλαδή όταν υπάρχει σφάλμα. Σε αυτήν την περίπτωση, τα βάρη του μοντέλου που επηρεάζουν το σήμα εισόδου ($x \neq 0$) θα υποστούν μεταβολή. Αντίθετα, τα βάρη που δεν επηρεάζουν το σήμα εισόδου ή που αντιστοιχούν σε σήμα εισόδου μηδέν, δεν θα υποστούν αλλαγή αφού για $x = 0$, θα έχουμε $\Delta w = 0$. Επιπλέον, για $t \neq y$ και με βάση τις τιμές 0/1 των t και y ο ρυθμός μεταβολής των βαρών γίνεται [31]:

$$\Delta w = L \cdot x \quad \text{ή} \quad \Delta w = -L \cdot x$$

Συνεπώς, αν το μοντέλο ταξινομεί λανθασμένα ένα πρότυπο, ο ρυθμός μεταβολής θα διορθώσει τα βάρη με τέτοιο τρόπο, ώστε την επόμενη φορά να ταξινομηθεί σωστά ή να πλησιάσει περισσότερο στη σωστή ταξινόμηση. Με κάθε επανάληψη της διαδικασίας εκπαίδευσης, ο ρυθμός μεταβολής επιτρέπει την προσαρμογή των βαρών, οδηγώντας σε σταδιακή βελτίωση της απόδοσης του μοντέλου. [38].

Μέχρι να γίνει αληθής η συνθήκη τερματισμού της εκπαίδευσης (μέχρι να σταματήσει η μεταβολή των βαρών ή να συμπληρωθεί ο αριθμός εποχών), κάνε τα παρακάτω:

Για κάθε ζευγάρι εισόδου x και εξόδου t του συνόλου εκπαίδευσης…

- Υπολόγισε την έξοδο y
- Αν $y = t$ τότε δεν γίνεται κάποια μεταβολή στα βάρη
- Αν $y \neq t$ τότε κάνε μεταβολή στα βάρη των εισόδων με σήμα $\neq 0$, σύμφωνα με τον τύπο:

$$\Delta w = L \cdot (t - y) \cdot x$$

προκειμένου το y να πλησιάσει το t

Εικόνα 5.3: Αλγόριθμος εκπαίδευσης Perceptron [31]

Η έννοια της εποχής (epoch)

Αναφέρεται σε έναν πλήρη κύκλο χρήσης των προτύπων – δειγμάτων [38], [59]. Για παράδειγμα, Αν το σύνολο των δεδομένων αποτελείται από 1000 δείγματα και εκπαιδεύουμε το δίκτυο για 10 εποχές, τότε σημαίνει ότι έχουμε προωθήσει τα 1000 δείγματα μέσα στο δίκτυο 10 φορές.

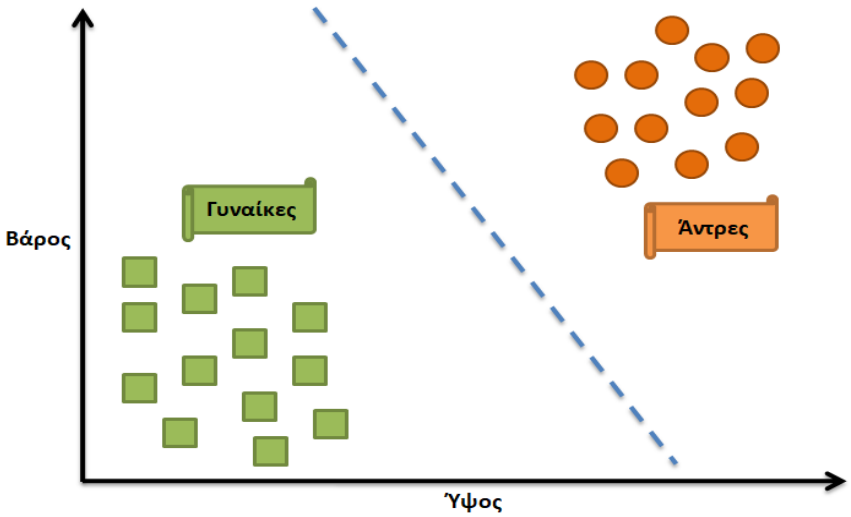
Εφαρμογή ταξινόμησης με Perceptron

Ένα απλό πρόβλημα ταξινόμησης με Perceptron μπορεί να είναι η αναγνώριση του φύλου (άντρας ή γυναίκα) με βάση τα χαρακτηριστικά βάρος και ύψος.

Οπότε, μπορούμε να εκπαιδεύσουμε ένα Perceptron να ταξινομεί τα άτομα ως άντρας (κλάση 1) ή γυναίκα (κλάση 0) ως εξής:

- Αν το ύψος του ατόμου είναι μεγαλύτερο από ένα κατώφλι-όριο, π.χ. 175 εκατοστά, και το βάρος είναι μεγαλύτερο από ένα άλλο κατώφλι, π.χ. 65 κιλά, τότε το άτομο ταξινομείται ως άντρας (κλάση 1).
- Διαφορετικά, αν το άτομο έχει ύψος μικρότερο ή ίσο με το κατώφλι και βάρος μικρότερο ή ίσο με το άλλο κατώφλι, τότε το άτομο ταξινομείται ως γυναίκα (κλάση 0).

Αν υπάρχει μια ευθεία γραμμή που διαχωρίζει τις δύο κλάσεις, τότε οι κλάσεις είναι **γραμμικά διαχωρίσιμες**. Στο Σχήμα 5.6, φαίνεται ο διαχωρισμός των δύο κλάσεων του προβλήματος.



Σχήμα 5.6: Ταξινόμηση γυναίκα / άντρα

5.7.4.2 Perceptron πολλών επιπέδων (Multilayer Perceptron - MLP)

Στην προηγούμενη ενότητα διαπιστώσαμε την αδυναμία του μοντέλου Perceptron να λύσει μη γραμμικά διαχωρίσιμα προβλήματα όπως είναι η πύλη XOR.

Στο Σχήμα 5.5, της προηγούμενης ενότητας, στην περίπτωση της πύλης XOR έχουμε δύο εισόδους που μπορούν να είναι 0 ή 1, και τον στόχο να παράγει μία έξοδο που είναι 1 (πράσινοι κύκλοι) μόνο όταν οι δύο είσοδοι διαφέρουν. Αν προσπαθήσουμε να εκπαιδεύσουμε ένα μοντέλο Perceptron για να

διαχωρίσουμε τους πορτοκαλί από τους πράσινους κύκλους θα διαπιστώσουμε πως δεν υπάρχει τέτοια ευθεία γραμμή για να το επιτύχουμε. Γενικά, για την επίλυση του προβλήματος XOR καθώς και για άλλα μη γραμμικά προβλήματα, όπως αναφέραμε, χρησιμοποιούνται πιο προηγμένες μέθοδοι ταξινόμησης, όπως νευρωνικά δίκτυα με πολλαπλά επίπεδα.

Έτσι, οι περιορισμοί του Perceptron του ενός νευρώνα, οδήγησε στην ανάπτυξη του μοντέλου **Perceptron με πολλά επίπεδα** [38].

Υλοποίηση πύλης XOR με Perceptron 2 επιπέδων

Για την υλοποίηση χρειαζόμαστε ένα Perceptron με 3 νευρώνες οργανωμένους σε 2 επίπεδα (Σχήμα 5.7). Αν και υπάρχει επίπεδο εισόδων, αυτό δεν θεωρείται ως ένα επίπεδο νευρώνων αφού οι νευρώνες του απλώς μεταφέρουν τιμές στο αμέσως επόμενο επίπεδο. Οπότε τα 2 επίπεδα είναι τα εξής:

- Ένα κρυφό επίπεδο με τους νευρώνες 1 και 2, με έξοδο a_1 και a_2 αντίστοιχα.
- Ένα επίπεδο εξόδου που έχει τον νευρώνα 3 με έξοδο y.

Επίσης, έχουμε τα βάρη των νευρώνων 1, 2, 3 (τιμές δίπλα στις συνδέσεις - ακμές), και τις πολώσεις τους w_0.

Η πύλη OR υλοποιείται από τον νευρώνα 1. Χρησιμοποιείται η **βηματική συνάρτηση** με κατώφλι $T = 0.5$. Οπότε για $x_1 = 0$ και $x_2 = 0$, θα έχουμε:

$$S = x_1 \cdot 1 + x_2 \cdot 1 = 0$$

$$S < 0.5$$

Οπότε είναι:

$$a_1 = g(S) \Rightarrow a_1 = 0$$

για $x_1 = 0$ και $x_2 = 1$, θα έχουμε:

$$S = x_1 \cdot 1 + x_2 \cdot 1 = 0 + 1 = 1$$

$$S > 0.5$$

Οπότε είναι:

$$a_1 = g(S) \Rightarrow a_1 = 1$$

για $x_1 = 1$ και $x_2 = 0$, θα έχουμε:

$$S = x_1 \cdot 1 + x_2 \cdot 1 = 1 + 0 = 1$$

$$S > 0.5$$

Οπότε είναι:

$$a_1 = g(S) \Rightarrow a_1 = 1$$

Και τέλος, για $x_1 = 1$ και $x_2 = 1$, θα έχουμε:

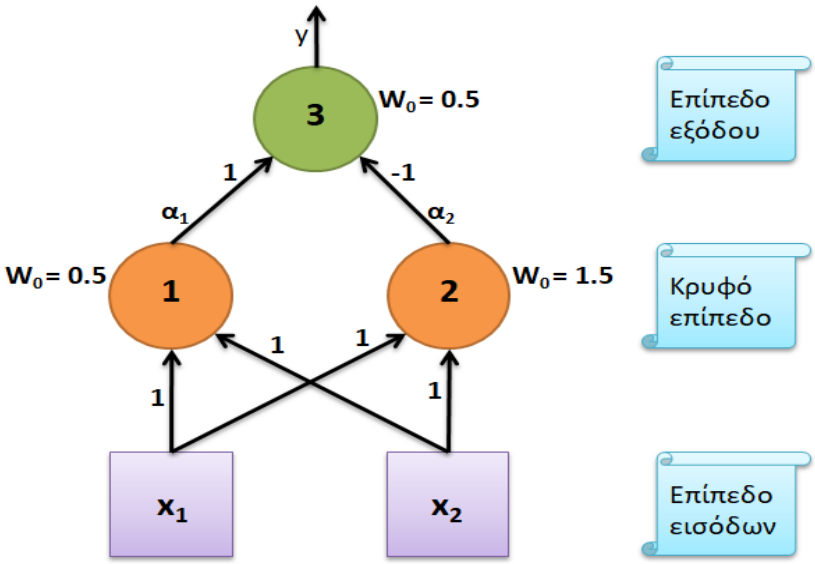
$$S = x_1 \cdot 1 + x_2 \cdot 1 = 1 + 1 = 2$$

$$S > 0.5$$

Οπότε είναι:

$$a_1 = g(S) \Rightarrow a_1 = 1$$

Με παρόμοιο τρόπο υλοποιείται και η πύλη AND από τον νευρώνα 2, με **βηματική συνάρτηση** και κατώφλι $T = 1.5$.



Σχήμα 5.7: Υλοποίηση πύλης XOR

Ο Πίνακας 5.1 παρακάτω δείχνει την σύνοψη των πυλών OR και AND.

Πίνακας 5.1: Πύλες OR και AND

x_1	x_2	a_1 OR	a_2 AND
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

Η πύλη XOR υλοποιείται από τον νευρώνα 3. Χρησιμοποιείται η **βηματική συνάρτηση** με κατώφλι $T = 0.5$.

Για $\alpha_1 = 0$ και $\alpha_2 = 0$, θα έχουμε:

$$S = \alpha_1 \cdot 1 + \alpha_2 \cdot (-1) = 0 + 0 = 0$$

$$S < 0.5$$

Οπότε είναι:

$$y = g(S) \Rightarrow y = 0$$

Για $\alpha_1 = 1$ και $\alpha_2 = 0$, θα έχουμε:

$$S = \alpha_1 \cdot 1 + \alpha_2 \cdot (-1) = 1 + 0 = 1$$

$$S > 0.5$$

Οπότε είναι:

$$y = g(S) \Rightarrow y = 1$$

Για $\alpha_1 = 1$ και $\alpha_2 = 1$, θα έχουμε:

$$S = \alpha_1 \cdot 1 + \alpha_2 \cdot (-1) = 1 + (-1) = 0$$

$$S < 0.5$$

Οπότε είναι:

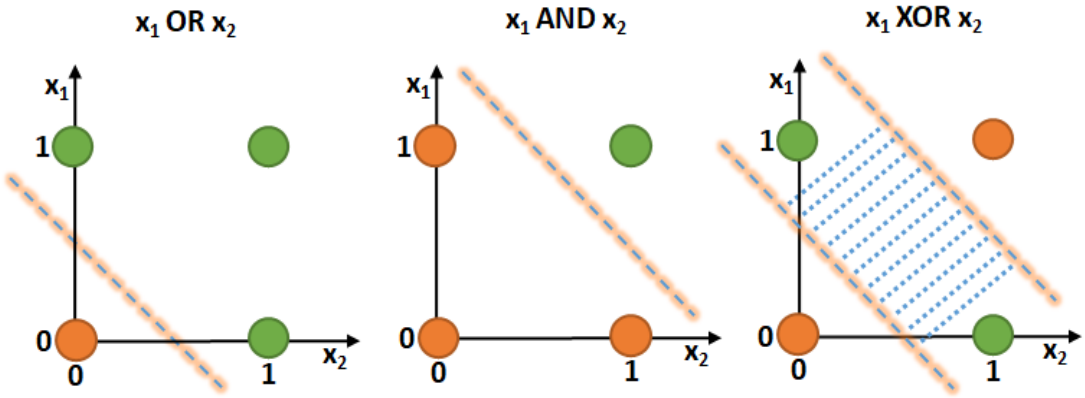
$$y = g(S) \Rightarrow y = 0$$

Ο Πίνακας 5.2 δείχνει την τελική σύνοψη των πυλών, συμπεριλαμβανομένης και της πύλης XOR.

Πίνακας 5.2: Πύλες OR, AND, και XOR

x_1	x_2	a_1 OR	a_2 AND	y XOR
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0

Συμπερασματικά, το επίπεδο εξόδου του ενός νευρώνα 3, συνδυάζει τις εξόδους του κρυφού επιπέδου και έτσι προκύπτει η συνάρτηση XOR (Σχήμα 5.8).



Σχήμα 5.8: Συνδυάζοντας τις ευθείες OR και AND, παίρνουμε την πύλη XOR

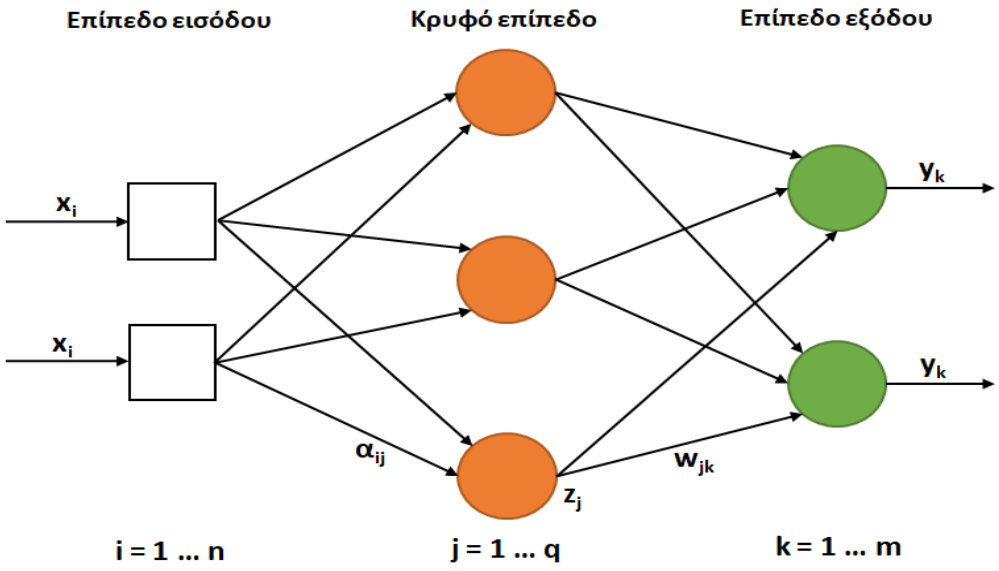
Αρχιτεκτονική MLP

Το βασικότερο χαρακτηριστικό των δικτύων MLP, είναι ότι κάθε νευρώνας σε ένα επίπεδο τροφοδοτεί μόνο τους νευρώνες του επόμενου επιπέδου και λαμβάνει εισόδους μόνο από τους νευρώνες του προηγούμενου επιπέδου. Αυτό σημαίνει ότι η πληροφορία ρέει από το πρώτο επίπεδο του δικτύου προς το τελευταίο, με κάθε επίπεδο νευρώνων να επεξεργάζεται και να μετατρέπει την είσοδο προτού την μεταβιβάσει στο επόμενο επίπεδο. Έτσι, τα δίκτυα MLP, ανήκουν στην κατηγορία **δικτύων πρόσθιας τροφοδότησης** του σήματος, εφόσον η ροή της πληροφορίας είναι μονής κατεύθυνσης [31], [38], [59].

Επομένως, τα δίκτυα MLP είναι ικανά να χειριστούν μη γραμμικές λειτουργίες και να επιλύσουν προβλήματα που περιλαμβάνουν περίπλοκες αποφάσεις. Αυτό επιτυγχάνεται μέσω της εφαρμογής μη γραμμικών συναρτήσεων ενεργοποίησης σε κάθε επίπεδο νευρώνων, επιτρέποντας την μάθηση και αναπαράσταση πιο πολύπλοκων δομών.

Όσον αφορά τις συναρτήσεις ενεργοποίησης, συνήθως αποφεύγεται η χρήση της βηματικής συνάρτησης καθώς **δεν είναι παραγωγίσιμη**. Αντί για αυτήν, συνήθως χρησιμοποιούνται άλλες συναρτήσεις ενεργοποίησης όπως η σιγμοειδής ή η υπερβολική εφαπτομένη, οι οποίες είναι παραγωγίσιμες. Οι περισσότεροι κανόνες εκπαίδευσης στα MLP, βασίζονται σε μεθόδους βελτιστοποίησης, όπως η μέθοδος της **βαθμωτής κατάβασης**. Αυτοί οι κανόνες εκπαίδευσης απαιτούν την ύπαρξη παραγώγων των συναρτήσεων ενεργοποίησης για τον υπολογισμό της μεταβολής των βαρών του δικτύου κατά τη διαδικασία της εκπαίδευσης [38], [59].

Έχει αποδειχθεί ότι δίκτυα MLP με **δύο επίπεδα** και έναν επαρκή αριθμό κρυφών νευρώνων μπορούν να προσεγγίσουν οποιαδήποτε συνάρτηση σε οποιονδήποτε επιθυμητό βαθμό ακρίβειας. Γι' αυτό τέτοιας δομής δίκτυα λέγονται και **καθολικοί προσεγγιστές** [31], [38], [70], [71]. Αυτή η ικανότητα των δικτύων MLP να προσεγγίζουν οποιαδήποτε συνάρτηση, καθιστά την αρχιτεκτονική MLP ιδιαίτερα ισχυρή και ευέλικτη στην αντιμετώπιση ποικίλων προβλημάτων μηχανικής μάθησης και αναγνώρισης προτύπων. Το Σχήμα 5.9 παρουσιάζει την αρχιτεκτονική ενός δικτύου MLP.



Σχήμα 5.9: Αρχιτεκτονική δικτύου MLP [31]

Εκπαίδευση MLP

Όπως στα απλά δίκτυα Perceptron έτσι και στα δίκτυα MLP, στόχος είναι η διαδικασία προσαρμογής των βαρών τους, προκειμένου να παραχθεί η επιθυμητή έξοδος. Μια βελτίωση του αλγορίθμου μάθησης Perceptron, ήταν ο κανόνας **Δέλτα** (Delta rule) που αναπτύχθηκε το 1960 από τους Marcian Hoff και Bernard Widrow [31]. Αν και εφαρμόστηκε σε δίκτυα Perceptron, το μειονέκτημά του ήταν ότι δεν μπορούσε να εφαρμοστεί σε δίκτυα πολλών επιπέδων, διότι δεν ήταν γνωστή η επιθυμητή έξοδος στους κρυφούς νευρώνες, αλλά μόνο σε αυτούς του επιπέδου εξόδου. Αυτό είχε ως συνέπεια την αναζήτηση πιο αποδοτικών τεχνικών [31].

Για πολλά χρόνια οι ερευνητές στον τομέα της τεχνητής νοημοσύνης δεν μπορούσαν να βρουν έναν μαθηματικό τρόπο ο οποίος να προωθεί τις μεταβολές των βαρών από επίπεδο σε επίπεδο. Ως αποτέλεσμα, η έρευνα επικεντρώθηκε κυρίως στην ανάπτυξη συμβολικών μεθόδων, όπου οι

αλγόριθμοι είχαν ορισμένους προκαθορισμένους κανόνες για την λήψη αποφάσεων. Η λύση βέβαια ήρθε 10 χρόνια αργότερα με τον αλγόριθμο **Back-Propagation** ο οποίος προτάθηκε από τον Paul Werbos το 1970 [31], [38], [72].

Ο **Back-Propagation** είναι ο πιο διαδεδομένος αλγόριθμος για την εκπαίδευση νευρωνικών δικτύων πολλών επιπέδων (MLP). Βασίζεται σε έναν κανόνα, γνωστό ως "γενικευμένο κανόνα Δέλτα", ο οποίος χρησιμοποιείται για τον υπολογισμό του ποσοστού του συνολικού σφάλματος που σχετίζεται με τα βάρη κάθε νευρώνα, συμπεριλαμβανομένων αυτών που ανήκουν σε κρυφά επίπεδα. Αυτό δίνει τη δυνατότητα να υπολογιστούν οι διορθώσεις που απαιτούνται για την ενημέρωση των βαρών του κάθε νευρώνα ξεχωριστά [31], [38].

Με βάση το Σχήμα 5.9, θα δείξουμε τον τρόπο εκπαίδευσης του MLP, δηλαδή τον τρόπο που αναπροσαρμόζονται τα βάρη των επιπέδων του δικτύου. Γίνεται η παραδοχή χρήσης της σιγμοειδούς συνάρτησης ως συνάρτηση ενεργοποίησης, καθώς ο αλγόριθμος Back-Propagation έχει αυτή την απαίτηση.

Κατά την εκκίνηση της διαδικασίας εκπαίδευσης εισάγονται τα διανύσματα εκπαίδευσης στην είσοδο του δικτύου. Οι νευρώνες στο επίπεδο εισόδου δεν κάνουν κάποιο υπολογισμό, η αρμοδιότητά τους είναι μόνο να μεταφέρουν την είσοδο στο κρυφό επίπεδο. Οι νευρώνες του κρυφού επιπέδου αφού λάβουν την είσοδο, χρησιμοποιούν την σιγμοειδή συνάρτηση για να παραγάγουν την έξοδο η οποία προωθείται στο επίπεδο εξόδου. Οι νευρώνες του επιπέδου εξόδου επεξεργάζονται την είσοδό τους και χρησιμοποιούν την δική τους σιγμοειδή συνάρτηση για να παραγάγουν την τελική έξοδο του δικτύου [31], [59].

Έτσι, η **είσοδος** ενός νευρώνα j του κρυφού επιπέδου δίνεται από τον τύπο:

$$in_j = \sum_{i=1}^{n} \alpha_{ij} x_i \tag{5.7}$$

Όπου α_{ij} το βάρος της σύνδεσης από τον νευρώνα i στον j. Το x_i είναι η είσοδος του νευρώνα i.

Η **έξοδος** του νευρώνα jείναι:

$$z_j = g(in_j) = g\left(\sum_{i=1}^{n} \alpha_{ij} x_i\right) \tag{5.8}$$

Έπειτα, η έξοδος z_j προωθείται σε όλους τους νευρώνες του αμέσως επόμενου επιπέδου (επίπεδο εξόδου στο παράδειγμά μας).

Η **είσοδος** ενός νευρώνα k, του επιπέδου εξόδου δίνεται απο τον τύπο:

$$in_k = \sum_{j=1}^{q} w_{jk} z_j \tag{5.9}$$

Η **έξοδος** του νευρώνα k είναι:

$$y_k = g(in_k) = g\left(\sum_{j=1}^{q} w_{jk} z_j\right) \tag{5.10}$$

Είναι αξιοσημείωτο ότι, τα αρχικά βάρη του δικτύου αρχικοποιούνται τυχαία και αλλάζουν με σκοπό τον περιορισμό του σφάλματος. Επειδή μόνο οι νευρώνες εξόδου γνωρίζουν την επιθυμητή έξοδο, μπορεί να χρησιμοποιηθεί ο γενικευμένος κανόνας Δέλτα προκειμένου να μεταβληθούν τα βάρη του επιπέδου εξόδου. Οπότε ισχύει:

$$\Delta w_{jk} = L \cdot \delta_k \cdot z_j \tag{5.11}$$

Όπου L είναι ο ρυθμός μάθησης, δ_k ο ρυθμός μεταβολής του σφάλματος του νευρώνα k ή διόρθωση του νευρώνα k, και z_j είναι η έξοδος του νευρώνα j.

Έτσι, ο ρυθμός μεταβολής δ_k του **επιπέδου εξόδου** υπολογίζεται από τον τύπο:

$$\delta_k = (t_k - y_k) \cdot g'(in_k) \tag{5.12}$$

Όπου t_k η επιθυμητή έξοδος ενός νευρώνα k και y_k η έξοδος ενός νευρώνα k. Επίσης, χρησιμοποιείται και η παράγωγος της συνάρτησης ενεργοποίησης η οποία δέχεται την είσοδο ενός νευρώνα k.

Επίσης, όσον αφορά το **κρυφό επίπεδο** πρέπει να υπολογιστεί το δ_j, δηλαδή:

$$\delta_j = \sum_{k=1}^{m} \delta_k w_{jk} \cdot g'(in_j) \tag{5.13}$$

Τώρα, εφόσον είναι γνωστό το δ_j για έναν νευρώνα j, μπορεί να χρησιμοποιηθεί για την ενημέρωση των βαρών του κρυφού επιπέδου:

$$\Delta \alpha_{ij} = L \cdot \delta_j \cdot x_i \tag{5.14}$$

Στην παραπάνω περιγραφή ουσιαστικά δείξαμε πως ενημερώνονται τα βάρη που συνδέουν το επίπεδο εξόδου με το κρυφό, καθώς και την ενημέρωση των βαρών μεταξύ κρυφού και επιπέδου εισόδου. Αυτή η διαδικασία ενημέρωσης βαρών ονομάζεται **ανάστροφη μετάδοση** (back propagation) [31].

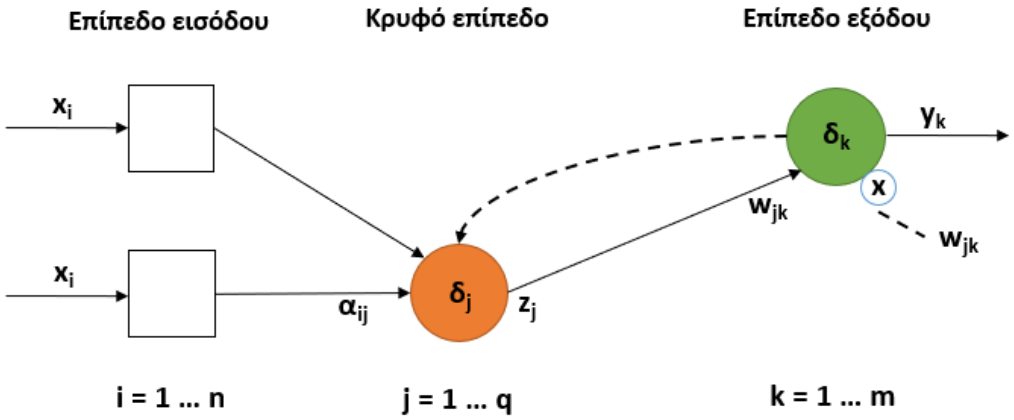
Ο αλγόριθμος Back-Propagation αντιπροσωπεύει μια τεχνική βελτιστοποίησης με τη μέθοδο της επικλινούς καθόδου, η οποία στοχεύει στην ελαχιστοποίηση του μέσου τετραγωνικού σφάλματος

μεταξύ της εξόδου του δικτύου και της επιθυμητής εξόδου για κάθε πρότυπο εκπαίδευσης. Το **μέσο τετραγωνικό σφάλμα** αποτελεί ένα κλασικό μέτρο κόστους και χρησιμοποιείται συχνά σε πολλά προβλήματα. Η εν λόγω μέθοδος ελαχιστοποίησης ουσιαστικά αναζητά το ολικό ελάχιστο του τετραγωνικού σφάλματος, με τα βάρη του δικτύου να λειτουργούν ως παράμετροι που αναπροσαρμόζονται. Ωστόσο, κατά τη διαδικασία αναζήτησης, υπάρχουν περιπτώσεις όπου το δίκτυο μπορεί να "παγιδευτεί" σε τοπικά ελάχιστα, περιορίζοντας έτσι την ικανότητά του να επιτύχει την βέλτιστη απόδοση [31], [38].

Το μέσο τετραγωνικό σφάλμα δίνεται απο τον τύπο:

$$J = \frac{1}{N}\sum_{p=1}^{N}\sum_{k=1}^{m}\left(t_k^{(p)} - y_k^{(p)}\right)^2 \tag{5.15}$$

Στο Σχήμα 5.10 φαίνεται πως για να υπολογίσουμε την διόρθωση δ_j του νευρώνα j του κρυφού επιπέδου, όλα τα σφάλματα του επιπέδου εξόδου μεταδίδονται πίσω προς τον νευρώνα j. Γίνεται δηλαδή ανάστροφη μετάδοση (back propagation).



Σχήμα 5.10: Ανάστροφη μετάδοση σφαλμάτων

5.7.4.3 Αναδρομικό μοντέλο του Jordan

Όπως αναφέραμε και σε προηγούμενη ενότητα, το μοντέλο του Jordan αποτελεί ένα από τα πρώτα αναδρομικά νευρωνικά δίκτυα πολλών επιπέδων νευρώνων. Η αρχιτεκτονική του δικτύου έχει ως εξής [38], [62]:

- Επίπεδο εισόδου
- Επίπεδο καθυστέρησης ή κατάστασης
- Κρυφό επίπεδο
- Επίπεδο εξόδου

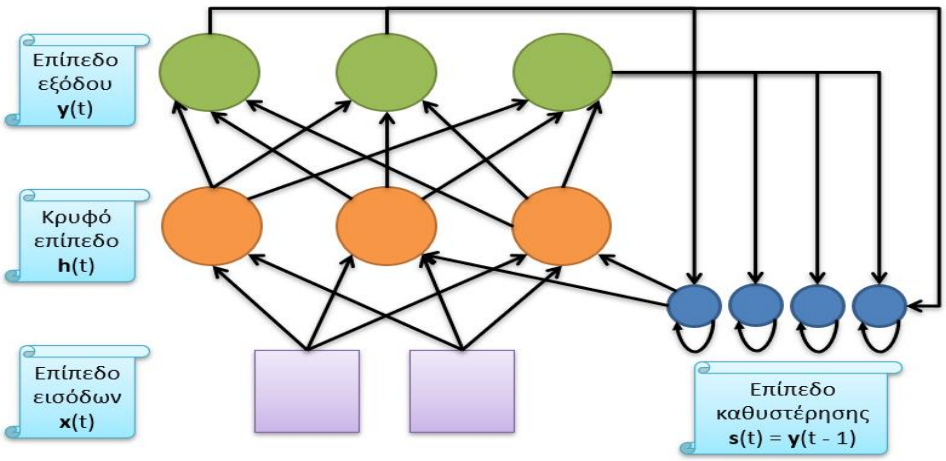
Βασικό χαρακτηριστικό του δικτύου είναι ότι, το επίπεδο εξόδου ανατροφοδοτεί το κρυφό επίπεδο μέσω του επιπέδου καθυστέρησης. Το επίπεδο καθυστέρησης αποτελείται από νευρώνες των οποίων

η έξοδος γίνεται είσοδος στον εαυτό τους. Με αυτό τον τρόπο δημιουργούν μια χρονική καθυστέρηση η οποία δίνεται από την σχέση:

$$\boldsymbol{s}(t) = \boldsymbol{y}(t-1) \tag{5.16}$$

Συνεπώς, η παραπάνω σχέση υποδεικνύει ότι η έξοδος του επιπέδου καθυστέρησης στον τρέχοντα χρόνο t είναι η ίδια με την έξοδο του δικτύου στον προηγούμενο χρόνο $t-1$. Γενικά, οι νευρώνες του επιπέδου καθυστέρησης διατηρούν ένα συνεχές ιστορικό των προηγούμενων χρονικών βημάτων. Όταν περνάει κάποιο χρονικό διάστημα, μεταδίδουν την έξοδό τους στο κρυφό επίπεδο του δικτύου, παρέχοντας έτσι πληροφορίες από προηγούμενες χρονικές περιόδους. Αυτό τους επιτρέπει να αποθηκεύουν και να ανακαλούν πληροφορίες από το παρελθόν και να διατηρούν κάποιο είδος μνήμης για την ακολουθία των εισόδων που έχουν λάβει [38], [62].

Όσον αφορά τις συναρτήσεις ενεργοποίησης, συνήθως χρησιμοποιείται η βηματική συνάρτηση στο επίπεδο εξόδου και στο κρυφό, ενώ το επίπεδο καθυστέρησης χρησιμοποιεί γραμμική συνάρτηση [62]. Παρακάτω, στο Σχήμα 5.11 φαίνεται η αρχιτεκτονική του μοντέλου (δεν έχουν συμπεριληφθεί όλες οι συνδέσεις - ακμές), καθώς και η μαθηματική ερμηνεία του.



Σχήμα 5.11: Αναδρομικό μοντέλο Jordan [38], [62]

Μαθηματική περιγραφή μοντέλου

Για κρυφό επίπεδο:

$$\boldsymbol{h}(t) = f(\boldsymbol{W}_{xh} \cdot \boldsymbol{x}(t) + \boldsymbol{W}_{sh} \cdot \boldsymbol{y}(t-1) + \boldsymbol{b}) \tag{5.17}$$

Για επίπεδο εξόδου:

$$\boldsymbol{y}(t) = g\left(\boldsymbol{W}_{hy} \cdot \boldsymbol{h}(t)\right) \tag{5.18}$$

Όπου:

$\boldsymbol{x}(t)$ είναι το επίπεδο εισόδου.

$\boldsymbol{h}(t)$ είναι το κρυφό επίπεδο.

$\boldsymbol{y}(t)$ είναι το επίπεδο εξόδου.

$\boldsymbol{W}_{xh}$ είναι τα βάρη μεταξύ του επιπέδου εισόδου με το κρυφό.

$\boldsymbol{W}_{sh}$ είναι τα βάρη μεταξύ επιπέδου καθυστέρησης και κρυφού.

$\boldsymbol{W}_{hy}$ τα βάρη μεταξύ κρυφού και επιπέδου εξόδου.

$\boldsymbol{b}$ είναι το διάνυσμα πολώσεων του κρυφού επιπέδου.

f είναι η συνάρτηση ενεργοποίησης του κρυφού επιπέδου.

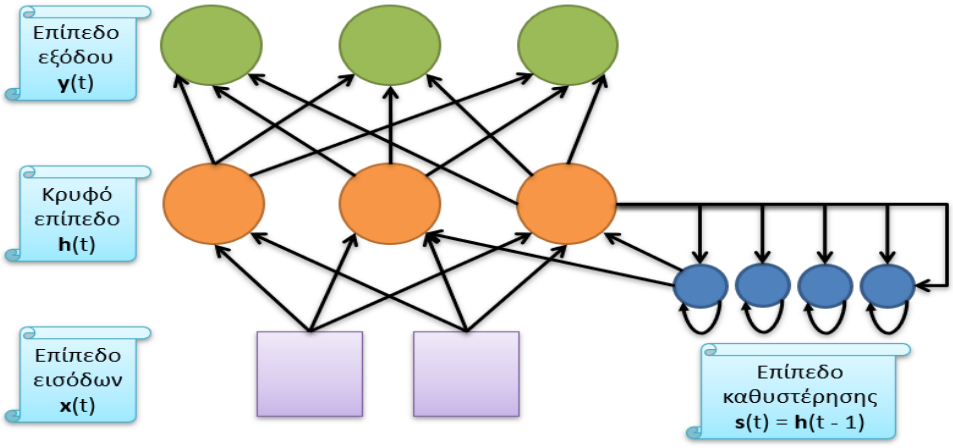
g συνάρτηση ενεργοποίησης του επιπέδου εξόδου.

5.7.4.4 Αναδρομικό μοντέλο του Elman

Το νευρωνικό μοντέλο Elman προτάθηκε από τον Jeffrey Elman το 1990. Είναι παρόμοιας αρχιτεκτονικής με το μοντέλο Jordan, απλά σε αυτό, το κρυφό επίπεδο ανατροφοδοτεί το επίπεδο καθυστέρησης και έπειτα το επίπεδο καθυστέρησης ανατροφοδοτεί το κρυφό επίπεδο. Όπως και στο Jordan, ο κάθε νευρώνας του επιπέδου καθυστέρησης συνδέεται αναδρομικά με τον εαυτό του και έτσι δημιουργεί την καθυστέρηση [38], [63]:

$$\boldsymbol{s}(t) = \boldsymbol{h}(t-1) \tag{5.19}$$

Το Σχήμα 5.12 δείχνει την αρχιτεκτονική του μοντέλου Elman.



Σχήμα 5.12: Αναδρομικό μοντέλο Elman [38], [63]

Μαθηματική περιγραφή μοντέλου

Για κρυφό επίπεδο:

$$\boldsymbol{h}(t) = f(\boldsymbol{W}_{xh} \cdot \boldsymbol{x}(t) + \boldsymbol{W}_{sh} \cdot \boldsymbol{h}(t-1) + \boldsymbol{b}) \tag{5.20}$$

Για επίπεδο εξόδου:

$$\boldsymbol{y}(t) = g\left(\boldsymbol{W}_{hy} \cdot \boldsymbol{h}(t)\right) \tag{5.21}$$

Μέθοδος εκπαίδευσης BPTT (Back – Propagation Through Time)

Η μέθοδος εκπαίδευσης BPTT (Back-Propagation Through Time) χρησιμοποιείται για την εκπαίδευση αναδρομικών νευρωνικών δικτύων. Η ιδέα πίσω από τη μέθοδο, είναι να εφαρμοστεί ο αλγόριθμος της ανάστροφης μετάδοσης (back-propagation) σε ένα RNN με διάρκεια χρόνου, ώστε να ενημερωθούν οι παράμετροι του δικτύου. Αυτό ουσιαστικά σημαίνει ότι, σε κάθε χρονική επανάληψη υπολογίζονται οι παράγωγοι της συνάρτησης κόστους και ενημερώνονται οι παράμετροι του δικτύου. Ως συνάρτηση κόστους μπορεί να χρησιμοποιηθεί οποιοδήποτε κριτήριο κόστους που χρησιμοποιεί και ο κλασσικός αλγόριθμος Back-Propagation, όπως το μέσο τετραγωνικό σφάλμα. Επειδή οι παράγωγοι υπολογίζονται σε κάθε κύκλο επανάληψης, η μέθοδος μπορεί να είναι απαιτητική από πλευράς υπολογιστικής ισχύος και μνήμης [38]. Ωστόσο, η χρήση της BPTT έχει αποδειχθεί αποτελεσματική για την εκπαίδευση αναδρομικών νευρωνικών δικτύων και έχει επιτρέψει την ανάπτυξη πολλών σημαντικών εφαρμογών.

Σημαντικό χαρακτηριστικό στα αναδρομικά δίκτυα είναι το γεγονός ότι, ένα επίπεδο νευρώνων που χρησιμοποιεί μια συνάρτηση ενεργοποίησης, ουσιαστικά αποτελεί ένα δίκτυο πολλών επιπέδων, με αποτέλεσμα την επανάληψη των ίδιων βαρών σε κάθε επίπεδο, και εφόσον κάθε επίπεδο αντιστοιχεί σε μια χρονική στιγμή, προκύπτει επανάληψη ίδιων βαρών μεταξύ διαφορετικών χρόνων [38].

Τέλος, είναι δυνατό να υπάρχουν στόχοι $\boldsymbol{a}(1), \boldsymbol{a}(2), \ldots, \boldsymbol{a}(T)$ για τις εξόδους εσωτερικών επιπέδων $\boldsymbol{y}(1), \boldsymbol{y}(2), \ldots, \boldsymbol{y}(T)$ [38].

Στην Εικόνα 5.4 φαίνεται ο αλγόριθμος BPTT και στο Σχήμα 5.13 η διαμόρφωση του αναδρομικού δικτύου Elman σε κάθε χρονική επανάληψη.

Είσοδος:

Εισαγωγή εισόδων $\boldsymbol{x}(1), \boldsymbol{x}(2), \boldsymbol{x}(3), \dots, \boldsymbol{x}(T)$ και στόχων $\boldsymbol{a}(1), \boldsymbol{a}(2), \boldsymbol{a}(3), \dots, \boldsymbol{a}(T)$.

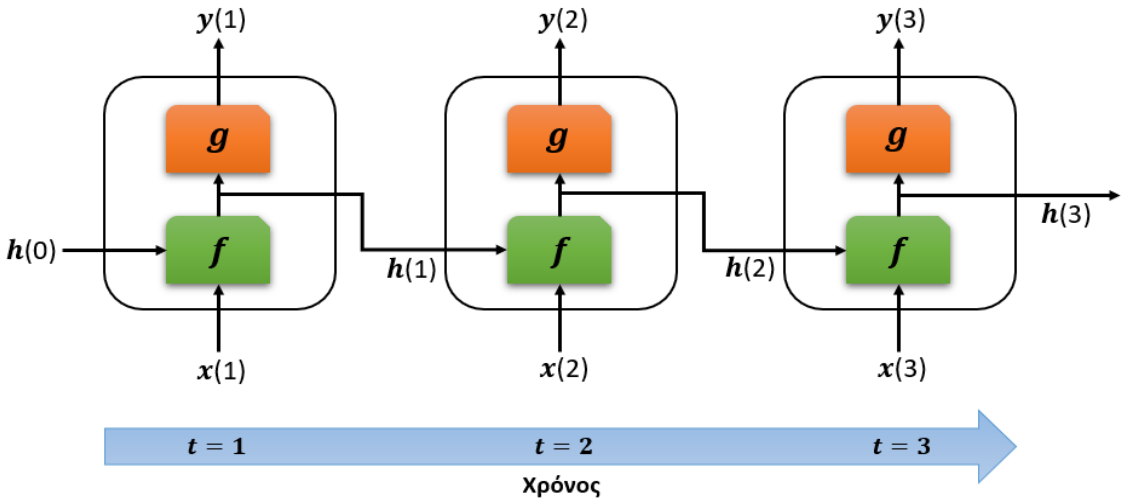
Έξοδος:

Τα βάρη $\boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{w}_2, \dots, \boldsymbol{w}_t$.

Μέθοδος:

```
Για κάθε εποχή από 1 μέχρι Max_Epochs {
  Για t από 1 μέχρι T {
     Αρχικοποίηση καταστάσεων h(t)
     Υπολογισμός ενδιάμεσων εξόδων y(t)
     Υπολογισμός σφάλματος E
     Ανάστροφη μετάδοση σφάλματος
     Ενημέρωση βαρών
  }
}
```

Εικόνα 5.4: Αλγόριθμος BPTT [38]



Σχήμα 5.13: Χρονικό ανάπτυγμα δικτύου Elman [38]

5.7.4.5 Μοντέλο LSTM (Long Short Term Memory)

Όπως αναφέρθηκε, η μέθοδος BPTT είναι μια διαδικασία εκπαίδευσης που χρησιμοποιείται σε αναδρομικά νευρωνικά δίκτυα. Ουσιαστικά, μπορούμε να πούμε ότι αποτελεί μια επέκταση του αλγορίθμου Back-Propagation. Παρόλο που η BPTT έχει εμφανίσει ικανοποιητικά αποτελέσματα σε πολλές εφαρμογές και προβλήματα, αντιμετωπίζει δυσκολίες όσον αφορά την διαχείριση των **εξασθενουσών παραγώγων**. Το εν λόγω πρόβλημα ορίζει ότι, όταν ο αριθμός των χρονικών στιγμών αυξάνεται, οι παράγωγοι έχουν την τάση να εξασθενούν καθώς προωθούνται πίσω στον χρόνο. Αυτό σημαίνει ότι η επίδραση των αρχικών χρονικών στιγμών στην ενημέρωση των βαρών, μειώνεται

σημαντικά κατά την εκπαίδευση. Κατά συνέπεια, το μοντέλο μπορεί να δυσκολεύεται στο να εκμεταλλευτεί την πλήρη πληροφορία του ιστορικού του πλαισίου [38].

Για την αντιμετώπιση αυτού του προβλήματος, έχουν αναπτυχθεί μέθοδοι, όπως το **LSTM** (Long Short-Term Memory), το οποίο προσπαθεί να ξεπεράσει το πρόβλημα του BPTT και να βελτιώσει την απόδοση των αναδρομικών δικτύων.

Στο άρθρο τους [73], οι Sepp Hochreiter και Jurgen Schmidhuber, παρουσιάζουν την αρχιτεκτονική του LSTM, η οποία βασίζεται σε ένα νευρωνικό δίκτυο με ειδικούς μηχανισμούς για τη διαχείριση και αποθήκευση μακροπρόθεσμης πληροφορίας. Το LSTM χρησιμοποιεί μια **ειδική μονάδα μνήμης**, γνωστή ως **κελί ελέγχου** (Cell State), η οποία μπορεί να αποθηκεύει και να ανακαλεί πληροφορία για μεγάλα χρονικά διαστήματα. Οι LSTM μονάδες χρησιμοποιούν **πύλες** (gates) για να ελέγχουν τη ροή των δεδομένων μέσα στο δίκτυο, συμπεριλαμβανομένης μιας πύλης εισόδου (input gate), μιας πύλης ανατροφοδότησης (forget gate) και μιας πύλης εξόδου (output gate). Αυτές οι πύλες επιτρέπουν στο LSTM να αποφασίσει ποιες πληροφορίες πρέπει να διατηρηθούν, ποιες νέες πληροφορίες πρέπει να εισαχθούν και πότε θα εξάγονται ή θα μπλοκάρονται πληροφορίες από τη μονάδα μνήμης. Κάθε πύλη είναι ένα **διάνυσμα πύλης** που περιλαμβάνει νευρώνες οι οποίοι παίρνουν τιμές από 0 έως 1.

Γενικά, το δίκτυο περιγράφεται απο ένα **διάνυσμα κατάστασης** $\boldsymbol{c}(t)$, το οποίο ανατροφοδοτεί τον εαυτό του μέσω μιας καθυστέρησης [38]:

$$c_i(t) = f_i(t)c_i(t-1) + g_i^{in}(t)in_i(t) \tag{5.22}$$

Πύλη ανατροφοδότησης μνήμης (Forget Gate)

Εκφράζεται ως διάνυσμα [38]:

$$\boldsymbol{f}(t) = [f_1(t), f_2(t), \dots, f_h(t)] \tag{5.23}$$

Η πύλη αυτή είναι υπεύθυνη για τον έλεγχο της προηγούμενης κατάστασης $\boldsymbol{c}(t-1)$. Αν η τιμή της $f_i(t)$ είναι 0, το σύστημα ξεχνά το στοιχείο $c_i(t-1)$. Επομένως, ο τύπος (5.22) γίνεται:

$$c_i(t) = g_i^{in}(t)in_i(t) \tag{5.24}$$

Διαφορετικά, αν δηλαδή η τιμή της $f_i(t)$ είναι 1, τότε το σύστημα διατηρεί το στοιχείο $c_i(t-1)$ και έτσι ο τύπος (5.22) γίνεται:

$$c_i(t) = c_i(t-1) + g_i^{in}(t)in_i(t) \tag{5.25}$$

Πύλη εισόδου (Input Gate)

Εκφράζεται ως διάνυσμα [38]:

$$\boldsymbol{g}^{in}(t) = \left[g_1^{in}(t), g_2^{in}(t), \dots, g_n^{in}(t)\right] \tag{5.26}$$

Η πύλη εισόδου ελέγχει την εισερχόμενη ακολουθία δεδομένων καθώς και την προηγούμενη κατάσταση του κελιού μνήμης. Αυτό επιτυγχάνεται με τη χρήση μιας **σιγμοειδούς συνάρτησης** ενεργοποίησης που δέχεται ως είσοδο τη συνδυασμένη πληροφορία από την εισερχόμενη ακολουθία δεδομένων και την προηγούμενη κατάσταση του κελιού μνήμης. Έτσι, όταν η πύλη εισόδου έχει τιμή 0, η εισερχόμενη πληροφορία αγνοείται και δεν επηρεάζει το κελί μνήμης, ενώ όταν έχει τιμή 1, η πληροφορία περνά στο κελί προς επεξεργασία.

Οπότε, αν η πύλη εισόδου $g_i^{in}(t) = 0$ τότε η έξοδος $in_i(t)$ είναι 0, καθώς αγνοείται η έξοδος της συνάρτησης υπερβολικής εφαπτομένης $\boldsymbol{tanh}$. Διαφορετικά, αν η πύλη εισόδου $g_i^{in}(t) = 1$ τότε, η έξοδος που προκύπτει από την επεξεργασία πληροφορίας της συνάρτησης $\boldsymbol{tanh}$, δηλαδή το διάνυσμα $in_i(t)$ περνάει κανονικά προς επεξεργασία.

Επομένως, σύμφωνα με τα παραπάνω έχουμε:

Για $g_i^{in}(t) = 0$:

$$in_i(t) = 0$$

Για $g_i^{in}(t) = 1$:

$$in_i(t) = tanh\left(\boldsymbol{w}_i^{in(T)}\boldsymbol{x}(t) + \boldsymbol{a}_i^{in(T)}\boldsymbol{y}(t-1) + b_i^{in(T)}\right) \tag{5.27}$$

Όπου $\boldsymbol{w}_i^{in(T)}$ τα βάρη από την είσοδο $\boldsymbol{x}(t)$ μέχρι έναν νευρώνα i που περιέχει την συνάρτηση ενεργοποίησης $\boldsymbol{tanh}$. Το $\boldsymbol{a}_i^{in(T)}$ είναι τα βάρη από το $\boldsymbol{y}(t-1)$ μέχρι έναν νευρώνα i που περιέχει την συνάρτηση ενεργοποίησης $\boldsymbol{tanh}$. Τέλος, $b_i^{in(T)}$ είναι οι πολώσεις. Αξίζει να σημειωθεί ότι, συμβολίσαμε τα βάρη χρησιμοποιώντας το κεφαλαίο γράμμα Τ, για να τονίσουμε την επανάληψη των βαρών κατά το ανάπτυγμα στο χρόνο, καθώς όταν αναπτύσσεται ένα αναδρομικό μοντέλο στον χρόνο, ουσιαστικά μιλάμε για ένα δίκτυο πολλών επιπέδων όπου τα βάρη τους παραμένουν ίδια.

Πύλη εξόδου (Output Gate)

Εκφράζεται ως διάνυσμα [38]:

$$\boldsymbol{g}^{out}(t) = [g_1^{out}(t), g_2^{out}(t), \dots, g_m^{out}(t)] \tag{5.28}$$

Έτσι, η πύλη εξόδου ελέγχει τη συμβολή του κελιού μνήμης στην τελική έξοδο του LSTM. Γίνεται χρήση μιας **σιγμοειδούς συνάρτησης ενεργοποίησης** που λαμβάνει ως είσοδο την πληροφορία από την εσωτερική κατάσταση του κελιού μνήμης, για να αποφασίσει τι πληροφορία θα περάσει στην έξοδο. Αν η πύλη εξόδου είναι 0, τότε η τελική έξοδος είναι κοντά στο 0. Αν η πύλη εξόδου είναι κοντά στο 1, τότε η τελική έξοδος αποδίδεται πλήρως από την έξοδο της συνάρτησης $\boldsymbol{tanh}$ η οποία δέχεται την εσωτερική κατάσταση του κελιού. Δηλαδή, στην έξοδο περνάει το αποτέλεσμα της συνάρτησης $\boldsymbol{tanh.}$

Επομένως, σύμφωνα με τα παραπάνω έχουμε:

Για $g_i^{out}(t) = 0$:

$$y_i(t) = 0$$

Για $g_i^{out}(t) = 1$:

$$y_i(t) = tanh\big(c_i(t)\big) \tag{5.29}$$

Έτσι, μπορούμε να πούμε ότι, ο νευρώνας i χρησιμοποιεί μια σιγμοειδή συνάρτηση ενεργοποίησης για να υπολογίσει την πύλη εξόδου $g_i^{out}(t)$, καθώς και μια συνάρτηση ενεργοποίησης υπερβολικής εφαπτομένης $tanh$ για τον υπολογισμό της εξόδου $y_i(t)$. Η σιγμοειδής συνάρτηση ως γνωστόν παίρνει μια συνολική είσοδο και παράγει μια τιμή μεταξύ 0 και 1. Αυτή η τιμή αντιπροσωπεύει το ποσοστό ενεργοποίησης της πύλης εξόδου $g_i^{out}(t)$. Όταν η τιμή είναι κοντά στο 0, η πύλη εξόδου είναι κλειστή, ενώ όταν είναι κοντά στο 1, η πύλη εξόδου είναι ανοικτή.

Από την άλλη πλευρά, η μη γραμμική συνάρτηση υπερβολικής εφαπτομένης $tanh$ παίρνει μια τιμή ως είσοδο και επιστρέφει μια τιμή μεταξύ -1 και 1. Η τιμή αυτή αντιπροσωπεύει την έξοδο $y_i(t)$ του νευρώνα i, που προκύπτει από το κελί μνήμης $c_i(t)$.

Συνοψίζοντας, ο νευρώνας i χρησιμοποιεί, τη σιγμοειδή συνάρτηση ενεργοποίησης για να αποφασίσει την κατάσταση της πύλης εξόδου, καθώς και την συνάρτηση υπερβολικής εφαπτομένης για να υπολογίσει την έξοδό του.

Γενικά, η πύλη $\boldsymbol{f}(t)$ είναι συνάρτηση των διανυσμάτων εισόδου, εξόδου, και κατάστασης. Οπότε παίρνει την μορφή [38]:

$$\boldsymbol{f}(t) = sigmoid\big(\boldsymbol{W}^f \boldsymbol{x}(t) + \boldsymbol{A}^f \boldsymbol{y}(t-1) + \boldsymbol{k}^f \circ \boldsymbol{c}(t-1) + \boldsymbol{b}^f\big) \tag{5.30}$$

Όπου:

$\boldsymbol{W}^f \in \mathbb{R}^{h\times n}$ και $\boldsymbol{A}^f \in \mathbb{R}^{h\times m}$ είναι πίνακες.

$\boldsymbol{k}^f$ και $\boldsymbol{b}^f$ είναι διανύσματα.

Επίσης οι πύλες εισόδου και εξόδου, είναι συναρτήσεις των διανυσμάτων εισόδου, εξόδου, και κατάστασης. Οπότε παίρνουν την μορφή [38]:

$$\boldsymbol{g}^{in}(t) = sigmoid\left(\boldsymbol{W}^{in}\boldsymbol{x}(t) + \boldsymbol{A}^{in}\boldsymbol{y}(t-1) + \boldsymbol{k}^{in} \circ \boldsymbol{c}(t-1) + \boldsymbol{b}^{in}\right) \quad (5.31)$$

και

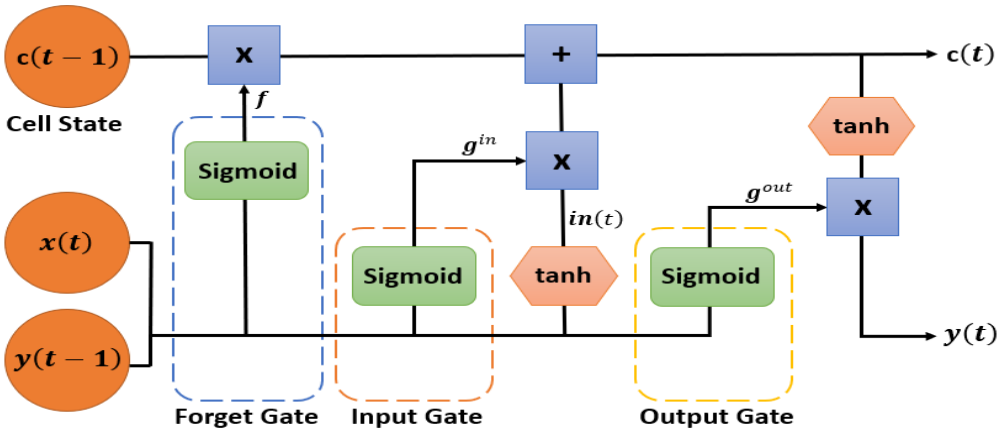
$$\boldsymbol{g}^{out}(t) = sigmoid(\boldsymbol{W}^{out}\boldsymbol{x}(t) + \boldsymbol{A}^{out}\boldsymbol{y}(t-1) + \boldsymbol{k}^{out} \circ \boldsymbol{c}(t-1) + \boldsymbol{b}^{out}) \quad (5.32)$$

Στις 3 παραπάνω συναρτήσεις, τα βάρη των πινάκων $\boldsymbol{W}, \boldsymbol{A}$, καθώς και των διανυσμάτων $\boldsymbol{k}, \boldsymbol{b}$ αναπροσαρμόζονται κατά την εκπαίδευση.

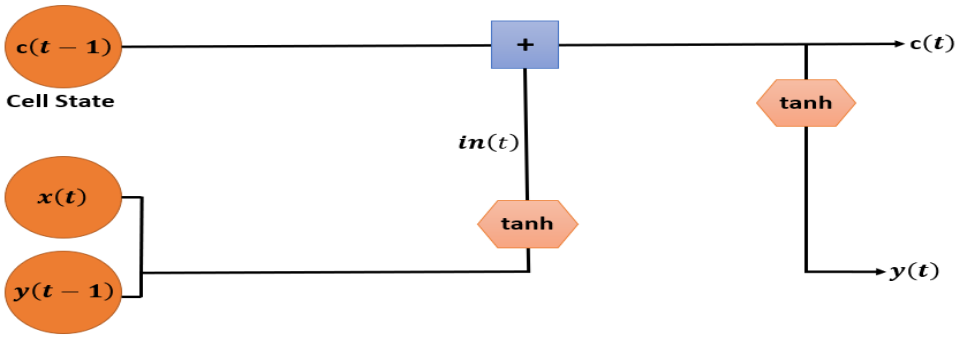
Η γενική αρχιτεκτονική του μοντέλου LSTM φαίνεται στο Σχήμα 5.14. Το Σχήμα 5.15 δείχνει μια απλοποιημένη αρχιτεκτονική του μοντέλου, δεδομένου ότι όλες οι πύλες είναι ανοιχτές. Δηλαδή ισχύει:

$$g_i^{in}(t) = 1,\ g_i^{out}(t) = 1,\ f_i(t) = 1$$

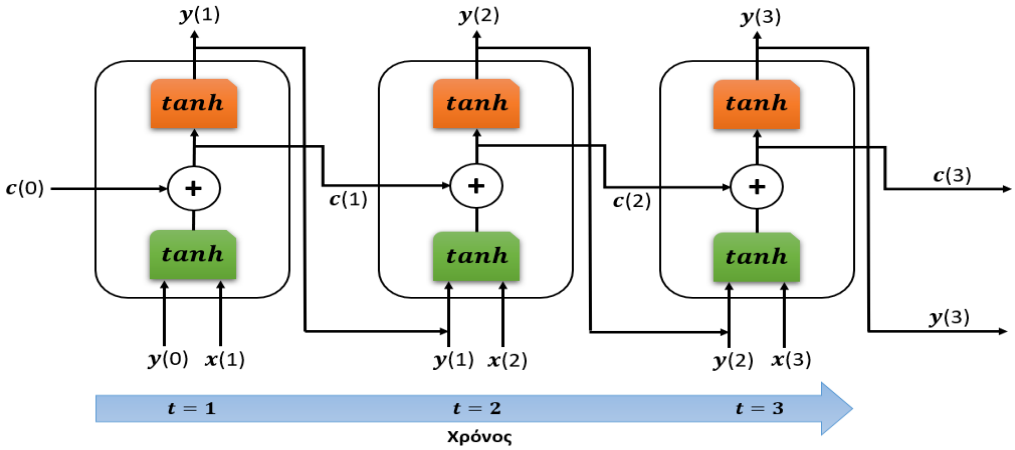
Το Σχήμα 5.16 δείχνει το ανάπτυγμα του μοντέλου στον χρόνο, δεδομένου ότι οι πύλες είναι ανοιχτές.



Σχήμα 5.14: Αρχιτεκτονική μοντέλου LSTM



Σχήμα 5.15: Αρχιτεκτονική μοντέλου LSTM – όλες οι πύλες ανοιχτές



Σχήμα 5.16: Ανάπτυγμα μοντέλου LSTM στον χρόνο

Μαθηματική περιγραφή μοντέλου LSTM

Με βάση την αρχιτεκτονική του μοντέλου, περιγράφεται ως εξής [38]:

Είσοδος: $\boldsymbol{x}(t)$

Διάνυσμα κατάστασης:

$$\boldsymbol{c}(t) = \boldsymbol{f}(t) \circ \boldsymbol{c}(t-1) + \boldsymbol{g}^{in}(t) \circ \boldsymbol{in}(t) \tag{5.33}$$

Είσοδος για διάνυσμα κατάστασης:

$$\boldsymbol{in}(t) = tanh\left(\boldsymbol{W}^{inx}\boldsymbol{x}(t) + \boldsymbol{A}^{iny}\boldsymbol{y}(t-1) + \boldsymbol{b}^{inb}\right) \tag{5.34}$$

Έξοδος:

$$\boldsymbol{y}(t) = \boldsymbol{g}^{out}(t) \circ tanh\left(\boldsymbol{c}(t)\right) \tag{5.35}$$

5.7.5 Νευρωνικό δίκτυο Softmax

Το νευρωνικό δίκτυο Softmax [38] είναι ένα είδος τεχνητού νευρωνικού δικτύου που χρησιμοποιείται κυρίως για προβλήματα ταξινόμησης. Σκοπός του είναι η μετατροπή μιας εισόδου σε ένα διάνυσμα πιθανοτήτων. Συχνά, χρησιμοποιείται ως το τελευταίο επίπεδο ενός νευρωνικού δικτύου για να παράγει προβλέψεις. Χρησιμοποιεί τον αλγόριθμο ταξινόμησης της λογιστικής παλινδρόμησης για κλάσεις $C > 2$. Γενικά, η αρχιτεκτονική του αποτελείται από ένα επίπεδο νευρώνων όπου η έξοδος του k νευρώνα υποδηλώνει την πιθανότητα ένα πρότυπο (διάνυσμα) εισόδου $\boldsymbol{x}$ να ανήκει στην κλάση k. Δηλαδή ισχύει:

$$y_k = p(t = k|\boldsymbol{x}) = \frac{exp\left(\boldsymbol{w}_k^T \boldsymbol{x} + w_{k,0}\right)}{\sum_{i=1}^{C} exp\left(\boldsymbol{w}_i^T \boldsymbol{x} + w_{i,0}\right)} \ , \ \ k = 1,2,\dots,C \tag{5.36}$$

5.7.6 Word2Vec και νευρωνικά δίκτυα

Στον τομέα του NLP μας ενδιαφέρει να δημιουργήσουμε αριθμητικές αναπαραστάσεις των λέξεων ώστε να γίνουν κατανοητές από τις μηχανές. Αυτές οι αναπαραστάσεις, τα λεγόμενα **word embeddings**, μπορούν να προκύψουν με διάφορους τρόπους, όπως με χρήση του αλγορίθμου TF-IDF και του μοντέλου Word2Vec που είδαμε στο κεφάλαιο 4.

Ο αλγόριθμος Word2Vec, προτάθηκε απο τον Tomas Mikolov και την ομάδα του το 2013 και βασίζεται σε δύο τεχνικές, τους αλγόριθμους CBOW και Skip-Gram [38], [74], [75]. Και οι δύο αλγόριθμοι είναι ουσιαστικά νευρωνικά δίκτυα και στοχεύουν στην εκμάθηση κατανεμημένων αναπαραστάσεων λέξεων με βάση το περιεχόμενό τους μέσα σε ένα σώμα κειμένου.

5.7.6.1 Μοντέλο CBOW (Continuous Bag-of-Words)

Ο σκοπός του CBOW είναι να μάθει να προβλέπει μια λέξη μέσα σε ένα κείμενο, βασιζόμενο στις γειτονικές λέξεις της [38], [67], [75]. Η λειτουργία του είναι η εξής:

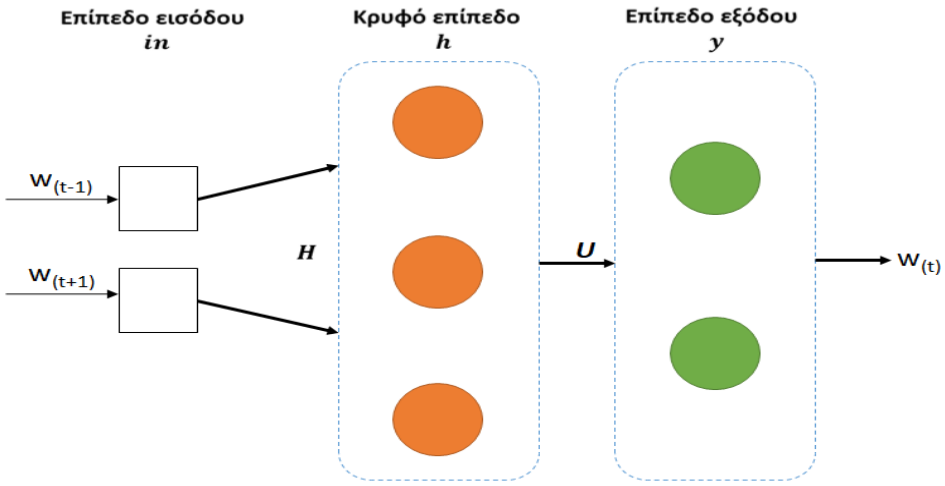
Κατασκευάζει ένα λεξικό από τις μοναδικές λέξεις του κειμένου και δημιουργεί ένα "παράθυρο" γύρω από κάθε λέξη. Το παράθυρο αυτό περιλαμβάνει τις γειτονικές λέξεις που χρησιμοποιούνται για να προβλεφθεί η συγκεκριμένη λέξη. Στη συνέχεια, το CBOW παίρνει τις αναπαραστάσεις των γειτονικών λέξεων και εκπαιδεύει ένα νευρωνικό δίκτυο για να προβλέπει τη συγκεκριμένη λέξη στο κέντρο του παραθύρου. Για την έξοδο χρησιμοποιεί ένα επίπεδο Softmax με N νευρώνες και πίνακα βαρών $\boldsymbol{U}$ μεταξύ κρυφού επιπέδου και επιπέδου εξόδου [75]:

$$\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \dots, \boldsymbol{u}_N]$$

Οπότε, δεδομένου ενός λεξιλογίου V, το μοντέλο δέχεται στην είσοδο πολλά διανύσματα της μορφής $(w_1, w_2, \dots, w_V)$ και σαν έξοδο παράγει ένα διάνυσμα w_t που είναι η ζητούμενη λέξη. Τα διανύσματα αυτά είναι **one-hot** κωδικοποιημένα. Δηλαδή, μόνο μια θέση τους περιέχει τον αριθμό 1, ενώ οι υπόλοιπες θέσεις περιέχουν τον αριθμό 0. Αν για παράδειγμα, η λέξη car είναι η τέταρτη λέξη μέσα στο λεξικό, τότε η λέξη αυτή θα αναπαρασταθεί σαν διάνυσμα της μορφής:

$$\boldsymbol{v}(car) = [0,0,0,1,0,0,0,\dots]$$

Όπου, όλα τα στοιχεία του διανύσματος είναι 0 εκτός απο ένα 1 που βρίσκεται στην τέταρτη θέση. Το Σχήμα 5.17 δείχνει την αρχιτεκτονική του μοντέλου.



Σχήμα 5.17: Αρχιτεκτονική CBOW [75]

Μαθηματική ερμηνεία μοντέλου

Η είσοδος του μοντέλου αποτελείται από τις κωδικοποιημένες αναπαραστάσεις των γειτονικών λέξεων της ζητούμενης λέξης. Οπότε μπορούμε να ορίσουμε την είσοδο με την σχέση [38]:

$$\boldsymbol{in} = \sum_{j=-c}^{c} \boldsymbol{v}(w_{t+j}) \quad , \quad j \neq 0 \tag{5.37}$$

Όπου $\boldsymbol{v}(w_{t+j})$ είναι διάνυσμα του οποίου (όπως είπαμε παραπάνω), κάθε στοιχείο έχει τιμή 0 εκτός απο την θέση η οποία υποδηλώνει την θέση της λέξης w_{t+j} στο λεξικό, και έτσι η εν λόγω θέση θα πάρει τιμή 1.

Το κρυφό επίπεδο περιλαμβάνει τον πίνακα βαρών $\boldsymbol{H}$. Άρα η τιμή του κρυφού επιπέδου είναι:

$$\boldsymbol{h} = \boldsymbol{H} \cdot \boldsymbol{in} \tag{5.38}$$

Για την έξοδο (όπως αναφέρθηκε), χρησιμοποιεί ένα επίπεδο Softmax με N νευρώνες και πίνακα βαρών $\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \dots, \boldsymbol{u}_N]$.

Οπότε η έξοδος είναι:

$$y_i = \frac{exp(-\boldsymbol{u}_i^T \cdot \boldsymbol{h})}{\sum_j exp(-\boldsymbol{u}_j^T \cdot \boldsymbol{h})} \tag{5.39}$$

Το διάνυσμα στόχος είναι:

$$\boldsymbol{target} = \boldsymbol{v}(w_t) \tag{5.40}$$

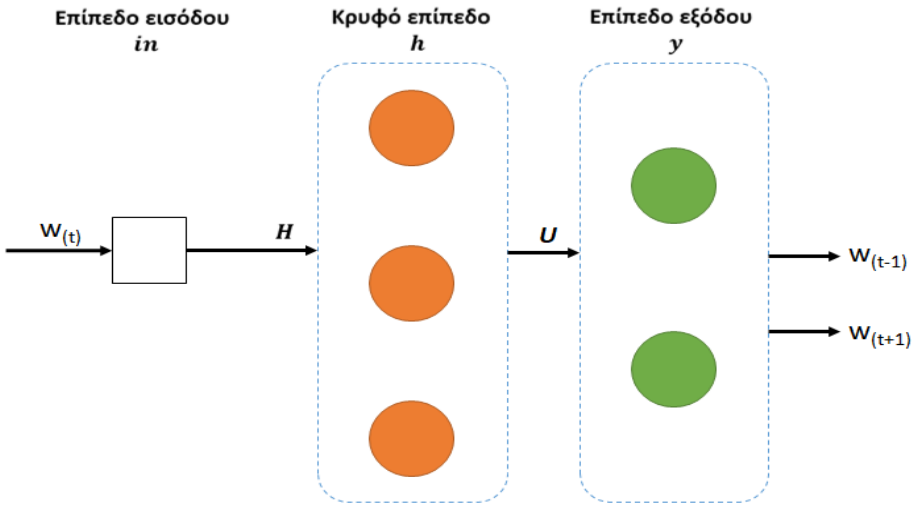
Ως συνάρτηση κόστους χρησιμοποιείται η εντροπία:

$$J_{CBOW} = \frac{1}{T}\sum_{t=1}^{T} log\, y_{k(w_t)} \tag{5.41}$$

Τέλος, η εκπαίδευση γίνεται με την μέθοδο βελτιστοποίησης επικλινούς καθόδου (gradient descent optimization procedure).

5.7.6.2 Μοντέλο Skip-Gram

Στόχος του αλγορίθμου Skip-Gram είναι να προβλέψει τα συμφραζόμενα μιας δεδομένης λέξης. Δηλαδή, λειτουργεί ανάποδα σε σχέση με τον αλγόριθμο CBOW [38], [67], [75]. Οπότε, δεδομένου ενός λεξιλογίου V, το μοντέλο δέχεται στην είσοδο ένα διάνυσμα w_t, και προσπαθεί να προβλέψει στην έξοδο διανύσματα της μορφής $(w_1, w_2, \ldots, w_V)$. Το Σχήμα 5.18 δείχνει την αρχιτεκτονική του μοντέλου.



Σχήμα 5.18: Αρχιτεκτονική Skip-Gram [75]

Μαθηματική ερμηνεία μοντέλου

Είσοδος η δεδομένη λέξη:

$$\boldsymbol{in} = \boldsymbol{v}(w_t) \tag{5.42}$$

Στόχος οι γειτονικές λέξεις:

$$\boldsymbol{target} = \sum_{j=-c}^{c} \boldsymbol{v}(w_{t+j}) \quad , \; j \neq 0 \tag{5.43}$$

Η περιγραφή του κρυφού καθώς και του επιπέδου εξόδου είναι παρόμοια με το CBOW.

Ως συνάρτηση κόστους χρησιμοποιείται η μορφή:

$$J_{SG} = \frac{1}{T}\sum_{t=1}^{T}\sum_{j=-c}^{c} log\, y_{k(w_{t+j})} \quad , \; j \neq 0 \tag{5.44}$$

Όπως το μοντέλο CBOW, έτσι και το Skip-Gram εκπαιδεύεται με την μέθοδο βελτιστοποίησης επικλινούς καθόδου.

5.7.7 Προκλήσεις στην εκπαίδευση νευρωνικών δικτύων και τεχνικές αντιμετώπισης

Όπως συμβαίνει σε όλες τις στατιστικές μεθόδους πρόβλεψης, έτσι και στα νευρωνικά δίκτυα μπορεί να εμφανιστούν δύο κύρια φαινόμενα που αποτελούν πρόκληση, η **υπερπροσαρμογή** (overfitting) και η **υποπροσαρμογή** (underfitting). Και τα δύο σχετίζονται με την πολυπλοκότητα του μοντέλου και την ικανότητά του να **γενικεύσει** σωστά σε νέα δεδομένα.

Έτσι, όταν το μοντέλο είναι υπερβολικά πολύπλοκο μπορεί να μάθει ακόμα και τα θορυβώδη χαρακτηριστικά των δεδομένων εκπαίδευσης. Το αποτέλεσμα είναι ότι το μοντέλο προσαρμόζεται πάρα πολύ στα δεδομένα εκπαίδευσης, αλλά αποτυγχάνει να γενικεύσει σε νέα δεδομένα (υπερπροσαρμογή). Από την άλλη, η υποπροσαρμογή συμβαίνει όταν το μοντέλο δεν είναι ικανό να μάθει την πολυπλοκότητα των δεδομένων εκπαίδευσης και αποτυγχάνει να γενικεύσει σωστά σε νέα δεδομένα [1], [31], [59].

Για τον περιορισμό αυτών των προκλήσεων μπορούμε να λάβουμε τα παρακάτω μέτρα:

- **Κατάλληλη αρχιτεκτονική μοντέλου**. Επιλογή μιας αρχιτεκτονικής μοντέλου που αντιστοιχεί στην πολυπλοκότητα των δεδομένων. Πρέπει να βρεθεί μια ισορροπία μεταξύ πολυπλοκότητας και γενίκευσης.
- **Μέγεθος δεδομένων εκπαίδευσης**. Το μέγεθος των δεδομένων εκπαίδευσης είναι ένας σημαντικός παράγοντας για την επίτευξη καλής απόδοσης στην εκπαίδευση των νευρωνικών δικτύων. Συνήθως, όσο πιο πολλά δεδομένα εκπαίδευσης έχουμε, τόσο καλύτερα μπορεί να γενικεύσει το μοντέλο σε νέα δεδομένα.
- **Καθαρισμός δεδομένων από θόρυβο**. Ο καθαρισμός των δεδομένων από θόρυβο μπορεί να θεωρηθεί ως μια τεχνική κανονικοποίησης που βοηθά στην αντιμετώπιση της

υπερπροσαρμογής. Στη περίπτωση του NLP, ο θόρυβος αναφέρεται σε ανεπιθύμητες παρεμβολές που επηρεάζουν την ποιότητα του κειμένου, όπως για παράδειγμα, τα λάθη στην ορθογραφία ή τα σημεία στίξης.

- **Πλήθος εποχών εκπαίδευσης**. Κατά την διαδικασία εκπαίδευσης ενός νευρωνικού δικτύου, τα δεδομένα παρουσιάζονται στο δίκτυο για επαναλαμβανόμενες εποχές, προκειμένου να προσαρμοστούν οι παράμετροι του μοντέλου. Ο αριθμός των εποχών είναι μια σημαντική υπερπαράμετρος που καθορίζει την έκταση της εκπαίδευσης του μοντέλου. Καθώς το μοντέλο επεξεργάζεται τα δεδομένα εκπαίδευσης κατά τις επαναλήψεις, προσαρμόζει τις εσωτερικές του παραμέτρους ώστε να μειώσει το **σφάλμα πρόβλεψης** και να βελτιστοποιήσει την απόδοσή του [31]. Άρα, η επιλογή κατάλληλου αριθμού εποχών είναι σημαντική, καθώς μπορεί να επηρεάσει την ικανότητα του μοντέλου να γενικεύει σε νέα δεδομένα. Η βιβλιοθήκη Keras διαθέτει την μέθοδο **fit()** για την εκπαίδευση νευρωνικών δικτύων σε ένα πλήθος εποχών [1].

5.8 Μετασχηματιστές (Transformers)

Η αρχιτεκτονική των μετασχηματιστών (Transformers), έχει επαναπροσδιορίσει τον τρόπο με τον οποίο οι μηχανές αντιλαμβάνονται και επεξεργάζονται τη φυσική γλώσσα. Τα μοντέλα αυτά γνωστά και ως **μεγάλα μοντέλα γλώσσας** (Large Language Models - LLMs), αποτελούν την πρόσφατη εξέλιξη στον τομέα της μηχανικής μάθησης και έχουν καταφέρει να επιτύχουν εντυπωσιακά αποτελέσματα σε πολλές γλωσσικές εργασίες [68], [76].

Βασική διαφορά των LLMs με τους προκατόχους τους, είναι ότι επιτρέπουν την **παράλληλη επεξεργασία** όλων των στοιχείων μίας πρότασης, με αποτέλεσμα τα μοντέλα αυτά να έχουν καλύτερη επίγνωση των σημασιολογικών σχέσεων μεταξύ των λέξεων. Βέβαια, σε αυτό βοηθά και ο **μηχανισμός προσοχής** (attention mechanism) που εφαρμόζεται, ο οποίος επιτρέπει στο μοντέλο να επικεντρώνεται σε συγκεκριμένα τμήματα του κειμένου κατά την επεξεργασία της γλώσσας [67].

Όπως έχουμε προαναφέρει, στη παρούσα εργασία χρησιμοποιήσαμε τέσσερα διακριτά μοντέλα τύπου transformers, ώστε να αναλύσουμε το συναίσθημα προτάσεων στα ελληνικά. Πριν αναλύσουμε την αρχιτεκτονική τους και τον τρόπο που εκπαιδεύτηκαν, είναι σημαντικό να εξετάσουμε πιο αναλυτικά τον μηχανισμό προσοχής.

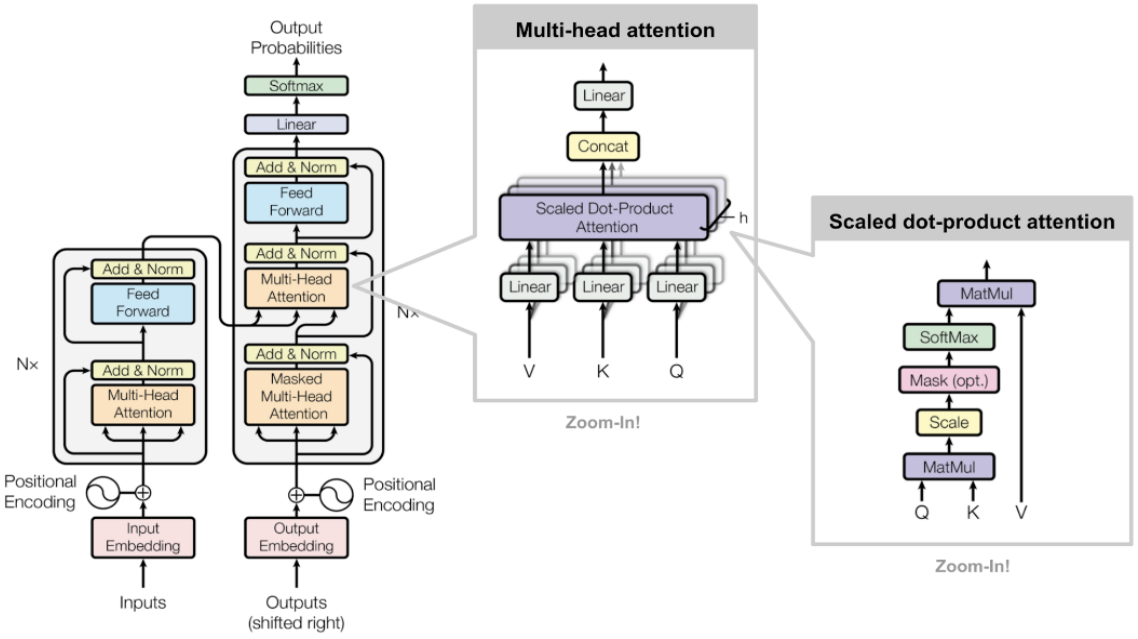
5.8.1 Μηχανισμός προσοχής (Attention Mechanism)

Ας μελετήσουμε εις βάθος την αρχιτεκτονική των transformers (Εικόνα 5.5) με σκοπό να κατανοήσουμε τον τρόπο λειτουργίας τους. Κατά κύριο λόγο, ένα transformer χρειάζεται εισόδους, αριθμητικές ακολουθίες, και εξόδους για την ανάθεση σημασιολογικής σημασίας σε αυτές τις ακολουθίες. Οι είσοδοι και οι έξοδοι αποτελούνται από διαφορετικές λέξεις (tokens) κάποιου αυθαίρετου πλήθους seq_length, το οποίο μπορεί να είναι και διαφορετικό μεταξύ τους. Στη συνέχεια, καθένα από τα παραπάνω tokens αναπαρίσταται με word embeddings διαστάσεων 512.

Για παράδειγμα, έστω η λέξη $t = "\tau\acute{\rho}\chi\omega"$. Η λέξη αυτή θα έχει αναπαράσταση:

$$\boldsymbol{x} = (0.1, 0.2, -0.12, \ldots, 0.22)$$

όπου η διάσταση του $\boldsymbol{x}$ θα είναι 512. Ας ονομάσουμε την παράμετρο της διάστασης των embeddings d_{model}.



Εικόνα 5.5: Αρχιτεκτονική Transformer [77]

Τα transformers όπως είπαμε, επεξεργάζονται παράλληλα τις πληροφορίες που τους παρέχονται. Ο τρόπος με τον οποίο λειτουργούν έχει σαν αποτέλεσμα να μην γνωρίζουν πληροφορίες σχετικά με την θέση μιας λέξης μέσα στην πρόταση. Προκύπτει λοιπόν η ανάγκη για κωδικοποίηση της θέσης (Position Encoding - PE). Με τη λογική αυτή, στα διανύσματα εισόδου προστίθενται διανύσματα d_{model} διάστασης που περιέχουν τις πληροφορίες αυτές. Στην περίπτωση του transformer γίνεται χρήση των τριγωνομετρικών συναρτήσεων, όπου pos η θέση της λέξης στην πρόταση και i η θέση των embeddings:

$$PE_{(pos,2i)} = sin\frac{pos}{10000^{\frac{2i}{d_{model}}}} \tag{5.45}$$

$$PE_{(pos,2i+1)} = cos\frac{pos}{10000^{\frac{2i}{d_{model}}}} \tag{5.46}$$

$pos = 1, \dots, seq\_length$ και $i = 1, \dots, d_{model}$

Βασικό στοιχείο ενός transformer, είναι η δομή **κωδικοποιητή-αποκωδικοποιητή**. Ο κωδικοποιητής αποτελείται από 6 όμοια στρώματα, όπου κάθε στρώμα έχει δύο υποστρώματα: ένα στρώμα πολλαπλής προσοχής (multi-head attention) και ένα στρώμα ενός απλού νευρωνικού δικτύου πρόσθιας τροφοδότησης (feed-forward neural network) [67], [77]. Σε καθένα από τα υποστρώματα

αυτά εφαρμόζονται αναδρομικές συνδέσεις και κανονικοποίηση του στρώματος. Ο **αποκωδικοποιητής** έχει παρόμοια αρχιτεκτονική με τη διαφορά ότι σε αυτόν προστίθεται ένα δεύτερο υπόστρωμα πολλαπλής προσοχής με διαφορετική λειτουργία από το πρώτο [67], [77]. Σκοπός του είναι να αποτρέψει το μοντέλο από το να βασίζεται στις επόμενες λέξεις για να κάνει προβλέψεις μιας συγκεκριμένης λέξης. Για να το πετύχει αυτό, το δεύτερο υπόστρωμα multi-head attention μετακινεί τις τιμές των διανυσμάτων εξόδου μια θέση δεξιά και εφαρμόζει μια **μάσκα** (mask) σε αυτές, προκειμένου να μην είναι γνωστές οι λέξεις που βρίσκονται μετά την επιθυμητή τιμή. Η μάσκα αυτή προκαλεί μια αδυναμία στο μοντέλο να "αντιγράψει" λέξεις από την είσοδο στην έξοδο, ώστε να επιτυγχάνεται μια πιο σωστή μετάφραση ή περίληψη του κειμένου, ανάλογα με το πεδίο εφαρμογής του transformer.

Όλη η ουσία των transformers έγκειται στα επίπεδα που διαθέτουν τον μηχανισμό προσοχής. Πρακτικά, ο μηχανισμός προσοχής λειτουργεί αναθέτοντας ένα βάρος w σε κάθε λέξη στην είσοδο $x = (x_1, x_2, \dots, x_n)$, όπου $n = seq\_length$, και $x \in \mathbb{R}^{n \times d_{model}}$. Στις σημαντικότερες λέξεις δίνεται μεγαλύτερη βαρύτητα, ενώ στις λιγότερο σημαντικές λέξεις, μικρότερη. Στην συνέχεια, συνδυάζει όλες τις λέξεις στην είσοδο με βάση τα βάρη τους, για να πάρει μια πιο εστιασμένη αναπαράσταση της εισόδου $y = (y_1, y_2, \dots, y_n)$. Άρα, $y \in \mathbb{R}^{n \times d_{model}}$.

Επομένως ισχύει:

$$y_i = \sum_{j=1}^{n} w_{ij} x_j \tag{5.47}$$

Όπου $i = 1,2,3,\dots,n$

Έτσι, διαπιστώνουμε πως ο μηχανισμός προσοχής είναι ένας ευφάνταστος τρόπος να περιγράψουμε μια απλή διαδικασία, αυτή του σταθμισμένου αθροίσματος. Τα βάρη αυτά ονομάζονται και σκορ προσοχής. Ας εξετάσουμε τον τρόπο που αναθέτονται τα βάρη στις εκάστοτε λέξεις.

Για να κατανοήσουμε καλύτερα τον μηχανισμό που προτάθηκε από τους Vaswani et al. [68], ας προσπαθήσουμε να τον ορίσουμε από την αρχή. Σαν βάρη στη σχέση (5.47), θα θεωρήσουμε το εσωτερικό γινόμενο του διανύσματος αναπαράστασης κάθε λέξης με το αντίστοιχο διάνυσμα όλων των λέξεων στην πρόταση. Άρα:

$$w_{ij} = x_i x_j^T \tag{5.48}$$

Θεωρητικά, διανύσματα με μικρό εσωτερικό γινόμενο βρίσκονται κοντά. Συνεπώς, αυτός είναι ο βασικός τρόπος με τον οποίο τα transformers αναγνωρίζουν τη σημασιολογία, το εσωτερικό γινόμενο. Βέβαια, στην πράξη το εσωτερικό γινόμενο μπορεί να πάρει οποιαδήποτε πραγματική τιμή. Για να αποφευχθούν τυχόν αποκλίσεις στα αποτελέσματα για μεγάλους αριθμούς, στα παραπάνω βάρη εφαρμόζεται η συνάρτηση softmax, ώστε το σύνολο τιμών τους να είναι το [0,1]. Επίσης, γίνεται μία διαίρεση με το $\sqrt{d_{model}}$, προκειμένου να αποφευχθεί στην εκπαίδευση ο μηδενισμός της παραγώγου της softmax, κάτι το οποίο συμβαίνει για μεγάλες τιμές της συνάρτησης. Έτσι, τα νέα βάρη θα είναι:

$$w'_{ij} = softmax\left(\frac{x_i x_j^T}{\sqrt{d_{model}}}\right) \tag{5.49}$$

Άρα, τελικά θα έχουμε:

$$y_i = \sum_{j=1}^{n} w'_{ij} x_j \tag{5.50}$$

Στην παραπάνω προσέγγιση δεν υπάρχουν πουθενά τιμές στις οποίες να γίνεται δυνατή η τροποποίησή τους. Οπότε δεν μπορεί να εφαρμοστεί ο Back Propagation. Με άλλα λόγια, το παραπάνω σύστημα δεν μπορεί να μάθει. Πρέπει να οριστούν κάποια βάρη για εκπαίδευση. Παρατηρούμε πως τα διανύσματα εισόδου χρησιμοποιούνται τρεις φορές στην διαδικασία υπολογισμού της εξόδου. Δυο φορές ώστε να υπολογιστούν τα βάρη και άλλη μία φορά για να υπολογιστούν τα διανύσματα εξόδου. Έτσι, εντάσσουμε τρεις γραμμικούς μετασχηματισμούς $W^K, W^Q, W^V \in \mathbb{R}^{d_{model} \times d_{model}}$ με εκπαιδεύσιμες παραμέτρους τέτοιες ώστε:

$$q_i = x_i W^Q,\ k_i = x_i W^K,\ v_i = x_i W^V$$

Ορίζουμε:

$$w_{ij} = softmax\left(\frac{q_i \cdot k_j^T}{\sqrt{d_{model}}}\right) \tag{5.51}$$

και

$$y_i = \sum_{j=1}^{n} w_{ij} v_j \tag{5.52}$$

Ας θεωρήσουμε σαν $K = Concat(k_i)$, $Q = Concat(q_i)$, και $V = Concat(v_i)$ όπου $i = 1,2,3,\ldots,n$ και Concat η συνάρτηση ενοποίησης πινάκων. Οι πίνακες αυτοί είναι διαστάσεων $K,\ Q,\ V \in \mathbb{R}^{n \times d_{model}}$. Με χρήση όλων των παραπάνω, καταλήγουμε στον τύπο για τον μηχανισμό προσοχής που χρησιμοποιήθηκε στον ορισμό των transformers:

$$Attention(K, Q, V) = softmax\left(\frac{QK^T}{\sqrt{d_{model}}}\right) \cdot V \tag{5.53}$$

Οι παραπάνω πίνακες περιέχουν βάρη τα οποία μαθαίνονται κατά την εκπαίδευση και ο ρόλος τους είναι ο εξής [68]:

- **Key** (K). Το κλειδί αντιπροσωπεύει τις πληροφορίες που χρησιμοποιούνται για τον υπολογισμό της συνάφειας κάθε στοιχείου στην ακολουθία εισόδου σε σχέση με άλλα στοιχεία.
- **Query** (Q). Το ερώτημα αντιπροσωπεύει τις πληροφορίες που χρησιμοποιούνται για τον υπολογισμό της ομοιότητας μεταξύ της ακολουθίας εισόδου και της ακολουθίας στόχου.
- **Value** (V). Η τιμή αντιπροσωπεύει τις πληροφορίες που χρησιμοποιούνται για τον υπολογισμό του σταθμισμένου αθροίσματος των στοιχείων της ακολουθίας εισόδου με βάση τη συνάφειά τους με την ακολουθία-στόχο.

Για την ακρίβεια, τα transformers χρησιμοποιούν μια παραλλαγή του (5.53), την πολλαπλή προσοχή.

$$Multihead(Q,K,V) = Concat(head_1,\dots,head_h)W^O \tag{5.54}$$

Όπου, $head_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right)$

Στην περίπτωση αυτή γίνεται γραμμική προβολή των πινάκων K, Q, V, h φορές σε υποχώρους μικρότερης διάστασης d_k, d_k, d_v αντίστοιχα, όπου υπολογίζονται h διαφορετικά σετ σκορ προσοχής. Ισχύουν τα εξής:

$$W_i^Q \in \mathbb{R}^{d_{model}\times d_k}, W_i^K \in \mathbb{R}^{d_{model}\times d_k}, W_i^V \in \mathbb{R}^{d_{model}\times d_v}, W^O \in \mathbb{R}^{hd_v\times d_{model}}$$

Στην συνέχεια, οι πινάκες $head_i$ ενώνονται και πολλαπλασιάζονται με W^O, έναν γραμμικό μετασχηματισμό ώστε τελικά το αποτέλεσμα να έχει τις ίδιες διαστάσεις με την είσοδο. Έτσι, το μοντέλο μπορεί να αντλεί πληροφορίες από διαφορετικούς υποχώρους των αναπαραστάσεων των λέξεων, αναλόγως με το πρόβλημα που έχει να αντιμετωπίσει. Όλες οι παράμετροι των πινάκων μαθαίνονται κατά την εκπαίδευση.

Στην αρχική περίπτωση των transformers ισχύουν: $h = 8, d_{model} = 512, d_k = d_v = \frac{d_{model}}{h} = 64$

Τέλος, ας αναφέρουμε τον τρόπο με τον οποίο ο κωδικοποιητής και ο αποκωδικοποιητής επικοινωνούν μεταξύ τους, τον λεγόμενο μηχανισμό **cross-attention** [78]. Όμοια με τον αρχικό ορισμό, ο μηχανισμός cross-attention χρειάζεται πίνακες K, Q και V για τον υπολογισμό της προσοχής. Η διαφορά εδώ έγκειται στον τρόπο που τους αποκτά. Πιο συγκεκριμένα, ο αποκωδικοποιητής χρησιμοποιεί τις εξόδους του κωδικοποιητή, ας τις ονομάσουμε $c \in \mathbb{R}^{n\times d_{model}}$, για να υπολογίσει τους πίνακες K και V. Για τον υπολογισμό του πίνακα Q γίνεται χρήση της εξόδου του προηγούμενου επιπέδου προσοχής, ας την ονομάσουμε $\alpha \in \mathbb{R}^{m\times d_{model}}$. Αξίζει να σημειωθεί πως m είναι το μήκος ακολουθίας της επιθυμητής εξόδου και δεν είναι πάντοτε $m = n$ (π.χ. η περίληψη ενός κειμένου). Άρα, ισχύουν τα εξής:

$$Q = \alpha W^Q, K = c\,W^K, V = c\,W^V$$

Όπου W^Q, W^K, W^V είναι πίνακες μετασχηματισμού όμοιοι με πριν, διάστασης $d_{model} \times d_{model}$. Τελικά, οι έξοδοι θα έχουν μήκος ακολουθίας m γιατί:

$$QK^T \in \mathbb{R}^{m \times n}, V \in \mathbb{R}^{n \times d_{model}}$$

Επομένως:

$$softmax\left(\frac{QK^T}{\sqrt{d_{model}}}\right)V \in \mathbb{R}^{m \times d_{model}} \quad (5.55)$$

Φυσικά, όλα όσα είπαμε για την πολλαπλή προσοχή εκτελούνται και στην περίπτωση της cross-attention.

Συμπερασματικά, όλα τα παραπάνω συντέλεσαν στη καθιέρωση των transformers ως την τελευταία λέξη της τεχνολογίας στο NLP. Πολλοί ερευνητές [79-82], έκαναν χρήση της παραπάνω αρχιτεκτονικής (ή ένα κομμάτι της) ώστε να δημιουργήσουν πολύ δυνατά μοντέλα ικανά να αντιμετωπίσουν μια μεγάλη γκάμα προβλημάτων NLP. Ένα από αυτά τα προβλήματα είναι και η ανάλυση συναισθήματος σε προτάσεις. Παρακάτω, θα εξετάσουμε τα μοντέλα που χρησιμοποιήθηκαν σε αυτή την εργασία.

5.8.2 Μοντέλο BERT (Bidirectional Encoder Representations from Transformers)

Το μοντέλο του BERT [79], προτάθηκε από τους Jacob Devlin et. al το 2018 και έφερε πολλές καινοτομίες στον τομέα του NLP. Σύμφωνα με την αρχιτεκτονική του, έχει δύο εκδοχές: την $BERT_{BASE}$ και την $BERT_{LARGE}$. Η βασική έκδοση αποτελείται από $N = 12$ επίπεδα κωδικοποιητή transformers, η διάσταση των embeddings είναι $d_{model} = 768$ και διαθέτει $h = 12$ επίπεδα στο μηχανισμό προσοχής. Αντίστοιχα, τα νούμερα για την μεγάλη έκδοση είναι $N = 24$, $d_{model} = 1024$ και $h = 16$.

Είναι σημαντικό να επισημάνουμε την απουσία του αποκωδικοποιητή στην αρχιτεκτονική του BERT, καθώς αποτελεί ένα βασικό στοιχείο που διακρίνει αυτό το μοντέλο. Όπως υποδεικνύει και το όνομά του, το BERT είναι ένα αμφίδρομο (bidirectional) transformer. Αυτό σημαίνει πως κατά την ανάλυση των πληροφοριών που του παρέχονται, έχει πλήρη επίγνωση για τα tokens ανεξαρτήτως της θέσης τους. Όπως αναλύσαμε και παραπάνω, ο αποκωδικοποιητής ενός transformer έχει τη δυνατότητα να εφαρμόζει μια μάσκα στις εξόδους, ώστε να μην επιτρέπει στο μοντέλο να κάνει προβλέψεις με βάση μεταγενέστερες λέξεις. Αυτό το χαρακτηριστικό απουσιάζει από τον BERT με βασική συνέπεια την αδυναμία του για παραγωγή κειμένου. Θα έλεγε κανείς πως η προσέγγιση αυτή ίσως δημιουργήσει προβλήματα, καθώς το μοντέλο θα μπορεί να κάνει σωστές προβλέψεις χωρίς όμως να αντιλαμβάνεται το πραγματικό νόημά τους. Έτσι, οι Jacob Devlin et. al. [79], βρήκαν έναν έξυπνο τρόπο να αντιμετωπίσουν το παραπάνω πρόβλημα, εισάγοντας την έννοια του **Masked Language Model** (MLM).

Όσον αφορά τα embeddings των λέξεων [79], ένα BERT μοντέλο χρησιμοποιεί τα Word Piece embeddings με λεξιλόγιο 30.000 λέξεων, τα embeddings θέσης (Position Encoding - PE, σχέσεις 5.45 και 5.46), και τα embeddings τμήματος. Επίσης, εισάγονται και τρία ειδικά tokens. Το [CLS] που

δηλώνει την αρχή της ακολουθίας εισόδου, το [SEP] που δηλώνει ένα διαχωριστικό σημείο στην ακολουθία εισόδου (π.χ. σε ζευγάρια ερώτησης απάντησης, το [SEP] token θα μπει ακριβώς μετά το πέρας της ερώτησης) και το [MASK] token, που σχετίζεται με την έννοια του MLM και θα δούμε παρακάτω. Τα embeddings τμήματος σχετίζονται με το [SEP] token και δηλώνουν σε ποιό τμήμα της εισόδου αντιστοιχούν τα διάφορα tokens.

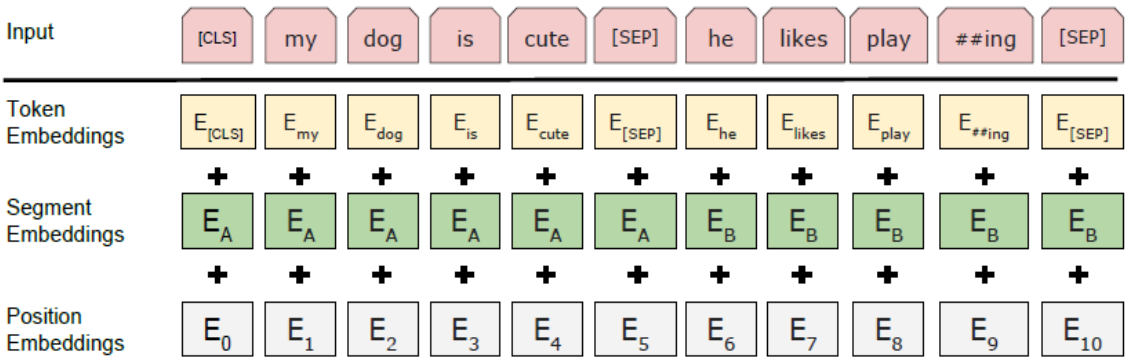
Η εκπαίδευση του BERT μοντέλου αποτελείται από δύο βήματα, την **προ-εκπαίδευση** (pre-training) και την **βελτιστοποίηση** (fine-tuning) [79]. Η προ-εκπαίδευση δεν απαιτεί δεδομένα με ετικέτες, οπότε ανήκει στην κατηγορία των μεθόδων μηχανικής μάθησης χωρίς επίβλεψη. Κατά την φάση προεκπαίδευσης, το μοντέλο εκπαιδεύεται σε ένα μεγάλο σύνολο κειμένων. Μετά το τέλος της φάσης αυτής, το μοντέλο έχει κατανοήσει κάποιες βασικές σημασιολογικές εξαρτήσεις. Έπειτα, ακολουθεί το βήμα της βελτιστοποίησης, όπου το προ-εκπαιδευμένο πλέον μοντέλο λαμβάνει ένα πιο συγκεκριμένο σύνολο δεδομένων για εκπαίδευση που αφορά μια συγκεκριμένη εργασία, όπως η ταξινόμηση κειμένου. Ουσιαστικά στη φάση αυτή, το μοντέλο προσαρμόζεται στην συγκεκριμένη εργασία με χρήση της αντίστοιχης διαθέσιμης ετικέτας, δηλαδή εφαρμόζεται η έννοια της μηχανικής μάθησης με επίβλεψη. Προφανώς, το βήμα της βελτιστοποίησης απαιτεί πολύ μικρότερο όγκο δεδομένων, καθώς και λιγότερες εποχές εκπαίδευσης συγκριτικά με την προ-εκπαίδευση. Άρα, βλέπουμε πως το BERT κάνει ένα συνδυασμό των βασικών τεχνικών μηχανικής μάθησης και ουσιαστικά ανήκει στη κατηγορία μοντέλου με ημι-επίβλεψη.

Πιο συγκεκριμένα, στην προ-εκπαίδευση ένα μοντέλο BERT έχει δύο βασικούς στόχους [79]. Πρώτον, πρέπει να γίνει ένα MLM. Ουσιαστικά σαν είσοδο στο transformer εισέρχονται προτάσεις στις οποίες κάποιες από τις λέξεις αντικαθίστανται με το [MASK] token. Σκοπός του BERT σε πρώτη φάση είναι να προβλέψει αυτές τις λέξεις. Ο τρόπος με τον οποίο γίνεται η ανάθεση της μάσκας [MASK] είναι τυχαίος και προκύπτει από προκαθορισμένα ποσοστά. Τα βάρη στα επίπεδα προσοχής, καθώς και στο feed-forward neural network, ρυθμίζονται καταλλήλως και προχωράμε στο δεύτερο στόχο της προεκπαίδευσης, όπου πρέπει να γίνει πρόβλεψη επόμενης πρότασης (Next Sentence Prediction - NSP). Στο σημείο αυτό, δημιουργείται ένα dataset από το ήδη υπάρχον και συντάσσονται ζευγάρια προτάσεων προκειμένου να κατηγοριοποιηθούν ως ***isNext***, εάν μια πρόταση Β συνδέεται με μια πρόταση Α, δηλαδή αν η πρόταση Β ακολουθεί την Α, ή ως ***NotNext***, εάν η πρόταση Β δεν συνδέεται με την Α. Αυτό βοηθά το μοντέλο στην κατανόηση της σημασίας των προτάσεων και της σχέσης μεταξύ τους.

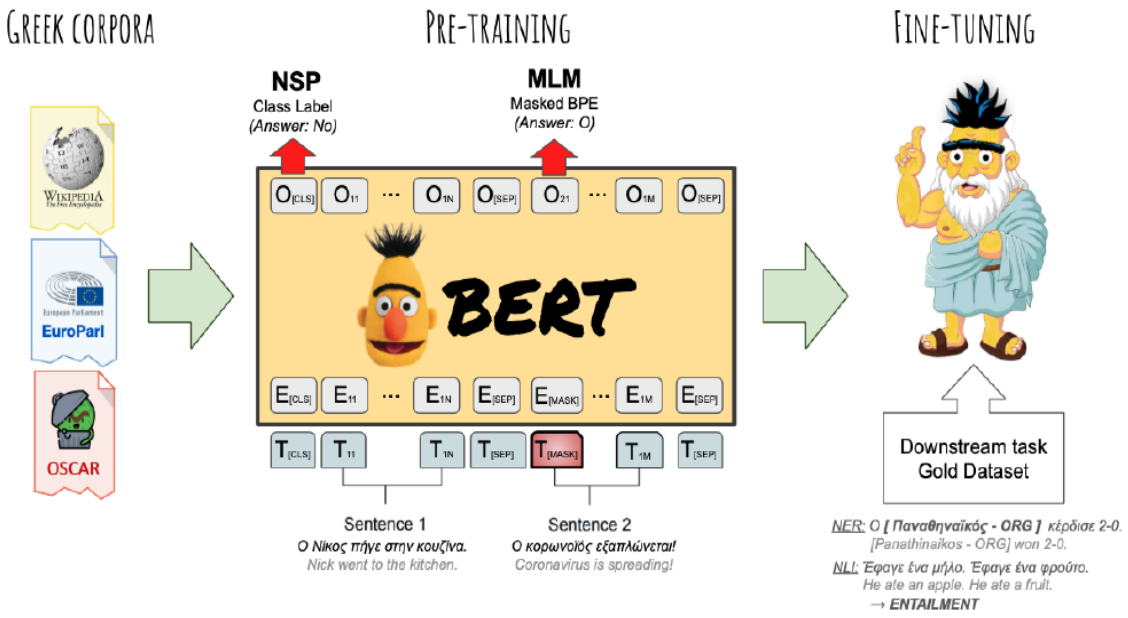
Στο βήμα της βελτιστοποίησης γίνεται προσαρμογή των παραμέτρων του BERT αναλόγως το πρόβλημα που έχει να αντιμετωπίσει. Στην περίπτωση την κατηγοριοποίησης κειμένου, που είναι και το πρόβλημα που μας ενδιαφέρει, προστίθενται ένας αριθμός από γραμμικούς νευρώνες στο τέλος, ανάλογος με τον αριθμό των κλάσεων κατηγοριοποίησης. Στην ανάλυση συναισθήματος, για την δική μας εργασία προσθέτουμε έναν νευρώνα σε περίπτωση δυαδικής κατηγοριοποίησης και τρεις νευρώνες σε περίπτωση τριαδικής κατηγοριοποίησης.

Εφόσον επιθυμούμε να αναλύσουμε το συναίσθημα στην ελληνική γλώσσα, χρησιμοποιούμε την ελληνική έκδοση του BERT [83], που εκπαιδεύτηκε από τον Γιάννη Κουτσικάκη και την ομάδα του. Η φάση της προεκπαίδευσης του συγκεκριμένου μοντέλου έγινε ακριβώς με τον ίδιο τρόπο με τον αρχικό BERT. Χρησιμοποιήθηκε η $BERT_{BASE}$ έκδοση και έγινε χρήση των εξής βάσεων δεδομένων: α) η ελληνική Wikipedia, β) η βάση δεδομένων από τα πρακτικά του Ευρωπαϊκού Κοινοβουλίου και γ) η ελληνική έκδοση του OSCAR, μιας open source βάσης δεδομένων με κείμενα που χρηματοδοτείται από το ίδρυμα Common Crawl Foundation.

Στην Εικόνα 5.6 φαίνεται η αναπαράσταση εισόδου του BERT και στην Εικόνα 5.7 η διαδικασία προ-εκπαίδευσης και βελτιστοποίησης.



Εικόνα 5.6: Αναπαράσταση εισόδου του BERT [79]



Εικόνα 5.7: Διαδικασία προ-εκπαίδευσης και βελτιστοποίησης [83]

5.8.3 Μοντέλο RoBERTa (Robustly Optimized BERT Approach)

Όπως δηλώνει και το όνομά του, το μοντέλο RoBERTa [81] είναι βασισμένο στο BERT. Οι ερευνητές του RoBERTa χρησιμοποιούν την μεγάλη έκδοσή του, $BERT_{LARGE}$ για προ-εκπαίδευση. Όσον αφορά την αρχιτεκτονική του transformer, διαφέρει από το BERT ως προς τον τρόπο προ-εκπαίδευσης.

Οι Yinhan Liu et al. [81], αναφέρουν τις βασικές διαφορές μεταξύ του μοντέλου τους, με την αρχική πρόταση του BERT. Πρώτον, στο μοντέλο RoBERTa έχει καταργηθεί ο στόχος προ-εκπαίδευσης που αφορά την πρόβλεψη επόμενης πρότασης (NSP). Η αφαίρεση αυτού του στόχου, προέκυψε από

πειράματα που έδειξαν ότι η προ-εκπαίδευση με χρήση μόνο του MLM ήταν πιο αποτελεσματική. Αυτή η αλλαγή επέτρεψε στο μοντέλο RoBERTa να επιτύχει καλύτερα αποτελέσματα σε διάφορα κείμενα και βάσεις δεδομένων, σε σχέση με το αρχικό μοντέλο BERT. Δεύτερον, γίνεται δυναμική χρήση [MASK] token σε κάθε εποχή εκπαίδευσης σε σύγκριση με τον BERT όπου η χρήση μάσκας είναι η ίδια σε κάθε εποχή. Τρίτον, το RoBERTa δεν χρησιμοποιεί embeddings τμήματος (Segment Embeddings) ώστε να διευκρινιστεί ποιό token ανήκει σε ποιό τμήμα. Αρκεί μόνο η χρήση του token [SEP] για τον διαχωρισμό των τμημάτων. Τέλος, το RoBERTa εκπαιδεύεται με μεγαλύτερα μεγέθη παρτίδας (batches) και χρησιμοποιεί κωδικοποίηση (Byte Pair Encoding - BPE), δηλαδή κωδικοποιεί σε επίπεδο συλλαβών [84].

Στο πλαίσιο της εργασίας μας, χρησιμοποιήθηκε μια πολυγλωσσική έκδοση του RoBERTa [85]. Το μοντέλο αυτό εκπαιδεύτηκε σε 100 γλώσσες, μεταξύ των οποίων και τα ελληνικά, από κείμενα του Common Crawl Foundation. Οι δημιουργοί του μοντέλου αυτού βασίστηκαν σε τεχνικές μηχανικής μάθησης χωρίς επίβλεψη που προτάθηκε από τους [86], ώστε το μοντέλο να μπορεί να αναγνωρίζει τις διαφορετικές γλώσσες. Τέλος, αξίζει να σημειωθεί πως η συγκεκριμένη έκδοση του μοντέλου βασίζεται στην αρχιτεκτονική του $BERT_{BASE}$.

5.8.4 Μοντέλο DistilBERT

Το BERT, ένα ξεχωριστό μοντέλο στο πεδίο του NLP, διαθέτει εκτεταμένη αρχιτεκτονική με 12 επίπεδα μετασχηματιστή και 110 εκατομμύρια παραμέτρους. Αντίθετα, το DistilBERT [80] αποτελεί μια παραλλαγή του BERT, συμπιέζοντας το μοντέλο σε 6 επίπεδα transformer και μειώνοντας τον αριθμό παραμέτρων στα 66 εκατομμύρια. Το χαρακτηριστικό αυτό επιτρέπει μια σημαντική μείωση σε απαιτήσεις μνήμης.

Για την ανάπτυξη του DistilBERT, μπαίνει στο παιχνίδι η απόσταξη γνώσης, μια καινοτόμος μέθοδος εκπαίδευσης. Αξιοποιώντας ως δάσκαλο ένα μεγαλύτερο μοντέλο όπως το BERT, το DistilBERT εκπαιδεύεται να μιμείται τη συμπεριφορά του μεγαλύτερου, εκμεταλλευόμενο τη γνώση που έχει ήδη αποκτηθεί.

Έτσι με βάση τα παραπάνω, πρόκειται για ένα μικρότερο και ελαφρύτερο μοντέλο σε σύγκριση με το BERT και το RoBERTa. Η μείωση του μεγέθους του μοντέλου οδηγεί σε ταχύτερους χρόνους συμπερασμάτων. Αυτό το καθιστά μια ελκυστική επιλογή σε περιβάλλοντα όπου οι περιορισμοί πόρων είναι ανησυχία, επιτρέποντας στις εφαρμογές NLP να λειτουργούν γρήγορα και αποτελεσματικά. Ωστόσο, παρά τα σημαντικά πλεονεκτήματα που προσφέρει, παρουσιάζει και μειονεκτήματα όπως:

- **Χαμηλή απόδοση**. Λόγω της μείωσης του αριθμού των επιπέδων και των παραμέτρων σε σχέση με το αρχικό BERT μοντέλο, το DistilBERT μπορεί να παρουσιάσει μια μικρή μείωση στην ακρίβεια και στην ικανότητά του να αντιληφθεί λεπτομέρειες σε κείμενο.
- **Μικρή γενικότητα**. Η απόσταξη γνώσης απο το BERT, μπορεί να οδηγήσει σε ένα μοντέλο με περιορισμένες δυνατότητες, με συνέπεια να μην μπορεί να αντιμετωπίσει πιο ιδιαίτερα προβλήματα τα οποία δεν έχουν καλυφθεί απο το BERT. Με άλλα λόγια έχει μεγάλη εξάρτηση απο το BERT.

Για τις ανάγκες της εργασίας μας, έγινε χρήση μιας πολυγλωσσικής έκδοσης του DistilBERT [87], εκπαιδευμένη στη Wikipedia σε 104 γλώσσες.

5.8.5 Μοντέλο GPT (Generative Pre-trained Transformer)

Τα μοντέλα τύπου GPT [88], σε αντίθεση με όλα τα LLMs που έχουμε εξετάσει μέχρι στιγμής, βασίζονται στη αρχιτεκτονική του αποκωδικοποιητή των transformers. Ως συνέπεια, τα μοντέλα αυτά έχουν την δυνατότητα παραγωγής κειμένου και άρα συγκλίνουν στην παραδοσιακή έννοια των μοντέλων γλώσσας (Language Models ή LMs).

Δοθείσας μιας ακολουθίας tokens $U = (u_1, \dots, u_n)$, βασικός σκοπός ενός GPT μοντέλου είναι να μεγιστοποιήσει την σχέση [88]:

$$L_1(U) = \sum_i log\, P(u_i | u_{i-k}, \dots, u_{i-1}; \Theta) \tag{5.56}$$

Όπου k είναι το μέγεθος του περιεχομένου που μπορεί να εξετάσει το μοντέλο, και P η δεσμευμένη πιθανότητα που μοντελοποιείται από νευρωνικό δίκτυο με παραμέτρους Θ. Οι παράμετροι αυτές εκπαιδεύονται με την μέθοδο στοχαστικής βαθμωτής κατάβασης.

Όπως και στο BERT, η εκπαίδευση ενός GPT μοντέλου χωρίζεται σε 2 φάσεις: την **προ-εκπαίδευση** και την **βελτιστοποίηση**. Στην φάση της προ-εκπαίδευσης [88], σκοπός του GPT είναι να προβλέψει σωστά μια λέξη $u_i \in U$. Φυσικά, λόγω της δομής του αποκωδικοποιητή των transformer τα GPT μοντέλα όταν προβλέπουν, γνωρίζουν το μέγιστο, k λέξεις πριν της ζητούμενης (στις επόμενες εφαρμόζεται η μάσκα του μηχανισμού προσοχής). Έτσι, τα βάρη στα επίπεδα προσοχής και στα επίπεδα του Feed Forward παραμετροποιούνται, ώστε τελικά το μοντέλο να προβλέπει σωστά τις λέξεις έχοντας υπόψη του ένα μεγάλο περιεχόμενο της πρότασης. Εδώ πρέπει να σημειώσουμε πως η βελτιστοποίηση σχετίζεται κατά πολύ με το τελικό πρόβλημα που καλείται να αντιμετωπίσει το GPT μοντέλο.

Γενικά, δοθείσας βάσης δεδομένων C, που αποτελείται από ακολουθίες λέξεων $(x^1, x^2, \dots, x^m)$ και ετικετών y, σκοπός του μοντέλου σε αυτή τη φάση είναι να μεγιστοποιήσει την ακόλουθη σχέση [88]:

$$L_2(C) = \sum_{(x,y)} log\, P(y | x^1, \dots, x^m) \tag{5.57}$$

Οι ερευνητές του GPT, παρατήρησαν ότι αν συνδυαστούν οι σχέσεις (5.56) και (5.57), με την διαφορά ότι στην συνάρτηση L_1 δοθεί η βάση δεδομένων C, θα προκύψει τελικά ότι στο βήμα της βελτιστοποίησης ο βασικός σκοπός είναι η μεγιστοποίηση της σχέσης [88]:

$$L_3(C) = L_2(C) + \lambda \cdot L_1(C) \tag{5.58}$$

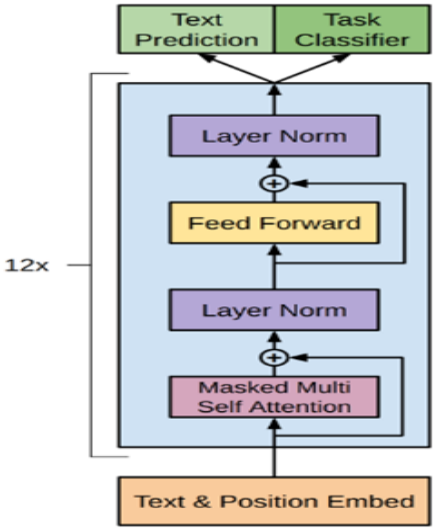
Όπου λ μια υπερπαράμετρος. Οι ερευνητές χρησιμοποίησαν $\lambda = 0.5$.

Για τις ανάγκες της εργασίας, χρησιμοποιήθηκε η δεύτερη γενιά των συγκεκριμένων μοντέλων, το GPT-2 [82]. Πιο συγκεκριμένα, χρησιμοποιήθηκε η μεγαλύτερη έκδοση του GPT-2 που περιέχει 1.5 δισεκατομμύρια παραμέτρους. Η έκδοση αυτή αποτελείται από $N = 48$ επίπεδα αποκωδικοποιητή

transformers, η διάσταση των embeddings είναι $d_{model} = 1600$, διαθέτει $h = 12$ επίπεδα στο μηχανισμό προσοχής και το μέγεθος του περιεχομένου είναι $k = 1024$.

Για την δημιουργία των embeddings, ένα GPT-2 μοντέλο κάνει χρήση της byte-pair κωδικοποίησης και προσθέτει τα embeddings θέσης. Τα συγκεκριμένα embeddings δεν είναι αυτά που αναφέρθηκαν στην ενότητα των transformers, αλλά προκύπτουν από ένα απλό πίνακα μετασχηματισμού του οποίου οι τιμές μαθαίνονται κατά την εκπαίδευση. Στην δική μας περίπτωση χρήσης, εφαρμόστηκε μία ελληνική έκδοση του GPT-2 [89], προσαρμοσμένη στο πρόβλημα της ανάλυσης συναισθήματος. Η συγκεκριμένη έκδοση βασίστηκε σε τεχνικές μεταφερόμενης μάθησης (transfer learning), ώστε να αξιοποιηθεί σε κάποιο βαθμό η κατανόηση που είχε το GPT-2 για την Αγγλική γλώσσα και να εφαρμοστεί στην Ελληνική. Ο τρόπος που γίνεται αυτό ξεφεύγει από τους σκοπούς της παρούσας εργασίας, αλλά παραθέτουμε την μεθοδολογία που βασίστηκαν οι ερευνητές [90].

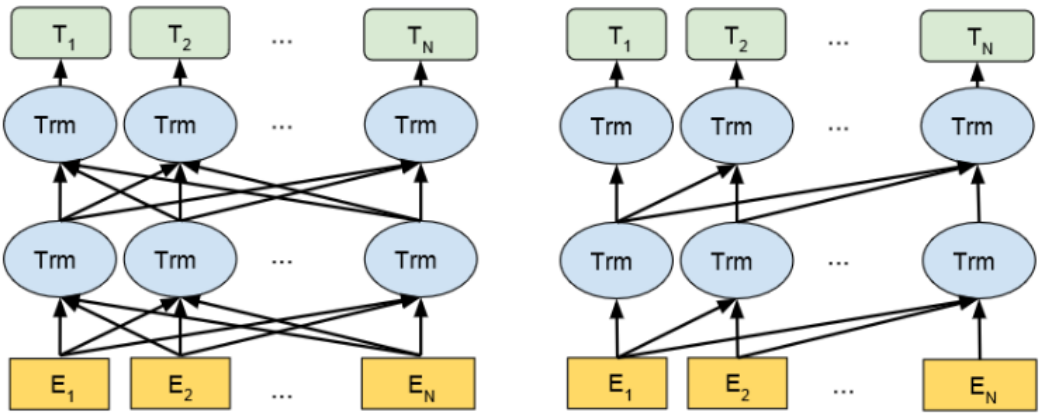
Το Σχήμα 5.19 δείχνει την αρχιτεκτονική του μοντέλου GPT.



Σχήμα 5.19: Αρχιτεκτονική μοντέλου GPT [88]

5.8.6 Σύγκριση BERT με GPT

Σύμφωνα με τα παραπάνω, έγινε ξεκάθαρο ότι, το μοντέλο BERT χρησιμοποιείται κυρίως για προβλήματα αναγνώρισης και κατηγοριοποίησης κειμένου. Μπορεί να εφαρμόσει μάσκα σε οποιαδήποτε λέξη της εισόδου και να προσπαθήσει να προβλέψει τη λέξη που βρίσκεται κάτω από τη μάσκα. Πρόκειται δηλαδή για ένα αμφίδρομο μοντέλο [67]. Από την άλλη πλευρά, το GPT δεν είναι αμφίδρομο, έχει σχεδιαστεί ειδικά για την παραγωγή κειμένου, και η μάσκα τοποθετείται μόνο σε επόμενες λέξεις, οπότε με βάση τις προηγούμενες λέξεις, το μοντέλο προσπαθεί να προβλέψει την επόμενη λέξη στην ακολουθία [67]. Και το BERT και το GPT ακολουθούν ένα παρόμοιο μοντέλο εκπαίδευσης με δύο φάσεις, την φάση προ-εκπαίδευσης (pre-training) χωρίς επίβλεψη και την φάση βελτιστοποίησης (fine-tuning) με επίβλεψη. Στο Σχήμα 5.20, φαίνεται η διαφορά των μοντέλων ως προς την αρχιτεκτονική προ-εκπαίδευσης. Αριστερά, το BERT χρησιμοποιεί έναν αμφίδρομο μετασχηματιστή, δεξιά το GPT χρησιμοποιεί μετασχηματιστή από αριστερά προς τα δεξιά [79].



Σχήμα 5.20: BERT (αριστερά) και GPT (δεξιά) [79]

5.9 Επίλογος

Με την ανάπτυξη των νευρωνικών δικτύων, έχουμε δει προηγμένες εφαρμογές σε πολλούς τομείς, όπως στο πεδίο του NLP. Η ικανότητά τους να μαθαίνουν και να παράγουν πολύπλοκες αναπαραστάσεις δεδομένων έχει αποδειχθεί καθοριστική για πολλές εφαρμογές. Τα νευρωνικά δίκτυα συνεχίζουν να εξελίσσονται με νέες αρχιτεκτονικές και μεθόδους εκπαίδευσης που αποδίδουν ακόμα καλύτερα αποτελέσματα. Παράλληλα, οι μετασχηματιστές εισήγαγαν μια νέα διάσταση στον τομέα του NLP. Η αρχιτεκτονική τους που βασίζεται στην αλληλεπίδραση μεταξύ των αναπαραστάσεων εισόδου και εξόδου, άνοιξε τον δρόμο για προηγμένες γλωσσικές εργασίες, όπως η αναγνώριση συναισθήματος και η σύνθεση κειμένου. Απέδειξαν ότι μπορούν να προσφέρουν εκπληκτικά αποτελέσματα, αλλά ταυτόχρονα απαιτούν μεγάλους υπολογιστικούς πόρους για την εκπαίδευσή τους, προκλήσεις οι οποίες προκαλούν το ενδιαφέρον της επιστημονικής κοινότητας.

Κεφάλαιο 6ο: Μεθοδολογία και Πειράματα

6.1 Εισαγωγή - Σχετικές εργασίες

Το πρόβλημα της ανάλυσης συναισθήματος έχει απασχολήσει εκτενώς την επιστημονική και ερευνητική κοινότητα. Η δυνατότητα που προσφέρει για γρήγορη και έμπιστη εκτίμηση της αρεσκείας των ανθρώπων γύρω από ένα θέμα, παροτρύνει πληθώρα επιχειρήσεων και οργανισμών να επενδύσουν στην συγκεκριμένη διαδικασία. Όπως αναφέρουν και οι M. Wankhade et al. [91], οι εφαρμογές της ανάλυσης συναισθήματος βρίσκουν εύφορο έδαφος σε τομείς όπως οι επιχειρήσεις (αξιολογήσεις προϊόντων, υπηρεσιών κ.λπ.), τα υγειονομικά, την τέχνη (κριτικές ταινιών, τραγουδιών κλπ.), το χρηματιστήριο και την παρακολούθηση της γνώμης του κοινού στα μέσα κοινωνικής δικτύωσης. Στην εργασία μας επιλέξαμε να ασχοληθούμε με δύο από τις παραπάνω κατηγορίες. Συγκεκριμένα, ασχοληθήκαμε με την ανάλυση συναισθήματος στην Ελληνική γλώσσα: α) σε αξιολογήσεις καταστημάτων από την ιστοσελίδα του Skroutz, και β) σε πολιτικά σχόλια από την πλατφόρμα του Twitter. Τα Ελληνικά είναι μια γλώσσα χαμηλού περιεχομένου, ή low resource γλώσσα όπως αναφέρεται συχνά στη βιβλιογραφία. Το γεγονός αυτό καθιστά δύσκολη τη μελέτη της σε πλαίσιο NLP, καθώς σε πολλές περιπτώσεις τα δεδομένα εκπαίδευσης απλώς δεν αρκούν. Ωστόσο, οι Έλληνες ερευνητές έχουν ασχοληθεί αρκετά με το πρόβλημα της ανάλυσης συναισθήματος. Οι προσεγγίσεις που χρησιμοποιούν μπορούν να διαχωριστούν σε δύο βασικές κατηγορίες: α) χρήση λεξικών και β) χρήση μεθόδων μηχανικής μάθησης.

Πριν την καθιέρωση των transformers στο NLP, η ανάλυση συναισθήματος γινόταν κατά βάση με χρήση λεξικών. Σημείο αναφοράς για την ελληνική γλώσσα είναι η βάση δεδομένων (λεξικό) που δημιουργήθηκε από τους [92]. Περιέχει 2315 ελληνικές λέξεις οι οποίες έχουν κατηγοριοποιηθεί με βάση την πόλωση του συναισθήματος, την υποκειμενικότητα και τα 6 βασικά συναισθήματα του Ekman [93]. Οι G. Kalamatianos at al. [94], κάνουν χρήση του παραπάνω λεξικού προκειμένου να κάνουν ανάλυση συναισθήματος σε βάση δεδομένων του Twitter, ώστε να εντοπίσουν το συναίσθημα που εκπίπτει από τα δημοφιλέστερα hashtags. Στη συγκεκριμένη εργασία σκοπός είναι η βαθύτερη ανάλυση στο συναίσθημα σε επίπεδο και των 6 συναισθημάτων του Ekman, όχι η εύρεση της πόλωσης. Από την άλλη, οι G. Petasis at al. [95], κάνουν χρήση ακόμα μεγαλύτερου λεξικού 6000 λέξεων οι οποίες είναι χαρακτηρισμένες σαν θετικές ή αρνητικές. Στην περίπτωση αυτή οι ερευνητές κάνουν χρήση ενός συστήματος με κανόνες για να κατηγοριοποιήσουν το συναίσθημα σε βάση δεδομένων με 2300 προτάσεις. Η ακρίβεια της συγκεκριμένης προσέγγισης έφτασε το 64%.

Πιο σύγχρονες τεχνικές, αξιοποιούν μεθόδους μηχανικής μάθησης, όπως αυτές που περιγράψαμε στα κεφάλαια 4 και 5. Οι προσεγγίσεις που θα συναντήσει κανείς στη βιβλιογραφία είναι πολλές. Κύριες διαφορές έγκειται στη δημιουργία των embeddings των λέξεων, καθώς και στην επιλογή του μοντέλου. Συνήθως, η επιλογή της χρήσης των transformers συνοδεύεται από προσαρμοσμένο διανυσματοποιητή (tokenizer) στην εκάστοτε περίπτωση.

Ο Δημήτρης Μπιλιάνος [96] στην εργασία του χρησιμοποιεί μία μικρή έκδοση του Skroutz dataset που περιέχει 480 προτάσεις κατηγοριοποιημένες ως θετικές ή αρνητικές. Κάνει χρήση δύο παραδοσιακών αλγορίθμων μηχανικής μάθησης του SVM και του GNB σε συνδυασμό με την μέθοδο διανυσματοποίησης TF-IDF. Επίσης, δοκιμάζει μια ελληνική έκδοση του BERT όπως και εμείς. Σε αυτό το μικρό dataset πετυχαίνει ακρίβεια 87% με τον SVM, 86% με τον GNB και 97% με τον BERT. Αυτή η σημαντική διαφορά 10-11% μας προϊδεάζει για την δύναμη των transformers.

Στην δουλειά των [97], συναντάμε μια ακόμη προσέγγιση μηχανικής μάθησης με χρήση του αλγόριθμου SVM σε δύο βάσεις δεδομένων από το Skroutz εστιασμένης μόνο στα κινητά τηλέφωνα. Για την ακρίβεια, στην εργασία τους οι ερευνητές μελέτησαν μια υβριδική προσέγγιση, όπου για την δημιουργία των embeddings κάνουν χρήση της Word2Vec μεθοδολογίας αλλά και λεξικών. Δοκιμάζουν διάφορες εκδοχές και συνδυασμούς στα embeddings και πετυχαίνουν μέγιστη ακρίβεια της τάξης του 83.6%.

Οι [98], [99] προσπαθούν όμοια με εμάς να κατηγοριοποιήσουν το συναίσθημα στο Skroutz Dataset και αποτελούν σημαντική πηγή. Και οι δύο κάνουν χρήση νευρωνικών δικτύων, με τον N. Avgero [98], να δημιουργεί ένα βαθύ νευρωνικό από την αρχή εντάσσοντας ένα στρώμα embedding το οποίο μετασχηματίζει την είσοδο 10000 διαστάσεων (το μέγεθος του λεξιλογίου) σε διανύσματα 16 διαστάσεων. Καταφέρνει ακρίβεια 92.6%. Ενώ ο N. Fragkis [99], προσεγγίζει την δική μας μεθοδολογία κάνοντας χρήση της ελληνικής έκδοσης του BERT. Το συγκεκριμένο μοντέλο πετυχαίνει ακρίβεια 95.7%.

Σημαντική είναι και η συνεισφορά των [100] στην Ελληνική ερευνητική κοινότητα. Στην συγκεκριμένη εργασία οι ερευνητές χρησιμοποιούν ένα τεράστιο, για τα ελληνικά δεδομένα, dataset που περιλαμβάνει, 7042 θετικά, 7977 αρνητικά και 44.791 ουδέτερα σχόλια. Συνολικά, 59.810 σχόλια. Δοκιμάζουν δυαδική και τριαδική κατηγοριοποίηση κάνοντας χρήση διάφορων transformers αλλά και διάφορων βαθιών μοντέλων. Μάλιστα, για την τριαδική κατηγοριοποίηση οι ερευνητές εκπαίδευσαν από την αρχή ένα μοντέλο BERT και ένα μοντέλο RoBERTa εξειδικευμένα στην Ελληνική γλώσσα. Πετυχαίνουν εξαιρετικά αποτελέσματα, όπου στη δυαδική κατηγοριοποίηση παίρνουν 99% μέσο F1-σκορ με χρήση ενός GPT-2 μοντέλου. Στην τριαδική περίπτωση, το μοντέλο τους που βασίζεται στο BERT υπερτερεί σε σχέση των υπολοίπων πετυχαίνοντας 80% μέσο F1-σκορ. Παρόλη τη διαθεσιμότητα των δύο μοντέλων που δημιούργησαν οι συγκεκριμένοι μελετητές, δεν αναρτούν τα δεδομένα εκπαίδευσης.

Η ανάλυση πολιτικών σχολίων δεν είναι κάτι καινοτόμο ακόμη και στην περίπτωση της Ελληνικής γλώσσας. Αναφέρουμε ενδεικτικά δύο σχετικές εργασίες [101], [102]. Στην πρώτη, οι ερευνητές χρησιμοποιούν τρεις αλγορίθμους μηχανικής μάθησης, τους Random Forest, Decision Tree και XGBoost. Για την δημιουργία των embeddings εξάγουν κάποια χαρακτηριστικά από σχόλια Twitter, όπως ο αριθμός των hashtags, και σε συνδυασμό με χαρακτηριστικά που εξάγουν από το λεξικό [92], όπως ο αριθμός των θετικών λέξεων, τελικά δημιουργούν το τελικό διάνυσμα χαρακτηριστικών. Η βάση που χρησιμοποιούν αποτελείται από 1640 tweets κατηγοριοποιημένα σε τρεις κλάσεις θετική, αρνητική και ουδέτερη. Τελικά, πετυχαίνουν ακρίβεια 80% με τον Random Forest, 70% με τον Decision Tree, και με τον XGBoost 79%.

Στην δεύτερη, παρατηρούμε μια πολυγλωσσική προσέγγιση κάτι που στη διεθνή βιβλιογραφία είναι σύνηθες. Εμάς μας ενδιαφέρει το κομμάτι της Ελληνικής γλώσσας, παρόλο που ίδιες προσεγγίσεις εφαρμόστηκαν στα Αγγλικά και στα Ισπανικά. Στη συγκεκριμένη εργασία, έγινε χρήση πολλών μοντέλων (SVM, RoBERTa, XML-RoBERTA κ.α.) με καλύτερη επίδοση (77% Average F1-Score) στην περίπτωση των Ελληνικών να πετυχαίνει ένα transformer ειδικά εκπαιδευμένο για ανάλυση συναισθήματος. Το dataset που χρησιμοποιήθηκε περιλαμβάνει 1000 tweets που κατηγοριοποιούνται σε θετικά, αρνητικά, ουδέτερα και απροσδιόριστα. Βλέπουμε εδώ μια περίπτωση χρήσης με τέσσερεις ετικέτες κατηγοριοποίησης.

Συνοψίζοντας, μπορεί κανείς εύκολα να αντιληφθεί από την μελέτη της Ελληνικής βιβλιογραφίας σχετικά με θέματα NLP, την έλλειψη σε δεδομένα εκπαίδευσης. Αντιλαμβανόμαστε την αναγκαιότητα κάποιων βάσεων δεδομένων να γίνουν σημεία αναφοράς με σκοπό να αναλυθούν από

την ερευνητική κοινότητα. Το Skroutz dataset είναι ένα από αυτά. Μια από τις συνεισφορές μας είναι και η δημιουργία ενός dataset τέτοιου τύπου, ώστε να εμπλουτίσει τη δυνατότητα των ερευνητών να αντιμετωπίσουν το γενικότερο πρόβλημα της ανάλυσης συναισθήματος. Το πολιτικό περιεχόμενο επιλέχθηκε καθώς την περίοδο συγγραφής της παρούσης (Φθινόπωρο 2022 - Άνοιξη 2023) η Ελλάδα βρισκόταν σε προεκλογική περίοδο.

Η συνέχεια του κεφαλαίου συνοψίζεται ως εξής: Στην ενότητα 6.2 συζητάμε την εξόρυξη tweets με τη βοήθεια του Twitter API. Στην ενότητα 6.3 δείχνουμε την διαδικασία που ακολουθήσαμε προκειμένου να καθαρίσουμε το dataset από σημεία στίξης και άλλο θόρυβο. Στην ενότητα 6.4, αναφέρουμε την διαδικασία ανάθεσης ετικετών η οποία συνιστά ένα πολύ σημαντικό βήμα πριν από εργασίες μηχανικής μάθησης. Ακολουθεί η ενότητα χαρτογράφησης tweets στην οποία προβάλλουμε tweets σε έναν διαδραστικό χάρτη, με την βοήθεια μεθόδων γεωκωδικοποίησης. Στην ενότητα 6.6, αναλύουμε τις βάσεις δεδομένων (datasets) που χρησιμοποιήσαμε παρουσιάζοντας και κάποιες στατιστικές μετρήσεις. Στη συνέχεια, γίνεται ανάλυση της διαδικασίας προ-επεξεργασίας των δεδομένων η οποία περιλαμβάνει σημαντικές μεθόδους όπως το lemmatization και η απαλοιφή λέξεων χωρίς αξία (stopwords). Στα επόμενα παρουσιάζουμε τις υπερ-παραμέτρους των μοντέλων που χρησιμοποιήσαμε, τις προ-εκπαιδευμένες ενσωματώσεις Word2Vec, την βιβλιοθήκη Tkinter, καθώς και τα αποτελέσματα που προέκυψαν από τα πειράματά μας, εφαρμόζοντας τον διαχωρισμό σε μηχανική και βαθιά μάθηση. Τέλος, προβαίνουμε σε σχολιασμό των αποτελεσμάτων.

Στο Παράρτημα Α, παραθέτουμε τον κώδικα όλου του κεφαλαίου σε Python.

6.2 Εξόρυξη δεδομένων και δημιουργία Politics Dataset

Η αναζήτηση για tweets είναι μια σημαντική λειτουργία που χρησιμοποιείται για την εμφάνιση συνομιλιών στο Twitter σχετικά με ένα συγκεκριμένο θέμα. Όπως είπαμε και στο κεφάλαιο 3, το Twitter API παρέχει δύο μεθόδους για αναζήτηση [25]. Την μέθοδο search_all_tweets(), η οποία επιστρέφει tweets των τελευταίων 30 ημερών [103], και την μέθοδο search_recent_tweets(), η οποία επιστρέφει tweets της τελευταίας εβδομάδας που ταιριάζουν με ένα συγκεκριμένο ερώτημα [104]. Στην εργασία μας χρησιμοποιήσαμε την δεύτερη μέθοδο με παραμέτρους όπως, το id χρήστη που αντιπροσωπεύει τον συγγραφέα του tweet (author_id), το όνομα χρήστη (username), την τοποθεσία που δηλώνει ο χρήστης η οποία δεν είναι πάντα έγκυρη (location), το κείμενο (text), την ημερομηνία δημοσίευσης (created_at), καθώς και την γλώσσα του tweet (lang) [105], [106].

Ο μέγιστος αριθμός αποτελεσμάτων αναζήτησης που πρέπει να επιστραφούν, ορίζεται στην παράμετρο max_results και είναι ένας αριθμός μεταξύ 10 και 100. Από προεπιλογή, μια απάντηση αιτήματος θα επιστρέψει 10 αποτελέσματα. Επίσης, κατά την φάση της εξόρυξης χρησιμοποιήσαμε την βιβλιοθήκη tweet-preprocessor προκειμένου να καθαρίσουμε τα tweets από urls, αναφορές σε hashtags (#topic) και σε χρήστες (mentions - @user). Τέλος, δημιουργούμε το αρχείο data.csv το οποίο περιέχει την στήλη ημερομηνία δημιουργίας του tweet, τον χρήστη, την τοποθεσία, το ερώτημα (πασόκ, σύριζα, κλπ), το κείμενο του tweet (TweetText), καθώς και την στήλη Sentiment που θα περιέχει το συναίσθημα που κρύβει το tweet (positive, negative, neutral). Το αρχείο data.csv, δεν αποτελεί την τελική βάση δεδομένων (dataset). Είναι απλώς μια πρώτη εικόνα των δεδομένων που έχουμε εξορύξει από το Twitter. Αν και σε πρώτη φάση καθαρίσαμε τα tweets, αυτά περιέχουν ακόμη θόρυβο (σημεία στίξης, emoji), όπως φαίνεται στην Εικόνα 6.1, που καλό είναι να εξαλειφθεί.

Location	Topic	TweetText
Κοζάνη	ΝΕΑ ΔΗΜΟΚΡΑΤΙΑ	τρείς εικόνες νέα δημοκρατία και [redacted]
Ρέθυμνο	ΠΑΣΟΚ	αυτό ήταν εταιρεία αυτό ήταν το πασόκ. αναβολι...
Δημοτική Ενότητα Χολαργού	ΣΥΡΙΖΑ	1985 εκλογές. θα βγάλω φωτογραφία τον μητσοτάκη
ΠάνωΣτιςΣτέγεςΜέσαΣτιςΚαρδιές	ΠΑΣΟΚ	Οι Άγιοι προφήτες του ΠΑΣΟΚ 😛 😛 😛 😛 😛 😛
Λάρισα	ΠΑΣΟΚ	πασόκ σημαίνει πρωτοπόρος
NaN	ΠΑΣΟΚ	που πηγε το "προτιμουμε να ανήκουμε στους ελλη...
Καρδίτσα	ΕΛΛΗΝΙΚΗ ΛΥΣΗ	ορίστε ομολογούν τα [redacted] θα ζητήσου...
Βέροια	ΠΑΣΟΚ	Εκλογές 2023 - ΠΑΣΟΚ σε Σκέρτσο και νέα δημοκρ...
Αθήνα	ΚΚΕ	το κκε στην [redacted] 😛 😛
Ναύπακτος	ΣΥΡΙΖΑ	Τσίπρας 👉Ο Σύριζα κατηγορεί τη νέα δημοκρατία ...
Χαλκίδα	ΚΚΕ	τα προβλήματα των αγροτών δεν οφείλονται σε φυ...
Δημοτική Ενότητα Θεσαλονίκης	ΚΚΕ	ΚΚΕ ΜΟΝΟ φιλοι μου.

Εικόνα 6.1: Τμήμα αρχείου data.csv (η στήλη TweetText περιέχει θόρυβο)

6.3 Καθαρισμός Dataset

Η βιβλιοθήκη tweet-preprocessor δεν είναι αρκετή για να καθαριστεί πλήρως το κείμενο. Πριν προχωρήσουμε στην ανάλυση συναισθήματος των tweets, πρέπει να εφαρμόσουμε διαδικασίες επιπλέον καθαρισμού στην στήλη TweetText, προκειμένου το τελικό dataset να απαλλαχθεί από κάποια σημεία στίξης ή περίεργα εικονίδια και να είναι πιο ευανάγνωστο. Όπως αναφέραμε παραπάνω, η στήλη Location δεν υποδηλώνει πάντα μια έγκυρη τοποθεσία χρήστη. Κάτι τέτοιο φαίνεται στην Εικόνα 6.1, στις γραμμές 4 και 6. Στην πρώτη περίπτωση έχουμε μια μη έγκυρη τοποθεσία, ενώ στην δεύτερη δεν έχει δηλωθεί τοποθεσία από τον χρήστη. Για όλους τους παραπάνω λόγους, αποφασίσαμε να απορρίπτουμε τέτοιου είδους tweets. Το γεγονός αυτό μας εξασφαλίζει την ανάκτηση συντεταγμένων τοποθεσίας για κάθε χρήστη που δημοσιεύει ένα tweet, προκειμένου όταν συλλέξουμε τα tweets και τα κατηγοριοποιήσουμε σε κλάσεις συναισθήματος (Negative, Positive, Neutral), να τα οπτικοποιήσουμε σε ένα διαδραστικό χάρτη.

Η διαδικασία που ακολουθήσαμε για όλα τα παραπάνω είναι η εξής:

- Διαγραφή όλων των εγγραφών με τιμή NaN
- Στη στήλη TweetText, διαγραφή πολλαπλών κενών (spaces), διαγραφή emoji, και ορισμένων σημείων στίξης (όχι όλων)
- Διαγραφή διπλότυπων εγγραφών. Λαμβάνουμε υπόψιν μόνο την στήλη TweetText, για τον εντοπισμό διπλότυπων.
- Μετατροπή κεφαλαίων γραμμάτων σε πεζά.

Στην Εικόνα 6.2 φαίνεται η νέα μορφή του αρχείου data.csv. Έχουν διαγραφεί σημεία στίξης, emoji και τυχόν διπλότυπες εγγραφές. Η εγγραφή με τιμή NaN στην στήλη Location έχει διαγραφεί αφού είναι εύκολα διαχειρίσιμη περίπτωση. Αντίθετα, η εγγραφή 4 δεν μπορεί να διαγραφεί στην παρούσα φάση διότι έχει μια απροσδιόριστη τιμή στην στήλη Location, και αυτό την καθιστά δύσκολα διαχειρίσιμη περίπτωση (δυστυχώς υπήρχαν πολλές τέτοιου τύπου εγγραφές). Ωστόσο, κατά την φάση της ανάκτησης συντεταγμένων καταφέραμε να απαλλαγούμε από τέτοιες ιδιαίτερες εγγραφές, αφού σε κάθε απροσδιόριστη τοποθεσία καταχωρήσαμε την τιμή None, και κατόπιν διαγράψαμε εγγραφές

με τιμές None. Για την ακρίβεια, όπως θα δούμε στην ενότητα 6.5, η στήλη Location αντικαθίσταται με 3 στήλες οι οποίες είναι, χώρα (country), πόλη (city), και περιφέρεια (state_district). Αυτές οι στήλες ουσιαστικά λάβανε την τιμή None.

Location	Topic	TweetText
Κοζάνη	ΝΕΑ ΔΗΜΟΚΡΑΤΙΑ	τρείς εικόνες νέα δημοκρατία και [redacted]
Ρέθυμνο	ΠΑΣΟΚ	αυτό ήταν εταιρεία αυτό ήταν το πασόκ. αναβολι...
Δημοτική Ενότητα Χολαργού	ΣΥΡΙΖΑ	1985 εκλογές. θα βγάλω φωτογραφία τον μητσοτάκη
ΠάνωΣτιςΣτέγεςΜέσαΣτιςΚαρδιές	ΠΑΣΟΚ	οι άγιοι προφήτες του πασοκ
Λάρισα	ΠΑΣΟΚ	πασόκ σημαίνει πρωτοπόρος
Καρδίτσα	ΕΛΛΗΝΙΚΗ ΛΥΣΗ	ορίστε ομολογούν τα [redacted] θα ζητήσου...
Βέροια	ΠΑΣΟΚ	εκλογές 2023 πασοκ σε σκέρτσο και νέα δημοκρατ...
Αθήνα	ΚΚΕ	το κκε[redacted]
Ναύπακτος	ΣΥΡΙΖΑ	τσίπρας ο σύριζα κατηγορεί τη νέα δημοκρατία γ...
Χαλκίδα	ΚΚΕ	τα προβλήματα των αγροτών δεν οφείλονται σε φυ...
Δημοτική Ενότητα Θεσαλονίκης	ΚΚΕ	κκε μονο φιλοι μου.

Εικόνα 6.2: Τμήμα αρχείου data.csv (η στήλη TweetText χωρίς θόρυβο)

6.4 Ανάθεση ετικετών (tagging)

Η λεγόμενη διαδικασία του ταγκαρίσματος (tagging), συνιστά ένα σημαντικό βήμα στην επεξεργασία και οργάνωση των δεδομένων. Στο πλαίσιο του NLP, το ταγκάρισμα αναφέρεται στην προσθήκη ετικετών σε φράσεις προκειμένου να κατηγοριοποιηθούν αυτές σε συγκεκριμένες κατηγορίες. Στην εργασία μας, η εν λόγω διαδικασία διήρκησε περίπου 2 μήνες και ταγκάραμε συνολικά πάνω από 4000 tweets.

Για την ανάθεση ετικετών ακολουθήθηκε η εξής διαδικασία. Όλα τα σχόλια κατηγοριοποιήθηκαν από εμάς με βάση το συναίσθημα που εκφράζουν. Όπως έχουμε δει, οι κατηγορίες είναι Positive, Negative και Neutral. Στη συνέχεια έγινε σύγκριση μεταξύ των ετικετών. Στις περισσότερες περιπτώσεις οι ετικέτες ήταν οι ίδιες. Σε περίπτωση διαφορών τα σχόλια περνούσαν από αξιολόγηση όπου τελικά αποφασιζόταν η τελική ετικέτα. Αν δεν μπορούσε να γίνει ξεκάθαρο ποια ετικέτα ήταν η ιδανικότερη, το σχόλιο αφαιρούνταν από την βάση δεδομένων.

Πρέπει να επισημανθεί πως η πολιτική είναι ένα θέμα υποκειμενικό. Πολλές φορές το συναίσθημα που εκφράζει ένα σχόλιο δεν είναι ξεκάθαρο και εξαρτάται κυρίως από τις πολιτικές αντιλήψεις του αναγνώστη. Στην δική μας περίπτωση προσπαθήσαμε να έχουμε όσο το δυνατόν αντικειμενικότερη άποψη. Ωστόσο, παροτρύνουμε τους ερευνητές που θα μελετήσουν το Politics dataset να έχουν το θέμα της υποκειμενικότητας στο μυαλό τους. Μετά την ολοκλήρωση του ταγκαρίσματος, η τελική όψη του dataset, πλέον περιλαμβάνει και την στήλη Sentiment με τιμές, όπως φαίνεται στην Εικόνα 6.3.

Location	Topic	TweetText	Sentiment
Κοζάνη	ΝΕΑ ΔΗΜΟΚΡΑΤΙΑ	τρείς εικόνες νέα δημοκρατία και [redacted]	Neutral
Ρέθυμνο	ΠΑΣΟΚ	αυτό ήταν εταιρεία αυτό ήταν το πασόκ. αναβολι...	Negative
Δημοτική Ενότητα Χολαργού	ΣΥΡΙΖΑ	1985 εκλογές. θα βγάλω φωτογραφία τον μητσοτάκη	Neutral
ΠάνωΣτιςΣτέγεςΜέσαΣτιςΚαρδιές	ΠΑΣΟΚ	οι άγιοι προφήτες του πασοκ	Neutral
Λάρισα	ΠΑΣΟΚ	πασόκ σημαίνει πρωτοπόρος	Positive
Καρδίτσα	ΕΛΛΗΝΙΚΗ ΛΥΣΗ	ορίστε ομολογούν τα[redacted] θα ζητήσου...	Negative
Βέροια	ΠΑΣΟΚ	εκλογές 2023 πασοκ σε σκέρτσο και νέα δημοκρατ...	Negative
Αθήνα	ΚΚΕ	το κκε στην[redacted]	Negative
Ναύπακτος	ΣΥΡΙΖΑ	τσίπρας ο σύριζα κατηγορεί τη νέα δημοκρατία γ...	Negative
Χαλκίδα	ΚΚΕ	τα προβλήματα των αγροτών δεν οφείλονται σε φυ...	Positive
Δημοτική Ενότητα Θεσαλονίκης	ΚΚΕ	κκε μονο φιλοι μου.	Positive

Εικόνα 6.3: Τμήμα dataset με στήλη Sentiment

6.5 Χαρτογράφηση tweets

Πολλά από τα tweets που συλλέξαμε, λόγω έλλειψης της πληροφορίας τοποθεσίας, δεν περιλαμβάνουν συντεταγμένες γεωγραφικού μήκους και πλάτους. Όπως αναφέραμε, τα εν λόγω tweets τα αγνοούμε. Συγκεκριμένα, ακολουθούμε την παρακάτω διαδικασία:

- Για κάθε τοποθεσία (Location), βρίσκουμε τις συντεταγμένες (αν υπάρχουν) και με βάση αυτές παίρνουμε την χώρα και την καταχωρούμε σε μια λίστα (lista_country).
- Παίρνουμε την πόλη και την καταχωρούμε σε μια λίστα (lista_city).
- Παίρνουμε την περιφέρεια και την καταχωρούμε σε μια λίστα (lista_state_district).
- Προβλέπουμε πως αν δεν βρεθούν συντεταγμένες για μια τοποθεσία, τότε οι λίστες lista_city, lista_country και lista_state_district, θα παίρνουν την τιμή None.
- Δημιουργούμε ένα νέο dataset, το οποίο πλέον δεν περιέχει την στήλη Location, αλλά 3 νέες στήλες, την λίστα country, city και state_district οι οποίες παραπέμπουν στις λίστες που δημιουργήσαμε νωρίτερα.
- Διαγραφή εγγραφών οι οποίες στις στήλες country, city και state_district έχουν την τιμή None. Με αυτό τον τρόπο οι απροσδιόριστες τιμές στο Location πλέον έχουν διαγραφεί, αφού δεν υπάρχουν συντεταγμένες για απροσδιόριστες τοποθεσίες.

Συνεπώς, ένα μέρος του τελικού dataset που προκύπτει φαίνεται παρακάτω στην Εικόνα 6.4.

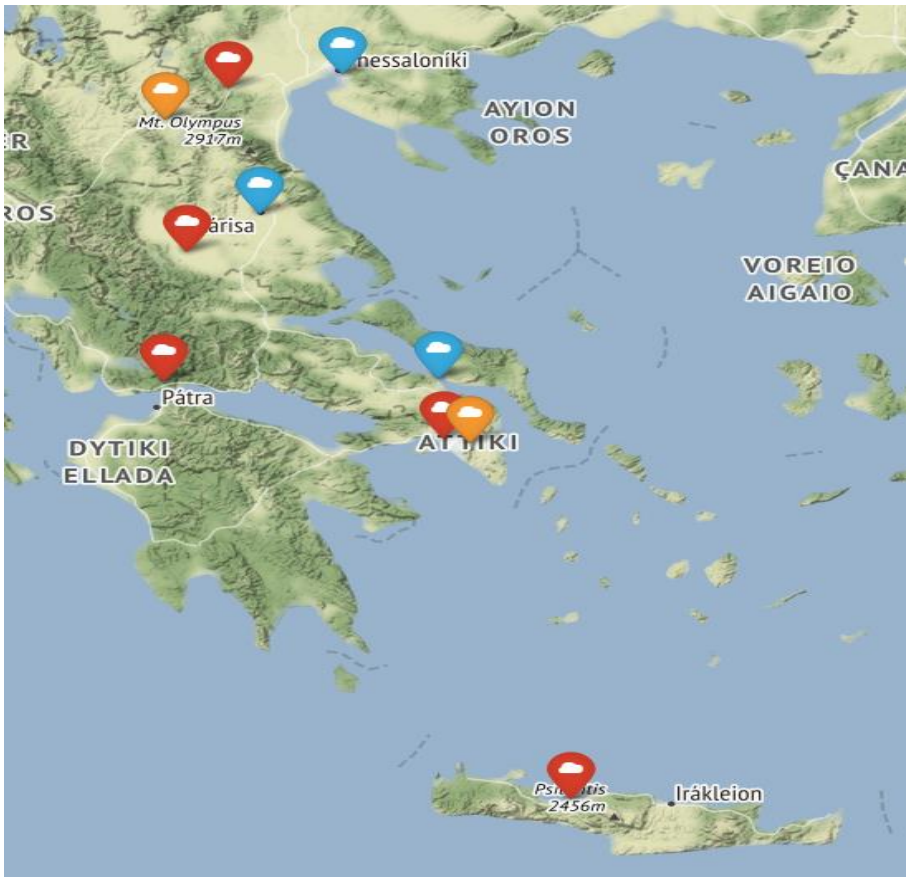
country	city	state_district	topic	tweet	sentiment
Ελλάς	Κοζάνη	Περιφέρεια Δυτικής Μακεδονίας	ΝΕΑ ΔΗΜΟΚΡΑΤΙΑ	τρείς εικόνες νέα δημοκρατία και [redacted]	Neutral
Ελλάς	Ρέθυμνο	Περιφέρεια Κρήτης	ΠΑΣΟΚ	αυτό ήταν εταιρεία αυτό ήταν το πασόκ. αναβολι...	Negative
Ελλάς	Δημοτική Ενότητα Χολαργού	Περιφέρεια Αττικής	ΣΥΡΙΖΑ	1985 εκλογές. θα βγάλω φωτογραφία τον μητσοτάκη	Neutral
Ελλάς	Λάρισα	Περιφέρεια Θεσσαλίας	ΠΑΣΟΚ	πασόκ σημαίνει πρωτοπόρος	Positive
Ελλάς	Καρδίτσα	Περιφέρεια Θεσσαλίας	ΕΛΛΗΝΙΚΗ ΛΥΣΗ	ορίστε ομολογούν τα [redacted] θα ζητήσου...	Negative
Ελλάς	Βέροια	Περιφέρεια Κεντρικής Μακεδονίας	ΠΑΣΟΚ	εκλογές 2023 πασοκ σε σκέρτσο και νέα δημοκρατ...	Negative
Ελλάς	Αθήνα	Περιφέρεια Αττικής	ΚΚΕ	το κκε στην [redacted]	Negative
Ελλάς	Δημοτική Ενότητα Ναυπάκτου	Περιφέρεια Δυτικής Ελλάδας	ΣΥΡΙΖΑ	τσίπρας ο σύριζα κατηγορεί τη νέα δημοκρατία γ...	Negative
Ελλάς	Χαλκίδα	Περιφέρεια Στερεάς Ελλάδας	ΚΚΕ	τα προβλήματα των αγροτών δεν οφείλονται σε φυ...	Positive
Ελλάς	Δημοτική Ενότητα Θεσαλονίκης	Περιφέρεια Κεντρικής Μακεδονίας	ΚΚΕ	κκε μονο φιλοι μου.	Positive

Εικόνα 6.4: Τμήμα τελικής μορφής dataset

Όπως βλέπουμε στην Εικόνα 6.4, η στήλη country έχει μόνο τιμές "Ελλάς". Αυτό συμβαίνει επειδή προβλέψαμε να κρατήσουμε tweets που προέρχονται μόνο από Ελλάδα. Η στήλη city, πήρε απλώς το λεκτικό του Location, και θα την χρησιμοποιήσουμε για τις ανάγκες της χαρτογράφησης. Η στήλη state_district αφορά την περιφέρεια, όπου δημοσιεύτηκε το κάθε tweet και την χρησιμοποιούμε στην ενότητα "Ανάλυση dataset". Οι στήλες tweet και sentiment, θα μας απασχολήσουν στα πειράματα μηχανικής μάθησης.

Στο σημείο αυτό έχουμε στην διάθεσή μας το τελικό dataset, το οποίο είναι πλέον απαλλαγμένο από θόρυβο και μπορεί να χρησιμοποιηθεί για περεταίρω εργασίες. Στο επόμενο στάδιο, έγινε η δημιουργία του διαδραστικού χάρτη που προβάλλει τα αρνητικά, θετικά και ουδέτερα tweets με την μορφή δεικτών (πινέζα) κόκκινων, μπλέ και πορτοκαλί αντίστοιχα. Με κλικ πάνω σε ένα δείκτη, θα εμφανίζεται το ψευδώνυμο του χρήστη και το κείμενο του tweet. Η Εικόνα 6.5 δείχνει τον χάρτη των tweets.

Για τις λειτουργίες γεωκωδικοποίησης, έγινε χρήση της βιβλιοθήκης geopy και της κλάσης Nominatim [107]. Η βιβλιοθήκη geopy παρέχει εργαλεία για γεωγραφική αναζήτηση (geocoding) και αντιστροφή γεωγραφικών αναφορών (reverse geocoding). Η κλάση Nominatim παρέχει δυνατότητες μετατροπής τοποθεσιών σε γεωγραφικές συντεταγμένες και μετατροπή συντεταγμένων σε τοποθεσίες. Επίσης, έγινε χρήση της βιβλιοθήκης Folium η οποία χρησιμοποιεί την βιβλιοθήκη χαρτογράφησης leaflet.js και προβάλει χάρτες ως αρχεία HTML [108].



Εικόνα 6.5: Χάρτης προβολής tweets

6.6 Ανάλυση βάσεων δεδομένων

Προφανώς, όλες οι παραπάνω εργασίες αφορούν τα tweets των πολιτικών κομμάτων (Politics). Στο Skroutz dataset [33] δεν εφαρμόστηκε κάποιος καθαρισμός, καθώς το συγκεκριμένο ήταν εξαρχής σε καλή κατάσταση. Παρακάτω, θα δείξουμε την δομή των datasets, κάποιες στατιστικές μετρήσεις καθώς και προβλήματα στο Politics που μας οδήγησαν στην κατασκευή νέας έκδοσης με ισορροπημένες (balanced) κλάσεις.

Όσον αφορά τα πολιτικά κόμματα, το σύνολο δεδομένων περιλαμβάνει συνολικά 8 στήλες (Created_at, Username, Country, City, State_District, Topic, Tweet, Sentiment), τις 6 από αυτές τις είδαμε στα προηγούμενα. Για λόγους ιδιωτικότητας δεν προβάλλαμε την στήλη Username. Η πρώτη στήλη (Created_at) αφορά την ημερομηνία δημοσίευσης του κάθε tweet. Συνολικά, το εν λόγω dataset περιέχει 4399 tweets. Από την άλλη το Skroutz dataset [33], περιλαμβάνει τρεις στήλες (id, Text, Sentiment) και περιέχει 6552 αξιολογήσεις χωρισμένες σε δύο κατηγορίες, θετικές και αρνητικές. Τμήμα του Politics είδαμε στις προηγούμενες ενότητες. Ο Πίνακας 6.1 δείχνει τμήμα του Skroutz dataset. Η στήλη id δεν μας ενδιαφέρει οπότε δεν εμφανίζεται.

Πίνακας 6.1: Αξιολογήσεις στο Skroutz dataset

Text	Sentiment
Όλα πήγαν καλά με την παραγγελία μου. Θα αγοράσω ξανά από το κατάστημα.	**Positive**
Πολύ ευχαριστημένος με την εταιρία!!! όλα τέλεια...ένα μικρό θέμα υπήρξε με την μεταφορική το οποίο λύθηκε άμεσα. Η εξυπηρέτηση σωστή και οι τιμές εξαιρετικές σε σχέση με άλλα καταστήματα. Ήταν η πρώτη φορά που παρήγγειλα από εκεί...στην δεύτερη θα επιβεβαιωθεί πραγματικά η θετική εμπειρία μου.	**Positive**
Έκανα τη παραγγελία πήρα άμεση επιβεβαίωση άμεσα έλαβα μήνυμα να κανονίσω ραντεβού παραλαβής από το κατάστημα που ήθελα πήγα στην έδρα τους και σε 10' είχα πληρώσει και παραλάβει. Όλη διαδικασία διήρκησε 2 εργάσιμες. Μπράβο σε όλους τους υπαλλήλους που ενήργησαν άμεσα και εξυπηρετικά.	**Positive**
Πήρα τηλέφωνο να παραγγείλω μια επιτραπέζια εστία την οποία έγραφαν ότι είναι άμεσα διαθέσιμη στην αποθήκη τους. Με ενημέρωσαν να πάω το απόγευμα να παραλάβω μετά τις 17.30.. Πήγα οχτώ πάρα και μετά από αναμονή με ενημέρωσαν ότι δεν είχε έρθει από την αποθήκη τους.. Απαράδεκτη συμπεριφορά όφειλαν να με ενημερώσουν.. Φυσικά δεν θα ξανα προτιμήσω το εν λόγω κατάστημα..	**Negative**
Πριν από 10 ημέρες αγόρασα μια TV lg 43 ιντσών και πλήρωσα με κάρτα. Η παράδοση έγινε 3 ημέρες μετά όμως όχι σε μένα αλλά σε γείτονα ο οποίος δεν είχε καμία εξουσιοδότηση από εμένα για να παραλάβει. Οι μεταφορείς άφησαν την TV στο πεζόδρομο και έφυγαν. Όταν η τηλεόραση ήρθε στα χέρια μου διαπίστωσα ότι η οθόνη ήταν σπασμένη. Κοιτάζοντας προσεκτικά τη συσκευασία στο ίδιο σημείο είχε ένα μικρό σχίσιμο. Μίλησα με την εταιρεία οι οποίοι άργησαν να επικοινωνήσουν μαζί μου και όχι μόνο δεν δέχτηκαν να αντικαταστήσουν το προϊόν αλλά έδειξαν και απαξίωση.	**Negative**

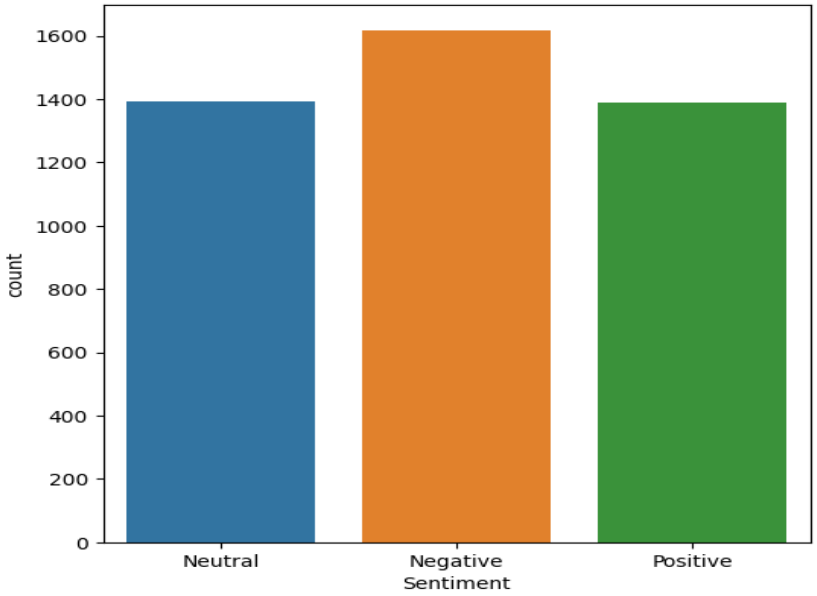
6.6.1 Στατιστικές μετρήσεις

Οι περισσότερες στατιστικές μετρήσεις αφορούν το Politics, καθώς για αυτό έχουμε συλλέξει αρκετές πληροφορίες. Αντίθετα, το Skroutz dataset με μόλις 2 στήλες, μας περιορίζει στο να οπτικοποιήσουμε κυρίως κατανομές λέξεων και ραβδογράμματα πλήθους tweets σε κάθε κλάση.

Όπως βλέπουμε στην Εικόνα 6.6 και Εικόνα 6.7, το Politics περιλαμβάνει 1618 Negative, 1393 Neutral και 1388 Positive. Πρόκειται για μη ισορροπημένες (unbalanced) κλάσεις, αφού η κάθε μια έχει διαφορετικό πλήθος tweets.

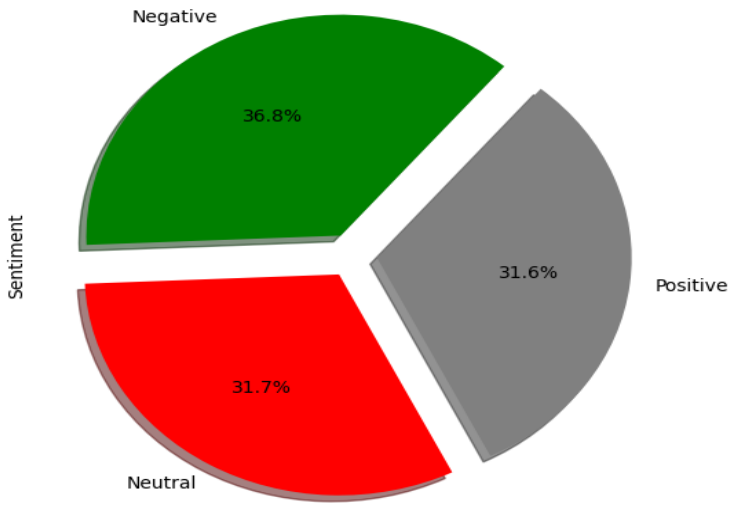
```
Sentiment
Negative    1618
Neutral     1393
Positive    1388
Name: count, dtype: int64
```

Εικόνα 6.6: Οι κλάσεις δεν είναι ισορροπημένες



Εικόνα 6.7: Οπτικοποίηση κλάσεων

Το διάγραμμα πίτας δείχνει το ποσοστό των tweets για κάθε κλάση.



Εικόνα 6.8: Ποσοστά κλάσεων

Προφανώς, το μεγαλύτερο ποσοστό το καταλαμβάνει η κλάση Negative, ενώ οι άλλες δύο κλάσεις είναι πολύ κοντά.

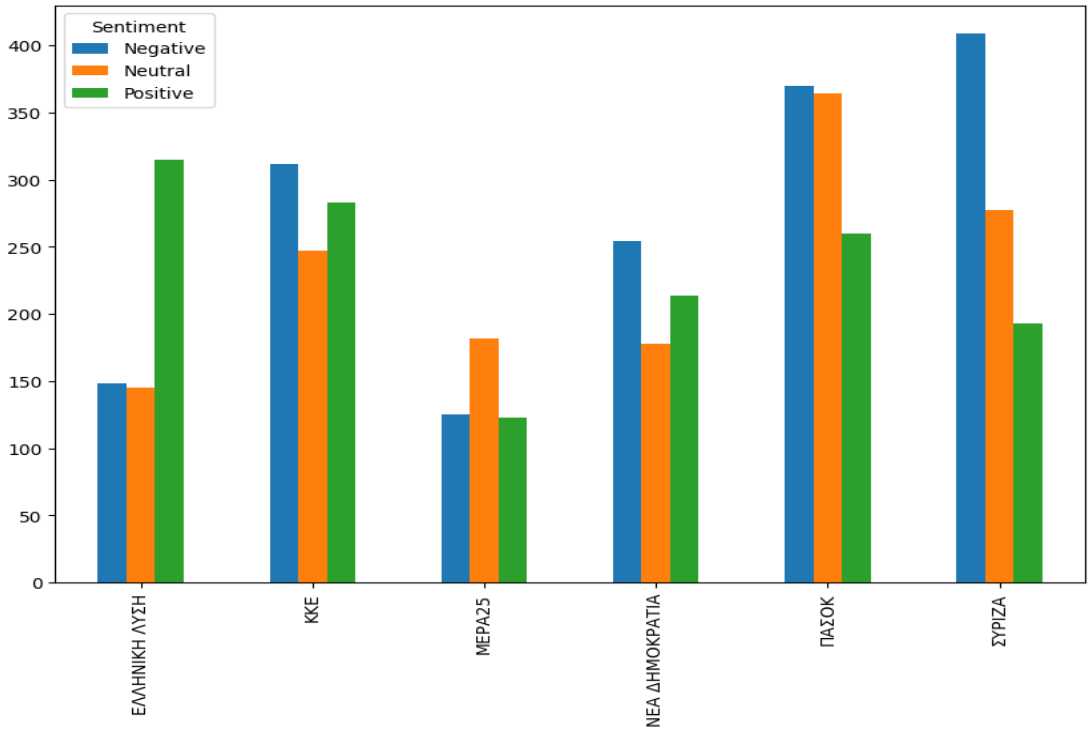
Στην Εικόνα 6.9, φαίνεται το πλήθος των tweets που έλαβε το κάθε πολιτικό κόμμα και στην Εικόνα 6.10, το πλήθος ανά κλάση, το οποίο οπτικοποιείται και στο ραβδόγραμμα της Εικόνας 6.11.

```
ΠΑΣΟΚ                994
ΣΥΡΙΖΑ               879
ΚΚΕ                  842
ΝΕΑ ΔΗΜΟΚΡΑΤΙΑ       646
ΕΛΛΗΝΙΚΗ ΛΥΣΗ        608
ΜΕΡΑ25               430
Name: Topic, dtype: int64
```

Εικόνα 6.9: Αριθμός tweets για κάθε κόμμα

Sentiment Topic	Negative	Neutral	Positive
ΕΛΛΗΝΙΚΗ ΛΥΣΗ	148	145	315
ΚΚΕ	312	247	283
ΜΕΡΑ25	125	182	123
ΝΕΑ ΔΗΜΟΚΡΑΤΙΑ	254	178	214
ΠΑΣΟΚ	370	364	260
ΣΥΡΙΖΑ	409	277	193

Εικόνα 6.10: Αριθμός tweets ανά κλάση

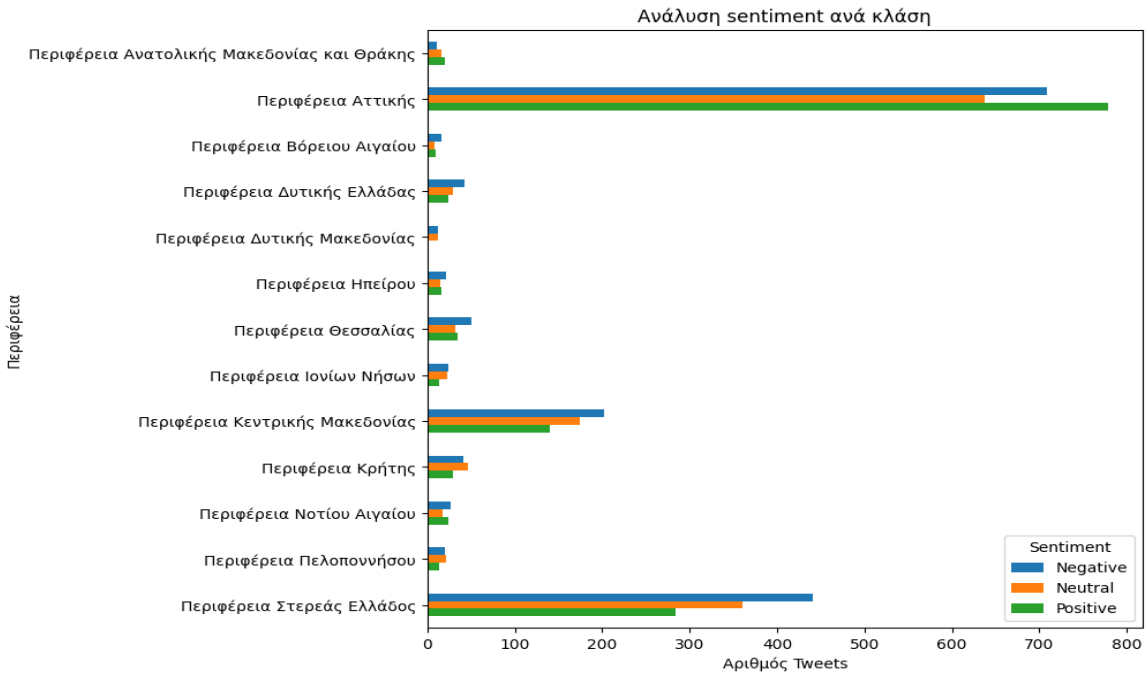


Εικόνα 6.11: Οπτικοποίηση αριθμού tweets ανά κλάση

Στην Εικόνα 6.12 φαίνεται ο αριθμός tweets ανά κλάση σε κάθε περιφέρεια της Ελλάδας. Το ίδιο οπτικοποιείται σε ραβδόγραμμα στην Εικόνα 6.13.

Sentiment State_District	Negative	Neutral	Positive
Περιφέρεια Ανατολικής Μακεδονίας και Θράκης	11	16	20
Περιφέρεια Αττικής	709	637	779
Περιφέρεια Βόρειου Αιγαίου	16	8	10
Περιφέρεια Δυτικής Ελλάδας	42	29	24
Περιφέρεια Δυτικής Μακεδονίας	12	12	1
Περιφέρεια Ηπείρου	22	15	16
Περιφέρεια Θεσσαλίας	51	32	34
Περιφέρεια Ιονίων Νήσων	24	23	14
Περιφέρεια Κεντρικής Μακεδονίας	202	174	140
Περιφέρεια Κρήτης	41	46	29
Περιφέρεια Νοτίου Αιγαίου	27	18	24
Περιφέρεια Πελοποννήσου	20	22	13
Περιφέρεια Στερεάς Ελλάδος	441	361	284

Εικόνα 6.12: Αριθμός tweets ανά κλάση για κάθε περιφέρεια

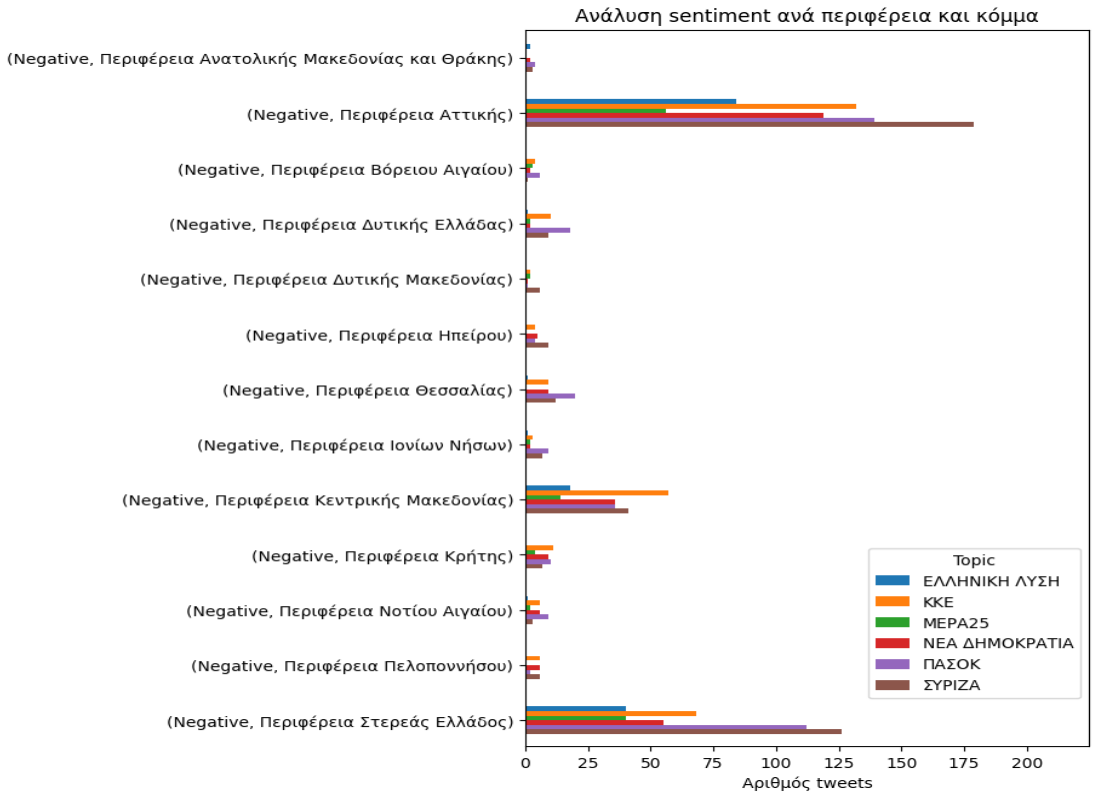


Εικόνα 6.13: Οπτικοποίηση αριθμού tweets για κάθε περιφέρεια

Οι επόμενες 6 εικόνες, παρουσιάζουν τον αριθμό των tweets ανά περιφέρεια και κόμμα για κάθε κλάση.

Sentiment	State_District	ΕΛΛΗΝΙΚΗ ΛΥΣΗ	ΚΚΕ	ΜΕΡΑ25	ΝΕΑ ΔΗΜΟΚΡΑΤΙΑ	ΠΑΣΟΚ	ΣΥΡΙΖΑ
Negative	Περιφέρεια Ανατολικής Μακεδονίας και Θράκης	2	0	0	2	4	3
	Περιφέρεια Αττικής	84	132	56	119	139	179
	Περιφέρεια Βόρειου Αιγαίου	0	4	3	2	6	1
	Περιφέρεια Δυτικής Ελλάδας	1	10	2	2	18	9
	Περιφέρεια Δυτικής Μακεδονίας	0	2	2	1	1	6
	Περιφέρεια Ηπείρου	0	4	0	5	4	9
	Περιφέρεια Θεσσαλίας	1	9	0	9	20	12
	Περιφέρεια Ιονίων Νήσων	1	3	2	2	9	7
	Περιφέρεια Κεντρικής Μακεδονίας	18	57	14	36	36	41
	Περιφέρεια Κρήτης	0	11	4	9	10	7
	Περιφέρεια Νοτίου Αιγαίου	1	6	2	6	9	3
	Περιφέρεια Πελοποννήσου	0	6	0	6	2	6
	Περιφέρεια Στερεάς Ελλάδος	40	68	40	55	112	126

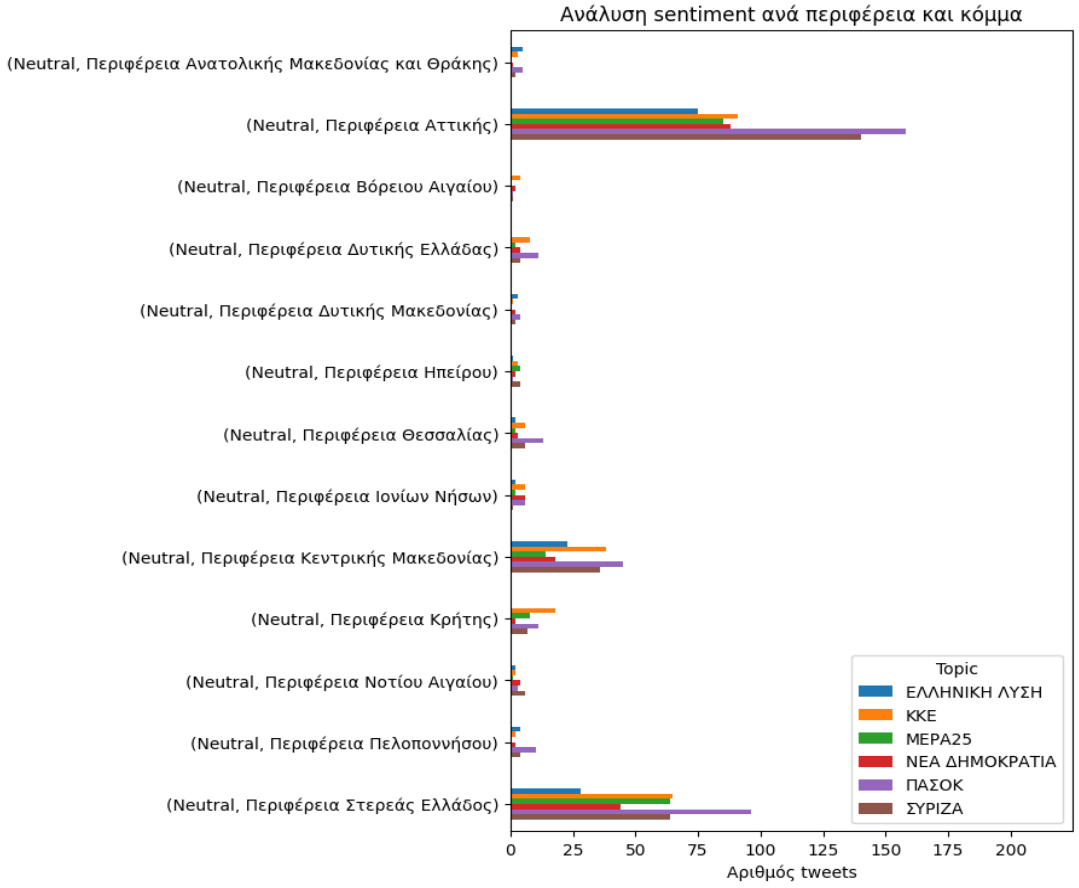
Εικόνα 6.14: Αριθμός των tweets ανά περιφέρεια και κόμμα (κλάση Negative)



Εικόνα 6.15: Οπτικοποίηση αριθμού tweets ανά περιφέρεια και κόμμα (κλάση Negative)

Sentiment	State_District	Topic	ΕΛΛΗΝΙΚΗ ΛΥΣΗ	ΚΚΕ	ΜΕΡΑ25	ΝΕΑ ΔΗΜΟΚΡΑΤΙΑ	ΠΑΣΟΚ	ΣΥΡΙΖΑ
Neutral	Περιφέρεια Ανατολικής Μακεδονίας και Θράκης		5	3	0	1	5	2
	Περιφέρεια Αττικής		75	91	85	88	158	140
	Περιφέρεια Βόρειου Αιγαίου		0	4	0	2	1	1
	Περιφέρεια Δυτικής Ελλάδας		0	8	2	4	11	4
	Περιφέρεια Δυτικής Μακεδονίας		3	1	0	2	4	2
	Περιφέρεια Ηπείρου		1	3	4	2	1	4
	Περιφέρεια Θεσσαλίας		2	6	2	3	13	6
	Περιφέρεια Ιονίων Νήσων		2	6	2	6	6	1
	Περιφέρεια Κεντρικής Μακεδονίας		23	38	14	18	45	36
	Περιφέρεια Κρήτης		0	18	8	2	11	7
	Περιφέρεια Νοτίου Αιγαίου		2	2	1	4	3	6
	Περιφέρεια Πελοποννήσου		4	2	0	2	10	4
	Περιφέρεια Στερεάς Ελλάδος		28	65	64	44	96	64

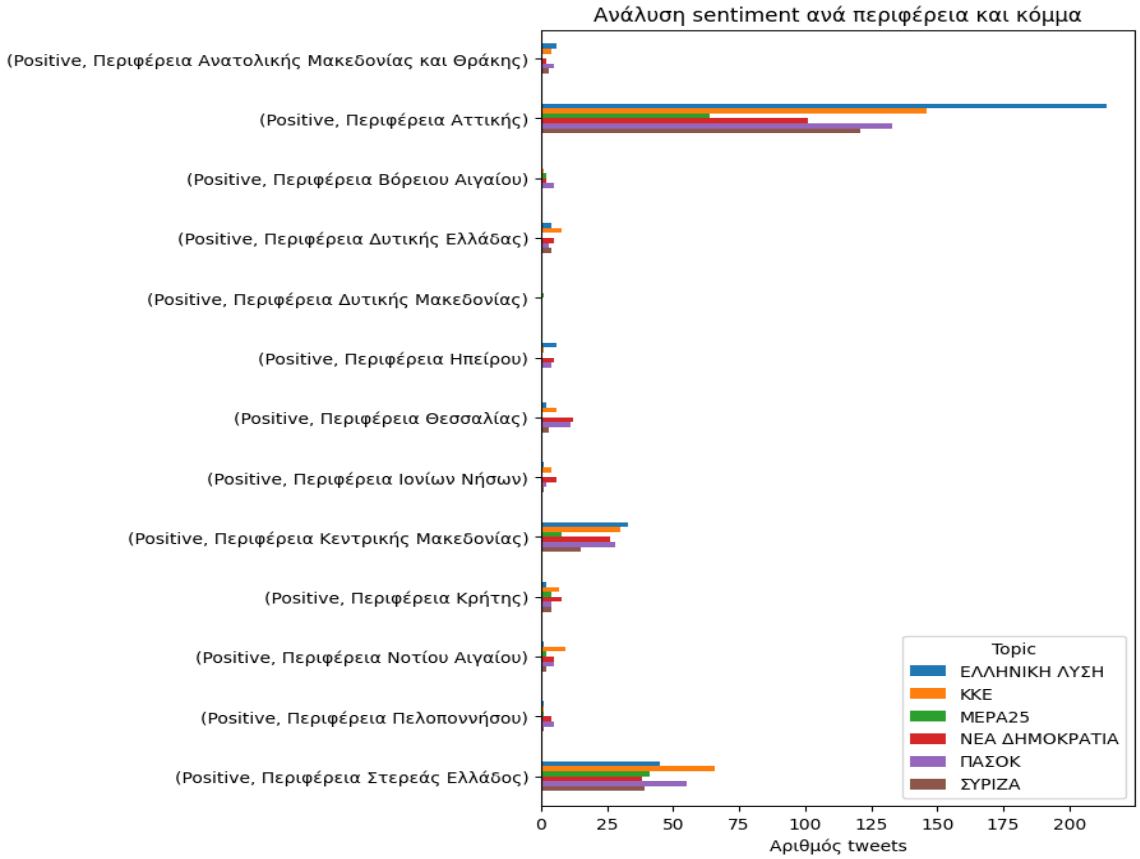
Εικόνα 6.16: Αριθμός των tweets ανά περιφέρεια και κόμμα (κλάση Neutral)



Εικόνα 6.17: Οπτικοποίηση αριθμού tweets ανά περιφέρεια και κόμμα (κλάση Neutral)

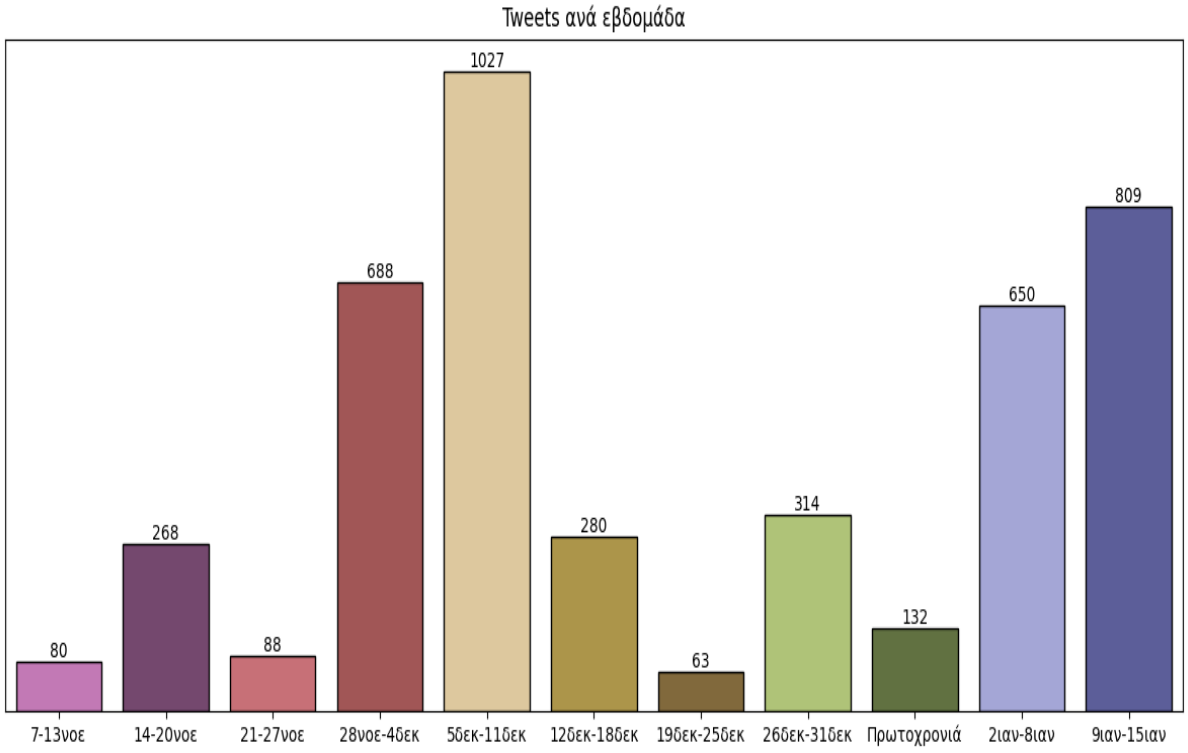
Sentiment	State_District	Topic ΕΛΛΗΝΙΚΗ ΛΥΣΗ	ΚΚΕ	ΜΕΡΑ25	ΝΕΑ ΔΗΜΟΚΡΑΤΙΑ	ΠΑΣΟΚ	ΣΥΡΙΖΑ
Positive	Περιφέρεια Ανατολικής Μακεδονίας και Θράκης	6	4	0	2	5	3
	Περιφέρεια Αττικής	214	146	64	101	133	121
	Περιφέρεια Βόρειου Αιγαίου	0	1	2	2	5	0
	Περιφέρεια Δυτικής Ελλάδας	4	8	0	5	3	4
	Περιφέρεια Δυτικής Μακεδονίας	0	0	1	0	0	0
	Περιφέρεια Ηπείρου	6	1	0	5	4	0
	Περιφέρεια Θεσσαλίας	2	6	0	12	11	3
	Περιφέρεια Ιονίων Νήσων	1	4	0	6	2	1
	Περιφέρεια Κεντρικής Μακεδονίας	33	30	8	26	28	15
	Περιφέρεια Κρήτης	2	7	4	8	4	4
	Περιφέρεια Νοτίου Αιγαίου	1	9	2	5	5	2
	Περιφέρεια Πελοποννήσου	1	1	1	4	5	1
	Περιφέρεια Στερεάς Ελλάδος	45	66	41	38	55	39

Εικόνα 6.18: Αριθμός των tweets ανά περιφέρεια και κόμμα (κλάση Positive)



Εικόνα 6.19: Οπτικοποίηση αριθμού tweets ανά περιφέρεια και κόμμα (κλάση Positive)

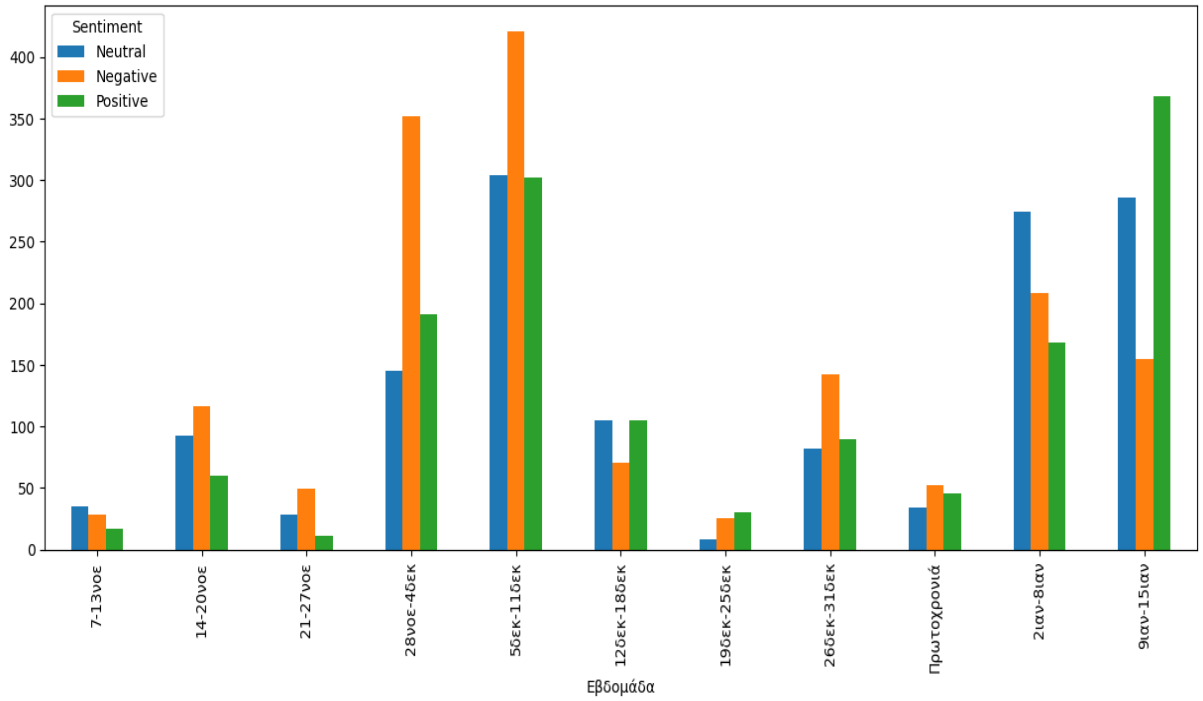
Η διαδικασία της εξόρυξης ξεκίνησε στις 07/11/2022 και ολοκληρώθηκε στις 15/01/2023. Η Εικόνα 6.20 δείχνει τον αριθμό των tweets συνολικά, και οι εικόνες 6.21 και 6.22, τον αριθμό tweets ανά κλάση για κάθε εβδομάδα.



Εικόνα 6.20: Οπτικοποίηση αριθμού tweets για κάθε εβδομάδα

Sentiment	Neutral	Negative	Positive
Created_at			
7-13νοε	35	28	17
14-20νοε	92	116	60
21-27νοε	28	49	11
28νοε-4δεκ	145	352	191
5δεκ-11δεκ	304	421	302
12δεκ-18δεκ	105	70	105
19δεκ-25δεκ	8	25	30
26δεκ-31δεκ	82	142	90
Πρωτοχρονιά	34	52	46
2ιαν-8ιαν	274	208	168
9ιαν-15ιαν	286	155	368

Εικόνα 6.21: Αριθμός tweets ανά κλάση για κάθε εβδομάδα

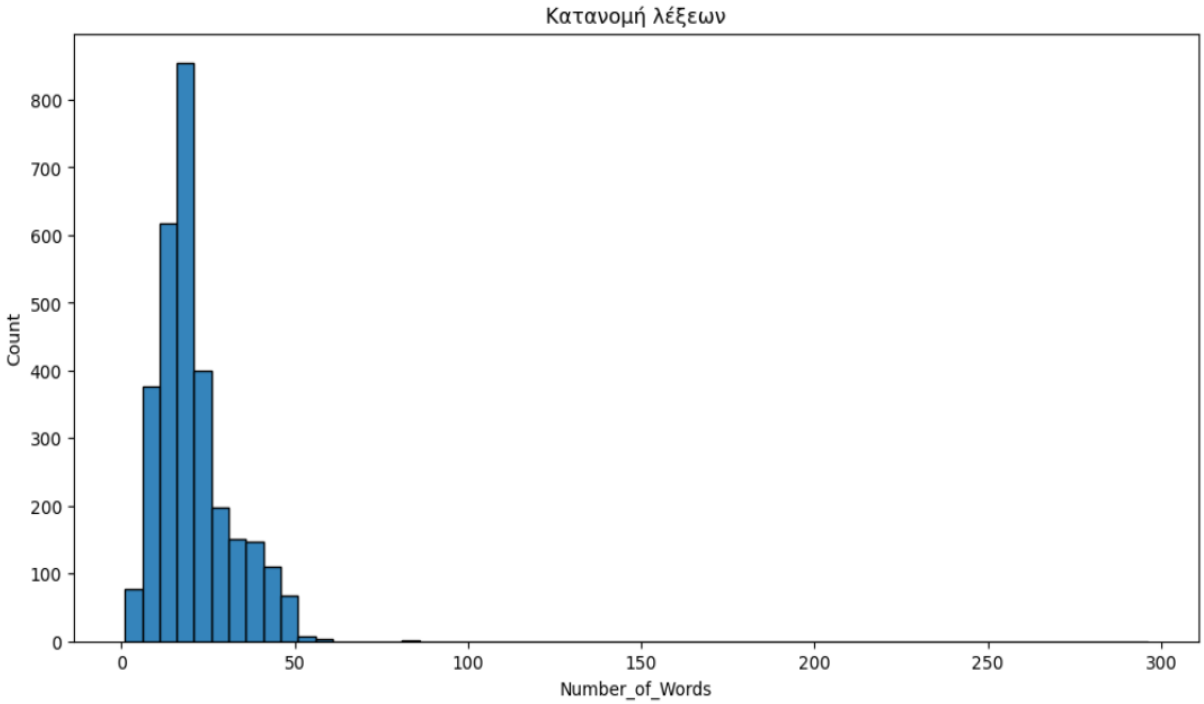


Εικόνα 6.22: Οπτικοποίηση αριθμού tweets ανά κλάση για κάθε εβδομάδα

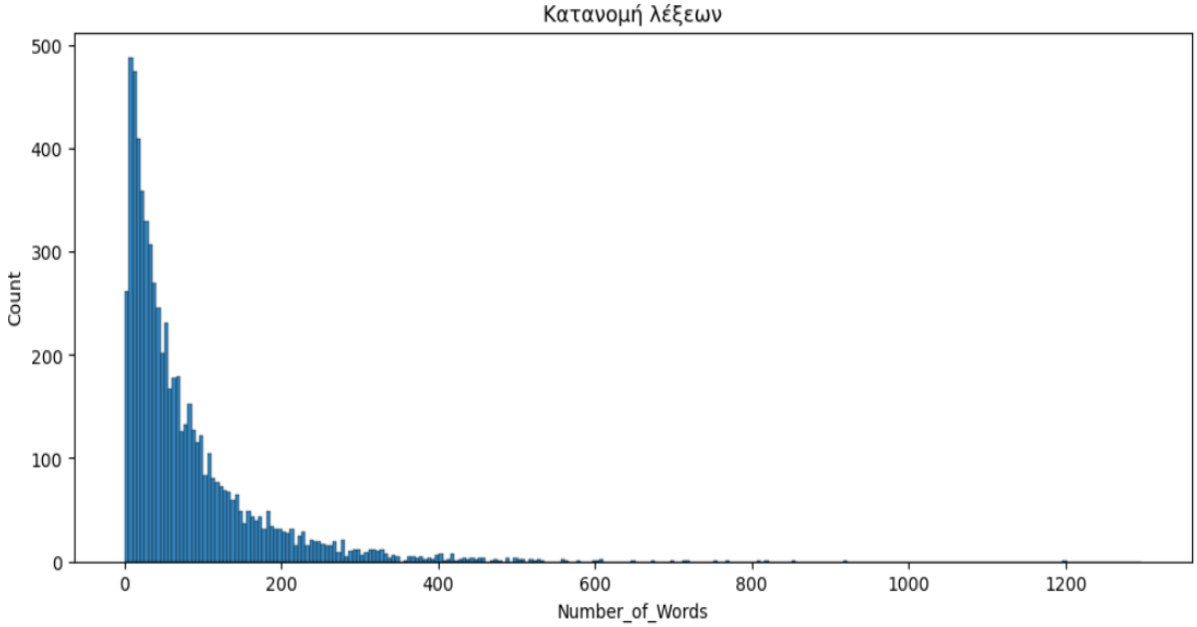
Ιδιαίτερη σημασία αξίζει να δοθεί στον πίνακα 6.2, όσον αφορά τον μέσο αριθμό tokens αλλά και τον μέγιστο αριθμό tokens. Η ανάλυση αυτών των μετρικών είναι σημαντική διότι μας συμβουλεύει για την επιλογή του μέγιστου μεγέθους ακολουθίας, αυτό που αναφέραμε σαν seq_length στο κεφαλαίο 5, στην εφαρμογή των Transformers αλλά και του Word2Vec διανυσματοποιητή. Βέβαια, για την καλύτερη δυνατή επιλογή της παραπάνω υπερπαραμέτρου είναι καίρια και η γνώση της κατανομής του πλήθους των λέξεων (εικόνες 6.23 και 6.24). Η παραπάνω διεργασία επιτυγχάνεται με διαίρεση του κειμένου σε λέξεις, με χρήση της βιβλιοθήκης NLTK. Οι τιμές που επιλέχθηκαν θα αναλυθούν στην ενότητα των πειραμάτων, όπου θα αναφέρουμε αναλυτικότερα τις υπερπαραμέτρους των πειραμάτων μας.

Πίνακας 6.2: Χαρακτηριστικά των Datasets

Dataset	Comments	Average Number Of Tokens	Max Number Of Tokens	Negative	Neutral	Positive
Skroutz	6552	84	1370	3276	-	3276
Unbalanced Politics	4399	21.515	91	1618	1393	1388

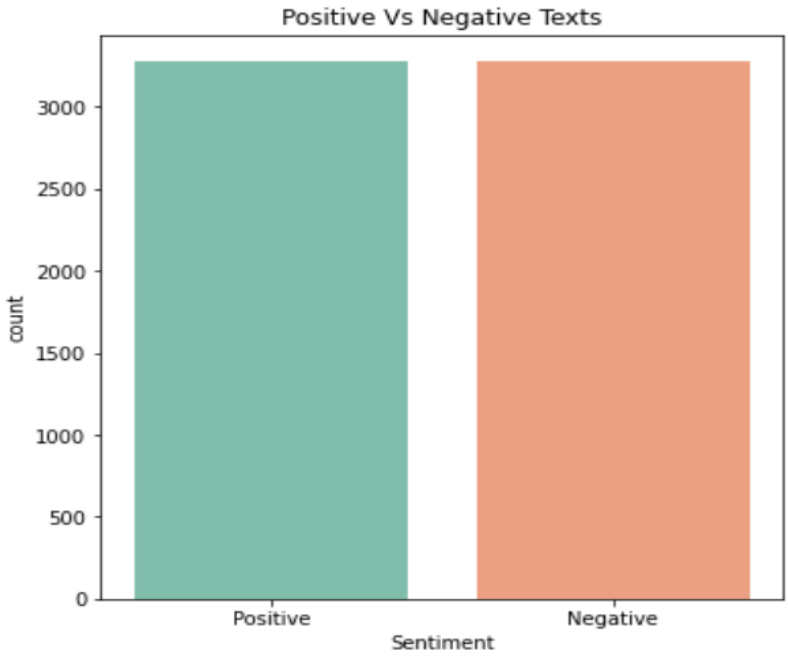


Εικόνα 6.23: Κατανομή λέξεων στο Politics (Unbalanced)



Εικόνα 6.24: Κατανομή λέξεων στο Skroutz

Ο Πίνακας 6.2, επίσης μας πληροφορεί για τον αριθμό των κριτικών στο Skroutz dataset. Είναι φανερό ότι πρόκειται για ένα ισορροπημένο (balanced) σύνολο δεδομένων. Μια οπτικοποίησή του φαίνεται στην Εικόνα 6.25.



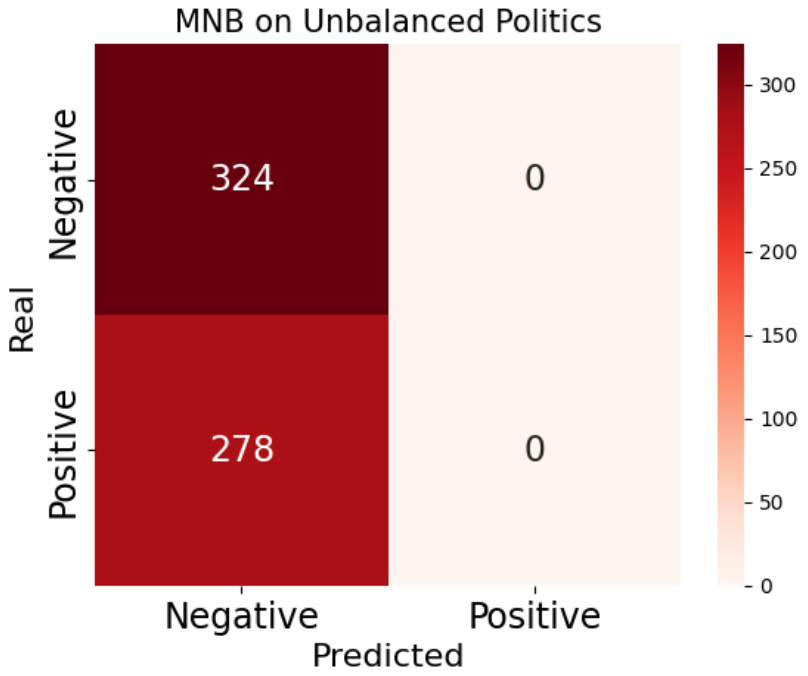
Εικόνα 6.25: Ισορροπημένο (Balanced) Skroutz dataset

Στη συνέχεια, με την βοήθεια της βιβλιοθήκης NLTK αφαιρούμε τα StopWords, εφαρμόζουμε λημματοποιήση στις λέξεις με το simplemma και κατασκευάζουμε τις οπτικοποιήσεις WordCloud (Εικόνα 6.26). Οι εν λόγω οπτικοποιήσεις, μας παρουσιάζουν τις συχνότερα εμφανιζόμενες λέξεις και μας πληροφορούν για την γενικότερη αλλά και ειδικότερη θεματολογία των βάσεων δεδομένων.

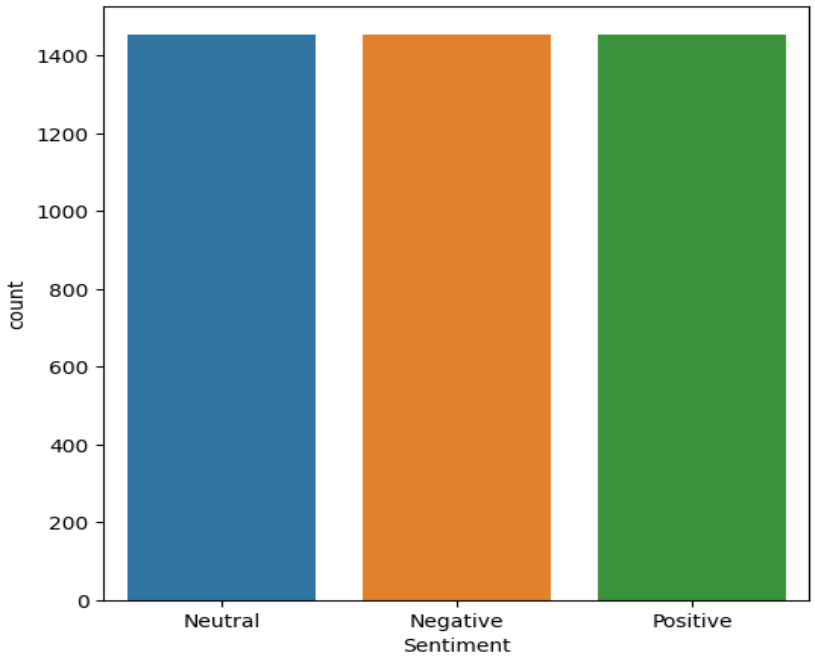
Εικόνα 6.26: Οπτικοποιήσεις WordCloud

6.6.2 Η Εξέλιξη του Politics Dataset: Από το Unbalanced στο Balanced

Όπως αναφέρθηκε, κατά την εξόρυξη και συλλογή των tweets προέκυψε ένα σύνολο δεδομένων με μη ισορροπημένες κλάσεις (unbalanced politics) εφόσον ο αριθμός των tweets μεταξύ των κλάσεων ήταν διαφορετικός. Κατά την διάρκεια των πειραμάτων, διαπιστώθηκε πως ο αλγόριθμος MNB κατηγοριοποιούσε μόνο την αρνητική κλάση, δηλαδή αυτή με τα περισσότερα δείγματα (Εικόνα 6.27). Ο ίδιος αλγόριθμος στο Skroutz Dataset (που είναι balanced), δεν παρουσίαζε την ίδια συμπεριφορά κατηγοριοποιώντας ορθότερα τα αποτελέσματα. Με αφορμή αυτή την παρατήρηση, δημιουργήσαμε μία δεύτερη έκδοση του Politics της οποίας οι κλάσεις είναι ίσες σε πλήθος προτάσεων (tweets). Έτσι, αφαιρέσαμε κάποια αρνητικά σχόλια, και σε συνδυασμό με μια στοχευμένη εξόρυξη για θετικά (από την οποία προέκυψαν και ουδέτερα) σχόλια τελικά κατασκευάστηκε η εναλλακτική μορφή του Politics με ισορροπημένες κλάσεις (Εικόνα 6.28). Ο Πίνακας 6.3 περιλαμβάνει τα χαρακτηριστικά των datasets συμπεριλαμβανομένης και της έκδοσης balanced.



Εικόνα 6.27: Παράδειγμα πίνακα σύγχυσης αλγορίθμου MNB σε δυαδική κατηγοριοποίηση στο Unbalanced Politics



Εικόνα 6.28: Politics με ισορροπημένες κλάσεις

Πίνακας 6.3: Χαρακτηριστικά συμπεριλαμβανομένου και του Balanced

Dataset	Comments	Average Number Of Tokens	Max Number Of Tokens	Negative	Neutral	Positive
Skroutz	6552	84	1370	3276	-	3276
Unbalanced Politics	4399	21.515	91	1618	1393	1388
Balanced Politics	4365	21.925	91	1455	1455	1455

6.7 Προ-επεξεργασία δεδομένων

Για να προχωρήσουμε στις εφαρμογές των πειραμάτων, έπρεπε να αποφασίσουμε τι άλλου είδους προ- επεξεργασία θα εφαρμοστεί στις βάσεις δεδομένων. Έτσι, αποφασίστηκε να δημιουργήσουμε τις εξής υποπεριπτώσεις:

- **Κατηγοριοποίηση**: Εφαρμόζουμε δυαδική και τριαδική κατηγοριοποίηση στα Politics. Στην δυαδική κατηγοριοποίηση απλώς αφαιρούμε τα ουδέτερα tweets. Το Skroutz Dataset περιέχει μόνο δύο κλάσεις, οπότε εκεί εφαρμόζεται μόνο δυαδική κατηγοριοποίηση.

- **Lemmatization/StopWords**: Επιλέγουμε στα πειράματα, είτε να εφαρμόσουμε λημματοποίηση των λέξεων και αφαίρεση των StopWords, είτε όχι. Η επιλογή αυτή έγινε ώστε να μειώσουμε των αριθμό των πειραμάτων.
- **Διαχωρισμός Δεδομένων**: Εφαρμόζουμε διαχωρισμό στα δεδομένα ώστε να τα χωρίσουμε σε δείγματα εκπαίδευσης και δείγματα ελέγχου. Εκτελέσαμε τρεις περιπτώσεις, όπου έγινε τυχαίος διαχωρισμός 70% (train set) / 30% (test set), τυχαίος διαχωρισμός 80% / 20% και ένας συγκεκριμένος διαχωρισμός 80% / 20%. Στην περίπτωση του συγκεκριμένου διαχωρισμού δεν γίνεται random split. Χρησιμοποιούνται 2 αρχεία, ένα για το train set και ένα για το test set, με τα μοντέλα μηχανικής μάθησης. Για τα μοντέλα transformers, χρησιμοποιείται επιπλέον και ένα αρχείο επικύρωσης (validation data). Αξίζει να αναφέρουμε πως στους τυχαίους διαχωρισμούς γίνεται χρήση του ίδιου SEED, για να δημιουργούνται κάθε φορά τα ίδια splits, προκειμένου να αποφευχθούν τυχόν αποκλίσεις στις συγκρίσεις των αποτελεσμάτων. Στην περίπτωση των transformers, γίνεται και επιπλέον διαχωρισμός στα δεδομένα εκπαίδευσης, πάντοτε 80% / 20 %, ώστε να προκύψουν και δεδομένα επικύρωσης (20%). Με αυτό τον τρόπο εξασφαλίζεται πως τα δείγματα εκπαίδευσης, τα δείγματα επικύρωσης και ελέγχου θα είναι πάντα τα ίδια για όλα τα μοντέλα, με συνέπεια να έχουμε ακριβείς συγκρίσεις. Για την εφαρμογή random split, χρησιμοποιούμε την συνάρτηση train_test_split() της βιβλιοθήκης Scikit-learn [109].
- **Διανυσματοποιητές (Tokenizers)**: Στην περίπτωση των transformers δεν διαχωρίζουμε υποπεριπτώσεις διανυσματοποιητών, καθώς κάθε μοντέλο που μας παρέχεται από το Hugging Face περιέχει και τον κατάλληλο διανυσματοποιητή λέξεων. Ωστόσο, όταν εκτελούμε πειράματα μηχανικής μάθησης κάνουμε χρήση δύο διακριτών διανυσματοποιητών του TF-IDF και του Word2Vec.

Για να γίνει πιο κατανοητή η παραπάνω μεθοδολογία, αναφέρουμε 2 παραδείγματα πειραμάτων.

Πείραμα 1 - επιλογές: Balanced_Politics, Multi Classification, No SL, Random_split_80_20, SVM, TF-IDF.

Πείραμα 2- επιλογές: Balanced_Skroutz, Binary Classification, Yes SL, Standard split, BERT.

Συγκεντρωτικά, εάν λάβουμε υπόψιν μας όλες τις υποπεριπτώσεις, ο αριθμός των τελικών διακριτών πειραμάτων που εκτελέσαμε ανέρχεται στα 468.

6.8 Αρχιτεκτονική και υπερ-παράμετροι μοντέλων

Μέχρι στιγμής έχουμε καλύψει το μεγαλύτερο μέρος της γνώσης για την εκτέλεση μεγάλης γκάμας πειραμάτων μηχανικής και βαθιάς μάθησης. Είναι σημαντικό τα πειράματα να μπορούν να αναπαραχθούν από άλλους ερευνητές. Για να επιτευχθεί αυτό, θα πρέπει να αναφέρουμε τις παραμέτρους που κάνουν χρήση τα μοντέλα μας προτού προβούν στη φάση της εκπαίδευσης. Οι παράμετροι αυτοί, όπως έχουμε ξανά αναφέρει, ονομάζονται και **υπερ-παράμετροι**. Καθορίζουν σε μεγάλο βαθμό την επιτυχία ενός πειράματος και η προσεκτική επιλογή τους είναι καίριας σημασίας. Όσα θα αναφερθούν στη παρούσα ενότητα, μπορείτε να τα μελετήσετε στον κώδικα του Παραρτήματος Α στην ενότητα "κώδικας 6ου κεφαλαίου".

Ας διαχωρίσουμε τις περιπτώσεις μεταξύ πειραμάτων μηχανικής μάθησης και βαθιάς μάθησης. Στην μηχανική μάθηση σε πρώτη φάση μας ενδιαφέρουν οι διανυσματοποιητές. Έχουμε αναφέρει στις ενότητες 4.9.1 και 4.9.2 για τις δυο μεθόδους διανυσματοποίησης που χρησιμοποιήσαμε. Για τις ανάγκες των βάσεων δεδομένων μας, επιλέγουμε στην μέθοδο TF-IDF σαν max_features το 500.000. Ο αριθμός αυτός υποδεικνύει τον μέγιστο αριθμό διαστάσεων που μπορεί να έχει το διάνυσμα των προτάσεων δοθείσας της βάσης δεδομένων. Για παράδειγμα, στην χρήση του TF-IDF

διανυσματοποιητή στο unbalanced politics, κάθε διάνυσμα πρότασης έχει 13.450 διαστάσεις, δηλαδή όσες και το πλήθος των λέξεων στο λεξικό.

Με σκοπό την εφαρμογή της Word2Vec μεθοδολογίας έγινε χρήση προ-εκπαιδευμένου μοντέλου [110] στα ελληνικά για τη βιβλιοθήκη fasttext. Με την βιβλιοθήκη gensim [11], γίνεται η φόρτωση των δεδομένων και στην συνέχεια η διανυσματοποίηση των προτάσεων. Χρησιμοποιούμε το όρισμα limit με τιμή 1000000 στην μέθοδο load_word2vec_format() για να δηλώσουμε τον μέγιστο αριθμό λέξεων που θα φορτωθούν από το προ-εκπαιδευμένο μοντέλο. Η προεπιλεγμένη τιμή είναι 2000000. Επίσης το μοντέλο για κάθε διάνυσμα λέξης, χρησιμοποιεί διάσταση 300 χαρακτηριστικών. Τέλος, στην περίπτωση των Politics Datasets ορίζουμε τον μέγιστο αριθμό λέξεων/tokens στο 80, ενώ για το Skroutz Dataset θέτουμε τον αντίστοιχο αριθμό στο 150. Οι τελευταίοι δύο αριθμοί προκύπτουν από την στατιστική μελέτη στα tokens των βάσεων δεδομένων μας, κάτι που είδαμε στην ενότητα 6.6.

Ολοκληρώνοντας για την περίπτωση της μηχανικής μάθησης, πρέπει να αναφέρουμε τις υπερπαραμέτρους στα μοντέλα του Scikit-learn που επιλέξαμε:

- Για το Random Forest επιλέγουμε n_estimators = 200. Αυτός ο αριθμός υποδεικνύει τον μέγιστο αριθμό δένδρων απόφασης που θα περιλαμβάνει το τυχαίο δάσος.
- Για το Decision Tree επιλέγουμε max_depth = 150, το μέγιστο βάθος του δένδρου απόφασης.
- Για τον K-Neighbors επιλέγουμε 3 εγγύτατους γείτονες στην περίπτωση χρήσης του TF-IDF διανυσματοποιητή, ενώ επιλέγουμε 2 εγγύτατους γείτονες στην περίπτωση χρήσης του Word2Vec διανυσματοποιητή.
- Για το Multinomial Naive Bayes χρησιμοποιούμε τις προεπιλεγμένες ρυθμίσεις. Κάνουμε έναν μετασχηματισμό στα δεδομένα ώστε οι τιμές των διανυσμάτων να είναι θετικές, κάτι που το απαιτεί η χρήση του αλγορίθμου.
- Για το Logistic Regression επιλέγουμε random_state = 5, solver='lbfgs' και max_iter=1000.
- Για το Support Vector Machine χρησιμοποιούμε τις προεπιλεγμένες ρυθμίσεις.
- Για το Gaussian Naive Bayes χρησιμοποιούμε επίσης τις προεπιλεγμένες ρυθμίσεις.

Στην βαθιά μάθηση, οι υπερ-παράμετροι που χρησιμοποιήσαμε συνοψίζονται ως εξής. Σε όλα τα βαθιά μοντέλα χρησιμοποιήσαμε BATCH_SIZE=32. Στην περίπτωση των τυχαίων διαχωρισμών, εκπαιδεύουμε τα μοντέλα μας για 8 εποχές στην δυαδική κατηγοριοποίηση, και στην περίπτωση της τριαδικής κατηγοριοποίησης, αν επιλεγεί ένα balanced dataset, τότε εκπαιδεύουμε με 6 εποχές, διαφορετικά (σε unbalanced), εκπαιδεύουμε με 8 εποχές. Στην περίπτωση του συγκεκριμένου διαχωρισμού, εκπαιδεύουμε για 8 εποχές στην δυαδική κατηγοριοποίηση, ενώ στην περίπτωση της τριαδικής κατηγοριοποίησης, αν το μοντέλο που έχει επιλεγεί είναι το GPT-2, τότε οι εποχές ορίζονται σε 10, διαφορετικά (για τα άλλα μοντέλα), οι εποχές ορίζονται σε 8. Όπως και για το Word2Vec, για τα Politics Datasets ορίζουμε τον μέγιστο αριθμό λέξεων/tokens στην τιμή 80, ενώ για το Skroutz Dataset στην τιμή 150. Τέλος, σε όλα τα μοντέλα μας χρησιμοποιούμε τον Adam optimizer [111], θέτοντας learning_rate=1e-05, χρησιμοποιούμε σαν βασική μετρική την ακρίβεια των μοντέλων (accuracy) και σαν συνάρτηση κόστους θέτουμε για την δυαδική κατηγοριοποίηση την binary_crossentropy, ενώ για την τριαδική χρησιμοποιούμε την categorical_crossentropy.

Αξίζει να σταθούμε και στην αρχιτεκτονική των μετασχηματιστών της εργασίας μας. Πέρα από την βασική αρχιτεκτονική των μοντέλων που αναλύσαμε στο κεφάλαιο 5, κάνουμε μερικές προσθήκες στα στρώματα των μοντέλων ώστε να βελτιώσουμε τα αποτελέσματα. Έτσι:

- Για το Greek BERT: προσθέτουμε ένα στρώμα Dropout [112] με ρυθμό 0.2, ένα πυκνό στρώμα Dense [113] 64 νευρώνων με συνάρτηση ενεργοποίησης την ReLU και ακόμα ένα στρώμα Dropout όμοιο με το πρώτο.
- Για το RoBERTa: προσθέτουμε ένα στρώμα Dropout με ρυθμό 0.2, ένα πυκνό στρώμα Dense 128 νευρώνων με συνάρτηση ενεργοποίησης την ReLU, ένα στρώμα Dropout, ένα πυκνό στρώμα Dense 64 νευρώνων με συνάρτηση ενεργοποίησης την ReLU και ένα ακόμα στρώμα Dropout.
- Για το DistilBERT: προσθέτουμε ένα στρώμα Dropout με ρυθμό 0.2, ένα πυκνό στρώμα Dense 256 νευρώνων με συνάρτηση ενεργοποίησης την ReLU, ένα στρώμα Dropout, ένα πυκνό στρώμα Dense 128 νευρώνων με συνάρτηση ενεργοποίησης την ReLU, ένα στρώμα Dropout, ένα πυκνό στρώμα 64 νευρώνων με συνάρτηση ενεργοποίησης την ReLU, ένα ακόμα στρώμα Dropout και ένα στρώμα Flatten [114].
- Για το GPT-2: προσθέτουμε απλώς ένα στρώμα Flatten.

Σε όλες τις παραπάνω περιπτώσεις, προσθέτουμε ένα τελικό στρώμα που σχετίζεται με την κατηγοριοποίηση. Στην δυαδική περίπτωση προσθέτουμε ένα στρώμα με 1 νευρώνα με συνάρτηση ενεργοποίησης τη sigmoid, ενώ στη τριαδική περίπτωση προσθέτουμε 3 νευρώνες με συνάρτηση ενεργοποίησης τη softmax. Όλα τα παραπάνω σχήματα αρχιτεκτονικής, αφορούν τους συγκεκριμένους διαχωρισμούς. Για τους τυχαίους διαχωρισμούς δεδομένων χρησιμοποιήθηκαν παρόμοιες αρχιτεκτονικές με ελάχιστες διαφορές. Στον κώδικα του κεφαλαίου 6, παραθέτουμε όλες τις λεπτομέρειες.

Τέλος, όσον αφορά τα μοντέλα μηχανικής μάθησης χρησιμοποιήσαμε τις μεθόδους fit() και predict() της βιβλιοθήκης Scikit learn, για εκπαίδευση και προβλέψεις, αντίστοιχα [115]. Όσον αφορά τα μοντέλα μετασχηματιστών, χρησιμοποιήσαμε τις μεθόδους fit() και predict() της κλάσης Model της Keras [116].

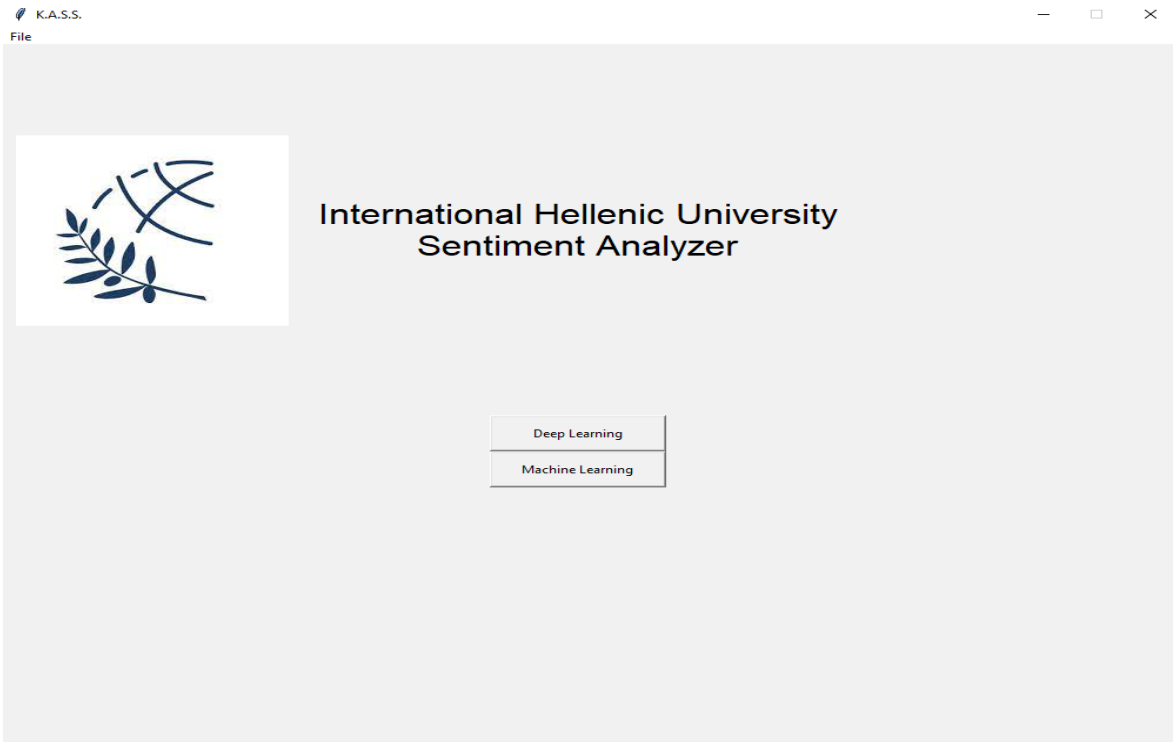
6.9 Βιβλιοθήκη Tkinter

Κατά την ανάπτυξη του κώδικα για την εργασία μας, προέκυψε επιτακτική η ανάγκη να συγκεντρώσουμε τις διάφορες τεχνικές μηχανικής και βαθιάς μάθησης που μας προσφέρονται. Συνεπώς, ακολουθώντας τεχνικές συναρτησιακού προγραμματισμού δημιουργήσαμε δύο αρχεία ‘utilities.py’ και ‘utilities_transformers.py’, τα οποία περιέχουν τις κατάλληλες μεθόδους για την προεπεξεργασία των δεδομένων, την εκτέλεση των πειραμάτων, την οπτικοποίηση και συλλογή των αποτελεσμάτων. Καθώς ήταν απαραίτητο να δίνονται κάποιοι παράμετροι από τον χρήστη (αλγόριθμος, βάση δεδομένων, διαχωρισμός, λημματοποίηση και StopWords, διανυσματοποιητές) συντάχθηκαν και Jupyter Notebooks, για μηχανική και για βαθιά μάθηση, στα οποία ο χρήστης παρέχει αυτές τις παραμέτρους και τελικά εκτελείται το πείραμα.

Αποφασίστηκε στη συνέχεια η δημιουργία μιας διεπαφής με χρήση της βιβλιοθήκης Tkinter [117], που έχει ως σκοπό την αύξηση του ρυθμού εκτέλεσης των πειραμάτων. Το Tkinter είναι μια βιβλιοθήκη προγραμματισμού γραφικού περιβάλλοντος (GUI) για την γλώσσα προγραμματισμού Python, η οποία παρέχει εργαλεία και στοιχεία για την δημιουργία διεπαφών χρήστη με γραφικά παράθυρα, κουμπιά, πεδία κειμένου και άλλα στοιχεία. Δώσαμε το όνομα K.A.S.S., που αντιπροσωπεύει τα αρχικά των ονομάτων μας.

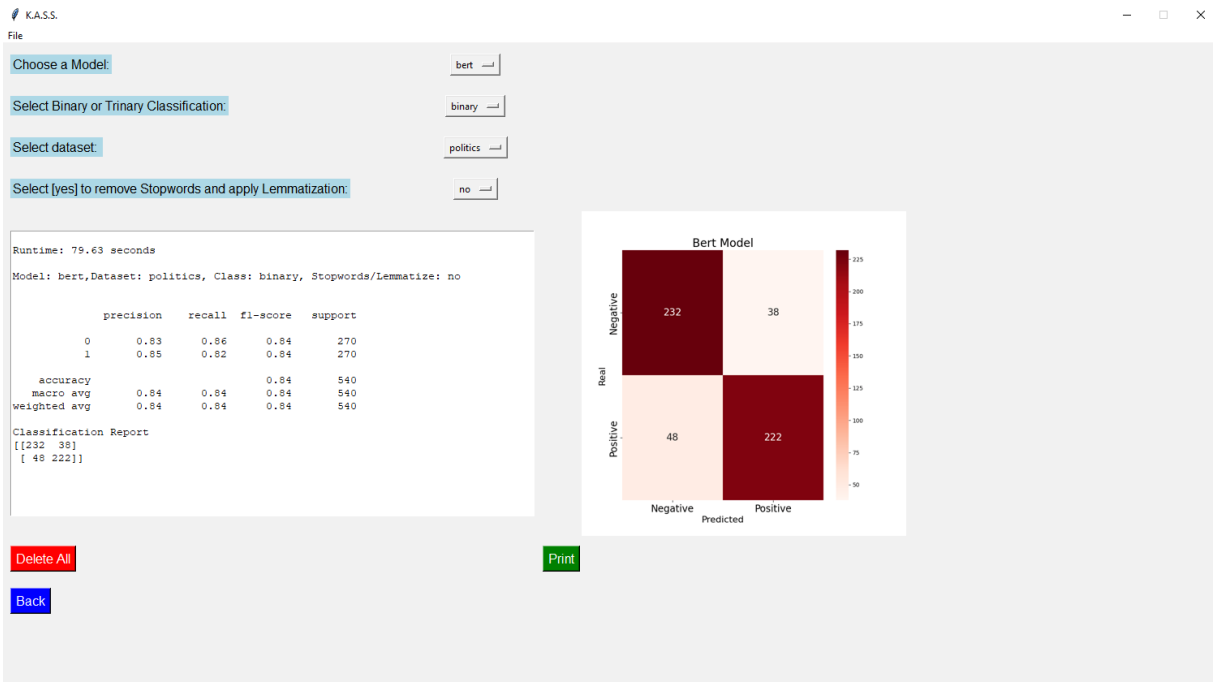
Το γραφικό περιβάλλον που δημιουργήσαμε βασίζεται σε τρία πλαίσια. Κατά την εκκίνηση της εφαρμογής βρίσκεται το βασικό πλαίσιο (Εικόνα 6.29), στο οποίο ο χρήστης επιλέγει αν θέλει να

εκτελέσει πείραμα μηχανικής ή βαθιάς μάθησης. Αναλόγως την επιλογή, κατευθύνεται στο αντίστοιχο πλαίσιο μηχανικής (Εικόνα 6.30) ή βαθιάς μάθησης (Εικόνα 6.31). Επιλέγει τις επιθυμητές παραμέτρους και εκτελείται το πείραμα.



Εικόνα 6.29: Το βασικό πλαίσιο K.A.S.S.

Εικόνα 6.30: Το machine learning πλαίσιο K.A.S.S.



Εικόνα 6.31: Το deep learning πλαίσιο K.A.S.S.

Πρέπει να επισημάνουμε πως στην παραπάνω εφαρμογή, δεν είναι δυνατή η εκπαίδευση μοντέλων μετασχηματιστών. Για την συγκεκριμένη διαδικασία είναι απαραίτητη η χρήση GPU, ώστε να γίνονται παράλληλα οι πολλαπλασιασμοί πινάκων, και συνεπώς η εκπαίδευση έγινε μέσω Kaggle. Το πλαίσιο deep learning λειτουργεί κάνοντας προβλέψεις με προ-εκπαιδευμένα μοντέλα. Για τις ανάγκες της εργασίας μας, εξαγάγαμε τα βάρη (αρχεία .h5) όλων των μοντέλων που εκπαιδεύτηκαν στην περίπτωση του συγκεκριμένου διαχωρισμού. Τα χρησιμοποιήσαμε ώστε να κάνουμε τις προβλέψεις με τα αντίστοιχα δεδομένα δοκιμής και τελικά να λάβουμε τα αποτελέσματά μας. Στην περίπτωση χρήσης αλγορίθμων μηχανικής μάθησης, ο χρήστης έχει τη δυνατότητα να επιλέξει αν θέλει να εφαρμόσει τυχαίο διαχωρισμό στα δεδομένα. Επιλέγει [yes] στο να εφαρμόσει random split και στη συνέχεια εμφανίζεται ένα αναδυόμενο παράθυρο, όπου του ζητείται να πληκτρολογήσει την τιμή 0.2 (80% για εκπαίδευση + 20% για test) ή την τιμή 0.3 (70% εκπαίδευση + 30% για test), αναλόγως τον διαχωρισμό σε δεδομένα εκπαίδευσης και ελέγχου που επιθυμεί.

Το K.A.S.S. αναπτύχθηκε χρησιμοποιώντας την Python 3.9 σε συμβατική CPU. Καθώς η συγκεκριμένη εφαρμογή κάνει χρήση πακέτων που απαιτούν πολλούς πόρους συνίσταται η χρήση εικονικού περιβάλλοντος (virtual environment). Έχουμε παραθέσει οδηγίες εγκατάστασης στο Παράρτημα Γ.

6.10 Αποτελέσματα πειραμάτων

Οι πίνακες 6.4 μέχρι 6.8, παρουσιάζουν τα συγκεντρωτικά αποτελέσματα ανά σύνολο δεδομένων και κατηγοριοποίηση. Η μετρική που επιλέχθηκε είναι η accuracy που μελετήθηκε στο Κεφάλαιο 4. Θα μπορούσαμε να χρησιμοποιήσουμε και την macro avg κυρίως στα unbalanced datasets, καθώς αποτελεί μια καλή επιλογή, καθώς λαμβάνει υπόψη την απόδοση κάθε κλάσης ανεξάρτητα από το μέγεθός της. Ωστόσο, στα πειράματά μας η διαφορά στις τιμές accuracy και macro avg ήταν αμελητέα, με αποτέλεσμα τελικά να επιλεγεί η μετρική accuracy.

Πίνακας 6.4: Μετρική accuracy για Unbalanced Politics με split 70/30

MODELS		UNBALANCED POLITICS			
		Multi Class		Binary Class	
		SL	SL no	SL	SL no
split train 70% / test 30%					
Transformers	TFBertModel	0.64	0.66	0.85	0.88
	RoBERTa	0.62	0.66	0.84	0.86
	DistilBert	0.48	0.51	0.74	0.76
	GPT-2	0.56	0.56	0.83	0.85
split train 70% / test 30%					
Word2Vec	RandomForest	0.49	0.45	0.62	0.63
	KNeighbors	0.41	0.37	0.55	0.53
	DecisionTree	0.39	0.38	0.56	0.53
	MultinomialNB	0.37	0.39	0.53	0.56
	Logistic Regression	0.51	0.48	0.70	0.68
	Support Vector Machines	0.47	0.46	0.65	0.62
	GaussianNB	0.32	0.34	0.54	0.54
TF-IDF	RandomForest	0.57	0.55	0.75	0.74
	KNeighbors	0.53	0.53	0.73	0.73
	DecisionTree	0.47	0.45	0.70	0.62
	MultinomialNB	0.59	0.59	0.78	0.76
	Logistic Regression	0.60	0.57	0.78	0.76
	Support Vector Machines	0.60	0.58	0.79	0.77
	GaussianNB	0.51	0.53	0.69	0.71

Πίνακας 6.5: Μετρική accuracy για Unbalanced Politics με split 80/20

MODELS		UNBALANCED POLITICS			
		Multi Class		Binary Class	
		SL	SL no	SL	SL no
split train 80% / test 20%					
Transformers	TFBertModel	0.66	0.69	0.84	0.88
	RoBERTa	0.65	0.64	0.82	0.86
	DistilBert	0.50	0.54	0.76	0.72
	GPT-2	0.60	0.62	0.80	0.83
split train 80% / test 20%					
Word2Vec	RandomForest	0.46	0.44	0.66	0.66
	KNeighbors	0.41	0.41	0.56	0.55
	DecisionTree	0.42	0.40	0.58	0.62
	MultinomialNB	0.37	0.40	0.54	0.56
	Logistic Regression	0.53	0.51	0.74	0.66
	Support Vector Machines	0.50	0.46	0.70	0.67
	GaussianNB	0.36	0.36	0.53	0.53
TF-IDF	RandomForest	0.56	0.57	0.76	0.75
	KNeighbors	0.52	0.51	0.71	0.71
	DecisionTree	0.48	0.45	0.72	0.66
	MultinomialNB	0.59	0.59	0.80	0.77
	Logistic Regression	0.60	0.57	0.77	0.76
	Support Vector Machines	0.60	0.58	0.78	0.78
	GaussianNB	0.53	0.53	0.72	0.73

Πίνακας 6.6: Μετρική accuracy για Balanced Politics/Skroutz με split 70/30

MODELS		BALANCED POLITICS				BALANCED SKROUTZ	
		Multi Class		Binary Class		Binary Class	
		SL	SL no	SL	SL no	SL	SL no
split train 70% / test 30%							
Transformers	TFBertModel	0.63	0.67	0.82	0.86	0.94	0.96
	RoBERTa	0.64	0.63	0.84	0.85	0.94	0.95
	DistilBert	0.46	0.46	0.70	0.74	0.90	0.91
	GPT-2	0.55	0.46	0.68	0.86	0.90	0.93
split train 70% / test 30%							
Word2Vec	RandomForest	0.45	0.47	0.64	0.63	0.78	0.79
	KNeighbors	0.38	0.37	0.55	0.55	0.53	0.53
	DecisionTree	0.39	0.35	0.56	0.54	0.69	0.68
	MultinomialNB	0.41	0.41	0.60	0.60	0.77	0.78
	Logistic Regression	0.54	0.49	0.70	0.70	0.84	0.84
	Support Vector Machines	0.48	0.47	0.67	0.65	0.82	0.82
	GaussianNB	0.35	0.33	0.34	0.33	0.82	0.77
TF-IDF	RandomForest	0.56	0.57	0.75	0.74	0.89	0.89
	KNeighbors	0.50	0.53	0.73	0.75	0.85	0.84
	DecisionTree	0.49	0.45	0.66	0.63	0.80	0.82
	MultinomialNB	0.60	0.58	0.78	0.78	0.87	0.87
	Logistic Regression	0.57	0.56	0.76	0.74	0.91	0.91
	Support Vector Machines	0.58	0.57	0.78	0.77	0.93	0.93
	GaussianNB	0.50	0.51	0.71	0.71	0.77	0.77

Πίνακας 6.7: Μετρική accuracy για Balanced Politics/Skroutz με split 80/20

MODELS		BALANCED POLITICS				BALANCED SKROUTZ	
		Multi Class		Binary Class		Binary Class	
		SL	SL no	SL	SL no	SL	SL no
split train 80% / test 20%							
Transformers	TFBertModel	0.66	0.70	0.88	0.88	0.95	0.96
	RoBERTa	0.64	0.67	0.84	0.86	0.93	0.95
	DistilBert	0.50	0.46	0.76	0.78	0.91	0.92
	GPT-2	0.57	0.56	0.85	0.87	0.94	0.94
split train 80% / test 20%							
Word2Vec	RandomForest	0.45	0.47	0.66	0.62	0.80	0.80
	KNeighbors	0.36	0.38	0.55	0.55	0.55	0.53
	DecisionTree	0.40	0.40	0.57	0.55	0.70	0.72
	MultinomialNB	0.43	0.44	0.62	0.62	0.78	0.79
	Logistic Regression	0.54	0.52	0.73	0.68	0.87	0.86
	Support Vector Machines	0.46	0.48	0.68	0.64	0.83	0.83
	GaussianNB	0.34	0.33	0.50	0.50	0.75	0.78
TF-IDF	RandomForest	0.57	0.59	0.76	0.74	0.90	0.89
	KNeighbors	0.51	0.53	0.71	0.73	0.87	0.86
	DecisionTree	0.47	0.46	0.69	0.64	0.83	0.83
	MultinomialNB	0.62	0.58	0.82	0.79	0.90	0.90
	Logistic Regression	0.59	0.58	0.79	0.76	0.93	0.93
	Support Vector Machines	0.60	0.61	0.81	0.79	0.94	0.94
	GaussianNB	0.52	0.53	0.72	0.73	0.76	0.78

Πίνακας 6.8: Μετρική accuracy για Balanced Politics/Skroutz με συγκεκριμένο split

MODELS		BALANCED POLITICS Train:2682δείγματα Test:810 δείγματα				BALANCED SKROUTZ Train: 3210 δείγματα Test: 1966δείγματα	
		Multi Class		Binary Class		Binary Class	
		SL	SL no	SL	SL no	SL	SL no
Transformers	TFBertModel	0.59	0.63	0.81	0.84	0.93	0.95
	RoBERTa	0.55	0.60	0.75	0.77	0.94	0.94
	DistilBERT	0.46	0.50	0.66	0.72	0.86	0.89
	GPT-2	0.58	0.53	0.78	0.79	0.90	0.91
Word2Vec	RandomForest	0.40	0.41	0.61	0.63	0.78	0.78
	KNeighbors	0.34	0.38	0.46	0.53	0.53	0.52
	DecisionTree	0.38	0.37	0.51	0.51	0.70	0.70
	MultinomialNB	0.41	0.42	0.60	0.58	0.76	0.77
	Logistic Regression	0.40	0.45	0.62	0.62	0.83	0.83
	Support Vector Machines	0.42	0.44	0.61	0.63	0.80	0.81
	GaussianNB	0.33	0.33	0.50	0.50	0.74	0.76
TF-IDF	RandomForest	0.41	0.48	0.62	0.65	0.87	0.87
	KNeighbors	0.40	0.49	0.62	0.68	0.79	0.87
	DecisionTree	0.41	0.42	0.57	0.59	0.69	0.80
	MultinomialNB	0.39	0.50	0.64	0.69	0.85	0.86
	Logistic Regression	0.41	0.49	0.64	0.65	0.89	0.90
	Support Vector Machines	0.42	0.49	0.64	0.67	0.89	0.91
	GaussianNB	0.41	0.45	0.55	0.69	0.73	0.77

Όπως αναφέρθηκε, ο συνολικός αριθμός των πειραμάτων που εκτελέστηκαν είναι 468. Στις εικόνες 6.32 – 6.44, παρουσιάζουμε τις μετρικές Precision, Recall και F1-score ανά κατηγοριοποίηση, μόνο για τα μοντέλα που σημειώνουν την καλύτερη απόδοση σύμφωνα με τους προηγούμενους πίνακες. Η επιλογή του καλύτερου έγινε σε κάθε στήλη ανά κατηγορία διανυσματοποιητή. Έτσι, για παράδειγμα στην τελευταία στήλη του πίνακα 6.8, στην κατηγορία Transformers, την καλύτερη απόδοση την

σημειώνει το BERT με 95%, ακολουθεί το SVM με 91% στην κατηγορία TF-IDF, και το Logistic Regression με 83% στην κατηγορία Word2Vec.

Για κατηγοριοποίηση Binary:

	Unbalanced Politics (split: 80/20), SL=YES					
	Precision		Recall		F1-Score	
MODELS	Positive	Negative	Positive	Negative	Positive	Negative
BERT	0.83	0.85	0.82	0.85	0.83	0.85
Word2Vec Logistic Regression	0.72	0.75	0.70	0.77	0.71	0.76
TF-IDF Multinomial NB	0.84	0.78	0.70	0.89	0.77	0.83
	Unbalanced Politics (split: 80/20), SL=NO					
	Precision		Recall		F1-Score	
MODELS	Positive	Negative	Positive	Negative	Positive	Negative
BERT	0.85	0.91	0.90	0.87	0.88	0.89
Word2Vec SVM	0.69	0.66	0.52	0.80	0.59	0.72
TF-IDF SVM	0.89	0.73	0.60	0.94	0.71	0.82

Εικόνα 6.32: Unbalanced Politics (split: 80/20 + SL=YES/NO)

	Unbalanced Politics (split: 70/30), SL=YES					
	Precision		Recall		F1-Score	
MODELS	Positive	Negative	Positive	Negative	Positive	Negative
BERT	0.82	0.88	0.86	0.84	0.84	0.86
Word2Vec Logistic Regression	0.68	0.72	0.67	0.73	0.67	0.72
TF-IDF SVM	0.87	0.75	0.64	0.92	0.73	0.82
	Unbalanced Politics (split: 70/30), SL=NO					
	Precision		Recall		F1-Score	
MODELS	Positive	Negative	Positive	Negative	Positive	Negative
BERT	0.87	0.88	0.86	0.89	0.87	0.89
Word2Vec Logistic Regression	0.65	0.70	0.65	0.71	0.65	0.70
TF-IDF SVM	0.86	0.73	0.60	0.91	0.71	0.81

Εικόνα 6.33: Unbalanced Politics (split: 70/30 + SL=YES/NO)

	Balanced Politics (test_set: 810), SL=YES					
	Precision		Recall		F1-Score	
MODELS	Positive	Negative	Positive	Negative	Positive	Negative
BERT	0.90	0.75	0.69	0.93	0.78	0.83
Word2Vec Logistic Regression	0.61	0.63	0.65	0.59	0.63	0.61
TF-IDF Multinomial NB	0.66	0.62	0.57	0.71	0.61	0.66
TF-IDF Logistic Regression	0.68	0.61	0.52	0.76	0.59	0.68
TF-IDF SVM	0.71	0.60	0.47	0.81	0.57	0.69
	Balanced Politics (test_set: 810), SL=NO					
	Precision		Recall		F1-Score	
MODELS	Positive	Negative	Positive	Negative	Positive	Negative
BERT	0.85	0.83	0.82	0.86	0.84	0.84
Word2Vec Random Forest	0.68	0.60	0.50	0.76	0.57	0.67
Word2Vec SVM	0.66	0.61	0.54	0.72	0.59	0.66
TF-IDF Multinomial NB	0.71	0.67	0.62	0.75	0.67	0.71
TF-IDF Gaussian NB	0.69	0.69	0.69	0.69	0.69	0.69

Εικόνα 6.34: Balanced Politics (test: 810 + SL=YES/NO)

	Balanced Skroutz (test_set: 1966), SL=YES					
	Precision		Recall		F1-Score	
MODELS	Positive	Negative	Positive	Negative	Positive	Negative
ROBERTA	0.94	0.93	0.93	0.94	0.94	0.94
Word2Vec Logistic Regression	0.86	0.81	0.79	0.87	0.82	0.84
TF-IDF Logistic Regression	0.85	0.93	0.94	0.83	0.89	0.88
TF-IDF SVM	0.86	0.92	0.93	0.85	0.89	0.88
	Balanced Skroutz (test_set: 1966), SL=NO					
	Precision		Recall		F1-Score	
MODELS	Positive	Negative	Positive	Negative	Positive	Negative
BERT	0.95	0.94	0.94	0.96	0.95	0.95
Word2Vec Logistic Regression	0.82	0.83	0.83	0.82	0.83	0.83
TF-IDF SVM	0.93	0.90	0.90	0.93	0.91	0.92

Εικόνα 6.35: Balanced Skroutz (test: 1966 + SL=YES/NO)

	Balanced Politics (split: 80/20), SL=YES					
	Precision		Recall		F1-Score	
MODELS	Positive	Negative	Positive	Negative	Positive	Negative
BERT	0.90	0.86	0.86	0.90	0.88	0.88
Word2Vec Logistic Regression	0.73	0.73	0.73	0.73	0.73	0.73
TF-IDF Multinomial NB	0.82	0.81	0.81	0.82	0.81	0.82
	Balanced Politics (split: 80/20), SL=NO					
	Precision		Recall		F1-Score	
MODELS	Positive	Negative	Positive	Negative	Positive	Negative
BERT	0.88	0.89	0.89	0.88	0.88	0.88
Word2Vec Logistic Regression	0.67	0.68	0.70	0.65	0.68	0.67
TF-IDF Multinomial NB	0.79	0.78	0.78	0.79	0.79	0.79
TF-IDF SVM	0.84	0.75	0.71	0.86	0.77	0.80

Εικόνα 6.36: Balanced Politics (split: 80/20 + SL=YES/NO)

	Balanced Skroutz (split: 80/20), SL=YES					
	Precision		Recall		F1-Score	
MODELS	Positive	Negative	Positive	Negative	Positive	Negative
BERT	0.95	0.96	0.96	0.95	0.95	0.95
Word2Vec Logistic Regression	0.86	0.87	0.87	0.86	0.87	0.87
TF-IDF SVM	0.93	0.94	0.94	0.93	0.94	0.94
	Balanced Skroutz (split: 80/20), SL=NO					
	Precision		Recall		F1-Score	
MODELS	Positive	Negative	Positive	Negative	Positive	Negative
BERT	0.95	0.96	0.96	0.95	0.96	0.96
Word2Vec Logistic Regression	0.87	0.86	0.86	0.87	0.86	0.86
TF-IDF SVM	0.93	0.94	0.95	0.93	0.94	0.94

Εικόνα 6.37: Balanced Skroutz (split: 80/20 + SL=YES/NO)

	Balanced Politics (split: 70/30), SL=YES					
	Precision		Recall		F1-Score	
MODELS	Positive	Negative	Positive	Negative	Positive	Negative
ROBERTA	0.84	0.83	0.83	0.84	0.83	0.84
Word2Vec Logistic Regression	0.69	0.70	0.72	0.68	0.70	0.69
TF-IDF Multinomial NB	0.78	0.78	0.78	0.78	0.78	0.78
TF-IDF SVM	0.83	0.74	0.70	0.86	0.76	0.79
	Balanced Politics (split: 70/30), SL=NO					
	Precision		Recall		F1-Score	
MODELS	Positive	Negative	Positive	Negative	Positive	Negative
BERT	0.91	0.83	0.81	0.92	0.86	0.87
GPT-2	0.86	0.85	0.85	0.87	0.86	0.86
Word2Vec Logistic Regression	0.68	0.72	0.75	0.65	0.71	0.68
TF-IDF Multinomial NB	0.79	0.78	0.78	0.79	0.78	0.79

Εικόνα 6.38: Balanced Politics (split: 70/30 + SL=YES/NO)

	Balanced Skroutz (split: 70/30), SL=YES					
	Precision		Recall		F1-Score	
MODELS	Positive	Negative	Positive	Negative	Positive	Negative
BERT	0.95	0.94	0.94	0.95	0.94	0.94
ROBERTA	0.94	0.94	0.94	0.94	0.94	0.94
Word2Vec Logistic Regression	0.84	0.84	0.84	0.84	0.84	0.84
TF-IDF SVM	0.94	0.93	0.93	0.94	0.93	0.93
	Balanced Skroutz (split: 70/30), SL=NO					
	Precision		Recall		F1-Score	
MODELS	Positive	Negative	Positive	Negative	Positive	Negative
BERT	0.96	0.95	0.95	0.96	0.96	0.96
Word2Vec Logistic Regression	0.83	0.84	0.84	0.83	0.84	0.84
TF-IDF SVM	0.93	0.93	0.92	0.93	0.93	0.93

Εικόνα 6.39: Balanced Skroutz (split: 70/30 + SL=YES/NO)

Για κατηγοριοποίηση Multi:

	Unbalanced Politics (split: 80/20), SL=YES								
	Precision			Recall			F1-Score		
MODELS	Negative	Neutral	Positive	Negative	Neutral	Positive	Negative	Neutral	Positive
BERT	0.70	0.52	0.78	0.68	0.58	0.71	0.69	0.55	0.75
Word2Vec Logistic Regression	0.56	0.45	0.56	0.55	0.44	0.59	0.55	0.45	0.58
TF-IDF Logistic Regression	0.58	0.53	0.68	0.72	0.44	0.60	0.64	0.49	0.64
TF-IDF SVM	0.55	0.55	0.76	0.83	0.37	0.55	0.66	0.44	0.64
	Unbalanced Politics (split: 80/20), SL=NO								
	Precision			Recall			F1-Score		
MODELS	Negative	Neutral	Positive	Negative	Neutral	Positive	Negative	Neutral	Positive
BERT	0.81	0.56	0.73	0.64	0.65	0.80	0.71	0.60	0.76
Word2Vec Logistic Regression	0.52	0.45	0.57	0.53	0.47	0.53	0.53	0.46	0.55
TF-IDF MultinomialNB	0.54	0.59	0.71	0.83	0.32	0.59	0.65	0.41	0.64

Εικόνα 6.40: Unbalanced Politics (split: 80/20 + SL=YES/NO + Multi Class)

	Unbalanced Politics (split: 70/30), SL=YES								
	Precision			Recall			F1-Score		
MODELS	Negative	Neutral	Positive	Negative	Neutral	Positive	Negative	Neutral	Positive
BERT	0.68	0.55	0.66	0.72	0.44	0.75	0.70	0.48	0.70
Word2Vec Logistic Regression	0.56	0.44	0.54	0.57	0.42	0.54	0.56	0.43	0.54
TF-IDF Logistic Regression	0.59	0.53	0.67	0.72	0.43	0.61	0.65	0.48	0.64
TF-IDF SVM	0.55	0.57	0.74	0.80	0.40	0.56	0.65	0.47	0.64
	Unbalanced Politics (split: 70/30), SL=NO								
	Precision			Recall			F1-Score		
MODELS	Negative	Neutral	Positive	Negative	Neutral	Positive	Negative	Neutral	Positive
BERT	0.77	0.56	0.66	0.66	0.52	0.80	0.71	0.54	0.72
ROBERTA	0.73	0.55	0.72	0.66	0.60	0.74	0.70	0.57	0.73
Word2Vec Logistic Regression	0.50	0.43	0.51	0.52	0.42	0.50	0.51	0.42	0.50
TF-IDF MultinomialNB	0.53	0.60	0.70	0.87	0.30	0.55	0.66	0.40	0.62

Εικόνα 6.41: Unbalanced Politics (split: 70/30 + SL=YES/NO + Multi Class)

	Balanced Politics (test_set: 810), SL=YES								
	Precision			Recall			F1-Score		
MODELS	Negative	Neutral	Positive	Negative	Neutral	Positive	Negative	Neutral	Positive
BERT	0.70	0.46	0.79	0.54	0.73	0.49	0.61	0.56	0.60
Word2Vec SVM	0.40	0.45	0.50	0.73	0.33	0.21	0.51	0.38	0.30
TF-IDF SVM	0.57	0.37	0.61	0.23	0.82	0.22	0.33	0.51	0.33
	Balanced Politics (test_set: 810), SL=NO								
	Precision			Recall			F1-Score		
MODELS	Negative	Neutral	Positive	Negative	Neutral	Positive	Negative	Neutral	Positive
BERT	0.68	0.52	0.73	0.74	0.61	0.55	0.71	0.56	0.63
Word2Vec Logistic Regression	0.46	0.42	0.48	0.44	0.44	0.48	0.45	0.43	0.48
TF-IDF MultinomialNB	0.53	0.42	0.54	0.57	0.41	0.51	0.55	0.42	0.52

Εικόνα 6.42: Balanced Politics (test: 810 + SL=YES/NO + Multi Class)

	Balanced Politics (split: 80/20), SL=YES								
	Precision			Recall			F1-Score		
MODELS	Negative	Neutral	Positive	Negative	Neutral	Positive	Negative	Neutral	Positive
BERT	0.63	0.63	0.70	0.82	0.41	0.74	0.71	0.50	0.72
Word2Vec Logistic Regression	0.57	0.47	0.57	0.52	0.48	0.61	0.54	0.48	0.59
TF-IDF MultinomialNB	0.64	0.55	0.64	0.65	0.48	0.71	0.65	0.51	0.67
	Balanced Politics (split: 80/20), SL=NO								
	Precision			Recall			F1-Score		
MODELS	Negative	Neutral	Positive	Negative	Neutral	Positive	Negative	Neutral	Positive
BERT	0.75	0.61	0.75	0.70	0.65	0.75	0.72	0.63	0.75
Word2Vec Logistic Regression	0.52	0.48	0.55	0.48	0.48	0.58	0.50	0.48	0.56
TF-IDF SVM	0.60	0.52	0.73	0.62	0.56	0.65	0.61	0.54	0.69

Εικόνα 6.43: Balanced Politics (split: 80/20 + SL=YES/NO + Multi Class)

	Balanced Politics (split: 70/30), SL=YES								
	Precision			Recall			F1-Score		
MODELS	Negative	Neutral	Positive	Negative	Neutral	Positive	Negative	Neutral	Positive
ROBERTA	0.59	0.58	0.77	0.83	0.43	0.66	0.69	0.49	0.71
Word2Vec Logistic Regression	0.57	0.48	0.58	0.49	0.52	0.61	0.53	0.50	0.59
TF-IDF MultinomialNB	0.60	0.56	0.62	0.63	0.45	0.71	0.61	0.50	0.66
	Balanced Politics (split: 70/30), SL=NO								
	Precision			Recall			F1-Score		
MODELS	Negative	Neutral	Positive	Negative	Neutral	Positive	Negative	Neutral	Positive
BERT	0.68	0.56	0.77	0.64	0.61	0.75	0.66	0.59	0.76
Word2Vec Logistic Regression	0.48	0.44	0.55	0.45	0.45	0.57	0.46	0.45	0.56
TF-IDF MultinomialNB	0.58	0.53	0.62	0.59	0.47	0.68	0.58	0.50	0.65

Εικόνα 6.44: Balanced Politics (split: 70/30 + SL=YES/NO + Multi Class)

6.11 Σχολιασμός - Επίλογος

Μελετώντας κανείς του πίνακες 6.4 - 6.8, μπορεί να βγάλει χρήσιμα συμπεράσματα. Λόγω του όγκου των πειραμάτων, θα συνοψίσουμε τις παρατηρήσεις μας στην παρούσα ενότητα. Σε ένα γενικότερο πλαίσιο, στο πρόβλημα της ανάλυσης συναισθήματος παρατηρούμε μια εμφανή υπεροχή των μετασχηματιστών και ειδικότερα του Greek BERT. Αυτό το αποτέλεσμα είναι αναμενόμενο, αν αναλογιστεί κανείς την δυνατότητα που έχουν τα transformers να αναλύουν σε βάθος το περιεχόμενο των προτάσεων με τον μηχανισμό προσοχής. Ωστόσο, οφείλουμε να σταθούμε και στα κόστη των μετασχηματιστών. Πρόκειται για μεγάλα νευρωνικά δίκτυα που απαιτούν πολλούς υπολογιστικούς πόρους για την εκπαίδευση, αλλά και την εφαρμογή τους. Σε πολλές περιπτώσεις, όπως θα δούμε και παρακάτω, ίσως να είναι συνετότερη η χρήση παραδοσιακών τεχνικών μηχανικής μάθησης.

Μπορούμε να αναλύσουμε τα πειράματά μας σε τρία βασικά επίπεδα: α) σε επίπεδο βάσης δεδομένων και κατηγοριοποίησης, β) σε επίπεδο προ-επεξεργασίας και γ) σε επίπεδο μοντέλου.

Και στις τρεις βάσεις δεδομένων μας, εξετάσαμε δυαδική κατηγοριοποίηση, ενώ στις βάσεις δεδομένων με τα πολιτικά σχόλια έγινε και κατηγοριοποίηση τριών κλάσεων. Είδαμε τους αλγορίθμους μας να πετυχαίνουν καλύτερες αποδόσεις στο Skroutz Dataset σε σύγκριση με τα άλλα datasets. Την καλύτερη απόδοση για το Skroutz Dataset 96% την είχε το Greek BERT, που αναγράφεται στον πίνακα 6.6 και 6.7 για split 70/30 και 80/20 αντίστοιχα, χωρίς την εφαρμογή lemmatize και την απαλοιφή stopwords. Το φαινόμενο αυτό πιθανώς οφείλεται στο γεγονός πως το Skroutz Dataset εμπεριέχει περισσότερα σχόλια, και μάλιστα με μεγαλύτερο αριθμό tokens ανά σχόλιο (Πίνακας 6.3). Για την τριαδική (multi) κατηγοριοποίηση παρατηρούμε ξανά το BERT να στέκεται στη κορυφή με καλύτερη ακρίβεια το 70% (Πίνακας 6.7 και Εικόνα 6.43). Ανάμεσα σε balanced και unbalanced Politics δεν παρατηρούνται σημαντικές διαφορές στην απόδοση. Ωστόσο, τα καλύτερα αποτελέσματα εμφανίζονται στα πειράματα με την ισορροπημένη έκδοση.

Είναι σημαντικό να αναλύσουμε και τις διάφορες περιπτώσεις προ-επεξεργασίας. Η αφαίρεση των StopWords και η εφαρμογή λημματοποίησης των λέξεων φαίνεται να βοηθά τους αλγορίθμους μηχανικής μάθησης να πετυχαίνουν καλύτερη γενίκευση. Αντίθετα, οι μετασχηματιστές δεν ευνοούνται από αυτή την επεξεργασία, πετυχαίνοντας καλύτερες αποδόσεις όταν δεν εφαρμόζεται.

Φαίνεται πως οι μετασχηματιστές βασίζονται στο πραγματικό νόημα των προτάσεων. Όσον αφορά τους διαχωρισμούς δεδομένων, είδαμε καλύτερες αποδόσεις στα μοντέλα που χρησιμοποιούν τους τυχαίους διαχωρισμούς. Ενδεχομένως η εφαρμογή cross validation στις βάσεις μας να βελτίωνε τα αποτελέσματα στην περίπτωση του συγκεκριμένου διαχωρισμού. Ανάμεσα στους τυχαίους διαχωρισμούς γενικότερα φαίνεται να υπερέχει ο 80/20. Τέλος, για την προ-επεξεργασία πρέπει να αναφερθούμε και στους διανυσματοποιητές της μηχανικής μάθησης. Η χρήση του TF-IDF προσφέρει εμφανώς καλύτερες αποδόσεις και πολύ μικρότερους χρόνους εκτέλεσης συγκριτικά με τον αλγόριθμο Word2Vec σε όλες τις περιπτώσεις. Ο μόνος αλγόριθμος που φαίνεται να συσχετίζεται καλά με το Word2Vec είναι ο Logistic Regression (Εικόνες 6.32 - 6.39).

Σε επίπεδο μοντέλων, όπως αναφερθήκαμε και παραπάνω, τα transformers, ειδικά το Greek BERT και το RoBERTa, αναλύουν ορθότερα το συναίσθημα. Αξίζει να σταθούμε όμως στο γεγονός πως στην περίπτωση του Skroutz dataset οι αλγόριθμοι SVM και Logistic Regression με χρήση του TF-IDF πλησιάζουν πολύ ή και ξεπερνούν πολλές φορές τις αποδόσεις των μετασχηματιστών. Στα Politics datasets το αποτέλεσμα αυτό δεν παρατηρήθηκε σε όλους τους πίνακες. Η βασική υπόθεσή μας για το φαινόμενο αυτό, βασίστηκε στο μεγαλύτερο πλήθος σχολίων που απαρτίζουν το Skroutz dataset καθώς και στο πλήθος των tokens κάθε σχολίου. Για να εξετάσουμε τις υποψίες μας, εκτελέσαμε 4 συμπληρωματικά πειράματα σε δυο γνωστά datasets για ανάλυση συναισθήματος της αγγλικής γλώσσας, το Movie Reviews [118] και το IMDB dataset [52]. Τα χαρακτηριστικά των βάσεων φαίνονται στον Πίνακα 6.9. Δοκιμάσαμε το Greek BERT και το SVM κάνοντας δυαδική κατηγοριοποίηση χωρίς εφαρμογή stopwords και lemmatize. Διαχωρίζουμε τα δεδομένα σε 80% για εκπαίδευση και 20% για έλεγχο. Στο IMDB dataset μετατρέπουμε τα γράμματα σε πεζά και επιλέγουμε 5000 θετικά και 5000 αρνητικά σχόλια, ώστε οι βάσεις μας τελικά να έχουν περίπου το ίδιο πλήθος σχολίων.

Πίνακας 6.9: Χαρακτηριστικά των Αγγλικών βάσεων δεδομένων

Dataset	Comments	Average Number Of Tokens	Max Number Of Tokens	Negative	Neutral	Positive
Movie Reviews	10662	22	62	5331	-	5331
IMDB Dataset	25000	282	2818	12500	-	12500

Για τις υπερ-παραμέτρους αναφέρουμε την περίπτωση της βαθιάς μάθησης, όπου κάνουμε χρήση του αγγλικού bert-based-uncased και εκπαιδεύουμε για 4 εποχές με MovieReviews_max_lenght = 60 και IMDb_max_lenght = 200. Στον Πίνακα 6.10, παραθέτουμε τα αποτελέσματα σε συνδυασμό με τα

πειράματα στα ελληνικά. Παρατηρούμε, πως η απόδοση του SVM γίνεται πολύ καλύτερη όταν στη βάση δεδομένων τα σχόλια περιέχουν μεγάλο αριθμό tokens κάτι που επιβεβαιώνεται και στα πειράματα της ελληνικής γλώσσας. Αυτό το αποτέλεσμα είναι ενδιαφέρον, καθώς οι μετασχηματιστές είναι υψηλού κόστους μοντέλα, τα οποία απαιτούν ισχυρούς υπολογιστικούς πόρους και μεγάλη μνήμη για την εκπαίδευση και την εκτέλεσή τους. Αν μια φθηνότερη εναλλακτική λύση όπως αυτή του SVM + TF-IDF προσφέρει τις ίδιες αποδόσεις, αμφισβητείται εκεί η χρήση των μετασχηματιστών. Εκτιμούμε πως πρέπει να διεξαχθεί περαιτέρω έρευνα σχετικά με αυτό το εύρημα.

Πίνακας 6.10: Αποτελέσματα πειραμάτων στην Αγγλική γλώσσα (Movie Reviews + IMDB)

	Skroutz	Unbalanced Politics	Balanced Politics	Movie Reviews	IMDB
BERT	0.96	0.88	0.88	**0.84**	**0.89**
SVM + TF-IDF	0.94	0.78	0.79	**0.76**	**0.89**

Στο κεφάλαιο αυτό αναλύσαμε διεξοδικά την μεθοδολογία που ακολουθήσαμε για την εκπόνηση της εργασίας μας. Είδαμε τις προσεγγίσεις διαφόρων Ελλήνων ερευνητών πάνω στο θέμα της ανάλυσης συναισθήματος. Εξετάσαμε τον τρόπο με τον οποίο εξαγάγαμε σχόλια από το Twitter, δημιουργώντας τα Politics Datasets, και παραθέσαμε όλη τη διαδικασία καθαρισμού και προ-επεξεργασίας των δεδομένων. Μελετήσαμε τις βάσεις δεδομένων και αναλύσαμε τελικά το συναίσθημα σε αυτές μέσω αλγορίθμων μηχανικής και βαθιάς μάθησης. Τέλος, παραθέσαμε τα αποτελέσματα και τα σχολιάσαμε.

Κεφάλαιο 7ο: Συμπεράσματα

7.1 Ανασκόπηση της εργασίας

Στην εργασία μας, ασχοληθήκαμε με την εξόρυξη δεδομένων (tweets) από το κοινωνικό δίκτυο Twitter, και με την χρήση αξιολογήσεων μιας βάσης δεδομένων της υπηρεσίας του Skroutz, προκειμένου να αναλύσουμε το συναισθηματικό περιεχόμενο των κειμένων. Σκοπός μας, όσον αφορά το Twitter ήταν η δημιουργία μιας βάσης δεδομένων που αφορά το πολιτικό περιεχόμενο στην Ελληνική επικράτεια. Για την δημιουργία της εν λόγω βάσης, εκμεταλλευτήκαμε τα APIs του Twitter. Έτσι, ανακτήσαμε έναν όγκο δεδομένων tweets, στα οποία εφαρμόσαμε διάφορες τεχνικές καθαρισμού πριν τα τροφοδοτήσουμε σε μοντέλα μηχανικής και βαθιάς μάθησης για περαιτέρω επεξεργασία. Επίσης, επειδή χρησιμοποιήσαμε επιτηρούμενη μάθηση ήταν απαραίτητο να γίνει ανάθεση ετικέτας σε κάθε tweet. Στην συνέχεια, αξιοποιήσαμε κλασσικά μοντέλα μηχανικής μάθησης καθώς και πιο μοντέρνα μοντέλα όπως τα Transformers. Μέσα απο τα πειράματά μας, διαπιστώσαμε την δυναμική των Transformers τα οποία όπως φάνηκε επιτυγχάνουν πολύ καλύτερη απόδοση σε νέα δεδομένα τα οποία δεν έχουν ξανα δει κατά την εκπαίδευση (γενίκευση). Με την συνεχή εξέλιξη αυτής της τεχνολογίας, αναμένουμε να δούμε περαιτέρω καινοτομίες που θα οδηγήσουν σε πρόοδο και ανακαλύψεις που θα μας εκπλήξουν.

Μπορούμε πλέον να πούμε ότι βρισκόμαστε σε μια συναρπαστική εποχή, όπου τεχνητή νοημοσύνη και μηχανική μάθηση, προσφέρουν απίστευτες δυνατότητες και ανοίγουν νέους ορίζοντες για το μέλλον της ανθρωπότητας.

7.2 Προτάσεις για βελτίωση

Παραθέτουμε ορισμένες συστάσεις για βελτίωση σε μελλοντικές σχετικές εργασίες:

- **Αύξηση του όγκου των δεδομένων**. Μια μεγαλύτερη συλλογή δεδομένων μπορεί να βελτιώσει την ακρίβεια και την γενίκευση των μοντέλων. Επιπλέον, μπορεί να εξεταστεί η συλλογή δεδομένων από άλλες κοινωνικές πλατφόρμες για μια πιο πλήρη ανάλυση του συναισθήματος.
- **Υπερ-δειγματοληψία** (over sampling) και η **υπο-δειγματοληψία** (under sampling). Αυτές οι τεχνικές χρησιμοποιούνται για την διαχείριση ανισορροπιών στα δεδομένα όταν υπάρχει ένα μη ισορροπημένο πλήθος δειγμάτων ανά κλάση. Αν και δοκιμάστηκαν ορισμένες σχετικές τεχνικές, ωστόσο δεν τις εφαρμόσαμε στα τελικά πειράματα, καθώς δεν πρόσφεραν σημαντική απόδοση στα αποτελέσματά μας. Βέβαια, οι εν λόγω τεχνικές μπορούν να ληφθούν υπόψιν σε μελλοντικές εργασίες.
- **Κατηγοριοποίηση περισσότερων κλάσεων**. Στην εργασία μας εφαρμόστηκε ταξινόμηση με 2 και 3 κλάσεις. Προφανώς, ο αριθμός των κλάσεων μπορεί να ποικίλλει ανάλογα με τον σκοπό του προβλήματος, περιλαμβάνοντας π.χ. συναισθήματα όπως θυμός, φόβος, έκπληξη.
- **Τεχνικές προ-επεξεργασίας**. Η προ-επεξεργασία των δεδομένων πριν από την εκπαίδευση των μοντέλων ενδέχεται να επηρεάσει σημαντικά την απόδοση. Προφανώς, η επιλογή της βέλτιστης τεχνικής προ-επεξεργασίας, εξαρτάται από τα δεδομένα και τον συγκεκριμένο σκοπό της κάθε εφαρμογής.
- **Επιλογή μοντέλου**. Σίγουρα η κατάλληλη επιλογή μοντέλου για ένα πρόβλημα δεν είναι εύκολη υπόθεση. Όπως είδαμε και στα αποτελέσματά μας, το κάθε μοντέλο επιτυγχάνει διαφορετική απόδοση η οποία εξαρτάται απο πολλές παραμέτρους, όπως το σύνολο

δεδομένων που χρησιμοποιείται, ο διανυσματοποιητής που έχει επιλεγεί καθώς και η ρύθμιση των υπερ-παραμέτρων των μοντέλων.

- **Αρχιτεκτονική μοντέλων**. Η χρήση πιο σύνθετων αρχιτεκτονικών νευρωνικών δικτύων (ως προς το πλήθος στρωμάτων και νευρώνων), σαφώς επηρεάζει την απόδοση των μοντέλων. Στην εργασία μας δοκιμάσαμε διάφορες αρχιτεκτονικές μέχρι να εντοπίσουμε αυτές που είναι οι πιο κατάλληλες.

Βιβλιογραφία

[1] D. Paul and D. Harvey, "*Intro to Python – for Computer Science and Data Science*", 1st Edition, Pearson Education, 2019.

[2] C. Uslu, "What is Kaggle?", 2022, [Online]

Available: https://www.datacamp.com/blog/what-is-kaggle. [Accessed: 7/11/2022]

[3] P. Raja, "What is Google Colab?", 2022, [Online] Available: https://www.scaler.com/topics/what-is-google-colab/. [Accessed: 7/11/2022]

[4] S. Bird, E. Klein, and E. Loper, "*Natural Language Processing with Python*", 1st Edition, O'Reilly, 2019.

[5] C. D. Manning, P. Raghavan, and S. Hinrich, "*Introduction to information retrieval*", 1st Edition, Cambridge University Press, 2008.

[6] A. Barbaresi, "Simplemma: a simple multilingual lemmatizer for Python", 2023, [Online] Available: https://pypi.org/project/simplemma/. [Accessed: 9/11/2022]

[7] D. Paul and D. Harvey, "*Python for Programmers*", 1st edition, Pearson Education, 2019.

[8] Matplotlib, "Matplotlib: Visualization with Python", [Online] Available: https://matplotlib.org/. [Accessed: 9/11/2022]

[9] Seaborn, "seaborn: statistical data visualization", [Online] Available: https://seaborn.pydata.org/index.html. [Accessed: 9/11/2022]

[10] Hugging Face, "Transformers", [Online] Available: https://huggingface.co/docs/transformers/index. [Accessed: 9/11/2022]

[11] Geeksforgeeks, "NLP Gensim Tutorial – Complete Guide For Beginners", 2022, [Online] Available: https://www.geeksforgeeks.org/nlp-gensim-tutorial-complete-guide-for-beginners/. [Accessed: 10/11/2022]

[12] A. P. Jain and V. D. Katkar, "Sentiments analysis of Twitter data using data mining", International Conference on Information Processing (ICIP), IEEE, Pune, India, 2015, pp. 807-810.

[13] Twitter Developer Platform, "Twitter API", [Online] Available: https://developer.twitter.com/en/docs/twitter-api. [Accessed: 10/11/2022]

[14] Twitter Developer Platform, "About the Twitter API", [Online] Available: https://developer.twitter.com/en/docs/twitter-api/getting-started/about-twitter-api. [Accessed: 10/11/2022]

[15] Twitter Developer Platform, "Twitter API v2", [Online] Available: https://developer.twitter.com/en/portal/products/essential. [Accessed: 10/11/2022]

[16] W3schools, "JSON - Introduction", [Online] Available: https://www.w3schools.com/js/js_json_intro.asp. [Accessed: 10/11/2022]

[17] J. Roesslein, "Tweepy", 2022, [Online] Available: https://docs.tweepy.org/en/stable/api.html#api-reference. [Accessed: 10/11/2022]

[18] Twitter Developer Platform, "Authentication", [Online] Available: https://developer.twitter.com/en/docs/authentication/overview. [Accessed: 12/11/2022]

[19] J. Roesslein, "Authentication Tutorial", 2009, [Online] Available: https://docs.tweepy.org/en/v3.5.0/auth_tutorial.html. [Accessed: 12/11/2022]

[20] J. Roesslein, "Authentication", 2022, [Online] Available: https://docs.tweepy.org/en/stable/authentication.html. [Accessed: 12/11/2022]

[21] Twitter, "What's new in the Twitter API," [Online] Available: https://developer.twitter.com/en/docs/twitter-api/migrate/whats-new. [Accessed: 12/11/2022]

[22] Twitter Developer Platform , "Conversation ID", [Online] Available: https://developer.twitter.com/en/docs/twitter-api/conversation-id. [Accessed: 13/11/2022]

[23] Tweepy Documentation, "API", [Online] Available: https://docs.tweepy.org/en/stable/api.html. [Accessed: 13/11/2022]

[24] Twitter Developers, "GET users/show", [Online] Available: https://developer.twitter.com/en/docs/twitter-api/v1/accounts-and-users/follow-search-get-users/api-reference/get-users-show. [Accessed: 13/11/2022]

[25] Tweepy Documentation, "Client", [Online] Available: https://docs.tweepy.org/en/stable/client.html. [Accessed: 13/11/2022]

[26] Twitter Developers, "Users lookup", [Online] Available: https://developer.twitter.com/en/docs/twitter-api/users/lookup/api-reference/get-users-by-username-username. [Accessed: 15/11/2022]

[27] Tweepy API Documentation, "API", [Online] Available: https://docs.tweepy.org/en/stable/api.html#tweepy.API.available_trends. [Accessed: 15/11/2022]

[28] Tweepy API Documentation, "Trends", [Online] Available: https://docs.tweepy.org/en/stable/api.html#tweepy.API.get_place_trends. [Accessed: 15/11/2022]

[29] Internet live stats, "Twitter Usage Statistics", [Online] Available: https://www.internetlivestats.com/twitter-statistics/. [Accessed: 15/11/2022]

[30] B. Mahesh, "Machine Learning Algorithms - A Review", *in International Journal of Science and Research (IJSR)*, vol. 7, no. 1, pp. 1-7, Jan. 2019.

[31] I. Vlahavas, P. Kefalas, N. Bassiliades, F. Kokkoras, and I. Sakellariou, "*Artificial Intelligence*", 3rd Edition, V.Giurdas Publications, 2006.

[32] Y. LeCun, C. Cortes, and C. Burges, "MNIST Handwritten Digit Database," in Encyclopedia of Machine Learning, 2nd ed., C. Sammut and G. I. Webb, Eds. New York, NY, USA: Springer, 2011, pp. 545–546. Available: https://en.wikipedia.org/wiki/MNIST_database. [Accessed: 2/4/2023]

[33] N. Fragkis, "Shops Reviews for Sentiment Analysis in Greek Language", [Online] Available: https://www.kaggle.com/datasets/nikosfragkis/skroutz-shop-reviews-sentiment-analysis [Accessed: 25/04/2023]

[34] Scikit Learn, "scikit-learn Machine Learning in Python", [Online] Available: https://scikit-learn.org/stable/ [Accessed: 16/11/2022]

[35] NumPy, "NumPy Documentation", [Online] Available: https://numpy.org/doc/ [Accessed: 18/11/2022]

[36] Pandas, "pandas documentation", [Online] Available: https://pandas.pydata.org/docs/ [Accessed: 18/11/2022]

[37] L. Ma and Y. Zhang, "Using Word2Vec to Process Big Text Data", IEEE International Conference on Big Data (Big Data), IEEE, Santa Clara, 2015, pp. 2380-2381.

[38] Κ. Διαμαντάρας και Δ. Μπότσης, "*Μηχανική Μάθηση*", Πρώτη Έκδοση, Κλειδάριθμος, 2019.

[39] K. Leung, "Micro, Macro & Weighted Averages of F1 Score, Clearly Explained", 2022, [Online] Available: https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f [Accessed: 10/5/2023]

[40] A. J. Myles, R. N. Feudale, Y. Liu, N. A. Woody and S. D. Brown, "An introduction to decision tree modeling", *J. Chemometrics*, vol. 18, pp. 275-285, 2004.

[41] UCI - Machine Learning Repository, "Iris Data Set", [Online] Available: https://archive.ics.uci.edu/ml/datasets/iris [Accessed: 19/11/2022]

[42] A. Heemann, D. Velinova, and M. Gastegger, "Decision Trees & Random Forests", 2017, [Online] Available: https://www.researchgate.net/publication/320384121_Textklassification_-_Decision_Trees_Random_Forests [Accessed: 19/11/2022]

[43] O. Kramer, "*Dimensionality Reduction with Unsupervised Nearest Neighbors*", Springer, 2013.

[44] G. Biau και E. Scornet, "A random forest guided tour", *Journal of Machine Learning Research*, vol. 16, pp. 1-42, Apr. 2016.

[45] A.C. Müller and S. Guido, "*Introduction to Machine Learning with Python: A Guide for Data Scientists*", 1st Edition, O'Reilly Media, 2016.

[46] J. Brownlee, "Support Vector Machines for Machine Learning", 2016, [Online] Available: https://machinelearningmastery.com/support-vector-machines-for-machine-learning/ [Accessed: 21/04/2023]

[47] J. Brownlee, "One-vs-Rest and One-vs-One for Multi-Class Classification", 2020, [Online] Available: https://machinelearningmastery.com/one-vs-rest-and-one-vs-one-for-multi-class-classification/ [Accessed: 21/04/2023]

[48] A. Shrestha and A. Mahmood, "Review of Deep Learning Algorithms and Architectures", *IEEE Access*, vol. 7, pp. 1-26, April 2019.

[49] Y. LeCun, D. Touresky, G. Hinton and T. Sejnowski, "A theoretical framework for back-propagation", in Proceedings of the 1988 Connectionist Models Summer School, Pittsburgh, 1988, pp. 21-28.

[50] Y. LeCun, Y. Bengio and G. Hinton, "Deep learning", *Nature*, vol. 521, pp. 436-444, May 2015.

[51] Keras, "About Keras", [Online] Available: https://keras.io/about/ [Accessed: 11/05/2023]

[52] Keras, "IMDB movie review sentiment classification dataset", [Online] Available: https://keras.io/api/datasets/imdb/ [Accessed: 11/05/2023]

[53] Keras, "MNIST digits classification dataset", [Online] Available: https://keras.io/api/datasets/mnist/ [Accessed: 11/05/2023]

[54] Keras, "Fashion MNIST dataset, an alternative to MNIST", [Online] Available: https://keras.io/api/datasets/fashion_mnist/ [Accessed: 11/05/2023]

[55] Keras, "Boston Housing price regression dataset", [Online] Available: https://keras.io/api/datasets/boston_housing/ [Accessed: 11/05/2023]

[56] Keras, "CIFAR10 small images classification dataset", [Online] Available: https://keras.io/api/datasets/cifar10/ [Accessed: 11/05/2023]

[57] Keras, "CIFAR100 small images classification dataset", [Online] Available: https://keras.io/api/datasets/cifar100/ [Accessed: 11/05/2023]

[58] A. Krizhevsky, "The CIFAR-10 dataset", [Online] Available: https://www.cs.toronto.edu/~kriz/cifar.html [Accessed: 11/05/2023]

[59] S. Russel and P. Norvig, "*Artificial Intelligence. A Modern Approach*", 2nd Edition, Pearson Education, 2003.

[60] D. W. Otter, J. R. Medina and J. K. Kalita, "A Survey of the Usages of Deep Learning for Natural Language Processing", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, pp. 604-624, Feb 2021.

[61] R. M. Larry and L. C. Jain, "Recurrent neural networks", *Design and Applications*, pp. 64-67, 2001.

[62] M. Jordan, "Serial Order: a Parallel Distributed Approach", 1986.

[63] J. L. Elman, "Finding Structure in time", *Cognitive Science*, vol. 14, pp. 179-211, 1990.

[64] Y. LeCun, Y. Bengio, L. Bottou and P.Haffner, "Gradient-Based Learning Applied to Document Recognition", *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, 1998.

[65] Z. Li, F. Liu, W. Yang, S. Peng and J. Zhou, "A survey of convolutional neural networks: analysis, applications, and prospects", IEEE transactions on neural networks and learning systems, pp. 1-22, June 2021.

[66] P. Mahapattanakul, "Convolutional Neural Network", *Becoming Human: Exploring Artificial Intelligence & What it Means to be Human*, Nov. 2019.

[67] Φ. Δουγαλή, "Αξιοποίηση του Γλωσσικού Μοντέλου BERT για την Αναγνώριση Ονοματικών Οντοτήτων σε Σώμα Τουριστικών Κειμένων στην Ελληνική", Μεταπτυχιακή Διατριβή, Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης, 2020.

[68] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, "Attention is All you Need", in Advances in Neural Information Processing Systems, vol. 30, 2017, pp. 5998–6008.

[69] F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain", Cornell Aeronautical Laboratory, *Psychological Review*, vol. 65, no. 6, pp. 386-408, 1958.

[70] G. Cybenko, "Approximation by Superpositions of a Sigmoidal Function", *Mathematics of Control, Signals, and Systems*, vol. 2, pp. 303-314, 1989.

[71] K. Funahashi, "On the Approximate Realization of Continuous Mappings by Neural Networks", *Neural Networks*, vol. 2, pp. 183-192, 1989.

[72] P. Werbos, "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences", Thesis (PhD), Harvard University, 1974.

[73] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, 1997.

[74] T. Mikolov, K. Chen, G. Corrado and J. Dean, "Efficient estimation of word representations in vector space", *arXiv preprint*, vol. arXiv:1301.3781, 2013.

[75] X. Rong, "word2vec parameter learning explained", *arXiv preprint*, arXiv:1411.2738, 2014.

[76] J. Hoffmann, "Training Compute-Optimal Large Language Models", *arXiv preprint*, arXiv:2203.15556, 2022.

[77] L. Weng, "The Transformer Family", 2020, [Online] Available: https://lilianweng.github.io/posts/2020-04-07-the-transformer-family/ [Accessed: 12/04/2023]

[78] L. Kuang-Huei, X. Chen, G. Hua, H. Houdong and H. Xiaodong, "Stacked cross attention for image-text matching", Proceedings of the European conference on computer vision (ECCV), Munich, Germany, 2018, pp. 212–228.

[79] D. Jacob, M. Chang, K. Lee and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding", *arXiv preprint*, arXiv:1810.04805, 2018.

[80] S. Victor, D. Lysandre, C. Julien and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter", *arXiv preprint*, arXiv:1910.01108, 2019.

[81] L. Yinhan, O. Myle, G. Naman, D. Jingfei, J. Mandar, D. Chen, O. Levy, M. Lewis, Z. Luke and V. Stoyanov, "Roberta: A Robustly Optimized BERT Pretraining Approach", *arXiv preprint*, arXiv:1907.11692, 2019.

[82] A. Radford, W. Jeffrey, R. Child, D. Luan, D. Amodei and I. Sutskever, "Language Models are Unsupervised Multitask Learners", 2019.

[83] J. Koutsikakis, I. Chalkidis, P. Malakasiotis, I. Androutsopoulos, "GREEK-BERT: The Greeks visiting Sesame Street", 11th Hellenic Conference on Artificial Intelligence, Greece, 2020, pp. 1-8.

[84] Hugging Face, "RoBERTa", [Online] Available: https://huggingface.co/docs/transformers/model_doc/roberta [Accessed: 24/05/2023]

[85] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, O. Myle, L. Zettlemoyer and V. Stoyanov, "Unsupervised cross-lingual representation learning at scale", *arXiv preprint*, arXiv:1911.02116, 2019.

[86] L. Guillaume and A. Conneau, "Cross-lingual language model pretraining", *arXiv preprint*, arXiv:1901.07291, 2019.

[87] Hugging Face, "Model Card for DistilBERT base multilingual (cased)", [Online] Available: https://huggingface.co/distilbert-base-multilingual-cased [Accessed: 24/05/2023]

[88] A. Radford, K. Narasimhan, T. Salimans and I. Sutskever, "Improving language understanding by generative pre-training", 2018.

[89] Hugging Face, "Greek (el) GPT2 model", [Online] Available: https://huggingface.co/lighteternal/gpt2-finetuned-greek [Accessed: 26/05/2023]

[90] T. Dehaene, "Dutch GPT2: Autoregressive Language Modelling on a budget", 2020, [Online] Available: https://blog.ml6.eu/dutch-gpt2-autoregressive-language-modelling-on-a-budget-cff3942dd020 [Accessed: 26/05/2023]

[91] M. Wankhade, A. C. S. Rao and C. Kulkarni, "A survey on sentiment analysis methods, applications, and challenges", *Artificial Intelligence Review*, vol. 55, pp. 5731-5780, 2022.

[92] A. Tsakalidis, S. Papadopoulos and I. Kompatsiaris, "An ensemble model for cross-domain polarity classification on twitter", Web Information Systems Engineering–WISE 2014: 15th International Conference, Springer International Publishing, Thessaloniki, Greece, 2014, pp. 1-11.

[93] P. Ekman, "An argument for basic emotions", *Cognition & Emotion*, vol. 6, pp. 169-200, 1992.

[94] G. Kalamatianos, D. S. Mallis, S. Symeonidis and A. Arampatzis, "Sentiment analysis of greek tweets and hashtags using a sentiment lexicon", Proceedings of the 19th panhellenic conference on informatics, Greece, 2015, pp. 63-68.

[95] G. Petasis, D. Spiliotopoulos, N. Tsirakis and P. Tsantilas, "Sentiment analysis for reputation management: Mining the greek web", Artificial Intelligence: Methods and Applications: 8th Hellenic Conference on AI, SETN 2014, Springer International Publishing, Ioannina, Greece, 2014, pp. 327-340.

[96] D. Bilianos, "Experiments in text classification: Analyzing the sentiment of electronic product reviews in greek", *Journal of Quantitative Linguistics*, vol. 29, pp. 374-386, 2022.

[97] M. Giatsoglou, M. G. Vozalis, K. Diamantaras, A. Vakali, G. Sarigiannidis and K. Ch. Chatzisavvas, "Sentiment analysis leveraging emotions and word embeddings", *Expert Systems with Applications*, vol. 69, pp. 214-224, 2017.

[98] N. Avgeros, "Skroutz Sentiment Analysis", 2022, [Online] Available: https://www.kaggle.com/code/nikosavgeros/skroutz-sentiment-analysis [Accessed: 20/12/2022]

[99] N. Fragkis, "Skroutz Sentiment Analysis with BERT (Greek)", 2022, [Online] Available: https://www.kaggle.com/code/nikosfragkis/skroutz-sentiment-analysis-with-bert-greek [Accessed: 20/12/2022]

[100] G. Alexandridis, I. Varlamis, K. Korovesis, G. Caridakis and P. Tsantilas, "A survey on sentiment analysis and opinion mining in Greek social media", *Information*, vol. 12, no. 8, pp. 1-17, 2021.

[101] D. Beleveslis, C. Tjortjis, D. Psaradelis and D. Nikoglou, "A hybrid method for sentiment analysis of election related tweets", in 2019 4th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM), IEEE, 2019, pp. 1-7.

[102] D. Antypas, A. Preece and J. Camacho-Collados, "Negativity spreads faster: A large-scale multilingual Twitter analysis on the role of sentiment in political communication", *Online Social Networks and Media*, vol. 33, 100242, pp. 1-15, 2023.

[103] Twitter Developer Platform, "Search Tweets", [Online] Available: https://developer.twitter.com/en/docs/twitter-api/tweets/search/api-reference/get-tweets-search-all [Accessed: 27/12/2022]

[104] Twitter Developer Platform, "Search Tweets", [Online] Available: https://developer.twitter.com/en/docs/twitter-api/tweets/search/api-reference/get-tweets-search-recent [Accessed: 27/12/2022]

[105] Twitter Developer Platform, "Twitter API v2 data dictionary", [Online] Available: https://developer.twitter.com/en/docs/twitter-api/data-dictionary/object-model/user [Accessed: 27/12/2022]

[106] Twitter Developer Platform, "Twitter API v2 data dictionary", [Online] Available: https://developer.twitter.com/en/docs/twitter-api/data-dictionary/object-model/tweet [Accessed: 27/12/2022]

[107] K. Esmukov, "geopy", 2022, [Online] Available: https://github.com/geopy/geopy/tree/master/geopy/geocoders [Accessed: 05/02/2023]

[108] github, "Folium", [Online] Available: https://python-visualization.github.io/folium/ [Accessed: 05/02/2023]

[109] Scikit-learn, "train test split", [Online] Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html [Accessed: 07/02/2023]

[110] fasttext, "Word vectors for 157 languages", [Online] Available: https://fasttext.cc/docs/en/crawl-vectors.html [Accessed: 10/02/2023]

[111] Keras, "Adam", [Online] Available: https://keras.io/api/optimizers/adam/ [Accessed: 15/02/2023]

[112] Keras, "Dropout layer", [Online] Available: https://keras.io/api/layers/regularization_layers/dropout/ [Accessed: 15/02/2023]

[113] Keras, "Dense layer", [Online] Available: https://keras.io/api/layers/core_layers/dense/ [Accessed: 15/02/2023]

[114] Keras, "Flatten layer", [Online] Available: https://keras.io/api/layers/reshaping_layers/flatten/ [Accessed: 15/02/2023]

[115] Scikit learn, "Supervised learning: predicting an output variable from high-dimensional observations", [Online] Available: https://scikit-learn.org/stable/tutorial/statistical_inference/supervised_learning.html [Accessed: 15/02/2023]

[116] Keras, "Model training APIs", [Online] Available: https://keras.io/api/models/model_training_apis/ [Accessed: 15/02/2023]

[117] Python, "tkinter", [Online] Available: https://docs.python.org/3/library/tkinter.html [Accessed: 06/04/2023]

[118] "Movie Review Data", [Online] Available: https://www.cs.cornell.edu/people/pabo/movie-review-data/ [Accessed: 09/06/2023]

Παράρτημα Α: Κώδικας σε Python

Αρχείο cover.ipynb

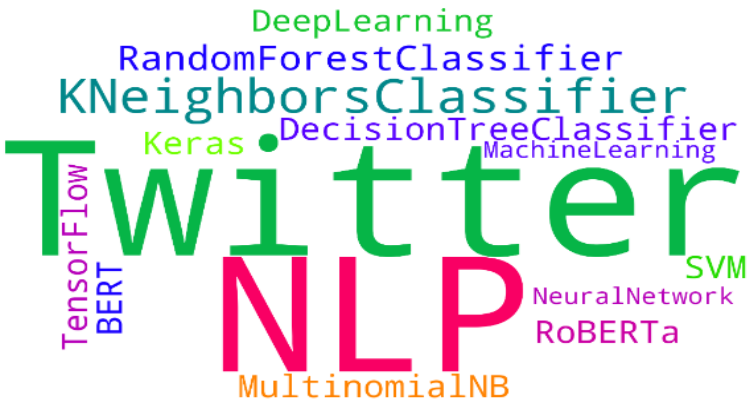
```
from wordcloud import WordCloud
import imageio
mask = imageio.imread('mask_oval.png')

combined_tweets = " ".join(["Twitter", "NLP", "KNeighborsClassifier", "RandomForestClassifier",
"DecisionTreeClassifier", "MultinomialNB", "BERT", "RoBERTa", "SVM", "Keras", "TensorFlow",
"DeepLearning", "MachineLearning",
"NeuralNetwork"])

wordcloud = WordCloud(colormap='prism', background_color='white', max_words=100, mask=mask)
wordcloud = wordcloud.generate(combined_tweets)
wordcloud = wordcloud.to_file(cover.png')

from IPython.display import Image
Image(filename='cover.png', width=500)
```

Output:



Twitter API v1.1 και ενημερωμένα Endpoints v2

Twitter API v1.1

user_timeline

Προβολή πρόσφατων tweets του λογαριασμού NASA. Ορίζεται count=5 για ανάκτηση των τελευταίων 5 tweets.

```
# Twitter API v1.1
# OAuth 1.0a
import tweepy
import my_key

auth = tweepy.OAuth1UserHandler(
  my_key.consumer_key,
  my_key.consumer_secret,
  my_key.access_token,
  my_key.access_token_secret
)
```

```
api = tweepy.API(auth)
tweets = api.user_timeline(screen_name='nasa', count=5)
for tweet in tweets:
    print(f'{tweet.user.screen_name}: {tweet.text}\n')
```

home_timeline

Προβολή των 5 πρόσφατων tweets και retweets του τρέχοντος χρήστη. Εμφανίζονται και τα tweets, retweets των φίλων του τρέχοντος χρήστη.

```
tweets = api.home_timeline()
for tweet in tweets:
    print(f'{tweet.user.screen_name}: {tweet.text}\n')
```

search_tweets

Χρήση μεθόδου search_tweets, για αναζήτηση 3 tweets σχετικά με ποδόσφαιρο. Χρησιμοποιείται το βοηθητικό αρχείο "tweet_utilities.py" το οποίο περιέχει μεθόδους για ταυτοποίηση εφαρμογής και εκτύπωση tweets.

Αρχείο tweet_utilities.py

```
import my_key
import tweepy
def get_API():
   # configure the OAuthHandler
   auth = tweepy.OAuth1UserHandler(
      my_key.consumer_key,
      my_key.consumer_secret,
      my_key.access_token,
      my_key.access_token_secret
   )
   return tweepy.API(auth)

def print_tweets(tweets):
   for tweet in tweets:
      print(f'{tweet.user.screen_name}:', end=' ')
      print(f'\n  ORIGINAL: {tweet.text}')

# main program
from tweet_utilities import print_tweets
from tweet_utilities import get_API
api = get_API()
tweets = api.search_tweets(q='football', count=3)
print_tweets(tweets)
```

Twitter API v2

get_users_following και get_users_followers

Προβολή των 20 **φίλων** (following) και **ακόλουθων** (followers) του λογαριασμού NASA (id=`11348282`). Χρησιμοποιείται η κλάση **Paginator** για σελιδοποίηση αποτελεσμάτων και η μέθοδος **flatten**, για επιπεδοποίηση των αποτελεσμάτων.

```
# Twitter API v2
# OAuth 2.0
import tweepy
import my_key
```

```
client = tweepy.Client(bearer_token=my_key.bearer_token,
                       wait_on_rate_limit=True)
# (limit * max_results) => 2 κλήσεις * 10 tweets = 20 tweets
paginator_friends = tweepy.Paginator(client.get_users_following, 11348282, max_results=10, limit=2)
for user in paginator_friends.flatten(limit=20):
    print(user.name)

paginator_followers = tweepy.Paginator(client.get_users_followers, 11348282, max_results=10, limit=2)
for user in paginator_followers.flatten(limit=20):
    print(user.name)
```

search_recent_tweets

Χρήση μεθόδου search_recent_tweets για την εξόρυξη tweets της τελευταίας εβδομάδας.

```
import tweepy
import my_key
client = tweepy.Client(bearer_token=my_key.bearer_token,
                       wait_on_rate_limit=True)


def print_tweets(tweets):
    for tweet, user in zip(tweets.data, tweets.includes['users']):
        print(f'{user.username}:', end=' ')
        print(f'\nTweet: {tweet.text}')

tweets = client.search_recent_tweets(
    query='MasterChefGR',
    expansions=['author_id'],
    max_results=10)
print_tweets(tweets)
```

Twitter Trends API

```
import tweepy
import my_key
auth = tweepy.OAuth2BearerHandler(my_key.bearer_token)
api = tweepy.API(auth=auth, wait_on_rate_limit=True)
available_trends = api.available_trends()
len(available_trends)
# Τυχαίνει η δεύτερη τοποθεσία με δημοφιλή θέματα να είναι ο Καναδάς.
available_trends[1]
# Το woeid της Θεσσαλονίκης είναι 963291
nyc_trends = api.get_place_trends(id=963291)
nyc_trends
# Η λίστα nyc_trends περιέχει ένα λεξικό με κλειδί trends. Το κλειδί αυτό παραπέμπει σε μια λίστα λεξικών.
Το κάθε λεξικό αποτελεί ένα δημοφιλές θέμα.
nyc_list = nyc_trends[0]['trends']
# λίστα nyc_list περιλαμβάνει τα λεξικά – θέματα.
# Σαρώνουμε την nyc_list, και όποιο θέμα έχει tweet_volume > 10000, το θέμα αυτό μπαίνει στη λίστα. Το
10000 είναι το όριο που θέτει το Twitter. Οπότε αν η παρακάτω συνθήκη ισχύει για ένα θέμα τότε αυτό
μπαίνει στη λίστα. Έτσι, η λίστα nyc_list ενημερώνεται με τα θέματα που έχουν πάνω από 10000 tweets.
nyc_list = [t for t in nyc_list if t['tweet_volume']]
# Κάνουμε import το itemgetter
from operator import itemgetter
# Κάνουμε φθίνουσα ταξινόμηση θεμάτων με βάση το tweet_volume.
nyc_list.sort(key=itemgetter('tweet_volume'), reverse=True)
# Εμφανίζουμε τα 30 ονόματα των πρώτων θεμάτων. Θέματα με κάτω από 10000 tweets δεν εμφανίζονται.
for trend in nyc_list[:30]:
    print(trend['name'])
```

Twitter Streaming API

Αρχείο tweetlistener.py

```
import tweepy
""" Υποκλάση της StreamingClient που επεξεργάζεται τα tweets καθώς φτάνουν """
class TweetListener(tweepy.StreamingClient):
    def __init__(self, bearer_token, limit=10):
        self.tweet_count = 0
        self.TWEET_LIMIT = limit

        super().__init__(bearer_token, wait_on_rate_limit=True)

    """Κλήση όταν η προσπάθεια σύνδεσης είναι επιτυχής."""
    def on_connect(self):
        print('Connection successful\n')

    """ Καλείται όταν το Twitter στέλνει ένα νέο tweet."""
    def on_response(self, response):

        try:
            # Λαμβάνει το όνομα χρήστη που έστειλε το tweet.
            username = response.includes['users'][0].username
            print(f'Username: {username}')
            print(f' Tweet text: {response.data.text}')
            print()
            self.tweet_count += 1

        except Exception as e:
            print(f'Exception occured: {e}')
            self.disconnect()

     # Αν το TWEET_LIMIT είναι ίσο με το tweet_count, τερματίζεται η ροή.
         if self.tweet_count == self.TWEET_LIMIT:
            self.disconnect()
```

Επεξήγηση κώδικα αρχείου tweetlistener.py

Αρχικά, δημιουργείται η κλάση TweetListener ως υποκλάση της StreamingClient. Αυτό εξασφαλίζει ότι η νέα κλάση θα διαθέτει τις προεπιλεγμένες υλοποιήσεις των μεθόδων της υπερκλάσης. Όταν δημιουργηθεί ένα αντικείμενο TweetListener, καλείται η μέθοδος __init__ της κλάσης TweetListener. Η παράμετρος bearer_token χρησιμοποιείται για τον έλεγχο ταυτότητας. Η παράμετρος limit ορίζει το πλήθος των tweets προς ανάκτηση. Από προεπιλογή έχει τιμή 10. Επίσης, δηλώνουμε τις μεταβλητές tweet_count και TWEET_LIMIT. Στην TWEET_LIMIT αποθηκεύεται το όριο που θέτουμε όσον αφορά το πλήθος tweets προς ανάκτηση. Κάθε φορά που η εφαρμογή λαμβάνει ένα tweet, ο μετρητής tweet_count αυξάνει κατά ένα. Όταν ο μετρητής φτάσει στο όριο των tweets, δηλαδή ισχύει `self.tweet_count == self.TWEET_LIMIT`,τότε τερματίζεται η ροή. Η μέθοδος on_connect καλείται όταν η εφαρμογή συνδεθεί με επιτυχία στη ροή Twitter. Παρακάμπτουμε την προεπιλεγμένη υλοποίηση της μεθόδου για να εμφανιστεί το μήνυμα "Connection successful". Η μέθοδος **on_response** καλείται από όταν λαμβάνεται κάθε tweet. Η **response** παράμετρος, είναι ένα αντικείμενο **Response()** που περιλαμβάνει μια λίστα **data** η οποία έχει το id του tweet, το κείμενο κλπ, ένα λεξικό **includes** το οποίο έχει κλειδί **users** που παραπέμπει σε μια λίστα χρηστών, μια λίστα **errors** που αποθηκεύει τυχόν σφάλματα που μπορεί να προκύψουν και μία λίστα **matching_rules** η οποία περιλαμβάνει ένα αντικείμενο μεταδεδομένων που υποδεικνύει ποιος κανόνας ή κανόνες ταίριαζαν με τα tweets που ελήφθησαν. Έτσι, για να ανακτήσουμε το username του χρήστη γράφουμε:

```
response.includes['users'][0].username
```

Ουσιαστικά, η παραπάνω γραμμή παραπέμπει σε μια λίστα **ενός χρήστη** όπου ανακτούμε το username. Για την ανάκτηση του κειμένου του tweet, γράφουμε:

```
response.data.text
```

Εκκίνηση επεξεργασίας ροής

```
import tweepy
import my_key
from tweetlistener import TweetListener
tweet_listener = TweetListener(
    bearer_token=my_key.bearer_token, limit=3)

# Δημιουργία και προσθήκη κανόνα ροής
rule = tweepy.StreamRule('messi')  # Ερώτημα αναζήτησης
tweet_listener.add_rules(rule)
# Εκκίνηση ροής tweets
tweet_listener.filter(
    expansions=['author_id']
)
```

Δημιουργία αντικειμένου **TweetListener** το οποίο αρχικοποιείται με το bearer_token και τον αριθμό tweets που θέλουμε να λάβουμε (limit=3). Θέλουμε να φιλτράρουμε τη ζωντανή ροή αναζητώντας tweets σχετικά με τον messi. Για να το κάνουμε αυτό, δημιουργούμε ένα **StreamRule**. Έπειτα, καλούμε την μέθοδο **add_rules** περνώντας ως όρισμα το StreamRule. Τέλος, χρησιμοποιούμε την μέθοδο **filter**, προκειμένου να ξεκινήσει η διαδικασία της ροής. Στη λίστα **expansions** δηλώνουμε την λέξη κλειδί **author_id** για να υποδείξουμε ότι θέλουμε στην απάντηση για κάθε tweet, να συμπεριλαμβάνεται και το αντικείμενο **user**. Έτσι, έχουμε πρόσβαση στο γνώρισμα **username** κάθε χρήστη. Η έξοδος του προγράμματος είναι η εξής:

```
Connection successful

Username: HoodieOnVeshti
 Tweet text: Maruti Suzuki recalls over 11,000 Grand Vitara cars. Sigh.... secon
d recall in a week. This time, potential defect in the rear seatbelt mounting br
ackets.

Username: thapashvin007
 Tweet text: RT @CNBCTV18Live: Maruti Suzuki recalls Grand Vitara as it suspects
defect in Rear Seat Belt mounting brackets “which in a rare case, may l…

Username: CNBC_Awaaz
 Tweet text: Maruti Suzuki । Grand Vitara के 11,177 यूनिट रीकॉल करेगी https://t.co/
jmMtYSW4W0

Stream connection closed by Twitter
```

Να σημειωθεί πως αν θέλουμε να τρέξουμε ξανά το πρόγραμμα με **νέο ερώτημα αναζήτησης**, πρέπει πρώτα να διαγράψουμε τους υπάρχοντες κανόνες ως εξής:

```
rules = tweet_listener.get_rules().data
ids = [rule.id for rule in rules]
tweet_listener.delete_rules(ids)
```

Κώδικας 6ου Κεφαλαίου

Εξόρυξη tweets (Data_Mining_Twitter.ipynb)

```
# Εισαγωγή απαραίτητων βιβλιοθηκών & αρχείο my_key.py που περιλαμβάνει τον κωδικό bearer_token για την ταυτοποίησή μας στο Twitter.
import tweepy
import preprocessor as p
import sys
from textblob import TextBlob
import csv
import my_key
from deep_translator import GoogleTranslator
import os
import pandas as pd


def terminate():
    print("\nΗ εξόρυξη tweets ολοκληρώθηκε με επιτυχία!")


# Δημιουργια του Object Client
client = tweepy.Client(bearer_token=my_key.bearer_token,
                       wait_on_rate_limit=True)

# Twitter V2: Αναζήτηση tweets με API Search
# Δημιουργία συνάρτησης Sentiment_Analysis()

def Sentiment_Analysis(tweets, query):

    # Εμφάνιση ενός μηνύματος εκκίνησης της διαδικασίας εξόρυξης
    print("Εκκίνηση διαδικασίας εξόρυξης tweets.\n")
    print("LOADING")
    print("Please Wait...\n")

    # Χρήση tweet-preprocessor για διαγραφη URLs και δεσμευμένων λέξεων (reserved words)
    # Ρυθμίζουμε τις επιλογές καθαρισμού με την συνάρτηση set_options()
    p.set_options(p.OPT.URL, p.OPT.RESERVED, p.OPT.HASHTAG, p.OPT.MENTION)

    # Παίρνουμε τις τοποθεσίες για κάθε χρήστη.
    # Χρήση λίστας για να μαζεύουμε όλες τις τοποθεσίες
    lista_topothesion = []

    # Παίρνουμε τα created_at για κάθε tweet
    lista_created_at = []


    # Μετρητής
    i = 0
    # Σαρώνουμε όλους τους χρήστες με for και γεμίζουμε την λίστα lista_topothesion με τις τοποθεσίες τους
    for u in tweets.includes['users']:
        lista_topothesion.append(u.location)
    # Σαρώνουμε τα tweets, για να πάρουμε την ημερομηνία δημιουργίας των tweets
    for t in tweets.data:
        lista_created_at.append(t.created_at)

    print("Η λίστα τοποθεσιών που προέκυψε είναι η εξής:")
    print(lista_topothesion)
    print('\n')
    print('Tweets που βρέθηκαν:\n')

    for tweet, user in zip(tweets.data, tweets.includes['users']):
```

```
            user = user.username
            tweet_text = tweet.text


        # Χρηση μετρητή i, για να παίρνω την τοποθεσία του τρέχοντος χρήστη
            # και το created_at του τρέχοντος tweet
            location = lista_topothesion[i]
            created_at = lista_created_at[i]

            # Καθαρισμός tweet απο δεσμευμένες λέξεις RT και links
            if not tweet_text.startswith('RT'):

                translatorEN = GoogleTranslator(source='auto', target='en')
                textEN = translatorEN.translate(tweet_text)
                # Καθαρισμός tweet απο δεσμευμένες και links
                textEN = p.clean(textEN)
                tweet_text = p.clean(tweet_text)

                # Ενημέρωση του λεξικού sentiment_dict με το polarity
                blob = TextBlob(textEN)
                if blob.sentiment.polarity > 0:
                    sentiment = 'Positive'
                    #sentiment_dict['positive'] += 1
                elif blob.sentiment.polarity == 0:
                    sentiment = 'Neutral'
                    #sentiment_dict['neutral'] += 1
                else:
                    sentiment = 'Negative'
                    #sentiment_dict['negative'] += 1


                # Εμφάνιση του tweet
                print(f'{created_at} {user}: {tweet_text} [{sentiment}]\n')

            # Εξαγωγή των tweets σε αρχείο csv
            # Άρα, θα εισάγουμε στο csv την μεταβλητή tweet_text (Όπως το λάβαμε αρχικά)
            # Το dataset, αρχείο data.csv, θα έχει την μορφή
            # [ Created_at,User,Location,Topic,TweetText,Sentiment ]
            # Αργότερα το εν λόγω dataset θα υποστεί περαιτέρω επεξεργασία με σκοπό την
            # ταξινόμηση συναισθήματος tweets με άλλες μεθόδους μηχανικής μάθησης.


                with open('data.csv', mode='a', newline='') as data:
                    writer = csv.writer(data)
                    writer.writerow([created_at, user, location, query, tweet_text, sentiment])

                i += 1  # Αύξηση του μετρητή i κατά ένα, για να δείχνει στον επόμενο χρήστη και tweet.

        terminate()


# ΜΕΘΟΔΟΣ search_recent_tweets()
# Σύμφωνα με την τεκμηρίωση της μεθόδου, το Twitter διατηρεί τον δείκτη αναζήτησής του μόνο
# για τα tweets των τελευταίων 7 ημερών, και μια αναζήτηση δεν επιστρέφει απαραίτητα όλα τα
# tweets που ταιριάζουν. Ο μέγιστος αριθμός αποτελεσμάτων αναζήτησης που πρέπει να
# επιστραφούν ορίζεται στην παράμετρο max_results και είναι ένας αριθμός μεταξύ 10 και 100.
# Από προεπιλογή, μια απάντηση αιτήματος θα επιστρέψει 10 αποτελέσματα.


def  main():

    # Στην εκκίνηση της διαδικασία εξόρυξης, διαγράφεται το προσωρινό
```

```
    # αρχείο data.csv, ώστε πάντα να περιέχει φρέσκα tweets, τα οποία
    # θα καθαριστούν και θα αναλυθούν μέχρι τελικά να μπούν στο αρχείο #unbalanced_politics.csv
    os.remove("data.csv")

    # Το ερώτημα αναζήτησης
    query = input('Για ποιό θέμα ψάχνετε tweets?')
    # π.χ. query = 'ΕΛΛΗΝΙΚΗ ΛΥΣΗ'

    # Μπορούμε να αναζητήσουμε τουλάχιστον 10 tweets.
    # Κάτω απο 10 tweets δίνει error.
    total_tweets = int(input('Πόσα tweets θέλετε?'))
    #total_tweets = 40

    tweets = client.search_recent_tweets(query=query,
                                         expansions=['author_id','referenced_tweets.id'],
                                         user_fields=['location'],
                                         tweet_fields=['lang','created_at','text'],
                                         max_results=total_tweets)

    Sentiment_Analysis(tweets, query)


if __name__ == '__main__':
    main()
```

Καθαρισμός Dataset (Cleaning_Dataset.ipynb)

```
import pandas as pd
import warnings
import my_key
import re
from geopy import OpenMapQuest
geo = OpenMapQuest(api_key = my_key.mapquest_key)
headers=['Created_at','User','Location','Topic','TweetText','Sentiment']
dataset = pd.read_csv('data.csv', sep=',', names=headers)
dataset


# Διαγραφή όλων των εγγραφών με τιμή NaN.
dataset = dataset.dropna()
dataset
import string
import emoji
EMOJI = re.compile('[\U00010000-\U0010ffff]', flags=re.UNICODE)

def strip_emoji(tweet):
    return EMOJI.sub(r'', tweet)

# Διαγραφή πολλων κενών spaces
def remove_multiple_spaces(tweet):
    return re.sub('\s\s+', ' ', tweet)


# Καθαρισμός απο ειδικούς χαρακτήρες όπως $ και & που υπάρχουν σε κάποιες λέξεις
"""
def filter_chars(tweet):
    my_list = []
    for word in tweet.split():
        if ('$' in word) | ('&' in word):
            my_list.append('')
        else:
```

```
            my_list.append(word)

    return ' '.join(my_list)
"""


# Διαγραφή σημείων στίξης και \r\n χαρακτήρων γραμμής
def strip_chars(tweet):
    # Διαγραφή \n και \r και μετατροπή σε πεζά γράμματα
    tweet = tweet.replace('\n', ' ').replace('\r', ' ').lower()

    # Διαγραφή σημείων στίξης
    bad_chars = 'Ã'+'±'+'ã'+'¼'+'â'+'»'+'«'+'§'+'′′'+'‘'+'’'+'–'+'$'+':'+'"'+'+'+'/'+'*'+'-
'+'<'+'>'+'='+'['+']'+'^'+'_'+'`'+'~'+'{'+'}'+'|'

    # Θέλουμε στο tweet χαρακτήρες εκτός απο τους bad_chars
    my_table_antistoixisis = tweet.maketrans('', '', bad_chars)
    tweet = tweet.translate(my_table_antistoixisis)

    #bad_chars = string.punctuation + 'Ã'+'±'+'ã'+'¼'+'â'+'»'+'«'+'§'+'′′'+'‘'+'’'+'–'
    #my_table_antistoixisis = str.maketrans('', '', bad_chars)   # Θέλουμε στο tweet, χαρακτήρες εκτός απο τους
bad_chars
    #tweet = tweet.translate(my_table_antistoixisis)

    return tweet


list_tweet_text = list(dataset['TweetText'])
nea_lista_tweet = []

for tweet in list_tweet_text:
    #nea_lista_tweet.append(remove_multiple_spaces(filter_chars(strip_chars(strip_emoji(tweet)))))
    nea_lista_tweet.append(remove_multiple_spaces(strip_chars(strip_emoji(tweet))))

dataset.loc[:, 'TweetText'] = nea_lista_tweet
dataset


# Διαγραφή διπλότυπων tweets. Λαμβάνουμε υπόψιν μόνο την στήλη TweetText,
# για τον εντοπισμό διπλότυπων.
# δηλώνουμε ότι η τροποποίηση αφορά το αρχικό dataframe και οχι κάποιο νέο.
# Η μέθοδος duplicated επιστρέφει μια σειρά (Series) με τιμές True για τις εγγραφές που είναι διπλότυπες και
False για τις μοναδικές εγγραφές. Το σύμβολο "~" (not) χρησιμοποιείται για να αντιστρέψει τις τιμές, δηλαδή
να επιστρέψει True για τις μοναδικές εγγραφές.
# Στη συνέχεια, η μέθοδος .loc χρησιμοποιείται για να επιλέξει μόνο τις μοναδικές εγγραφές, αναθέτοντας το
αποτέλεσμα στη μεταβλητή dataset. Με τη χρήση του ~ μέσα στις αγκύλες, επιλέγονται οι γραμμές όπου η
τιμή είναι True.
# Με αυτόν τον τρόπο, η αντικατάσταση των διπλών εγγραφών γίνεται απευθείας στο αρχικό DataFrame
dataset.

dataset = dataset.loc[~dataset.duplicated(subset='TweetText')]
# Για κάθε location του DataFrame, βρίσκουμε τις συντεταγμένες (αν υπάρχουν)
# και με βάση αυτές παίρνουμε την Χώρα και την καταχωρούμε στη λίστα lista_country
# Επίσης, χρησιμοποιούμε το Nominatim API για να χρησιμοποιήσουμε την μέθοδο
# reverse. Η μέθοδος αυτή δίνει περισσότερες πληροφορίες για μια τοποθεσία
# και παίρνει 2 ορίσματα το latitude και το longitude.
# Το αποτέλεσμα της reverse το μετατρέπουμε σε λεξικό raw['address'].
# Απο το λεξικό παίρνουμε το country και city.

from geopy.geocoders import Nominatim
geolocator = Nominatim(user_agent="D_Ergasia")
```

```
list_location = list(dataset['Location'])
lista_country = []
lista_city = []

# Περιφέρεια
lista_state_district = []

def city_country():

    for topothesia in list_location:
        #geo_location = geo.geocode(topothesia)
        geo_location = geolocator.geocode(topothesia)

        if geo_location:
            lat = str(geo_location.latitude)
            lng = str(geo_location.longitude)
            location = geolocator.reverse(lat+","+lng)
            print(location)
            address = location.raw['address']
            print(address)

            # Αν το city υπάρχει στο λεξικό address, τότε βάλε το address['city']
            # αλλιώς βάλε το address['state'] ή address['village']
            if "city" in address:
                city = address['city']
            elif "state" in address:
                city = address['state']
            elif "village" in address:
                city = address['village']
            else:
                city = None


            if "country" in address:
                country = address['country']
            elif "state" in address:
                country = address['state']
            else:
                country = None

            # Κρατάμε μόνο τις Ελληνικές Περιφέρειες
            if "country_code" in address and address['country_code'] == 'gr':
                if "state_district" in address:
                    state_district = address['state_district']
                elif "state" in address:
                    state_district = address['state']
                elif "country" in address:
                    state_district = address['country']
                else:
                    state_district = None
            else:
                state_district = None
            lista_city.append(city)
            lista_country.append(country)
            lista_state_district.append(state_district)

        else:
            lista_city.append(None)
            lista_country.append(None)
            lista_state_district.append(None)
```

```
# Κλήση της μεθόδου city_country()
city_country()
# Δημιουργία dataset "dataset_with_country_city_perifereia"
# Αυτό το dataset θα έχει επιπλέον τις στήλες country, city και state_district (περιφέρεια)
# dataset_with_country_city_perifereia
column_created_at = list(dataset['Created_at'])
column_user = list(dataset['User'])
column_topic = list(dataset['Topic'])
column_tweet_text = list(dataset['TweetText'])
column_sentiment = list(dataset['Sentiment'])
lexiko_country_city_perifereia = { 'created_at':column_created_at, 'username':column_user,
'country':lista_country, 'city':lista_city,
                    'state_district':lista_state_district, 'topic':column_topic, 'tweet':column_tweet_text,
'sentiment':column_sentiment }
dataset_with_country_city_perifereia = pd.DataFrame(lexiko_country_city_perifereia)
dataset_with_country_city_perifereia


# Παρατηρούμε στο dataset_with_country_city_perifereia παραπάνω ότι (ισως) έχουν προκύψει τιμές None
στην στήλη city και country κλπ.
# Κάποιες περιοχές δεν ήταν έγκυρες και δεν πήραμε συντεταγμένες για άκυρες τοποθεσίες.
# Στην μέθοδο city_country, είχαμε προβλέψει πως αν δεν βρεθούν συντεταγμένες για μια τοποθεσία
# τότε στις λίστες lista_city και lista_country και lista_state_district, να βάζει την τιμή None. Και οι λίστες
αυτές γίνανε
# στήλες στο dataset dataset_with_country_city_perifereia.
# Πρέπει να διαγράψουμε εγγραφές με None
dataset_with_country_city_perifereia = dataset_with_country_city_perifereia.dropna()
dataset_with_country_city_perifereia
import csv
for t in dataset_with_country_city_perifereia.itertuples():
    with open(unbalanced_politics.csv', mode='a', newline='') as data:
        writer = csv.writer(data)
        created_at = t.created_at
        username = t.username
        country = t.country
        city = t.city
        state_district = t.state_district
        topic = t.topic
        tweet = t.tweet
        sentiment = t.sentiment
        writer.writerow([created_at, username, country, city, state_district, topic, tweet, sentiment])


for t in dataset_with_country_city_perifereia.itertuples():
    with open('cleaning_dataset_for_map_VER2.csv', mode='a', newline='') as data:
        writer = csv.writer(data)
        created_at = t.created_at
        username = t.username
        country = t.country
        city = t.city
        state_district = t.state_district
        topic = t.topic
        tweet = t.tweet
        sentiment = t.sentiment
        writer.writerow([created_at, username, country, city, state_district, topic, tweet, sentiment])
# Σε αυτό το σημείο έχουμε καθαρίσει οσο γίνεται το αρχικό dataset data.csv.
# Τελικά, έχει προκύψει το dataframe dataset_with_countries_and_cities
# το οποιο μάλιστα μετατράπηκε σε 2 αρχεία csv.
# file1 ->  unbalanced_politics.csv ->  για ανάλυση δεδομένων και για machine learning
# file2 ->  cleaning_dataset_for_map_VER2.csv  ->  για δημιουργία χάρτη που δείχνει τα θετικά, αρνητικά και
ουδέτερα tweets.
```

Χαρτογράφηση tweets (Create_Map.ipynb)

```
import pandas as pd
import tweepy
import my_key
import preprocessor as p
from geopy.geocoders import Nominatim
geolocator = Nominatim(user_agent="D_Ergasia")
from geopy import OpenMapQuest
geo = OpenMapQuest(api_key=my_key.mapquest_key)
import folium
headers=['created_at','username','country','city','state_district','topic','tweet','sentiment']
datasetMAP = pd.read_csv('cleaning_dataset_for_map_VER2.csv', sep=',', names=headers)
datasetMAP


latitude = []
longitude = []
list_location = list(datasetMAP['city'])

for topothesia in list_location:
    #geo_location = geo.geocode(topothesia)
    geo_location = geolocator.geocode(topothesia)


    if geo_location:
        latitude.append(geo_location.latitude)
        longitude.append(geo_location.longitude)
    else:
        latitude.append(None)
        longitude.append(None)

username = list(datasetMAP['username'])
tweet = list(datasetMAP['tweet'])
sentiment = list(datasetMAP['sentiment'])

my_lexiko = { 'username':username, 'lat':latitude, 'lng':longitude, 'tweet':tweet, 'sentiment':sentiment }
df_map = pd.DataFrame(my_lexiko)
df_map
df_map = df_map.dropna()
df_map
# Διαγραφή διπλότυπων εγγραφών
df_map.drop_duplicates(inplace=True)
df_map


# Το όρισμα location της folium, καθορίζει το κέντρο του χάρτη.
# Θέλουμε με την φόρτωση του χάρτη να γίνει εστίαση σε σημεία που έχουμε tweets.
# Θα ελέγξουμε τυχαία το σημείο -> (latitude[1], longitude[1]) (δεύτερη γραμμή df_map)
# και αν υπάρχει σημείο, τότε θα μπούν αυτές οι συντεταγμένες στο όρισμα location
# του folium.Map(). Αν δεν υπάρχει σημείο, τότε θα βάλουμε τυχαία τις συντεταγμένες
# location=[39.8283, -98.5795] που αντιστοιχούν στο γεωγραφικό κέντρο των ηπειρωτικών ΗΠΑ.
# ( http://bit.ly/CenterOfTheUS )


location_center = []

if latitude[1] and longitude[1]:
    location_center.append(latitude[1])
    location_center.append(longitude[1])
else:
    location_center.append(39.8283)
```

```
    location_center.append(-98.5795)

mymap = folium.Map(location=location_center,
                   tiles='Stamen Terrain', zoom_start=5, detect_retina=True)


for t in df_map.itertuples():
    if t.sentiment == "Positive":
        text = ': '.join([t.username, t.tweet])
        pop_up = folium.Popup(text, parse_html=True)
        marker = folium.Marker((t.lat, t.lng), popup=pop_up, draggable=True,
icon=folium.Icon(icon="cloud",color="blue"))
        marker.add_to(mymap)
    elif t.sentiment == "Neutral":
        text = ': '.join([t.username, t.tweet])
        pop_up = folium.Popup(text, parse_html=True)
        marker = folium.Marker((t.lat, t.lng), popup=pop_up, draggable=True,
icon=folium.Icon(icon="cloud",color="orange"))
        marker.add_to(mymap)
    else:
        text = ': '.join([t.username, t.tweet])
        pop_up = folium.Popup(text, parse_html=True)
        marker = folium.Marker((t.lat, t.lng), popup=pop_up, draggable=True,
icon=folium.Icon(icon="cloud",color="red"))
        marker.add_to(mymap)


mymap.save('tweet_map_positive_negative_neutral.html')
mymap
```

Ανάλυση δεδομένων (Data_Analysis_Twitter.ipynb)

```
# Εισαγωγή βιβλιοθηκών
import pandas as pd
import seaborn as sns
!pip install matplotlib --upgrade
import matplotlib.pyplot as plt
import re
# Για αφαίρεση κοινών λέξεων
importnltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words = stopwords.words('greek')
headers=['Created_at','Username','Country','City','State_District','Topic','TweetText','Sentiment']
dataset = pd.read_csv('/content/drive/MyDrive/ΔΙΠΛΩΜΑΤΙΚΗ/unbalanced_politics.csv', sep=',',
names=headers)
dataset.head()
dataset.info()


# Καθαρισμός StopWords στην στήλη TweetText που περιέχει τα tweets μας.
list_tweet_text = list(dataset['TweetText'])
lista_without_stopwords = []
for tweet in list_tweet_text:
    lista_without_stopwords.append(' '.join([word for word in tweet.split() if word not in stop_words]))

dataset.TweetText = lista_without_stopwords
dataset
dataset['Sentiment'].value_counts()
```

```
# Πλοτάρει σε ραβδόγραμμα τα Positive, Neutral και Negative.
plt.figure(figsize = (6,6))
sns.countplot(x = dataset['Sentiment'])


# Δείχνει το πλήθος των tweets για το κάθε κόμμα.
dataset['Topic'].value_counts()
dataset['Sentiment'].value_counts().plot(kind='pie', autopct='%1.1f%%', figsize=(12,6), colors=["green", "red",
"gray"], explode=(0.1,0.1,0.1), shadow=True, startangle=50)


ts = dataset.groupby(['Topic', 'Sentiment']).Sentiment.count().unstack()
ts
ts.plot(kind='bar', figsize=(10,7))
sentiment_ana_perifereia = dataset.groupby(['State_District', 'Sentiment']).Sentiment.count().unstack()
sentiment_ana_perifereia


# Δημιουργία του ραβδογράμματος
sentiment_ana_perifereia.plot(kind='barh', figsize=(8, 8), ylabel='Περιφέρεια', xlabel='Αριθμός Tweets ανά
Περιφέρεια')

# Προσαρμογή των ετικετών και των τίτλων
plt.xlabel('Αριθμός Tweets')
plt.ylabel('Περιφέρεια')
plt.title('Ανάλυση sentiment ανά κλάση')

# Αντιστροφή της κατεύθυνσης των περιφερειών στον άξονα y
plt.gca().invert_yaxis()
# Εμφάνιση του ραβδογράμματος
plt.show()
s = dataset.groupby(['Sentiment', 'State_District', 'Topic']).Sentiment.count().unstack().fillna(0).astype(int)
s
# Δημιουργία του ραβδογράμματος
s.plot(kind='barh', figsize=(6, 28), ylabel='Sentiment ανά Περιφέρεια και Κόμμα')

# Προσαρμογή των ετικετών και των τίτλων
plt.xlabel('Αριθμός tweets')
plt.ylabel('Περιφέρεια')
plt.title('Ανάλυση sentiment ανά περιφέρεια και κόμμα')

# Αντιστροφή της κατεύθυνσης των περιφερειών στον άξονα y
plt.gca().invert_yaxis()
# Εμφάνιση του ραβδογράμματος
plt.show()
# Μετατροπή ημερομηνίας της στήλης Created_at.
# Πρέπει να γίνει για να μπορώ να πάρω αργότερα το έτος, τον μήνα και την ημέρα.
dataset['Created_at'] = pd.to_datetime(dataset['Created_at'])
dataset.head()
# Πλήθος των tweets ανά ημέρα.
view_day = dataset['Created_at'].dt.strftime('%Y-%m-%d').value_counts()
view_day
# Ταξινόμηση με βάση την ημέρα.
view_day_sort = dataset['Created_at'].dt.strftime('%Y-%m-%d').value_counts().sort_index()
view_day_sort
# Υπολογισμός των tweets ανα εβδομάδα
# Η μέθοδος reset_index() δίνει το όνομα index στην στήλη εβδομάδας και το όνομα Πλήθος στην στήλη που
έχει το
# πλήθος των tweets της κάθε εβδομάδας.
tweets_ana_evdomada = dataset['Created_at'].dt.strftime('%Y-
%W').value_counts().sort_index().reset_index(name='Πλήθος')
tweets_ana_evdomada
```

```
# Το tweets_ana_evdomada είναι πλέον ένα dataframe
list_index = list(tweets_ana_evdomada['index'])
nea_lista = []

for i in list_index:
    if i == "2022-45":
        nea_lista.append("7-13νοε")
    elif i == "2022-46":
        nea_lista.append("14-20νοε")
    elif i == "2022-47":
        nea_lista.append("21-27νοε")
    elif i == "2022-48":
        nea_lista.append("28νοε-4δεκ")
    elif i == "2022-49":
        nea_lista.append("5δεκ-11δεκ")
    elif i == "2022-50":
        nea_lista.append("12δεκ-18δεκ")
    elif i == "2022-51":
        nea_lista.append("19δεκ-25δεκ")
    elif i == "2022-52":
        nea_lista.append("26δεκ-31δεκ")
    elif i == "2023-00":
        nea_lista.append("Πρωτοχρονιά")
    elif i == "2023-01":
        nea_lista.append("2ιαν-8ιαν")
    elif i == "2023-02":
        nea_lista.append("9ιαν-15ιαν")
    elif i == "2023-03":
        nea_lista.append("16ιαν-22ιαν")
    elif i == "2023-04":
        nea_lista.append("23ιαν-29ιαν")
    elif i == "2023-05":
        nea_lista.append("30ιαν-5φεβ")
    elif i == "2023-06":
        nea_lista.append("6φεβ-12φεβ")
    elif i == "2023-07":
        nea_lista.append("13φεβ-19φεβ")
    elif i == "2023-08":
        nea_lista.append("20φεβ-26φεβ")

tweets_ana_evdomada['index'] = nea_lista
tweets_ana_evdomada
# Σχεδιασμός ραβδογράμματος για οπτικοποίηση των tweets ανα εβδομάδα
plt.figure(figsize=(15,6))
ax = sns.barplot(x='index', y='Πλήθος', data=tweets_ana_evdomada, edgecolor='black', ci=False,
palette='tab20b_r')
plt.title('Tweets ανά εβδομάδα')
plt.yticks([])
ax.bar_label(ax.containers[0])
plt.ylabel('')
plt.xlabel('')
plt.show()
# Palette Values:
# Accent, Accent_r, Blues, Blues_r, BrBG, BrBG_r, BuGn, BuGn_r, BuPu, BuPu_r, CMRmap, CMRmap_r,
Dark2, Dark2_r, GnBu, GnBu_r, Greens, Greens_r, Greys, Greys_r, OrRd, OrRd_r, Oranges, Oranges_r,
PRGn, PRGn_r, Paired, Paired_r, Pastel1, Pastel1_r, Pastel2, Pastel2_r, PiYG,
# PiYG_r, PuBu, PuBuGn, PuBuGn_r, PuBu_r, PuOr, PuOr_r, PuRd, PuRd_r, Purples, Purples_r, RdBu,
RdBu_r, RdGy, RdGy_r, RdPu, RdPu_r, RdYlBu, RdYlBu_r, RdYlGn, RdYlGn_r, Reds, Reds_r, Set1,
Set1_r, Set2, Set2_r, Set3, Set3_r, Spectral, Spectral_r, Wistia, Wistia_r, YlGn, YlGnBu, YlGnBu_r,
```

```
# YlGn_r, YlOrBr, YlOrBr_r, YlOrRd, YlOrRd_r, afmhot, afmhot_r, autumn, autumn_r, binary, binary_r,
bone, bone_r, brg, brg_r, bwr, bwr_r, cividis, cividis_r, cool, cool_r, coolwarm, coolwarm_r, copper, copper_r,
cubehelix, cubehelix_r, flag, flag_r, gist_earth, gist_earth_r, gist_gray, gist_gray_r,
# gist_heat, gist_heat_r, gist_ncar, gist_ncar_r, gist_rainbow, gist_rainbow_r, gist_stern, gist_stern_r,
gist_yarg, gist_yarg_r, gnuplot, gnuplot2, gnuplot2_r, gnuplot_r, gray, gray_r, hot, hot_r, hsv, hsv_r, icefire,
icefire_r, inferno, inferno_r, jet, jet_r, magma, magma_r, mako,
# mako_r, nipy_spectral, nipy_spectral_r, ocean, ocean_r, pink, pink_r, plasma, plasma_r, prism, prism_r,
rainbow, rainbow_r, rocket, rocket_r, seismic, seismic_r, spring, spring_r, summer, summer_r, tab10, tab10_r,
tab20, tab20_r, tab20b, tab20b_r, tab20c, tab20c_r, terrain, terrain_r,
# twilight, twilight_r, twilight_shifted, twilight_shifted_r, viridis, viridis_r, vlag, vlag_r, winter, winter_r
# *** *** *** ***
# Απο την στήλη Created_at, παίρνω την εβδομάδα
dataset['Created_at'] = dataset['Created_at'].dt.strftime('%Y-%W')
dataset.sort_values(by=['Created_at'], inplace = True)
dataset
list_index = list(dataset['Created_at'])
nea_lista = []

for i in list_index:
    if i == "2022-45":
        nea_lista.append("7-13νοε")
    elif i == "2022-46":
        nea_lista.append("14-20νοε")
    elif i == "2022-47":
        nea_lista.append("21-27νοε")
    elif i == "2022-48":
        nea_lista.append("28νοε-4δεκ")
    elif i == "2022-49":
        nea_lista.append("5δεκ-11δεκ")
    elif i == "2022-50":
        nea_lista.append("12δεκ-18δεκ")
    elif i == "2022-51":
        nea_lista.append("19δεκ-25δεκ")
    elif i == "2022-52":
        nea_lista.append("26δεκ-31δεκ")
    elif i == "2023-00":
        nea_lista.append("Πρωτοχρονιά")
    elif i == "2023-01":
        nea_lista.append("2ιαν-8ιαν")
    elif i == "2023-02":
        nea_lista.append("9ιαν-15ιαν")
    elif i == "2023-03":
        nea_lista.append("16ιαν-22ιαν")
    elif i == "2023-04":
        nea_lista.append("23ιαν-29ιαν")
    elif i == "2023-05":
        nea_lista.append("30ιαν-5φεβ")
    elif i == "2023-06":
        nea_lista.append("6φεβ-12φεβ")
    elif i == "2023-07":
        nea_lista.append("13φεβ-19φεβ")
    elif i == "2023-08":
        nea_lista.append("20φεβ-26φεβ")


dataset['Created_at'] = nea_lista
dataset
c = dataset.groupby(['Created_at', 'Sentiment'], sort=False).Sentiment.count().unstack()
c
c.plot(kind='bar', figsize=(15,6), xlabel='Εβδομάδα')
```

Μηχανική και Βαθιά μάθηση

Εκπαίδευση και προβλέψεις μοντέλων Transformers (bert-gpt-roberta-distilbert-split70-30-80-20.ipynb)

Στο αρχείο αυτό, γίνεται **εκπαίδευση και πρόβλεψη** μοντέλων Transformers σε **τυχαίους διαχωρισμούς** (random splits).

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import os
# TF_CPP_MIN_LOG_LEVEL=2: φιλτράρει πληροφορίες και προειδοποιητικά μηνύματα.
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
import keras
from keras.layers import LSTM, Dense, Input, Dropout, Embedding, Flatten
from keras.models import Model, load_model
from keras.callbacks import EarlyStopping
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.utils import plot_model
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.python.client import device_lib
from tensorflow.keras.optimizers import Adam, SGD
from tensorflow.keras.optimizers.experimental import AdamW
from keras import regularizers
!pip3 install transformers
import torch
import transformers
from transformers import TFBertModel, AutoTokenizer, TFAutoModel, BertTokenizer, TFDistilBertModel
from transformers import AutoModelForMaskedLM, TFXLMRobertaModel, DistilBertTokenizer
from transformers import TFGPT2LMHeadModel, GPT2Model, AutoModel

# Για αφαίρεση κοινών λέξεων (stop words)
import nltk
from nltk.corpus import stopwords
stop_words = stopwords.words('greek')


ans_model = input('Ποιό μοντέλο θέλετε να εκπαιδεύσετε? [bert] [roberta] [distilbert] [gpt]')
# Επιλογή Dataset
ans_dataset = input('Επιλέξτε dataset [politics] ή [skroutz]')
headers=['Created_at','Username','Country','City','State_District','Topic','Text','Sentiment']
headers_skroutz=['id','Text','Sentiment']

if ans_dataset == 'politics':
    ans_un_bal = input('Θέλετε [unbalanced] ή [balanced]?')
    if ans_un_bal == 'unbalanced':
        dataset = pd.read_csv('/kaggle/input/unbalanced-politics/unbalanced_politics.csv', sep=',',
names=headers)
    else:
        dataset = pd.read_csv('/kaggle/input/balanced-politics/balanced_politics.csv', sep=',', names=headers)
else:
    ans_un_bal = 'balanced'
    dataset = pd.read_csv('/kaggle/input/myskroutzdataset/skroutz_dataset.csv', sep=',', names=headers_skroutz)
```

```
dataset.info()
dataset['Sentiment'].value_counts()


if ans_dataset == 'politics':
    ans_class = input('Επιλέξτε αν θέλετε [binary] η [multi] class: ')
else:
    ans_class = 'binary'


# Το dataset περιέχει στην στήλη Sentiment τις ετικέτες neutral, negative και positive.
# Θα χρησιμοποιήσουμε το μοντέλο BERT για να κάνουμε δυαδική ταξινόμηση συναισθήματος,
# οπότε θα κρατήσουμε μόνο 2 ετικέτες, Positive και Negative.
# Tweets που είναι neutral, θα φύγουν με την μέθοδο dropna()


def delete_neutral(dataset):
    list_tweet_text = list(dataset['Text'])
    list_sentiment = list(dataset['Sentiment'])

    nea_lista_tweet_text = []
    nea_lista_sentiment = []

    for sent, tweet in zip(list_sentiment, list_tweet_text):
        if sent == 'Neutral':
            nea_lista_tweet_text.append(None)
            nea_lista_sentiment.append(None)
        else:
            nea_lista_tweet_text.append(tweet)
            nea_lista_sentiment.append(sent)


    dataset['Text'] = nea_lista_tweet_text
    dataset['Sentiment'] = nea_lista_sentiment
    dataset = dataset.dropna()
    return dataset


if ans_class == 'binary' and ans_dataset == 'politics':
    dataset = delete_neutral(dataset)


dataset.info()
ans_stopword_lemma = input('Επιλέξτε αν θέλετε να αφαιρέσετε λέξεις χωρία αξία (stopwords) και αν θέλετε
να εφαρμοστεί λημματοποίηση [yes] ή [no]: ')


# Καθαρισμός Stop Words στην στήλη Text.
if ans_stopword_lemma == 'yes':
    list_tweet_text = list(dataset['Text'])
    lista_without_stopwords = []

    for tweet in list_tweet_text:
        lista_without_stopwords.append(' '.join([word for word in tweet.split() if word not in stop_words]))

    dataset['Text'] = lista_without_stopwords
dataset


# Λημματοποίηση είναι η διαδικασία ομαδοποίησης των κλιτών μορφών μιας λέξης,
# ώστε να μπορούν να αναλυθούν ως ένα ενιαίο στοιχείο που προσδιορίζεται
# από το λήμμα ή τη μορφή λεξικού της λέξης.
if ans_stopword_lemma == 'yes':
!pip install simplemma
```

```
    import simplemma
    list_tweet_text = list(dataset['Text'])
    lista_lemmatize = []

    for text in list_tweet_text:
        lista_lemmatize.append(' '.join([simplemma.lemmatize(word, lang='el') for word in text.split()]))

dataset.Text = lista_lemmatize
dataset
# Έλεγχος για χαμένες τιμές
dataset.isna().sum()


# Κατασκευάζουμε μια νέα στήλη Number_of_Words, που κρατάει τον αριθμό των λέξεων για κάθε κείμενο-tweet.
dataset['Number_of_Words'] = dataset['Text'].apply(lambda x: len(x.split()))
dataset.head(10)


# Δημιουργία γραφήματος Κατανομής λέξεων
if ans_dataset == 'politics':
plt.figure(figsize = (12,6))
sns.histplot(data=dataset, x='Number_of_Words', bins=range(1, 300, 5), alpha=0.9)
plt.title('Κατανομή λέξεων')
else:
plt.figure(figsize = (12,5))
sns.histplot(data=dataset, x='Number_of_Words', bins=range(1, 1300, 5), alpha=0.9)
plt.title('Κατανομή λέξεων')


# Δήλωση μεταβλητών
SEED_70_30 = 15
SEED_80_20 = 12
BATCH_SIZE = 32
VALIDATION_SPLIT = 0.2  #Αν θέλουμε τυχαίο σύνολο επικύρωσης


if ans_class == 'binary':
    EPOCHS = 8
else:
    if ans_un_bal == 'balanced':
        EPOCHS = 6
    else:
        EPOCHS = 8


if ans_dataset == 'politics':
    MAX_LENGTH = 80
else:
    MAX_LENGTH = 150


# Γράφημα [Positive - Negative] ή [Positive - Neutral - Negative]
plt.figure(figsize = (9,5))
if ans_un_bal == 'balanced':
plt.title('Balanced Tweets')
else:
plt.title('Unbalanced Tweets')
sns.countplot(x = dataset['Sentiment'], palette = 'Set2', alpha = 0.9)


# Κωδικοποίηση ετικετών σε αριθμητική μορφή.
if ans_class == 'binary':
```

```
dataset.Sentiment.replace("Positive", 1, inplace = True)
dataset.Sentiment.replace("Negative", 0, inplace = True)
    y = dataset['Sentiment']
    X = dataset['Text']
else:
    ohe = preprocessing.OneHotEncoder()
    y = ohe.fit_transform(np.array(dataset['Sentiment']).reshape(-1, 1)).toarray()
    X = dataset['Text']


# Μέθοδος δημιουργίας μοντέλου create_model()
def create_model(model, ans_model):

    if ans_model == 'bert':
        dropout_rate = 0.2
        input_ids = Input(shape = (MAX_LENGTH,), dtype = tf.int32, name = 'input_ids')
        attention_mask = Input(shape = (MAX_LENGTH,), dtype = tf.int32, name = 'attention_mask')
        inputs = model([input_ids, attention_mask])[1]
        print(inputs)
        out = Dropout(dropout_rate)(inputs)


        if ans_class == 'binary':
            # Χρήση πυκνού στρώματος 128 νευρώνων
            out = Dense(128, activation = 'relu')(out)
            out = Dropout(dropout_rate)(out)

            # Χρήση πυκνού στρώματος 64 νευρώνων
            out = Dense(64, activation = 'relu')(out)
            out = Dropout(dropout_rate)(out)

            # Χρήση πυκνού στρώματος 32 νευρώνων
            out = Dense(32, activation = 'relu')(out)
            out = Dropout(dropout_rate)(out)


        if ans_class == 'binary':
            # Απαιτείται ένας νευρώνας, εφόσον το tweet είναι θετικό ή αρνητικό
            # Χρήση συνάρτησης ενεργοποίησης sigmoid η οποία προτιμάται στην δυαδική ταξινόμηση
            out = Dense(1, activation = 'sigmoid')(out)
        else:
          # Απαιτούνται 3 νευρώνες, εφόσον έχουμε 3 κλάσεις.
          out = Dense(3, activation="softmax")(out)


        my_model = Model(inputs=[input_ids, attention_mask], outputs = out)

        # Χρήση optimizer Adam  (learning_rate = 0.00001)
        optimizer = Adam(learning_rate = 0.00001, epsilon = 1e-08)

# Μεταγλώττιση του μοντέλου
if ans_class == 'binary':
     my_model.compile(optimizer = optimizer, metrics = 'accuracy', loss = 'binary_crossentropy')
 else:
     my_model.compile(optimizer = optimizer, metrics = 'accuracy', loss = 'categorical_crossentropy')

        return my_model


    if ans_model == 'roberta':
        dropout_rate = 0.2
        input_ids = Input(shape = (MAX_LENGTH,), dtype = 'int32', name = 'input_ids')
        attention_mask = Input(shape = (MAX_LENGTH,), dtype = 'int32', name = 'attention_mask')
```

```
    inputs = model([input_ids, attention_mask])[1]
    print(inputs)
    out = Dropout(dropout_rate)(inputs)


    # Χρήση πυκνού στρώματος 128 νευρώνων
    out = Dense(128, activation = 'relu')(out)
    out = Dropout(dropout_rate)(out)

    # Χρήση πυκνού στρώματος 64 νευρώνων
    out = Dense(64, activation = 'relu')(out)
    out = Dropout(dropout_rate)(out)


    if ans_class == 'binary':
       # Απαιτείται ένας νευρώνας, εφόσον το tweet είναι θετικό ή αρνητικό
       # Χρήση συνάρτησης ενεργοποίησης sigmoid η οποία προτιμάται στην δυαδική ταξινόμηση
       out = Dense(1, activation = 'sigmoid')(out)
    else:
       # Απαιτούνται 3 νευρώνες
       out = Dense(3, activation="softmax")(out)


    my_model = Model(inputs=[input_ids, attention_mask], outputs = out)

    # Χρήση optimizer Adam
    optimizer = Adam(learning_rate = 1e-05, epsilon = 1e-08)

    # Μεταγλώττιση του μοντέλου
    if ans_class == 'binary':
      my_model.compile(optimizer = optimizer, metrics = 'accuracy', loss = 'binary_crossentropy')
    else:
      my_model.compile(optimizer = optimizer, metrics = 'accuracy', loss = 'categorical_crossentropy')

    return my_model


if ans_model == 'distilbert':
    dropout_rate = 0.2
    input_ids = Input(shape = (MAX_LENGTH,), dtype = 'int32', name = 'input_ids')
    attention_mask = Input(shape = (MAX_LENGTH,), dtype = 'int32', name = 'attention_mask')
    inputs = model([input_ids, attention_mask])[0]
    print(inputs)
    out = Dropout(dropout_rate)(inputs)

    if ans_class == 'binary':
       # Χρήση πυκνού στρώματος 256 νευρώνων
       out = Dense(256, activation = 'relu')(out)
       out = Dropout(dropout_rate)(out)

       # Χρήση πυκνού στρώματος 128 νευρώνων
       out = Dense(128, activation = 'relu')(out)
       out = Dropout(dropout_rate)(out)


       # Χρήση πυκνού στρώματος 64 νευρώνων
       out = Dense(64, activation = 'relu')(out)
       out = Dropout(dropout_rate)(out)

   if ans_class == 'multi':
      # Χρήση πυκνού στρώματος 8 νευρώνων
```

```
            out = Dense(8, activation = 'relu')(out)
            out = Dropout(dropout_rate)(out)
        out = Flatten()(out)

        if ans_class == 'binary':
            # Απαιτείται ένας νευρώνας, εφόσον το tweet είναι θετικό ή αρνητικό
            # Χρήση συνάρτησης ενεργοποίησης sigmoid η οποία προτιμάται στην δυαδική ταξινόμηση
            out = Dense(1, activation = 'sigmoid')(out)
        else:
            # Απαιτούνται 3 νευρώνες
            out = Dense(3, activation="softmax")(out)

        my_model = Model(inputs=[input_ids, attention_mask], outputs = out)

        # Χρήση optimizer Adam
        optimizer = Adam(learning_rate = 1e-05, epsilon = 1e-08, clipnorm = 1.0)


       # Μεταγλώττιση του μοντέλου
        if ans_class == 'binary':
            my_model.compile(optimizer = optimizer, metrics = 'accuracy', loss = 'binary_crossentropy')
        else:
            my_model.compile(optimizer = optimizer, metrics = 'accuracy', loss = 'categorical_crossentropy')

        return my_model


    if ans_model == 'gpt':

        #dropout_rate = 0.2
        input_ids = Input(shape = (MAX_LENGTH,), dtype=tf.int32, name='input_ids')
        outputs = model(input_ids)[0][:, -1, :]

        outputs = Flatten()(outputs)

        if ans_class == 'binary':
            # Απαιτείται ένας νευρώνας, εφόσον το tweet είναι θετικό ή αρνητικό
            # Χρήση συνάρτησης ενεργοποίησης sigmoid η οποία προτιμάται στην δυαδική ταξινόμηση
            outputs = Dense(1, activation = 'sigmoid')(outputs)
        else:
            # Απαιτούνται 3 νευρώνες
            outputs = Dense(3, activation="softmax")(outputs)

        classifier_model = Model(inputs=input_ids, outputs=outputs)
        optimizer = AdamW(learning_rate=1e-05, epsilon=1e-08, clipnorm=1.0)


        if ans_class == 'binary':
            classifier_model.compile(optimizer = optimizer, metrics = 'accuracy', loss = 'binary_crossentropy')
        else:
            classifier_model.compile(optimizer = optimizer, metrics = 'accuracy', loss = 'categorical_crossentropy')
        return classifier_model


# Δήλωση της tokenization μεθόδου για BERT, RoBERTa και DistilBERT.
def get_tokens(samples):
    my_dict = tokenizer(text = list(samples),
                add_special_tokens = True,
                max_length = MAX_LENGTH,
                truncation = True,
                padding = 'max_length',
                return_tensors = 'tf',
```

```
            return_token_type_ids = False,
            return_attention_mask = True,
            verbose = True)
    return my_dict


# Δήλωση της tokenization μεθόδου για GPT-2.
def get_tokens_GPT(samples):
    my_dict = tokenizer(text = list(samples),
            add_special_tokens = False,
            max_length = MAX_LENGTH,
            truncation = True,
            padding = 'max_length',
            return_tensors = 'tf',
            verbose = False)
    return my_dict


if ans_model == 'bert':
    model = TFAutoModel.from_pretrained("nlpaueb/bert-base-greek-uncased-v1")
    tokenizer = BertTokenizer.from_pretrained("nlpaueb/bert-base-greek-uncased-v1")
if ans_model == 'roberta':
    model = TFXLMRobertaModel.from_pretrained("jplu/tf-xlm-roberta-base")
    tokenizer = AutoTokenizer.from_pretrained("xlm-roberta-base")
if ans_model == 'distilbert':
    model = TFDistilBertModel.from_pretrained("distilbert-base-multilingual-cased", from_pt=True)
    tokenizer = DistilBertTokenizer.from_pretrained("distilbert-base-multilingual-cased")
if ans_model == 'gpt':
    #from transformers import TFAutoModelWithLMHead
    from transformers import TFAutoModelForCausalLM
    from transformers import GPT2Tokenizer
    model = TFAutoModelForCausalLM.from_pretrained("lighteternal/gpt2-finetuned-greek", from_pt=True)
    tokenizer = GPT2Tokenizer.from_pretrained("lighteternal/gpt2-finetuned-greek", from_pt=True)


my_model = create_model(model, ans_model)
my_model.summary()
plot_model(my_model, show_layer_names=True)


# Χρήση μεθόδου train_test_split
ans_test_size = float(input('Εισάγετε test size: [0.2] ή [0.3]'))
if ans_test_size == 0.2:
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = ans_test_size, random_state =
SEED_80_20, stratify = y)
else:
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = ans_test_size, random_state =
SEED_70_30, stratify = y)


# Για να κρατήσουμε το ίδιο validation set
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2, random_state=42)
X_train.shape
y_train.shape
X_test.shape
y_test.shape
X_train[:30]
X_test[:30]
X_val[:30]


# Επιστροφή των tokens των δειγμάτων εκπαίδευσης και ελέγχου, εκπαίδευση του μοντέλου με την μέθοδο
fit().
```

```
if ans_model == 'bert' or ans_model == 'roberta' or ans_model == 'distilbert':
    X_train_token = get_tokens(X_train)
    X_test_token = get_tokens(X_test)
    X_val_token = get_tokens(X_val)
print("Δείγματα Εκπαίδευσης - Το λεξικό για τα input_ids και attention_mask: ", X_train_token, "\n")

print("Εκκίνηση Εκπαίδευσης... Παρακαλώ περιμένετε!")
history_model = my_model.fit(x = {'input_ids':X_train_token['input_ids'],
'attention_mask':X_train_token['attention_mask']},
                    y = y_train,
                    epochs = EPOCHS,
                    validation_data = ({'input_ids':X_val_token['input_ids'],
'attention_mask':X_val_token['attention_mask']}, y_val),
                    batch_size = BATCH_SIZE,
                    callbacks = [EarlyStopping(monitor='val_accuracy', mode='max', patience=4,
restore_best_weights=True, verbose=True)])
else: #αν ειναι gpt-2
    X_train_token = get_tokens_GPT(X_train)
    X_test_token = get_tokens_GPT(X_test)
    X_val_token = get_tokens_GPT(X_val)
print("Δείγματα Εκπαίδευσης - Το λεξικό για τα input_ids: ", X_train_token, "\n")

print("Εκκίνηση Εκπαίδευσης... Παρακαλώ περιμένετε!")
history_model = my_model.fit(x = {'input_ids':X_train_token['input_ids']},
                    y = y_train,
                    epochs = EPOCHS,
                    validation_data = ({'input_ids':X_val_token['input_ids']}, y_val),
                    batch_size = BATCH_SIZE,
                    callbacks = [EarlyStopping(monitor='val_accuracy', mode='max', patience=4,
restore_best_weights=True, verbose=True)])


# Οπτικοποίηση ακρίβεια εκπαίδευσης (train accuracy) & ακρίβεια επικύρωσης (validation accuracy)
# Απώλεια εκπαίδευσης (train loss) & απώλεια επικύρωσης (validation loss)
def plot_diagramms(history_model, metric):

plt.plot(history_model.history[metric])
plt.plot(history_model.history['val_' + metric])
plt.xlabel("Epochs")
plt.ylabel(metric)
plt.legend([metric, 'val_' + metric])
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plot_diagramms(history_model, 'accuracy')
plt.subplot(1, 2, 2)
plot_diagramms(history_model, 'loss')


# Μέθοδος report για classification report
def report(real, prediction, ans_class):

    if ans_class == 'binary':
        tn = ['Negative', 'Positive']
    else:
        tn = ['Negative', 'Neutral', 'Positive']

print(classification_report(real, prediction, target_names=tn))
# Μέθοδος awx_confusion_matrix για προβολή χάρτη θερμότητας (HeatMap)
def awx_confusion_matrix(real, prediction, label, ans_class):


    fig, ax = plt.subplots(figsize=(6,6))
```

```
    if ans_class == 'binary':
        tn = ['Negative', 'Positive']
    else:
        tn = ['Negative', 'Neutral', 'Positive']

    labels = tn
    ax = sns.heatmap(confusion_matrix(real, prediction), annot=True, cmap="Reds", fmt='g', cbar=True,
annot_kws={"size":14})
plt.title(label, fontsize=15)
    ax.xaxis.set_ticklabels(labels, fontsize=15)
    ax.yaxis.set_ticklabels(labels, fontsize=15)
    ax.set_ylabel('Real', fontsize=14)
    ax.set_xlabel('Predicted', fontsize=14)
plt.show()


if ans_model == 'bert':
    label = 'Bert Model'
elif ans_model == 'roberta':
     label = 'RoBerta Model'
elif ans_model == 'distilbert':
     label = 'DistilBERT Model'
elif ans_model == 'gpt':
     label = 'GPT-2 Model'


# Χρήση μεθόδου predict()
if ans_model == 'bert' or ans_model == 'roberta' or ans_model == 'distilbert':
    if ans_class == 'binary':
        pred = np.where(my_model.predict([X_test_token['input_ids'], X_test_token['attention_mask']]) >= 0.5,
1, 0)
report(y_test, pred, ans_class)
        awx_confusion_matrix(y_test, pred, label, ans_class)
    else:
        pred = my_model.predict([X_test_token['input_ids'], X_test_token['attention_mask']])
        y_pred = np.zeros_like(pred)
        y_pred[np.arange(len(y_pred)), pred.argmax(1)] = 1


report(y_test, y_pred, ans_class)
        awx_confusion_matrix(y_test.argmax(1), y_pred.argmax(1), label, ans_class)
else:
    if ans_class == 'binary':
        pred = np.where(my_model.predict([X_test_token['input_ids']]) >= 0.5, 1, 0)
report(y_test, pred, ans_class)
        awx_confusion_matrix(y_test, pred, label, ans_class)
    else:
        pred = my_model.predict([X_test_token['input_ids']])
        y_pred = np.zeros_like(pred)
        y_pred[np.arange(len(y_pred)), pred.argmax(1)] = 1

report(y_test, y_pred, ans_class)
        awx_confusion_matrix(y_test.argmax(1), y_pred.argmax(1), label, ans_class)
# END #
```

Εκπαίδευση μοντέλων Transformers (bert-gpt-roberta-distilbert-train-only.ipynb)

Στο αρχείο αυτό, γίνεται **μόνο εκπαίδευση** μοντέλων Transformers σε **συγκεκριμένους διαχωρισμούς**. Αποθηκεύουμε τα βάρη, προκειμένου στην εφαρμογή Tkinter να γίνουν μόνο προβλέψεις.

Για Politics: train_set_politics2682.csv, validation_data873.csv

Για Skroutz: train_set_skroutz3210.csv, validation_data1376.csv

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
importos
# TF_CPP_MIN_LOG_LEVEL=2 ώστε να φιλτράρει πληροφορίες και προειδοποιητικά μηνύματα.
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
import keras
from keras.layers import LSTM, Dense, Input, Dropout, Embedding, Flatten
from keras.models import Model, load_model
from keras.callbacks import EarlyStopping
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.utils import plot_model
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.python.client import device_lib
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.optimizers.experimental import AdamW
!pip3 install transformers
import torch
import transformers
from transformers import TFBertModel, AutoTokenizer, TFAutoModel, BertTokenizer
from transformers import AutoModelForMaskedLM, TFXLMRobertaModel, DistilBertTokenizer,
TFDistilBertModel
from transformers import TFGPT2LMHeadModel, GPT2Model, AutoModel


# Για αφαίρεση κοινών λέξεων (stopwords)
import nltk
from nltk.corpus import stopwords
stop_words = stopwords.words('greek')
ans_model = input('Ποιό μοντέλο θέλετε να εκπαιδεύσετε? [bert] [roberta] [distilbert] [gpt]')
# Επιλογή Dataset
ans_dataset = input('Επιλέξτε dataset [politics] ή [skroutz]')
headers=['Text','Sentiment']


if ans_dataset == 'politics':
    dataset = pd.read_csv('/kaggle/input/train-set-politics2682/train_set_politics2682.csv', sep=',',
names=headers)
    val_data = pd.read_csv('/kaggle/input/validation-data873/validation_data873.csv', sep=',', names=headers)
else:
    dataset = pd.read_csv('/kaggle/input/train-set-skroutz3210/train_set_skroutz3210.csv', sep=',',
names=headers)
    val_data = pd.read_csv('/kaggle/input/validation-data1376/validation_data1376.csv', sep=',',
names=headers)
dataset.head()
dataset.info()
```

```
dataset['Sentiment'].value_counts()


if ans_dataset == 'politics':
    ans_class = input('Επιλέξτε αν θέλετε [binary] η [multi] class')
else:
    ans_class = 'binary'


# Το dataset περιέχει στη στήλη Sentiment τις ετικέτες neutral, negative και positive.
# Θα χρησιμοποιήσουμε το μοντέλο BERT για να κάνουμε δυαδική ταξινόμηση συναισθήματος,
# οπότε θα κρατήσουμε μόνο 2 ετικέτες, Positive και Negative.
# Tweets που είναι neutral, θα φύγουν με την μέθοδο dropna()


def delete_neutral(dataset):
    list_tweet_text = list(dataset['Text'])
    list_sentiment = list(dataset['Sentiment'])
    nea_lista_tweet_text = []
    nea_lista_sentiment = []

    for sent, tweet in zip(list_sentiment, list_tweet_text):
        if sent == 'Neutral':
            nea_lista_tweet_text.append(None)
            nea_lista_sentiment.append(None)
        else:
            nea_lista_tweet_text.append(tweet)
            nea_lista_sentiment.append(sent)

    dataset['Text'] = nea_lista_tweet_text
    dataset['Sentiment'] = nea_lista_sentiment
    dataset = dataset.dropna()
    return dataset


if ans_class == 'binary' and ans_dataset == 'politics':
    dataset = delete_neutral(dataset)
    val_data = delete_neutral(val_data)


# Μετά την περικοπή των Neutral στο dataset. (μόνο politics ->binary γινεται περικοπή)
dataset.info()
ans_stopword_lemma = input('Επιλέξτε αν θέλετε να αφαιρέσετε λέξεις χωρία αξία (stopwords) και αν θέλετε
να εφαρμοστεί λημματοποίηση [yes] ή [no]: ')


# Καθαρισμός Stop Words στην στήλη Text.
if ans_stopword_lemma == 'yes':
    list_tweet_text = list(dataset['Text'])
    lista_without_stopwords = []

    for tweet in list_tweet_text:
        lista_without_stopwords.append(' '.join([word for word in tweet.split() if word not in stop_words]))

    dataset['Text'] = lista_without_stopwords
dataset


# Λημματοποίηση είναι η διαδικασία ομαδοποίησης των κλιτών μορφών μιας λέξης,
# ώστε να μπορούν να αναλυθούν ως ένα ενιαίο στοιχείο, που προσδιορίζεται
# από το λήμμα ή τη μορφή λεξικού της λέξης.
if ans_stopword_lemma == 'yes':
!pip install simplemma
```

```
    import simplemma

    list_tweet_text = list(dataset['Text'])
    lista_lemmatize = []

    for text in list_tweet_text:
        lista_lemmatize.append(' '.join([simplemma.lemmatize(word, lang='el') for word in text.split()]))

dataset.Text = lista_lemmatize
dataset


# Έλεγχος για χαμένες τιμές
dataset.isna().sum()
# Δήλωση μεταβλητών
SEED = 15  # Δεν ειναι απαραίτητο αφόσον δεν χρησιμοποιειται η train_test_split
BATCH_SIZE = 32

if ans_dataset == 'politics':
    MAX_LENGTH = 80
else:
    MAX_LENGTH = 150


if ans_class == 'binary':
    EPOCHS = 8
else:
    if ans_model == 'gpt':
EPOCHS = 10
    else:
        EPOCHS = 8


# MODEL_NAME για BERT
if ans_model == 'bert':
    if ans_dataset == 'politics' and ans_class == 'binary' and ans_stopword_lemma == 'yes':
        MODEL_NAME = 'train_bert_bin_politics_with_sl.h5'
    if ans_dataset == 'politics' and ans_class == 'binary' and ans_stopword_lemma == 'no':
        MODEL_NAME = 'train_bert_bin_politics_without_sl.h5'
    if ans_dataset == 'politics' and ans_class == 'multi' and ans_stopword_lemma == 'yes':
        MODEL_NAME = 'train_bert_multi_politics_with_sl.h5'
    if ans_dataset == 'politics' and ans_class == 'multi' and ans_stopword_lemma == 'no':
        MODEL_NAME = 'train_bert_multi_politics_without_sl.h5'
    if ans_dataset == 'skroutz' and ans_class == 'binary' and ans_stopword_lemma == 'yes':
        MODEL_NAME = 'train_bert_bin_skroutz_with_sl.h5'
    if ans_dataset == 'skroutz' and ans_class == 'binary' and ans_stopword_lemma == 'no':
        MODEL_NAME = 'train_bert_bin_skroutz_without_sl.h5'


# MODEL_NAME για RoBERTa
if ans_model == 'roberta':
    if ans_dataset == 'politics' and ans_class == 'binary' and ans_stopword_lemma == 'yes':
        MODEL_NAME = 'train_roberta_bin_politics_with_sl.h5'
    if ans_dataset == 'politics' and ans_class == 'binary' and ans_stopword_lemma == 'no':
        MODEL_NAME = 'train_roberta_bin_politics_without_sl.h5'
    if ans_dataset == 'politics' and ans_class == 'multi' and ans_stopword_lemma == 'yes':
        MODEL_NAME = 'train_roberta_multi_politics_with_sl.h5'
    if ans_dataset == 'politics' and ans_class == 'multi' and ans_stopword_lemma == 'no':
        MODEL_NAME = 'train_roberta_multi_politics_without_sl.h5'
    if ans_dataset == 'skroutz' and ans_class == 'binary' and ans_stopword_lemma == 'yes':
        MODEL_NAME = 'train_roberta_bin_skroutz_with_sl.h5'
    if ans_dataset == 'skroutz' and ans_class == 'binary' and ans_stopword_lemma == 'no':
```

```
        MODEL_NAME = 'train_roberta_bin_skroutz_without_sl.h5'


# MODEL_NAME για DistilBERT
if ans_model == 'distilbert':
    if ans_dataset == 'politics' and ans_class == 'binary' and ans_stopword_lemma == 'yes':
        MODEL_NAME = 'train_distilbert_bin_politics_with_sl.h5'
    if ans_dataset == 'politics' and ans_class == 'binary' and ans_stopword_lemma == 'no':
        MODEL_NAME = 'train_distilbert_bin_politics_without_sl.h5'
    if ans_dataset == 'politics' and ans_class == 'multi' and ans_stopword_lemma == 'yes':
        MODEL_NAME = 'train_distilbert_multi_politics_with_sl.h5'
    if ans_dataset == 'politics' and ans_class == 'multi' and ans_stopword_lemma == 'no':
        MODEL_NAME = 'train_distilbert_multi_politics_without_sl.h5'
    if ans_dataset == 'skroutz' and ans_class == 'binary' and ans_stopword_lemma == 'yes':
        MODEL_NAME = 'train_distilbert_bin_skroutz_with_sl.h5'
    if ans_dataset == 'skroutz' and ans_class == 'binary' and ans_stopword_lemma == 'no':
        MODEL_NAME = 'train_distilbert_bin_skroutz_without_sl.h5'


# MODEL_NAME για GPT
if ans_model == 'gpt':
    if ans_dataset == 'politics' and ans_class == 'binary' and ans_stopword_lemma == 'yes':
        MODEL_NAME = 'train_gpt_bin_politics_with_sl.h5'
    if ans_dataset == 'politics' and ans_class == 'binary' and ans_stopword_lemma == 'no':
        MODEL_NAME = 'train_gpt_bin_politics_without_sl.h5'
    if ans_dataset == 'politics' and ans_class == 'multi' and ans_stopword_lemma == 'yes':
        MODEL_NAME = 'train_gpt_multi_politics_with_sl.h5'
    if ans_dataset == 'politics' and ans_class == 'multi' and ans_stopword_lemma == 'no':
        MODEL_NAME = 'train_gpt_multi_politics_without_sl.h5'
    if ans_dataset == 'skroutz' and ans_class == 'binary' and ans_stopword_lemma == 'yes':
        MODEL_NAME = 'train_gpt_bin_skroutz_with_sl.h5'
    if ans_dataset == 'skroutz' and ans_class == 'binary' and ans_stopword_lemma == 'no':
        MODEL_NAME = 'train_gpt_bin_skroutz_without_sl.h5'


dataset['Sentiment'].value_counts()
# Γράφημα Positive - Negative
plt.figure(figsize = (10,6))
plt.title('Balanced Tweets')
sns.countplot(x = dataset['Sentiment'], palette = 'Set2', alpha = 0.9)


# Κωδικοποίηση ετικετών σε αριθμητική μορφή.
if ans_class == 'binary':
dataset.Sentiment.replace("Positive", 1, inplace = True)
dataset.Sentiment.replace("Negative", 0, inplace = True)
    val_data.Sentiment.replace("Positive", 1, inplace = True)
    val_data.Sentiment.replace("Negative", 0, inplace = True)
    y_train = dataset['Sentiment']
    y_val = val_data['Sentiment']
    X_train = dataset['Text']
    X_val = val_data['Text']
else:
    ohe = preprocessing.OneHotEncoder()
    y_train = ohe.fit_transform(np.array(dataset['Sentiment']).reshape(-1, 1)).toarray()
    y_val = ohe.fit_transform(np.array(val_data['Sentiment']).reshape(-1, 1)).toarray()
    X_train = dataset['Text']
    X_val = val_data['Text']


# Μέθοδος δημιουργίας μοντέλου create_model()
def create_model(model, ans_model):
```

```
if ans_model == 'bert':
    dropout_rate = 0.2
    input_ids = Input(shape = (MAX_LENGTH,), dtype = tf.int32, name = 'input_ids')
    attention_mask = Input(shape = (MAX_LENGTH,), dtype = tf.int32, name = 'attention_mask')

    inputs = model([input_ids, attention_mask])[1]
    print(inputs)
    out = Dropout(dropout_rate)(inputs)

    # Χρήση πυκνού στρώματος 64 νευρώνων
    out = Dense(64, activation = 'relu')(out)
    out = Dropout(dropout_rate)(out)


    if ans_class == 'binary':
        # Απαιτείται ένας νευρώνας, εφόσον το tweet είναι θετικό ή αρνητικό
        # Χρήση συνάρτησης ενεργοποίησης sigmoid η οποία προτιμάται στην δυαδική ταξινόμηση
        out = Dense(1, activation = 'sigmoid')(out)
    else:
      # Απαιτούνται 3 νευρώνες
      out = Dense(3, activation="softmax")(out)


    my_model = Model(inputs=[input_ids, attention_mask], outputs = out)

    # Χρήση optimizer Adam  (learning_rate = 0.00001)
    optimizer = Adam(learning_rate = 0.00001, epsilon = 1e-08)

    # Μεταγλώττιση του μοντέλου
    if ans_class == 'binary':
        my_model.compile(optimizer = optimizer, metrics = 'accuracy', loss = 'binary_crossentropy')
    else:
        my_model.compile(optimizer = optimizer, metrics = 'accuracy', loss = 'categorical_crossentropy')

    return my_model


if ans_model == 'roberta':
    dropout_rate = 0.2
    input_ids = Input(shape = (MAX_LENGTH,), dtype = 'int32', name = 'input_ids')
    attention_mask = Input(shape = (MAX_LENGTH,), dtype = 'int32', name = 'attention_mask')

    inputs = model([input_ids, attention_mask])[1]
    print(inputs)
    out = Dropout(dropout_rate)(inputs)

    # Χρήση πυκνού στρώματος 128 νευρώνων
    out = Dense(128, activation = 'relu')(out)
    out = Dropout(dropout_rate)(out)


    # Χρήση πυκνού στρώματος 64 νευρώνων
    out = Dense(64, activation = 'relu')(out)
    out = Dropout(dropout_rate)(out)

    if ans_class == 'binary':
        # Απαιτείται ένας νευρώνας, εφόσον το tweet είναι θετικό ή αρνητικό
        # Χρήση συνάρτησης ενεργοποίησης sigmoid η οποία προτιμάται στην δυαδική ταξινόμηση
        out = Dense(1, activation = 'sigmoid')(out)
    else:
        # Απαιτούνται 3 νευρώνες
        out = Dense(3, activation="softmax")(out)
```

```
    my_model = Model(inputs=[input_ids, attention_mask], outputs = out)

    # Χρήση optimizer Adam
    optimizer = Adam(learning_rate = 1e-05, epsilon = 1e-08)
    # Μεταγλώττιση του μοντέλου
    if ans_class == 'binary':
        my_model.compile(optimizer = optimizer, metrics = 'accuracy', loss = 'binary_crossentropy')
    else:
        my_model.compile(optimizer = optimizer, metrics = 'accuracy', loss = 'categorical_crossentropy')

    return my_model


if ans_model == 'distilbert':
    dropout_rate = 0.2
    input_ids = Input(shape = (MAX_LENGTH,), dtype = 'int32', name = 'input_ids')
    attention_mask = Input(shape = (MAX_LENGTH,), dtype = 'int32', name = 'attention_mask')
    inputs = model([input_ids, attention_mask])[0]
    print(inputs)
    out = Dropout(dropout_rate)(inputs)

    # Χρήση πυκνού στρώματος 256 νευρώνων
    out = Dense(256, activation = 'relu')(out)
    out = Dropout(dropout_rate)(out)

    # Χρήση πυκνού στρώματος 128 νευρώνων
    out = Dense(128, activation = 'relu')(out)
    out = Dropout(dropout_rate)(out)
    # Χρήση πυκνού στρώματος 64 νευρώνων
    out = Dense(64, activation = 'relu')(out)
    out = Dropout(dropout_rate)(out)
    out = Flatten()(out)

    if ans_class == 'binary':
        # Απαιτείται ένας νευρώνας, εφόσον το tweet είναι θετικό ή αρνητικό
        # Χρήση συνάρτησης ενεργοποίησης sigmoid η οποία προτιμάται στην δυαδική ταξινόμηση
        out = Dense(1, activation = 'sigmoid')(out)
    else:
        # Απαιτούνται 3 νευρώνες
        out = Dense(3, activation="softmax")(out)


    my_model = Model(inputs=[input_ids, attention_mask], outputs = out)
    # Χρήση optimizer Adam
    optimizer = Adam(learning_rate = 1e-05, epsilon = 1e-08, clipnorm = 1.0)

    # Μεταγλώττιση του μοντέλου
    if ans_class == 'binary':
        my_model.compile(optimizer = optimizer, metrics = 'accuracy', loss = 'binary_crossentropy')
    else:
        my_model.compile(optimizer = optimizer, metrics = 'accuracy', loss = 'categorical_crossentropy')


    return my_model


# Για GPT Model
if ans_model == 'gpt':

    dropout_rate = 0.2
    input_ids = Input(shape = (MAX_LENGTH,), dtype=tf.int32, name='input_ids')
    outputs = model(input_ids)[0][:, -1, :]
```

```
        outputs = Flatten()(outputs)

        if ans_class == 'binary':
            # Απαιτείται ένας νευρώνας, εφόσον το tweet είναι θετικό ή αρνητικό
            # Χρήση συνάρτησης ενεργοποίησης sigmoid η οποία προτιμάται στην δυαδική ταξινόμηση
            outputs = Dense(1, activation = 'sigmoid')(outputs)
        else:
            # Απαιτούνται 3 νευρώνες
            outputs = Dense(3, activation="softmax")(outputs)


        classifier_model = Model(inputs=input_ids, outputs=outputs)
        optimizer = Adam(learning_rate=1e-05, epsilon=1e-08, clipnorm=1.0)

        if ans_class == 'binary':
            classifier_model.compile(optimizer = optimizer, metrics = 'accuracy', loss = 'binary_crossentropy')
        else:
            classifier_model.compile(optimizer = optimizer, metrics = 'accuracy', loss = 'categorical_crossentropy')

        return classifier_model


# Δήλωση της tokenization μεθόδου
def get_tokens(samples):
    my_dict = tokenizer(text = list(samples),
                add_special_tokens = True,
                max_length = MAX_LENGTH,
                truncation = True,
                padding = 'max_length',
                return_tensors = 'tf',
                return_token_type_ids = False,
                return_attention_mask = True,
                verbose = True)
    return my_dict


# Δήλωση της tokenization μεθόδου για GPT-2.
def get_tokens_GPT(samples):
    my_dict = tokenizer(text = list(samples),
                add_special_tokens = False,
                max_length = MAX_LENGTH,
                truncation = True,
                padding = 'max_length',
                return_tensors = 'tf',
                verbose = False)
    return my_dict


if ans_model == 'bert':
    model = TFAutoModel.from_pretrained("nlpaueb/bert-base-greek-uncased-v1")
    tokenizer = BertTokenizer.from_pretrained("nlpaueb/bert-base-greek-uncased-v1")
if ans_model == 'roberta':
    model = TFXLMRobertaModel.from_pretrained("jplu/tf-xlm-roberta-base")
    tokenizer = AutoTokenizer.from_pretrained("xlm-roberta-base")
if ans_model == 'distilbert':
    model = TFDistilBertModel.from_pretrained("distilbert-base-multilingual-cased", from_pt=True)
    tokenizer = DistilBertTokenizer.from_pretrained("distilbert-base-multilingual-cased")
if ans_model == 'gpt':
    from transformers import TFAutoModelForCausalLM
    from transformers import GPT2Tokenizer
    model = TFAutoModelForCausalLM.from_pretrained("lighteternal/gpt2-finetuned-greek", from_pt=True)
    tokenizer = GPT2Tokenizer.from_pretrained("lighteternal/gpt2-finetuned-greek", from_pt=True)
```

```
model = create_model(model, ans_model)
model.summary()
plot_model(model, show_layer_names=True)
X_train.shape
y_train.shape
X_train[:10]
X_val[:10]


# Επιστροφή των tokens των δειγμάτων εκπαίδευσης και ελέγχου, εκπαίδευση του μοντέλου με την μέθοδο fit().
if ans_model == 'bert' or ans_model == 'roberta' or ans_model == 'distilbert':
    X_train_token = get_tokens(X_train)
    X_val_token = get_tokens(X_val)
print("Δείγματα Εκπαίδευσης - Το λεξικό για τα input_ids και attention_mask: ", X_train_token, "\n")
print("Εκκίνηση Εκπαίδευσης... Παρακαλώ περιμένετε!")
history_model = model.fit(x = {'input_ids':X_train_token['input_ids'],
'attention_mask':X_train_token['attention_mask']},
                    y = y_train,
                    epochs = EPOCHS,
                    validation_data = ({'input_ids':X_val_token['input_ids'],
'attention_mask':X_val_token['attention_mask']}, y_val),
                    batch_size = BATCH_SIZE,
                    callbacks = [EarlyStopping(monitor='val_accuracy', mode='max', patience=4,
restore_best_weights=True, verbose=True)])
else: #αν ειναι gpt-2
    X_train_token = get_tokens_GPT(X_train)
    X_val_token = get_tokens_GPT(X_val)
print("Δείγματα Εκπαίδευσης - Το λεξικό για τα input_ids: ", X_train_token, "\n")

print("Εκκίνηση Εκπαίδευσης... Παρακαλώ περιμένετε!")
history_model = model.fit(x = {'input_ids':X_train_token['input_ids']},
                    y = y_train,
                    epochs = EPOCHS,
                    validation_data = ({'input_ids':X_val_token['input_ids']}, y_val),
                    batch_size = BATCH_SIZE,
                    callbacks = [EarlyStopping(monitor='val_accuracy', mode='max', patience=4,
restore_best_weights=True, verbose=True)])


# Οπτικοποίηση ακρίβεια εκπαίδευσης (train accuracy) & ακρίβεια επικύρωσης (validation accuracy)
# Απώλεια εκπαίδευσης (train loss) & απώλεια επικύρωσης (validation loss)
def plot_diagramms(history_model, metric):
plt.plot(history_model.history[metric])
plt.plot(history_model.history['val_' + metric])
plt.xlabel("Epochs")
plt.ylabel(metric)
plt.legend([metric, 'val_' + metric])
plt.figure(figsize=(11, 5))
plt.subplot(1, 2, 1)
plot_diagramms(history_model, 'accuracy')
plt.subplot(1, 2, 2)
plot_diagramms(history_model, 'loss')
model.save(MODEL_NAME)
# Η display συνάρτηση εμφανίζει το αντικείμενο της σύνδεσης προς το αρχειο .h5
from IPython.display import FileLink
display(FileLink(MODEL_NAME))
# END #
```

Εφαρμογή Tkinter (main.py, utilities.py, utilities_transformers.py)

Το μόνο που απομένει όσον αφορά τα Transformers, είναι να γίνουν οι προβλέψεις (***αρχείο: utilities_transformers.py***) μέσω της εφαρμογής Tkinter. Όσον αφορά τα κλασσικά μοντέλα μηχανικής μάθησης, γίνεται εκπαίδευση και πρόβλεψη με βάση είτε τους τυχαίους διαχωρισμούς είτε τους συγκεκριμένους (***αρχείο: utilities.py***).

αρχείο utilities_transformers.py

```
# utilities_transformers.py
# βοηθητικέςμέθοδοι
import tensorflow as tf
import numpy as np
import pandas as pd
import transformers
from transformers import BertTokenizer, TFBertModel, AutoTokenizer, TFXLMRobertaModel,
TFAutoModel, DistilBertTokenizer, TFDistilBertModel
from transformers import TFAutoModelForCausalLM, GPT2Tokenizer
#from tensorflow.keras.models import load_model, Model
from sklearn import preprocessing
import torch
from utilities import report, delete_neutral, stopwords_lemmatize, awx_confusion_matrix
# Για αφαίρεση λέξεων χωρίς αξία (stopwords)
from nltk.corpus import stopwords
stop_words = stopwords.words('greek') #nltk stop words


# Μέθοδος επιλογής μοντέλου
def select_model_transformers(ans_dataset, ans_class, ans_stopword_lemma, ans_models):

    if ans_dataset == 'politics':
        MAX_LENGTH = 80
    else:
        MAX_LENGTH = 150


# BERT MODEL
    if ans_models == 'bert':
        label = 'Bert Model'
        headers=['Text','Sentiment']

        if ans_dataset == 'politics':
            dataset = pd.read_csv('Politics/train2682_test810/test_set_politics810.csv', sep=',', names=headers)
        else:
            dataset = pd.read_csv('Skroutz/test_set_skroutz1966.csv', sep=',', names=headers)

        if ans_dataset == 'politics' and ans_class == 'binary':
            dataset = delete_neutral(dataset)


        if ans_stopword_lemma == 'yes':
            dataset = stopwords_lemmatize(dataset)

        if ans_class == 'binary':
            dataset.Sentiment.replace("Positive", 1, inplace = True)
            dataset.Sentiment.replace("Negative", 0, inplace = True)
            y_test = dataset['Sentiment']
            X_test = dataset['Text']
        else:
            ohe = preprocessing.OneHotEncoder()
            y_test = ohe.fit_transform(np.array(dataset['Sentiment']).reshape(-1, 1)).toarray()
```

```
        X_test = dataset['Text']


    tokenizer = BertTokenizer.from_pretrained("nlpaueb/bert-base-greek-uncased-v1")
    X_test_token = get_tokens(X_test, tokenizer, MAX_LENGTH)


# Ορίζουμε το μοντέλο BERT ως υποκλάση της Model
class MyBertModel(tf.keras.Model):
        def __init__(self):
super(MyBertModel, self).__init__()
self.bert = TFAutoModel.from_pretrained('nlpaueb/bert-base-greek-uncased-v1')
self.dropout1 = tf.keras.layers.Dropout(0.2)
self.dense1 = tf.keras.layers.Dense(64, activation='relu')
self.dropout2 = tf.keras.layers.Dropout(0.2)
            if ans_class == 'binary':
self.dense2 = tf.keras.layers.Dense(1, activation='sigmoid')
            else:
self.dense2 = tf.keras.layers.Dense(3, activation='softmax')


        def call(self, inputs, **kwargs):
            input_ids = inputs[0]
            attention_mask = inputs[1]
            outputs = self.bert(input_ids, attention_mask, **kwargs)[1]
            outputs = self.dropout1(outputs)
            outputs = self.dense1(outputs)
            outputs = self.dropout2(outputs)
            outputs = self.dense2(outputs)
            return outputs

    model = MyBertModel()
    input_ids = tf.keras.layers.Input(shape=(MAX_LENGTH,), dtype=tf.int32, name="input_ids")
    attention_mask = tf.keras.layers.Input(shape=(MAX_LENGTH,), dtype=tf.int32,
name="attention_mask")
    outputs = model([input_ids, attention_mask])


# Φόρτωση των βαρών του προ-εκπαιδευμένου μοντέλου
if ans_dataset == 'politics':
        if ans_class == 'binary' and ans_stopword_lemma == 'yes':
model.load_weights('E:\\h5\\Politics\\train2682_test810\\train_bert_bin_politics_with_sl.h5')
        elif ans_class == 'binary' and ans_stopword_lemma == 'no':
model.load_weights('E:\\h5\\Politics\\train2682_test810\\train_bert_bin_politics_without_sl.h5')
        elif ans_class == 'multi' and ans_stopword_lemma == 'yes':
model.load_weights('E:\\h5\\Politics\\train2682_test810\\train_bert_multi_politics_with_sl.h5')
        elif ans_class == 'multi' and ans_stopword_lemma == 'no':
model.load_weights('E:\\h5\\Politics\\train2682_test810\\train_bert_multi_politics_without_sl.h5')

    if ans_dataset == 'skroutz':
        if ans_stopword_lemma == 'yes':
            model.load_weights('E:\\h5\\Skroutz\\train_bert_bin_skroutz_with_sl.h5')
        else:
            model.load_weights('E:\\h5\\Skroutz\\train_bert_bin_skroutz_without_sl.h5')
    # Χρήση μεθόδου predict()
    if ans_class == 'binary':
        pred = np.where(model.predict([X_test_token['input_ids'], X_test_token['attention_mask']]) >= 0.5, 1,
0)
        m,r = report(y_test, pred, ans_class)
        awx_confusion_matrix(y_test, pred, label, ans_class)
    else:
        pred = model.predict([X_test_token['input_ids'], X_test_token['attention_mask']])
```

```
            y_pred_bert = np.zeros_like(pred)
            y_pred_bert[np.arange(len(y_pred_bert)), pred.argmax(1)] = 1
            m,r = report(y_test, y_pred_bert, ans_class, transformers=True)
            awx_confusion_matrix(y_test.argmax(1), y_pred_bert.argmax(1), label, ans_class)


        # RoBERTa MODEL
        if ans_models == 'roberta':
            label = 'RoBERTa Model'
            headers=['Text','Sentiment']


            if ans_dataset == 'politics':
                dataset = pd.read_csv('Politics/train2682_test810/test_set_politics810.csv', sep=',', names=headers)
            else:
                dataset = pd.read_csv('Skroutz/test_set_skroutz1966.csv', sep=',', names=headers)

            if ans_dataset == 'politics' and ans_class == 'binary':
                dataset = delete_neutral(dataset)

            if ans_stopword_lemma == 'yes':
                dataset = stopwords_lemmatize(dataset)


            if ans_class == 'binary':
                dataset.Sentiment.replace("Positive", 1, inplace = True)
                dataset.Sentiment.replace("Negative", 0, inplace = True)
                y_test = dataset['Sentiment']
                X_test = dataset['Text']
            else:
                ohe = preprocessing.OneHotEncoder()
                y_test = ohe.fit_transform(np.array(dataset['Sentiment']).reshape(-1, 1)).toarray()
                X_test = dataset['Text']

            tokenizer = AutoTokenizer.from_pretrained("xlm-roberta-base")
            X_test_token = get_tokens(X_test, tokenizer, MAX_LENGTH)

# Ορίζουμε το μοντέλο RoBERTa ως υποκλάση της Model
class MyRoBERTaModel(tf.keras.Model):
            def __init__(self):
super(MyRoBERTaModel, self).__init__()
self.roberta = TFXLMRobertaModel.from_pretrained('jplu/tf-xlm-roberta-base')
self.dropout1 = tf.keras.layers.Dropout(0.2)
self.dense1 = tf.keras.layers.Dense(128, activation='relu')
self.dropout2 = tf.keras.layers.Dropout(0.2)
self.dense2 = tf.keras.layers.Dense(64, activation='relu')
self.dropout3 = tf.keras.layers.Dropout(0.2)


                if ans_class == 'binary':
                    self.dense3 = tf.keras.layers.Dense(1, activation='sigmoid')
                else:
                    self.dense3 = tf.keras.layers.Dense(3, activation='softmax')


            def call(self, inputs, **kwargs):
                input_ids = inputs[0]
                attention_mask = inputs[1]
                outputs = self.roberta(input_ids, attention_mask, **kwargs)[1]
                outputs = self.dropout1(outputs)
                outputs = self.dense1(outputs)
                outputs = self.dropout2(outputs)
```

```
            outputs = self.dense2(outputs)
            outputs = self.dropout3(outputs)
            outputs = self.dense3(outputs)
            return outputs


    model = MyRoBERTaModel()
    input_ids = tf.keras.layers.Input(shape=(MAX_LENGTH,), dtype=tf.int32, name="input_ids")
    attention_mask = tf.keras.layers.Input(shape=(MAX_LENGTH,), dtype=tf.int32,
name="attention_mask")
    outputs = model([input_ids, attention_mask])


# Φόρτωση των βαρών του προ-εκπαιδευμένου μοντέλου
if ans_dataset == 'politics':
        if ans_class == 'binary' and ans_stopword_lemma == 'yes':
          model.load_weights('E:\\h5\\Politics\\train2682_test810\\train_roberta_bin_politics_with_sl.h5')
        elif ans_class == 'binary' and ans_stopword_lemma == 'no':
          model.load_weights('E:\\h5\\Politics\\train2682_test810\\train_roberta_bin_politics_without_sl.h5')
        elif ans_class == 'multi' and ans_stopword_lemma == 'yes':
          model.load_weights('E:\\h5\\Politics\\train2682_test810\\train_roberta_multi_politics_with_sl.h5')
        elif ans_class == 'multi' and ans_stopword_lemma == 'no':
          model.load_weights('E:\\h5\\Politics\\train2682_test810\\train_roberta_multi_politics_without_sl.h5')


    if ans_dataset == 'skroutz':
        if ans_stopword_lemma == 'yes':
           model.load_weights('E:\\h5\\Skroutz\\train_roberta_bin_skroutz_with_sl.h5')
        else:
           model.load_weights('E:\\h5\\Skroutz\\train_roberta_bin_skroutz_without_sl.h5')


    # Χρήση μεθόδου predict()
    if ans_class == 'binary':
        pred = np.where(model.predict([X_test_token['input_ids'], X_test_token['attention_mask']]) >= 0.5, 1,
0)
        m,r = report(y_test, pred, ans_class)
        awx_confusion_matrix(y_test, pred, label, ans_class)
    else:
        pred = model.predict([X_test_token['input_ids'], X_test_token['attention_mask']])
        y_pred_roberta = np.zeros_like(pred)
        y_pred_roberta[np.arange(len(y_pred_roberta)), pred.argmax(1)] = 1

        m,r = report(y_test, y_pred_roberta, ans_class, True)
        awx_confusion_matrix(y_test.argmax(1), y_pred_roberta.argmax(1), label, ans_class)


  # DistilBERT MODEL
  if ans_models == 'distilbert':
    label = 'DistilBERT Model'
    headers=['Text','Sentiment']

    if ans_dataset == 'politics':
      dataset = pd.read_csv('Politics/train2682_test810/test_set_politics810.csv', sep=',', names=headers)
    else:
      dataset = pd.read_csv('Skroutz/test_set_skroutz1966.csv', sep=',', names=headers)

    if ans_dataset == 'politics' and ans_class == 'binary':
      dataset = delete_neutral(dataset)

    if ans_stopword_lemma == 'yes':
      dataset = stopwords_lemmatize(dataset)
```

```
        if ans_class == 'binary':
            dataset.Sentiment.replace("Positive", 1, inplace = True)
            dataset.Sentiment.replace("Negative", 0, inplace = True)
            y_test = dataset['Sentiment']
            X_test = dataset['Text']
        else:
            ohe = preprocessing.OneHotEncoder()
            y_test = ohe.fit_transform(np.array(dataset['Sentiment']).reshape(-1, 1)).toarray()
            X_test = dataset['Text']

        tokenizer = DistilBertTokenizer.from_pretrained("distilbert-base-multilingual-cased")
        X_test_token = get_tokens(X_test, tokenizer, MAX_LENGTH)


# Ορίζουμε το μοντέλο DistilBERT ως υποκλάση της Model
class MyDistilBERTModel(tf.keras.Model):
            def __init__(self):
super(MyDistilBERTModel, self).__init__()
self.distilbert = TFDistilBertModel.from_pretrained('distilbert-base-multilingual-cased', from_pt=True)
self.dropout1 = tf.keras.layers.Dropout(0.2)
self.dense1 = tf.keras.layers.Dense(256, activation='relu')
self.dropout2 = tf.keras.layers.Dropout(0.2)
self.dense2 = tf.keras.layers.Dense(128, activation='relu')
self.dropout3 = tf.keras.layers.Dropout(0.2)
self.dense3 = tf.keras.layers.Dense(64, activation='relu')
self.dropout4 = tf.keras.layers.Dropout(0.2)
self.flatten = tf.keras.layers.Flatten()


                if ans_class == 'binary':
                    self.dense4 = tf.keras.layers.Dense(1, activation='sigmoid')
                else:
                    self.dense4 = tf.keras.layers.Dense(3, activation='softmax')

            def call(self, inputs, **kwargs):
                input_ids = inputs[0]
                attention_mask = inputs[1]
                outputs = self.distilbert(input_ids, attention_mask, **kwargs)[0]
                outputs = self.dropout1(outputs)
                outputs = self.dense1(outputs)
                outputs = self.dropout2(outputs)
                outputs = self.dense2(outputs)
                outputs = self.dropout3(outputs)
                outputs = self.dense3(outputs)
                outputs = self.dropout4(outputs)
                outputs = self.flatten(outputs)
                outputs = self.dense4(outputs)
                return outputs


        model = MyDistilBERTModel()
        input_ids = tf.keras.layers.Input(shape=(MAX_LENGTH,), dtype=tf.int32, name="input_ids")
        attention_mask = tf.keras.layers.Input(shape=(MAX_LENGTH,), dtype=tf.int32,
name="attention_mask")
        outputs = model([input_ids, attention_mask])

# Φόρτωση των βαρών του προ-εκπαιδευμένου μοντέλου
if ans_dataset == 'politics':
            if ans_class == 'binary' and ans_stopword_lemma == 'yes':
model.load_weights('E:\\h5\\Politics\\train2682_test810\\train_distilbert_bin_politics_with_sl.h5')
            elif ans_class == 'binary' and ans_stopword_lemma == 'no':
```

```
model.load_weights('E:\\h5\\Politics\\train2682_test810\\train_distilbert_bin_politics_without_sl.h5')
        elif ans_class == 'multi' and ans_stopword_lemma == 'yes':
model.load_weights('E:\\h5\\Politics\\train2682_test810\\train_distilbert_multi_politics_with_sl.h5')
        elif ans_class == 'multi' and ans_stopword_lemma == 'no':
model.load_weights('E:\\h5\\Politics\\train2682_test810\\train_distilbert_multi_politics_without_sl.h5')

    if ans_dataset == 'skroutz':
        if ans_stopword_lemma == 'yes':
          model.load_weights('E:\\h5\\Skroutz\\train_distilbert_bin_skroutz_with_sl.h5')
        else:
          model.load_weights('E:\\h5\\Skroutz\\train_distilbert_bin_skroutz_without_sl.h5')


    # Χρήση μεθόδου predict()
    if ans_class == 'binary':
        pred = np.where(model.predict([X_test_token['input_ids'], X_test_token['attention_mask']]) >= 0.5, 1,
0)
        m,r = report(y_test, pred, ans_class)
        awx_confusion_matrix(y_test, pred, label, ans_class)
    else:
        pred = model.predict([X_test_token['input_ids'], X_test_token['attention_mask']])
        y_pred_distilbert = np.zeros_like(pred)
        y_pred_distilbert[np.arange(len(y_pred_distilbert)), pred.argmax(1)] = 1

        m,r = report(y_test, y_pred_distilbert, ans_class, True)
        awx_confusion_matrix(y_test.argmax(axis=1), y_pred_distilbert.argmax(axis=1), label, ans_class)


  # GPT-2 MODEL
  if ans_models == 'gpt':
    label = 'GPT-2 Model'
    headers=['Text','Sentiment']

    if ans_dataset == 'politics':
        dataset = pd.read_csv('Politics/train2682_test810/test_set_politics810.csv', sep=',', names=headers)
    else:
        dataset = pd.read_csv('Skroutz/test_set_skroutz1966.csv', sep=',', names=headers)

    if ans_dataset == 'politics' and ans_class == 'binary':
        dataset = delete_neutral(dataset)

    if ans_stopword_lemma == 'yes':
        dataset = stopwords_lemmatize(dataset)


    if ans_class == 'binary':
         dataset.Sentiment.replace("Positive", 1, inplace = True)
         dataset.Sentiment.replace("Negative", 0, inplace = True)
         y_test = dataset['Sentiment']
         X_test = dataset['Text']
    else:
         ohe = preprocessing.OneHotEncoder()
        y_test = ohe.fit_transform(np.array(dataset['Sentiment']).reshape(-1, 1)).toarray()
        X_test = dataset['Text']
    tokenizer = GPT2Tokenizer.from_pretrained("lighteternal/gpt2-finetuned-greek", from_pt=True)
    X_test_token = get_tokens_GPT(X_test, tokenizer, MAX_LENGTH)

# Ορίζουμε το μοντέλο GPT-2 ως υποκλάση της Model
class MyGPTModel(tf.keras.Model):
        def __init__(self):
super(MyGPTModel, self).__init__()
```

```
            self.gpt = TFAutoModelForCausalLM.from_pretrained('lighteternal/gpt2-finetuned-greek',
from_pt=True)
self.flatten = tf.keras.layers.Flatten()

            if ans_class == 'binary':
                self.dense1 = tf.keras.layers.Dense(1, activation='sigmoid')
            else:
                self.dense1 = tf.keras.layers.Dense(3, activation='softmax')
        def call(self, inputs, **kwargs):
            input_ids = inputs
            outputs = self.gpt(input_ids, **kwargs)[0][:, -1, :]
            outputs = self.flatten(outputs)
            outputs = self.dense1(outputs)
            return outputs

    model = MyGPTModel()
    input_ids = tf.keras.layers.Input(shape=(MAX_LENGTH,), dtype=tf.int32, name="input_ids")
    outputs = model(input_ids)

    # Φόρτωση των βαρών του προ-εκπαιδευμένου μοντέλου
if ans_dataset == 'politics':
        if ans_class == 'binary' and ans_stopword_lemma == 'yes':
          model.load_weights('E:\\h5\\Politics\\train2682_test810\\train_gpt_bin_politics_with_sl.h5')
        elif ans_class == 'binary' and ans_stopword_lemma == 'no':
          model.load_weights('E:\\h5\\Politics\\train2682_test810\\train_gpt_bin_politics_without_sl.h5')
        elif ans_class == 'multi' and ans_stopword_lemma == 'yes':
          model.load_weights('E:\\h5\\Politics\\train2682_test810\\train_gpt_multi_politics_with_sl.h5')
        elif ans_class == 'multi' and ans_stopword_lemma == 'no':
          model.load_weights('E:\\h5\\Politics\\train2682_test810\\train_gpt_multi_politics_without_sl.h5')

    if ans_dataset == 'skroutz':
        if ans_stopword_lemma == 'yes':
            model.load_weights('E:\\h5\\Skroutz\\train_gpt_bin_skroutz_with_sl.h5')
        else:
            model.load_weights('E:\\h5\\Skroutz\\train_gpt_bin_skroutz_without_sl.h5')

    # Χρήση μεθόδου predict()
    if ans_class == 'binary':
        pred = np.where(model.predict([X_test_token['input_ids']]) >= 0.5, 1, 0)
        m,r = report(y_test, pred, ans_class)
        awx_confusion_matrix(y_test, pred, label, ans_class)
    else:
        pred = model.predict([X_test_token['input_ids']])
        y_pred_gpt = np.zeros_like(pred)
        y_pred_gpt[np.arange(len(y_pred_gpt)), pred.argmax(1)] = 1
        m,r = report(y_test, y_pred_gpt, ans_class, True)
        awx_confusion_matrix(y_test.argmax(1), y_pred_gpt.argmax(1), label, ans_class)
  return r, m
# Μέθοδος get_tokens για ανάκτηση αριθμητικών χαρακτηριστικών
def get_tokens(samples, tokenizer, MAX_LENGTH):
  my_dict = tokenizer(text = list(samples),
            add_special_tokens = True,
            max_length = MAX_LENGTH,
            truncation = True,
            padding = 'max_length',
            return_tensors = 'tf',
            return_token_type_ids = False,
            return_attention_mask = True,
            verbose = True)
  return my_dict
```

```
# Δήλωση της get_tokens_GPT μεθόδου για GPT-2.
def get_tokens_GPT(samples, tokenizer, MAX_LENGTH):
    my_dict = tokenizer(text = list(samples),
                add_special_tokens = False,
                max_length = MAX_LENGTH,
                truncation = True,
                padding = 'max_length',
                return_tensors = 'tf',
                verbose = False)
    return my_dict
```

Αρχείο utilities.py

```
# utilities.py
# βοηθητικές μέθοδοι
import tensorflow as tf
import numpy as np
import pandas as pd
# Για αφαίρεση λέξεων χωρίς αξία (stopwords)
from nltk.corpus import stopwords
stop_words = stopwords.words('greek') #nltk stop words
# Πακέτα οπτικοποίησης
import matplotlib.pyplot as plt
import matplotlib
matplotlib.use('agg')
import seaborn as sns
from sklearn import preprocessing
# Προεπεξεργασία δεδομένων
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
# Μοντέλα μηχανικής μάθησης
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn import svm

# Αναφορές - μετρικές
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
# Μέθοδος διατήρησης positive και negative (αποκοπή neutral)
def delete_neutral(dataset):
    list_text = list(dataset['Text'])
    list_sentiment = list(dataset['Sentiment'])

    nea_lista_text = []
    nea_lista_sentiment = []
```

```
    for sent, tweet in zip(list_sentiment, list_text):

        if sent == 'Neutral':
            nea_lista_text.append(None)
            nea_lista_sentiment.append(None)
        else:
            nea_lista_text.append(tweet)
            nea_lista_sentiment.append(sent)

    d = {'Text':nea_lista_text,'Sentiment':nea_lista_sentiment}
    df = pd.DataFrame(d)
    df = df.dropna()
    return df


# Μέθοδος για stopwords και lemmatize
def stopwords_lemmatize(dataset):
    import simplemma
    list_text = list(dataset['Text'])
    lista_without_stopwords = []

    for text in list_text:
        lista_without_stopwords.append(' '.join([word for word in text.split() if word not in stop_words]))

    dataset['Text'] = lista_without_stopwords

    list_text = list(dataset['Text'])
    lista_lemmatize = []

    for text in list_text:
        lista_lemmatize.append(' '.join([simplemma.lemmatize(word, lang='el') for word in text.split()]))

    dataset['Text'] = lista_lemmatize
    return dataset

# Μέθοδος tfidf_labelEncoder
def tfidf_labelEncoder(X, y, random_split, ans_test_size=0):
print(X, y)
    if not random_split:
        X_train, X_test = X
        y_train, y_test = y
print(X_train, X_test)
        tfidf_vectorizer = TfidfVectorizer(max_features=500000)
        tfidf_vectorizer.fit(X_train)
        X_train = tfidf_vectorizer.transform(X_train)
        X_test = tfidf_vectorizer.transform(X_test)

        LE = LabelEncoder()
        y_train = LE.fit_transform(y_train)
        y_test = LE.fit_transform(y_test)
```

```
print('Σχήμα δειγμάτων εκπαίδευσης: ', X_train.shape)
print('Σχήμα ετικετών εκπαίδευσης: ', y_train.shape)
print('Σχήμα δειγμάτων ελέγχου: ', X_test.shape)
print('Σχήμα ετικετών ελέγχου: ', y_test.shape)
print('\n')
   else:
      tfidf_vectorizer = TfidfVectorizer(max_features=500000)
      X = tfidf_vectorizer.fit_transform(X)

      LE = LabelEncoder()
      y = LE.fit_transform(y)

      if ans_test_size == 0.2:
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=ans_test_size, stratify=y,
random_state=12)
      elif ans_test_size == 0.3:
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=ans_test_size, stratify=y,
random_state=15)
      else:
         return ValueError

print('Σχήμα δειγμάτων εκπαίδευσης: ', X_train.shape)
print('Σχήμα ετικετών εκπαίδευσης: ', y_train.shape)
print('Σχήμα δειγμάτων ελέγχου: ', X_test.shape)
print('Σχήμα ετικετών ελέγχου: ', y_test.shape)
print('\n')

   return X_train, X_test, y_train, y_test


# Μέθοδος word2vec_labelEncoder
def  word2vec_labelEncoder(X, y, random_split, ans_dataset, ans_test_size=0):
   if not random_split:
      X_train, X_test = X

      y_train, y_test = y
      LE = LabelEncoder()
      y_train = LE.fit_transform(y_train)
      y_test = LE.fit_transform(y_test)

print('Σχήμα δειγμάτων εκπαίδευσης: ', X_train.shape)
print('Σχήμα ετικετών εκπαίδευσης: ', y_train.shape)
print('Σχήμα δειγμάτων ελέγχου: ', X_test.shape)
print('Σχήμα ετικετών ελέγχου: ', y_test.shape)
print('\n')

      import gensim
      model = gensim.models.KeyedVectors.load_word2vec_format('cc.el.300.vec', limit=1000000)
#limit=2000000 default
```

```
# Μήκος λέξης κάθε tweet. Σε αυτό το μήκος διατηρείται κάθε δείγμα tweet.
if ans_dataset == 'politics':
        MAX_LEN = 80
    else:
        MAX_LEN = 150
    # Πλήθος αριθμών μέσα στο Vector κάθε λέξης. Κάθε λέξη έχει ένα αντίστοιχο Vector που περιέχει αριθμούς.
VEC_LEN = 300

    Xtrain = tokenize(X_train, MAX_LEN, model)
    print('\n')
    Xtest = tokenize(X_test, MAX_LEN, model)

print('Σχήμα δειγμάτων εκπαίδευσης 3D: ', Xtrain.shape)

    # Μετατροπή πίνακα 3D σε 2D
Xtrain = np.reshape(Xtrain,(Xtrain.shape[0],MAX_LEN*VEC_LEN))
print('Σχήμα δειγμάτων εκπαίδευσης 2D: ', Xtrain.shape)
print('Σχήμα δειγμάτων ελέγχου 3D: ', Xtest.shape)

    # Μετατροπή πίνακα 3D σε 2D
Xtest = np.reshape(Xtest,(Xtest.shape[0],MAX_LEN*VEC_LEN))

print('Σχήμα δειγμάτων ελέγχου 2D: ', Xtest.shape)
else:
    import gensim
    LE = LabelEncoder()
    y = LE.fit_transform(y)



    if ans_test_size == 0.2:
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=ans_test_size, stratify=y, random_state=12)
    elif ans_test_size == 0.3:
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=ans_test_size, stratify=y, random_state=15)
    else:
        return ValueError

print('Σχήμα δειγμάτων εκπαίδευσης: ', X_train.shape)
print('Σχήμα ετικετών εκπαίδευσης: ', y_train.shape)
print('Σχήμα δειγμάτων ελέγχου: ', X_test.shape)
print('Σχήμα ετικετών ελέγχου: ', y_test.shape)
print('\n')

    model = gensim.models.KeyedVectors.load_word2vec_format('cc.el.300.vec', limit=100000)
#limit=2000000 default

# Μήκος λέξης κάθε tweet. Σε αυτό το μήκος διατηρείται κάθε δείγμα tweet.
if ans_dataset == 'politics':
```

```
            MAX_LEN = 80
        else:
            MAX_LEN = 150
        # Πλήθος αριθμών μέσα στο Vector κάθε λέξης. Κάθε λέξη έχει ένα αντίστοιχο Vector που περιέχει αριθμούς.
VEC_LEN = 300

        Xtrain = tokenize(X_train, MAX_LEN, model)
        print('\n')
        Xtest = tokenize(X_test, MAX_LEN, model)

print('Σχήμα δειγμάτων εκπαίδευσης 3D: ', Xtrain.shape)

# Μετατροπή πίνακα 3D σε 2D
Xtrain = np.reshape(Xtrain,(Xtrain.shape[0],MAX_LEN*VEC_LEN))

print('Σχήμα δειγμάτων εκπαίδευσης 2D: ', Xtrain.shape)
print('Σχήμα δειγμάτων ελέγχου 3D: ', Xtest.shape)

# Μετατροπή πίνακα 3D σε 2D
Xtest = np.reshape(Xtest,(Xtest.shape[0],MAX_LEN*VEC_LEN))
print('Σχήμα δειγμάτων ελέγχου 2D: ', Xtest.shape)
return Xtrain, Xtest, y_train, y_test



# Μέθοδος για τον πίνακα αντιστοιχίσεων (αριθμών) απο τις ενσωματώσεις λέξεων.
def tokenize(samples, max_len, model):
    list_token = []
    text = list(samples)
    for tweet in text:
        list_word = []
        for word in tweet.split():

            if word in model:
                my_vec = model.get_vector(word)
                #my_vec = my_vec * 100000000
                list_word.append(my_vec)

        list_token.append(list_word)

    pin = tf.keras.preprocessing.sequence.pad_sequences(list_token, maxlen=max_len, dtype='float32')
    print(pin)
return pin

# Μέθοδος επιλογής μοντέλου
def select_model(X_train, X_test, y_train, y_test, ans_class, ans_models, ans_embeddings):

    if ans_models == 'rf':
        label = 'RandomForest'
        model = RandomForestClassifier(n_estimators = 200)
```

```
            prediction, real = processing(model, X_train, X_test, y_train, y_test)
        elif ans_models == 'dt':
            label = 'DecisionTree'
            model = DecisionTreeClassifier(max_depth = 150)
            prediction, real = processing(model, X_train, X_test, y_train, y_test)
        elif ans_models == 'kn':
            label = 'KNeighbors'
            if ans_embeddings == 'tfidf':
                model = KNeighborsClassifier(n_neighbors = 3) # για tfidf 3 γείτονες
            else:
                model = KNeighborsClassifier(n_neighbors = 2, weights='distance') # για word2vec 2 γείτονες
            prediction, real = processing(model, X_train, X_test, y_train, y_test)
        elif ans_models == 'mnb':
            label = 'MultinomialNB'
            model = MultinomialNB()
            XtrainX = X_train - np.min(X_train)
            XtestX = X_test - np.min(X_test)
            prediction, real = processing(model, XtrainX, XtestX, y_train, y_test)
        elif ans_models == 'lr':
            label = 'LogisticRegression'
            model = LogisticRegression(random_state = 5, solver='lbfgs', max_iter=1000)
            prediction, real = processing(model, X_train, X_test, y_train, y_test)
        elif ans_models == 'svm':
            label = 'SupportVectorMachine'
            model = svm.SVC()
            prediction, real = processing(model, X_train, X_test, y_train, y_test)
        elif ans_models == 'gnb':
            label = 'GaussianNB'
            model = GaussianNB()
            if ans_embeddings == 'tfidf':
                trainX = X_train.toarray()
                model.fit(trainX, y_train)
                testX = X_test.toarray()
                prediction = model.predict(testX)
                print('15 ετικέτες πρόβλεψης:     ', prediction[:15])
                real = y_test
                print('15 πραγματικές ετικέτες: ', real[:15], '\n')
            else: # αν ans_embeddings == word2vec
                prediction, real = processing(model, X_train, X_test, y_train, y_test)

        m,r = report(real, prediction, ans_class)
        awx_confusion_matrix(real, prediction, label, ans_class)
        return r,m

    # Μέθοδος processing
    def processing(model, X_train, X_test, y_train, y_test):
        model.fit(X_train, y_train)
        prediction = model.predict(X_test)
        print('15 ετικέτες πρόβλεψης:     ', prediction[:15])
        real = y_test
```

```
    print('15 πραγματικές ετικέτες: ', real[:15], '\n')
    return prediction, real

# Μέθοδος report για classification report
def report(real, prediction, ans_class, transformers=False):
    if ans_class == 'binary':
        tn = ['Negative', 'Positive']
    else:
        tn = ['Negative', 'Neutral', 'Positive']
    if transformers:
        cm = confusion_matrix(real.argmax(axis=1), prediction.argmax(axis=1))
    else:
        cm = confusion_matrix(real, prediction)
    cr = str(classification_report(real, prediction))
    cm = np.array2string(cm)
    return cm, cr

# Μέθοδος awx_confusion_matrix για προβολή χάρτη θερμότητας
def  awx_confusion_matrix(real, prediction, label, ans_class):

    fig, ax = plt.subplots(figsize=(8,8))

    if ans_class == 'binary':
        tn = ['Negative', 'Positive']
    else:
        tn = ['Negative', 'Neutral', 'Positive']

    labels = tn
    ax = sns.heatmap(confusion_matrix(real, prediction), annot=True, cmap="Reds", fmt='g', cbar=True,
annot_kws={"size":17})
plt.title(label, fontsize=20)
    ax.xaxis.set_ticklabels(labels, fontsize=17)
    ax.yaxis.set_ticklabels(labels, fontsize=17)
    ax.set_ylabel('Real', fontsize=16)
    ax.set_xlabel('Predicted', fontsize=16)
ax.figure.savefig('plot.png')
    return
```

Αρχείο main.py

```
import tkinter as tk
import pandas as pd
from tkinter import PhotoImage, filedialog
from utilities import *
from utilities_transformers import select_model_transformers
tf.autograph.set_verbosity(0)
import tensorflow as tf
from PIL import Image, ImageTk
import time
import threading
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
files_list = []

#Μέθοδος εναλλαγής πλαισίου eg. start frame --> main_frame
def switch_frame(origin,goal):
goal.pack(fill="both", expand=True)
    if goal!=start_frame:
root.geometry('1500x850')
origin.pack_forget()
    else:
root.geometry('1000x800')
origin.pack_forget()

#Μέθοδος διάσπασης threads στη CPU  TODO: Optimization for GPU
def thread(machine_learning=True):
    global start_time, t1
    if machine_learning:
        button.config(state=tk.DISABLED)
        label3.grid(row=7, column=3)
        t2=threading.Thread(target= lambda: print_time(label3, main_frame))
        t2.start()
        t1=threading.Thread(target=run_ML_models)
        t1.start()
    else:
        start_time = time.time()
        button3.config(state=tk.DISABLED)
        label4.grid(row=7, column=3)
        t2=threading.Thread(target= lambda: print_time(label4, transformers_frame))
        t2.start()
        t1=threading.Thread(target=run_transformers)
        t1.start()

#Μέθοδος που εμφανίζει και εξαφανίζει το κείμενο Please wait... κάθε ένα δευτερόλεπτο
def print_time(loading_label, frame):
    if loading_label.cget("text")=="Please wait...":
        loading_label.configure(text="")
    else:
        loading_label.configure(text="Please wait...")
```

```
#αν η thread 1 δεν τρέχει τότε επιστρέφει
if not t1.is_alive():
        return
    # Εμφανίζει το κείμενο κάθε 1 δευτερόλεπτο καλώντας τον εαυτό της
frame.after(1000, lambda: print_time(loading_label, frame))

#Μέθοδος για εκτέλεση Machine Learning πειραμάτων
def run_ML_models():
    main_frame.after(0, lambda: print_time(label3, main_frame)) #Ξεκινά την μέθοδο print_time, δηλαδή την
t2 thread
    #Συλλογη μεταβλητών
    ans_test_size = ""
    ans_dataset = dataset_var.get()
    ans_class = check_var.get()
    ans_stopword_lemma = stopwords_var.get()
    ans_embeddings = vectorizer_var.get()
    ans_models = model_var.get()
    ans_split = random_split_var.get()
    if ans_models != 'Select a model' and ans_class != "Select Classification" and ans_dataset !='Select Dataset'
and ans_stopword_lemma != 'Stopwords/Lemmatization' and ans_embeddings !='Select Vectorizer' and
ans_split !='Random Split':
#Φόρτωση της βάσης δεδομένων
ifans_dataset == 'politics':
headers=['Text','Sentiment']
            headers2=['Created_at','Username','Country','City','State_District','Topic','Text','Sentiment']
            politics = pd.read_csv('Politics/balanced_politics.csv', sep=',', names=headers2)
            train_politics = pd.read_csv('Politics/train2682_test810/train_set_politics2682.csv', sep=',',
names=headers)
            test_politics = pd.read_csv('Politics/train2682_test810/test_set_politics810.csv', sep=',',
names=headers)
        elif ans_dataset == 'skroutz':
            headers=['Text','Sentiment']
            headers2=['id','Text','Sentiment']
            skroutz = pd.read_csv('Skroutz/skroutz_dataset.csv', sep=',', names=headers)
            train_skroutz = pd.read_csv('Skroutz/train_set_skroutz3210.csv', sep=',', names=headers)
            test_skroutz = pd.read_csv('Skroutz/test_set_skroutz1966.csv', sep=',', names=headers)
        else:
print('Δώστε ένα απο τα παραπάνω dataset!')
#Προεπεξεργασία
        if ans_dataset == 'politics':
            if ans_class == 'binary':
                politics = delete_neutral(politics)
                train_politics = delete_neutral(train_politics)
                test_politics = delete_neutral(test_politics)
        if ans_stopword_lemma == 'yes' and ans_dataset == 'politics':
            politics = stopwords_lemmatize(politics)
            train_politics = stopwords_lemmatize(train_politics)
            test_politics = stopwords_lemmatize(test_politics)
        if ans_stopword_lemma == 'yes' and ans_dataset == 'skroutz':
            skroutz=stopwords_lemmatize(skroutz)
```

```
            train_skroutz = stopwords_lemmatize(train_skroutz)
            test_skroutz = stopwords_lemmatize(test_skroutz)
#Δημιουργία test και trainset, αναλόγως την περίπτωση χρήσης
if ans_dataset == 'politics':
            X = politics['Text']
            y = politics['Sentiment']
            X_train = train_politics['Text']
            y_train = train_politics['Sentiment']
            X_test = test_politics['Text']
            y_test = test_politics['Sentiment']
        else:
            X = skroutz['Text']
            y = skroutz['Sentiment']
            X_train = train_skroutz['Text']
            y_train = train_skroutz['Sentiment']
            X_test = test_skroutz['Text']
            y_test = test_skroutz['Sentiment']
        if ans_embeddings == 'tfidf':
            if ans_split == 'no':
                X_train, X_test, y_train, y_test = tfidf_labelEncoder((X_train, X_test), (y_train, y_test), False)
            else:
                ans_test_size = open_popup(root)
                if ans_test_size != 'error':
                    X_train, X_test, y_train, y_test = tfidf_labelEncoder(X, y, True, ans_test_size)
                else:
                  label.delete(1.0, "end-1c")
                  label.insert(1.0, str('Please insert 0.2 or 0.3'))
                   button.config(state=tk.NORMAL)
                   label3.grid_forget()
                   return
        if ans_embeddings == 'word2vec':
            if ans_split == 'no':
                X_train, X_test, y_train, y_test = word2vec_labelEncoder((X_train, X_test), (y_train, y_test), False,
ans_dataset)
            else:
                ans_test_size = open_popup(root)
                if ans_test_size != 'error':
                    X_train, X_test, y_train, y_test = word2vec_labelEncoder(X, y, True, ans_dataset, ans_test_size)
                else:
                  clear()
label.insert(1.0, str('Please insert 0.2 or 0.3'))
                   button.config(state=tk.NORMAL)
                   label3.grid_forget()
                   return
        #Εκτέλεση πειράματος
        report, matrix  = select_model(X_train, X_test, y_train, y_test, ans_class, ans_models, ans_embeddings)
label.delete(1.0, "end-1c")
label.insert(1.0,f'Model: {ans_models},Dataset: {ans_dataset}, Class: {ans_class}, Stopwords/Lemmatize:
{ans_stopword_lemma}, Vectorizer: {ans_embeddings}\n\n\n'+str(report)+'\n'+'Classification Report
\n'+matrix)
```

```
        #Αποθήκευση confusion matrix σαν plot.png
        new_image = Image.open('plot.png')
        size = (400, 400)
        scaled_img = new_image.resize(size)
        tk_image2 = ImageTk.PhotoImage(scaled_img)
        label3.grid_forget()
        button.config(state=tk.NORMAL)
#αλλαγή της εικόνας στην plot.png
confusion_matrix_image.configure(image=tk_image2)
        confusion_matrix_image.image = tk_image2
    else:
label.delete(1.0, "end-1c")
label.insert(1.0, str('Please select a model and try again!'))
    end_time = time.time()
    runtime = end_time - start_time
label.insert(1.0,f"Runtime: {round(runtime,2)} seconds \n\n")
    content = label.get("1.0", "end-1c")
#Αποθήκευση περιοχομένων του textlabel σε txt αρχείο με κατάλληλο όνομα
with
open("files/"+ans_models+"_"+ans_dataset+"_"+ans_class+"_"+ans_stopword_lemma+"SL_"+ans_embeddin
gs+"_split"+ans_split+str(ans_test_size)+".txt", "w") as file:
file.write(content)

#Μέθοδος για εκτέλεση Deep Learning πειραμάτων
def run_transformers():
    transformers_frame.after(0, lambda: print_time(label4, transformers_frame))
    #Συλλογη μεταβλητών
    ans_dataset = dataset_var2.get()
    ans_class = class_var.get()
    ans_stopword_lemma = stopwords_var2.get()
    ans_models = transformers_var.get()
    if ans_models != 'Select a model' and ans_class != "Select Classification" and ans_dataset !='Select Dataset'
and ans_stopword_lemma != 'Stopwords/Lemmatization':
        #Εκτέλεση πειραμάτων
        report, matrix = select_model_transformers(ans_dataset, ans_class, ans_stopword_lemma, ans_models)
        transformers_label.delete(1.0, "end-1c")
        transformers_label.insert(1.0,f'Model: {ans_models},Dataset: {ans_dataset}, Class: {ans_class},
Stopwords/Lemmatize: {ans_stopword_lemma}\n\n\n'+str(report)+'\n'+'Classification Report \n'+matrix)
        new_image = Image.open('plot.png')
        size = (400, 400)
        scaled_img = new_image.resize(size)
        tk_image2 = ImageTk.PhotoImage(scaled_img)
        label4.grid_forget()
        button3.config(state=tk.NORMAL)
        confusion_matrix_image2.configure(image=tk_image2)
        confusion_matrix_image2.image = tk_image2
    else:
        transformers_label.delete(1.0, "end-1c")
        transformers_label.insert(1.0, str('Please select a model and try again!'))
    end_time = time.time()
```

```
    runtime = end_time - start_time
    transformers_label.insert(1.0,f"Runtime: {round(runtime,2)} seconds \n\n")
#TODO:Αποθήκευση περιοχομένων του textlabel σε txt αρχείο με κατάλληλο όνομα

#Μέθοδος καθαρισμού text label
def clear(text_label, matrix_label):
    text_label.delete(1.0, "end-1c")
    matrix_label.configure(image=tk_img)
    matrix_label.image = tk_img

#TODO: Μέθοδος φόρτωσης μιας τυχαίας βάσης δεδομένων
def load_csv():
    file_path = filedialog.askopenfilename(defaultextension=".csv", filetypes=[("CSV Files", "*.csv")])
    if file_path:
        txt = str(file_path)
        print(txt)
        df = pd.read_csv(file_path)
dataset.append(txt)
        print(dataset_box['menu'])
        #label2.insert(1.0, txt)
        #label2.config(state="normal")
        #label2.delete(1.0, "end-1c")
        #label2.insert(1.0, txt)
        #label2.config(state="disabled")
        return df

#Ενα αναδυόμενο παράθυρο, για εισαγωγή του ποσοστού που θα έχει το test set
def open_popup(parent):
    top, entry_var = popUpText(parent)
parent.wait_window(top)
    try:
        value = float(entry_var.get())
    except:
value = "error"
returnvalue
#Μέθοδος αποθήκευσης τιμής για το ποσοστό του testset
def popUpText(parent):
    top = tk.Toplevel(parent)
top.resizable(0,0)
    #τοποθέτηση του αναδυόμενου παραθύρου στο κέντρο της οθόνης
width = 300
height = 80
    #μήκος και πλάτος της οθόνης
screen_width = root.winfo_screenwidth()
    screen_height = root.winfo_screenheight()
#υπολογισμός x και y συντεταγμένων
x = (screen_width/2) - (width/4)
y = (screen_height/2) - (height/2)
top.geometry('%dx%d+%d+%d' % (width, height, x, y))
    text_label = tk.Label(top, text='Insert a decimal 0.2 or 0.3 to apply the random split')
```

```
    text_label.pack()
    entry_var = tk.StringVar()
    entry = tk.Entry(top, textvariable=entry_var)
entry.pack()
    save_button = tk.Button(top, text="Save", command=lambda: save_entry(entry_var, top))
    save_button.pack()
    return top, entry_var

def save_entry(entry_var, top):
    entry_text = entry_var.get()
top.destroy()
    return entry_text

#-----------------------------------Αρχή Εφαρμογής Tkinter-----------------------------------------------------

root = tk.Tk()
root.title("K.A.S.S.")
root.geometry("1000x800")
root.configure(bg='#ADD8E6')
root.resizable(0,0) #ρυθμιση ώστε να μην είναι μεγαλώνει ο χρήστης τις διαστάσεις του παραθύρου

#TODO:Μενού αρχείων και διασύνδεση με τη load_csv μέθοδο
# Create the menu widget
menu = tk.Menu(root)
root.config(menu=menu)

#Δημιουργία μενού αρχείων
file_menu = tk.Menu(menu)
menu.add_cascade(label="File", menu=file_menu)

#επιλογές μενού αρχείων
file_menu.add_command(label="Open", command=load_csv)
file_menu.add_command(label="Save", command=lambda: print("Save"))
file_menu.add_separator()
file_menu.add_command(label="Exit", command=root.quit)

#-----------------------------------Αρχή βασικού πλαισίου-----------------------------------------------------


start_frame = tk.Frame(root)
start_frame.pack(fill="both", expand=True)
title = tk.Label(start_frame, text="International Hellenic University\nSentiment Analyzer", font=("Helvetica",
24), fg='black')
image1 = PhotoImage(file="files/univercity_logo.png")
image_label = tk.Label(start_frame, image=image1)
#label2 = tk.Text(start_frame, wrap="word", state="disabled")
start_button = tk.Button(start_frame,height=2,width=20,activebackground='red', text="Machine Learning",
command=lambda: switch_frame(start_frame, main_frame))
load_button = tk.Button(start_frame,height=2,width=20,activebackground='red', text="Deep Learning",
command=lambda: switch_frame(start_frame, transformers_frame))
title.grid(row=0, column=1, pady=100, padx=10)
```

```
image_label.grid(row=0, column=0, pady=100, padx=10)
#label2.grid(row=1)
load_button.grid(row=4,column=1)
start_button.grid(row=5,column=1)


#-----------------------------------Τέλος βασικού πλαισίου-----------------------------------------------------

#-----------------------------------------------------Αρχή Machinelearning πλαισίου-----------------------------------
----------------------------


main_frame = tk.Frame(root)

#ορισμός από λίστες για τις υπερπαραμέτρους των πειραμάτων
models = ['rf', 'dt', 'kn', 'mnb', 'lr', 'svm', 'gnb']
stopwords = ['yes', 'no']
binary = ['binary', 'multi']
vectorizer = ['tfidf', 'word2vec']
dataset = ['politics', 'skroutz']
#ορισμός μεταβλητων tkinter τυπου string
model_var = tk.StringVar()
stopwords_var = tk.StringVar()
check_var = tk.StringVar()
dataset_var = tk.StringVar()
vectorizer_var = tk.StringVar()
random_split_var = tk.StringVar()
#Ταμπέλα Please wait...
label3 = tk.Label(main_frame, text='Please wait...')

#ορισμός ταμπέλας πίνακα σύγχησης
img = Image.open('files/blank.png')
new_size = (400, 400)
scaled_image = img.resize(new_size)
tk_img = ImageTk.PhotoImage(scaled_image)
confusion_matrix_image = tk.Label(main_frame,image=tk_img)

#Προεπιλεγμένες τιμες
model_var.set("Select Model")
stopwords_var.set('Stopwords/Lemmatization')
check_var.set("Select Classification")
dataset_var.set("Select Dataset")
vectorizer_var.set("Select Vectorizer")
random_split_var.set("Random Split")

#Περιγραφικές ταμπέλες
dropdown_label = tk.Label(main_frame, text="Choose a Model:", font=("Helvetica", 12), bg='#ADD8E6')
stopwords_box_label = tk.Label(main_frame, text="Select [yes] to remove Stopwords and apply
Lemmatization:", font=("Helvetica", 12), bg='#ADD8E6')
check_box_label = tk.Label(main_frame, text="Select Binary or Trinary Classification:", font=("Helvetica",
12), bg='#ADD8E6')
```

```
dataset_box_label = tk.Label(main_frame, text="Select dataset: ", font=("Helvetica", 12), bg='#ADD8E6')
vectorizer_box_label = tk.Label(main_frame, text="Select vectorizer: ", font=("Helvetica", 12),
bg='#ADD8E6')
random_split_label = tk.Label(main_frame, text="Select [yes] to apply random split: ", font=("Helvetica", 12),
bg='#ADD8E6')

#Ταμπέλες πολλαπλών επιλογών
check_box = tk.OptionMenu(main_frame, check_var, *binary)
dataset_box = tk.OptionMenu(main_frame, dataset_var, *dataset)
model_box = tk.OptionMenu(main_frame, model_var, *models)
stopwords_box = tk.OptionMenu(main_frame, stopwords_var,*stopwords )
vectorizer_box = tk.OptionMenu(main_frame,vectorizer_var,*vectorizer)
random_split_box = tk.OptionMenu(main_frame, random_split_var, *stopwords)
#Ταμπέλα text
label = tk.Text(main_frame, height=20, width=80, padx=2,pady=15)

#Κουμπια έναρξης, επιστροφής και εκαθάρισης
clear_button = tk.Button(main_frame, text="Delete All", command=lambda: clear(label,
confusion_matrix_image), bg='red', fg='white', font=("Helvetica", 12))
button = tk.Button(main_frame, text="Print", command=thread, bg='green', fg='white', font=("Helvetica", 12))
button2 = tk.Button(main_frame, text="Back", command=lambda: switch_frame(main_frame, start_frame),
bg='blue', fg='white', font=("Helvetica", 12))

#Παράθεση όλων των ταμπέλων στην οθόνη με χρήση του συστήματος πλέγματος(grid) του tkinter
dropdown_label.grid(row=0, column=0, padx=10, pady=10, sticky="w")
model_box.grid(row=0, column=1, padx=10, pady=10)
check_box_label.grid(row=1, column=0, padx=10, pady=10, sticky="w")
check_box.grid(row=1, column=1, padx=10, pady=10)
vectorizer_box_label.grid(row=2, column=0, padx=10, pady=10,sticky="w")
vectorizer_box.grid(row=2, column=1, padx=10, pady=10)
dataset_box_label.grid(row=3, column=0, padx=10, pady=10, sticky="w")
dataset_box.grid(row=3, column=1, padx=10, pady=10)
stopwords_box_label.grid(row=4, column=0, padx=10, pady=10, sticky="w")
stopwords_box.grid(row=4, column=1, padx=10, pady=10)
random_split_label.grid(row=5, column=0, padx=10, pady=10, sticky="w")
random_split_box.grid(row=5, column=1, padx=10, pady=10)
label.grid(row=6, column=0, columnspan=2,padx=10, pady=10)
confusion_matrix_image.grid(row=6, column=3)
clear_button.grid(row=7, column=0,padx=10, pady=10, sticky="w")
button.grid(row=7, column=2)
button2.grid(row=8, column=0,padx=10, pady=10, sticky="w")



#----------Τέλος Machine learning πλαισίου----------



#----------Αρχή Deep learning /  transformers πλαισίου----------
```

```
transformers_frame = tk.Frame(root)
#ορισμός από λίστες για τις υπερπαραμέτρους των πειραμάτων
transformers = ['bert', 'distilbert', 'roberta', 'gpt']
transformers_var = tk.StringVar()
stopwords_var2 = tk.StringVar()
class_var = tk.StringVar()
dataset_var2 = tk.StringVar()

#Ταμπέλα Please wait...
label4 = tk.Label(transformers_frame, text='Please wait...')
#ορισμός ταμπέλας πίνακα σύγχυσης
confusion_matrix_image2 = tk.Label(transformers_frame,image=tk_img)

#Προεπιλεγμένες τιμες
transformers_var.set("Select Model")
stopwords_var2.set('Stopwords/Lemmatization')
class_var.set("Select Classification")
dataset_var2.set("Select Dataset")

#Περιγραφικές ταμπέλες
dropdown_label2 = tk.Label(transformers_frame, text="Choose a Model:", font=("Helvetica", 12),
bg='#ADD8E6')
stopwords_box_label2 = tk.Label(transformers_frame, text="Select [yes] to remove Stopwords and apply
Lemmatization:", font=("Helvetica", 12), bg='#ADD8E6')
check_box_label2 = tk.Label(transformers_frame, text="Select Binary or Trinary Classification:",
font=("Helvetica", 12), bg='#ADD8E6')
dataset_box_label2 = tk.Label(transformers_frame, text="Select dataset: ", font=("Helvetica", 12),
bg='#ADD8E6')
#Ταμπέλες πολλαπλών επιλογών
class_box = tk.OptionMenu(transformers_frame, class_var, *binary)
dataset_box2 = tk.OptionMenu(transformers_frame, dataset_var2, *dataset)
model_box2 = tk.OptionMenu(transformers_frame, transformers_var, *transformers)
stopwords_box2 = tk.OptionMenu(transformers_frame, stopwords_var2,*stopwords )
#Ταμπέλα text
transformers_label = tk.Text(transformers_frame, height=20, width=80, padx=2,pady=15)
#Κουμπια έναρξης, επιστροφής και εκαθάρισης
clear_button2 = tk.Button(transformers_frame, text="Delete All", command=lambda:
clear(transformers_label, confusion_matrix_image2), bg='red', fg='white', font=("Helvetica", 12))
button3 = tk.Button(transformers_frame, text="Print", command=lambda: thread(machine_learning=False),
bg='green', fg='white', font=("Helvetica", 12))
button4 = tk.Button(transformers_frame, text="Back", command=lambda: switch_frame(transformers_frame,
start_frame), bg='blue', fg='white', font=("Helvetica", 12))

#Παράθεση όλων των ταμπέλων στην οθόνη με χρήση του συστήματος πλέγματος(grid) του tkinter
dropdown_label2.grid(row=0, column=0, padx=10, pady=10, sticky="w")
model_box2.grid(row=0, column=1, padx=10, pady=10)
check_box_label2.grid(row=1, column=0, padx=10, pady=10, sticky="w")
class_box.grid(row=1, column=1, padx=10, pady=10)
dataset_box_label2.grid(row=2, column=0, padx=10, pady=10, sticky="w")
dataset_box2.grid(row=2, column=1, padx=10, pady=10)
```

```
stopwords_box_label2.grid(row=3, column=0, padx=10, pady=10, sticky="w")
stopwords_box2.grid(row=3, column=1, padx=10, pady=10)
transformers_label.grid(row=4, column=0, columnspan=2,padx=10, pady=10)
confusion_matrix_image2.grid(row=4, column=3)
clear_button2.grid(row=5, column=0,padx=10, pady=10, sticky="w")
button3.grid(row=5, column=2)
button4.grid(row=6, column=0,padx=10, pady=10, sticky="w")


#----------------------------------------------------------Τέλος Deep learning /  transformers πλαισίου----------------------
-------------------------------------------
root.mainloop()
```

Παράρτημα Β: Εγκατάσταση πακέτων

Για τις εγκαταστάσεις όλων των βιβλιοθηκών, ανοίγουμε το Anaconda Prompt (Windows), Terminal (macOS, Linux) ή shell (Linux).

Διανομή Anaconda Python

Περιλαμβάνει πολλές βιβλιοθήκες Python οπτικοποίησης δεδομένων, τον διερμηνέα IPython, και τα τετράδια Jupyter. Για εγκατάσταση κατεβάζουμε το Python 3.x Anaconda για Windows, Linux, macOS, από την σελίδα:

```
https://www.anaconda.com/download
```

Μετά την ολοκλήρωση της εγκατάστασης, στην γραμμή εντολών εισάγουμε τις παρακάτω εντολές για να ενημερώσουμε τα πακέτα στις τελευταίες εκδόσεις τους.

```
conda update conda
conda update --all
```

Εγκατάσταση της βιβλιοθήκης TextBlob

```
conda install -c conda-forge textblob
```

Εγκατάσταση της Spacy

```
Pip install -U spacy

python -m spacy download<language_model>
```

Για την ελληνική γλώσσα μπορούμε να επιλέξουμε ένα εκ τριών μοντέλων el_core_news_sm, el_core_news_md και el_core_news_sm. Στη συνέχεια κάνουμε import και load.

```
import spacy

nlp = spacy.load("el_core_news_sm")
```

Εγκατάσταση του wordcloud

```
conda install -c conda-forge wordcloud
```

Εγκατάσταση της Tweepy

```
pip install tweepy == 4.6.0
```

Εγκατάσταση βιβλιοθήκης tweet-preprocessor

```
pip install tweet-preprocessor
```

Εγκατάσταση βιβλιοθήκης geopy

```
conda install -c conda-forge geopy
```

Εγκατάσταση βιβλιοθήκης Folium

```
pip install folium
```

Εγκατάσταση TensorFlow

```
pip install tensorflow
```

Η παραπάνω εντολή θα κατεβάσει και θα εγκαταστήσει την τελευταία έκδοση του TensorFlow. Μετά την εγκατάσταση, μπορείτε να ελέγξετε εάν η εγκατάσταση έγινε σωστά εκτελώντας τις παρακάτω εντολές στο κέλυφος εντολών ή στην κονσόλα:

```
python
import tensorflow as tf
tf.__version__
```

Εάν δεν εμφανιστεί κάποιο μήνυμα λάθους, τότε η εγκατάσταση του TensorFlow έχει ολοκληρωθεί με επιτυχία.

Παράρτημα Γ: Οδηγίες εγκατάστασης KASS

ΟΔΗΓΙΕΣ ΕΓΚΑΤΑΣΤΑΣΗΣ KASS με χρήση virtual environment στο Visual Studio Code

1. Μεταβείτε στο αποθετήριο https://github.com/stelsar123/kass και αποθηκεύστε τα αρχεία που παρέχονται σε έναν επιθυμητό φάκελο.

2. Ανοίξτε το Visual Studio Code.

3. Ανοίξτε τον φάκελο που περιέχει την εφαρμογή Tkinter, επιλέγοντας "File" -> "Open Folder" και πλοηγηθείτε στον επιθυμητό κατάλογο.

4. Αφού ο φάκελος του έργου σας είναι ανοιχτός στο Visual Studio Code, κάντε κλικ στο εικονίδιο του τερματικού (Terminal) στο οριζόντιο μενού επιλογών και επιλέξτε τη δημιουργία νέου τερματικού (New Terminal)

5. Στο τερματικό, δημιουργήστε ένα νέο virtual environment χρησιμοποιώντας την εντολή python με την επιλογή -m venv και καθορίζοντας ένα όνομα για το virtual environment. Για παράδειγμα, για να δημιουργήσετε ένα virtual environment με το όνομα "venv", χρησιμοποιήστε την παρακάτω εντολή:

```
python -m venv venv
```

6. Ενεργοποιήστε το virtual environment εκτελώντας το αρχείο ενεργοποίησης που βρίσκεται στον κατάλογο Scripts του virtual environment. Χρησιμοποιήστε την παρακάτω εντολή:

```
.\venv\Scripts\activate
```

7. Μετά την ενεργοποίηση του virtual environment, θα δείτε το όνομά του μέσα σε παρενθέσεις στην αρχή της γραμμής εντολών του τερματικού. Αυτό υποδηλώνει ότι τώρα εργάζεστε εντός του virtual environment.

8. Εγκαταστήστε τα απαιτούμενα πακέτα από το αρχείο requirements.txt χρησιμοποιώντας την εντολή pip. Εκτελέστε την παρακάτω εντολή:

```
pip install -r requirements.txt
```

Αυτή η εντολή θα διαβάσει το αρχείο requirements.txt και θα εγκαταστήσει όλα τα πακέτα που αναφέρονται σε αυτό.

9. Μόλις ολοκληρωθεί η εγκατάσταση, μπορείτε να εκτελέσετε την εφαρμογή Tkinter επιλέγοντας το αρχείο Python main.py. Ανοίξτε το αρχείο στο Visual Studio Code και κάντε κλικ στο κουμπί "Run" στο πάνω μέρος του επεξεργαστή ή χρησιμοποιήστε το πλήκτρο προσαρμογής Ctrl + F5.

Βεβαιωθείτε ότι έχετε ορίσει το σωστό διερμηνέα Python στο Visual Studio Code. Μπορείτε να το κάνετε αυτό επιλέγοντας τον διερμηνέα από τη γραμμή κατάστασης στο κάτω μέρος του επεξεργαστή.