ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

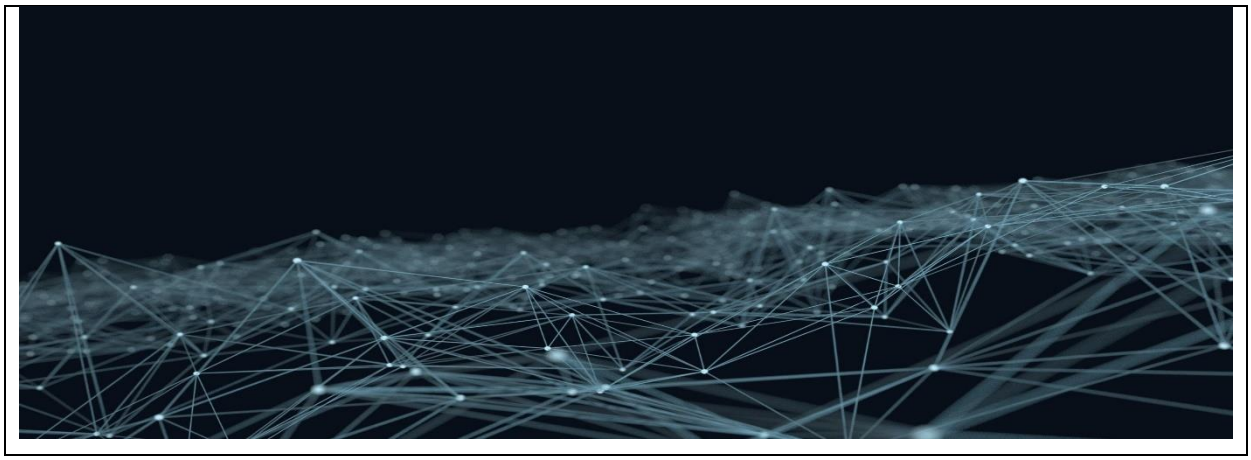
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη Συστήματος Αυτόματου Διαλόγου

(Chatbot)



Του φοιτητή
Νεοκάζη Παναγιώτη
Αρ. Μητρώου: 134059

Επιβλέπων
Κωνσταντίνος Διαμαντάρας
Καθηγητής

Ημερομηνία 05-02-2022

Τίτλος Π.Ε. Ανάπτυξη συστήματος αυτόματου διαλόγου (Chatbot)
Κωδικός Π.Ε. 20227
Νεοκάζης Παναγιώτης
Κωνσταντίνος Διαμαντάρας
Ημερομηνία ανάληψης Π.Ε. 03-11-2020
Ημερομηνία περάτωσης Π.Ε. 05-02-2022

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Περίληψη

Με την ταχύρρυθμη ανάπτυξη της τεχνολογίας και την άνθιση ποικίλων υπηρεσιών του διαδικτύου τα τελευταία χρόνια, και παράλληλα, με την εξέλιξη της τεχνητής νοημοσύνης (Artificial Intelligence - AI) και του Internet of Things (IoT) έρχονται στο προσκήνιο καινοτόμες εφαρμογές οι οποίες βρίσκουν πρόσφορο έδαφος σε διάφορους τομείς και μπορούν να εξυπηρετήσουν αλλά και να συναρπάσουν μικρούς και μεγάλους. Ένα εξαιρετικό επίτευγμα αυτών των τεχνολογιών είναι τα συστήματα αυτόματου διαλόγου, κοινώς Chatbots.

Η παρούσα πτυχιακή εργασία αφορά τη σχεδίαση και την υλοποίηση ενός συστήματος αυτόματου διαλόγου χρησιμοποιώντας και αναλύοντας σύγχρονες τεχνικές.

Στην αρχή γίνεται μια εισαγωγή στα chatbots και τις κατηγορίες που διαχωρίζονται βάση της δομής που ακολουθούν καθώς και μια σύντομη ιστορική αναδρομή για να δούμε την πορεία που ακολουθούν μέχρι σήμερα. Στη συνέχεια αναλύονται τεχνικές επεξεργασίας φυσικού λόγου μέσα από σύγχρονες βιβλιοθήκες που επικρατούν στο χώρο καθώς και τεχνικές, στατιστικοί αλγόριθμοι μηχανικής μάθησης για text classification και αλγόριθμοι βαθιάς μάθησης. Έπειτα αναλύεται η δομή του συστήματος ξεχωριστά για κάθε κομμάτι που απαρτίζεται και τέλος, η ενοποίηση των διεργασιών αυτών που οδηγούν στην τελική εφαρμογή.

Μέσω αυτού του συστήματος, ο χρήστης θα μπορεί να αναζητεί πληροφορίες για διαθέσιμα αεροπορικά εισιτήρια, για τις αεροπορικές εταιρείες που εξυπηρετούν αυτά τα δρομολόγια καθώς και τις φθηνότερες πτήσεις.

«Development of an Automatic Dialogue System»

«Neokazis Panagiotis»

Abstract

With the rapid development of technology and the flourishing of various internet services in recent years, and simultaneously, with the development of Artificial Intelligence (AI) and Internet of things, innovative applications which find breeding ground in different fields and are able to serve and fascinate people of all ages come to the fore. An outstanding achievement of these technologies is the automatic dialogue systems, the Chatbots.

This dissertation concerns the design and implementation of an automatic dialogue system using and analysing modern techniques.

In the beginning, there is an introduction about chatbots and the categories, separated based on the structure that they follow, as well as a brief historical background in order to see the course they follow to date. In continue, techniques of natural speech processing through modern libraries that prevail in the field as well as technique, statistical machine learning algorithms for text classification and deep learning algorithms are analysed. The analysis of the structure of the system separately for each piece that is captured follows, and finally, the integration of these processes that lead to the final implementation.

Through this system, the user will be able to search for information on available airline tickets, the airlines that serve these routes as well as the cheapest flights.

Ευχαριστίες

Με την ολοκλήρωση της παρούσας πτυχιακής εργασίας σηματοδοτείται η λήξη των προπτυχιακών μου σπουδών στο τμήμα Μηχανικών Πληροφορικής, σε αυτό το σημείο νιώθω την ανάγκη να εκφράσω την ευγνωμοσύνη μου για όλη την υποστήριξη που έλαβα από την οικογένεια μου για να καταφέρω να φτάσω μέχρι εδώ. Τους φίλους μου, την Σπυριδούλα που με ενθάρρυναν διαρκώς. Τέλος, θα ήθελα να ευχαριστήσω τον καθηγητή μου, κ. Κωνσταντίνο Διαμαντάρα, που ανέλαβε πρόθυμα την επίβλεψη της εργασίας καθώς και για την καθοδήγηση του.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Συντομογραφίες

ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία
AI	Artificial Intelligence
IOT	Internet of Things
NLP	Natural Language Processing
NLU	Natural Language Understanding
ML	Machine Learning
DL	Deep Learning
ANN	Artificial Neural Networks
POS	Part of Speech
NER	Named Entities Recognition
BOW	Bag of Words
TF-IDF	Term Frequency – Inverse Document Frequency
NB	Naive Bayes
SVM	Support Vector Machine
RNN	Recurrent Neural Network
LSTM	Long Short Term Memory
ATIS	Airline Travel Information Systems
IR	Intent Recognition
SF	Slot Filling

Κεφάλαιο 1ο: Εισαγωγή στα Chatbots

1.1 Εισαγωγή

Τα Chatbots έχουν αναδειχθεί ως ένας νέος τρόπος αλληλεπίδρασης με πολλές υπηρεσίες του διαδικτύου καθώς και με έξυπνες συσκευές, παρέχοντας διεπαφές συνομιλίας όπου δίνουν στον τελικό χρήστη μια ιδιαίτερη, πιο φυσική αίσθηση της επικοινωνίας σε σχέση με τις παραδοσιακές διεπαφές χρήστη. Γνωστά και ως Interactive Agents, Talkbots ή Virtual Assistants, τα Chatbots είναι εξειδικευμένα συστήματα λογισμικού τα οποία μπορούν και εξυπηρετούν συγκεκριμένες μορφές αναγκών των χρηστών όπως την αναζήτηση και την ανάκτηση πληροφορίας μέσα από μια πηγή δεδομένων, τη δέσμευση ραντεβού με κάποιον υπάλληλο μιας εταιρείας ή ακόμα και την πραγματοποίηση μιας αγοράς χωρίς την ύπαρξη κάποιου φυσικού προσώπου.

Εκ των πραγμάτων, ένα Chatbot είναι ένα πρόγραμμα υπολογιστή υλοποιημένο συνήθως με προδιαγραφές τεχνητής νοημοσύνης εξειδικεύοντας το κατ' αυτόν τον τρόπο σε ένα συγκεκριμένο φάσμα ρόλων – περσόνων, καθιστώντας το πιο ικανό να φέρει εις πέρας το ζητούμενο αποτέλεσμα. Σχεδιασμένο να συμπεριφέρεται όπως ο άνθρωπος κατά την αλληλεπίδραση του με τους τελικούς χρήστες, το chatbot μέσα από ένα κανάλι επικοινωνίας δέχεται σαν είσοδο κείμενα, φωνητικές οδηγίες ή αφή[4], επεξεργάζεται τα δεδομένα εισόδου αντιστοιχίζοντας τα με δεδομένα μέσα από τη βάση γνώσεων του (Knowledge Base) και εξάγει μια πρόταση με την αντίστοιχη μορφή για να συνεχιστεί ο διάλογος[1,2]. Ένα κανάλι επικοινωνίας θα μπορούσε να είναι μια ιστοσελίδα, ένα live chat, το Meta Messenger, η εφαρμογή WhatsApp κ.ο.κ [10].

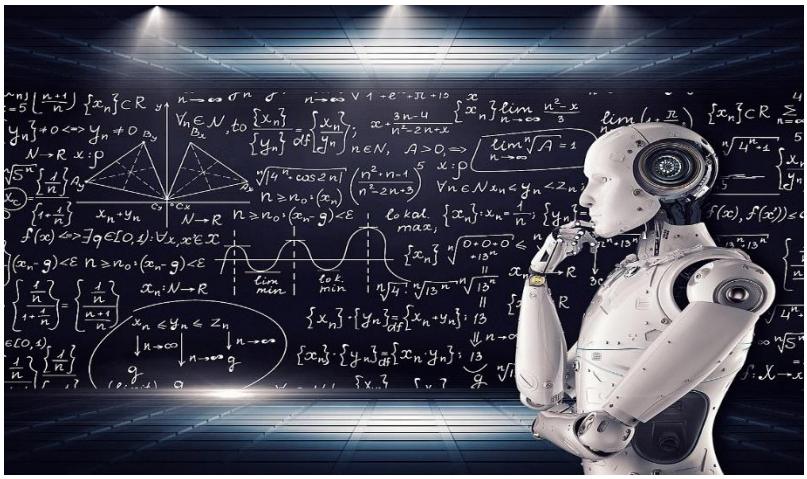
Σήμερα καθώς οι μηχανές ολοένα και μπαίνουν στη ζωή μας, επαγγελματικά ή και μέσα από την καθημερινή μας ρουτίνα ο τρόπος ζωής μας μετασχηματίζεται. Από τη μία μεριά στον επιχειρηματικό τομέα η αξία ενός Chatbot μπορεί να αποδειχθεί σε ένα μεγάλο εύρος βιομηχανιών που κυμαίνεται από υγειονομική περίθαλψη έως τους παραγωγούς ή ακόμα και το λιανικό εμπόριο [2,3] ενώ αντίθετα μέσα από τις καθημερινές μας δραστηριότητες μπορεί κανείς να συναντήσει chatbots όπως η Alexa της Amazon, η Cortana της Microsoft ή η Siri της Apple τα οποία μπορούν να σταθούν ως προσωπικοί βοηθοί (Personal Assistants) [4].

Τα Chatbots έρχονται σήμερα υποσχόμενα ότι θα αλλάξουν ριζικά τον τρόπο που αλληλεπιδρούμε με την τεχνολογίας.

1.2 Βασικές έννοιες

1.2.1 Τεχνητή Νοημοσύνη

Η τεχνητή νοημοσύνη είναι ο τομέας της επιστήμης των υπολογιστών όπου μπορεί κανείς να την περιγράψει ως τη νοημοσύνη των μηχανών. Επιτρέπει στα Chatbots να εκτελούν διεργασίες λήψης αποφάσεων, μετάφραση φυσικής γλώσσας ή και αναγνώριση ομιλίας, διεργασίες που σε διαφορετική περίπτωση θα απαιτούσαν τη φυσική παρουσία της ανθρώπινης νοημοσύνης [10].



Σχήμα 1.1 : Τεχνητή Νοημοσύνη. Source

1.2.2 Επεξεργασία φυσικής γλώσσας

Η φυσική γλώσσα αναφέρεται στον τρόπο που εμείς οι άνθρωποι επικοινωνούμε μεταξύ μας μέσω ομιλίας ή κειμένων.

Ο Επιστημονικός τομέας που μελετά τη γλώσσα συμπεριλαμβανομένης της γραμματικής, της σημασιολογίας και της φωνητικής είναι η γλωσσολογία. Μέσω της κλασικής γλωσσολογίας επινοήθηκαν και αξιολογήθηκαν διάφοροι κανόνες ως προς τη σύνταξη και τη σημασιολογία. Η σύγχρονη μελέτη όμως βασίζεται στην υπολογιστική γλωσσολογία, γνωστή και ως επεξεργασία φυσικής γλώσσας (Natural Language Processing – NLP) αντικατοπτρίζοντας έτσι και την πιο μηχανική προσέγγιση των στατιστικών μεθόδων στις οποίες και βασίζεται χρησιμοποιώντας εργαλεία της επιστήμης των υπολογιστών [11].

Η επεξεργασία φυσικής γλώσσας είναι ένα κομμάτι της τεχνητής νοημοσύνης το οποίο βοηθά τους υπολογιστές στην κατανόηση της ανθρώπινης γλώσσας [10].

1.2.3 Κατανόηση φυσικής γλώσσας

Η κατανόηση της φυσικής γλώσσας (Natural Language Understanding – NLU) είναι μια υποκατηγορία του NLP. Ενώ το NLP αναφέρεται συγκεκριμένα στο φάσμα εργαλείων που χρησιμοποιούν οι μηχανές για την επεξεργασία της ανθρώπινης γλώσσας, το NLU έχει ως απώτερο σκοπό του να κάνει το μηχάνημα να καταλαβαίνει τι πραγματικά σημαίνει το κείμενο που επεξεργάζεται [8].

Η μηχανική κατανόηση ανάγνωσης περιλαμβάνει πράγματα όπως η κατηγοριοποίηση κειμένων και η ανάλυση περιεχομένων [8].

1.2.4 Βάση γνώσεων

Μια συζήτηση με ένα chatbot μπορεί να ξεκινήσει με έναν απλό χαιρετισμό, ωστόσο καθώς αναπτύσσεται ο διάλογος, ο χρήστης μπορεί να περιπλέξει τα πράγματα με ένα σύνολο απαιτητικών ερωτήσεων οι οποίες μπορούν να προκαλέσουν "σύγχυση" στο chatbot. Ερωτήσεις για το πώς μπορεί να συνδεθεί σε μια εφαρμογή μέχρι το πώς να κλείσει ραντεβού στην αμέσως επόμενη διαθέσιμη ημερομηνία. Η λύση σε αυτό το σημείο είναι η βάση γνώσεων.

Μια βάση γνώσεων (Knowledge Base) είναι μια συλλογή καλά δομημένων, αλλά και στοχευμένων πληροφοριών ενός συγκεκριμένου εύρους που είναι απαραίτητες για την υλοποίηση ενός chatbot υποστηριζόμενο από AI. Συγκεκριμένα, ο λόγος ύπαρξης μιας βάσης γνώσεων είναι για να καλύπτεται όλο το εύρος των αναγκών μιας υπηρεσίας. Λειτουργεί σαν εγχειρίδιο εξειδικεύοντας το έτσι πάνω στην υπηρεσία αυτή. Με αυτόν τον τρόπο το chatbot αναλαμβάνει ένα συγκεκριμένο ρόλο και είναι σε θέση να επικοινωνήσει άμεσα με τον τελικό χρήστη [16].

1.2.5 Μηχανική μάθηση

Η μηχανική μάθηση (Machine Learning) είναι ένας κλάδος του AI, εστιάζει στη χρήση δεδομένων και αλγορίθμων για τη μίμηση του τρόπου με τον οποίο μαθαίνουν οι άνθρωποι βελτιώνοντας σταδιακά την ακρίβεια της. Όπως ο άνθρωπος μαθαίνει μέσα από εμπειρίες, έτσι μια μηχανή μπορεί να μάθει μέσα από δεδομένα, σκοπός της μηχανικής μάθησης είναι η εκμάθηση μέσα από δεδομένα και η μετατροπή σε αποφάσεις και προβλέψεις [10].

1.2.6 Βαθιά μάθηση

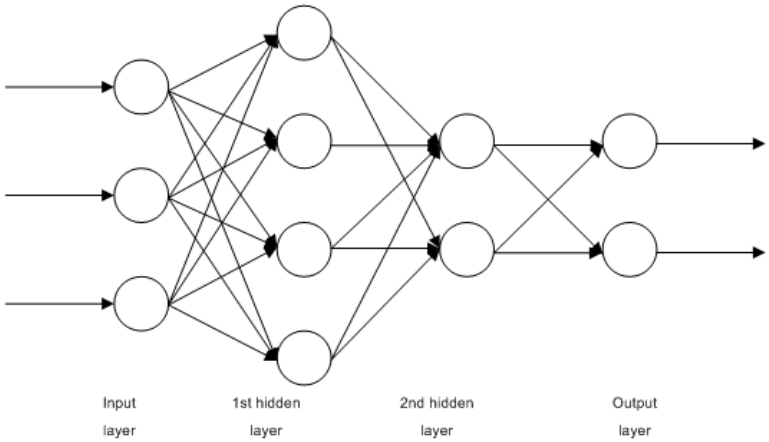
Η βαθιά μάθηση (Deep Learning) είναι ένα υποπεδίο της μηχανικής μάθησης όπου οι αλγόριθμοι της διαφέρουν στον τρόπο εκμάθησης από τους αλγόριθμους της μηχανικής μάθησης. Η βαθιά μάθηση αυτοματοποιεί μεγάλο μέρος του τμήματος εξαγωγής χαρακτηριστικών της διαδικασίας, περιορίζοντας ένα σημαντικό μέρος της χειροκίνητης ανθρώπινης παρέμβασης που θα χρειαζόταν σε διαφορετική περίπτωση κι επιτρέποντας τη χρήση μεγαλύτερων συνόλων δεδομένων [13].

1.2.7 Τεχνητά Νευρωνικά Δίκτυα

Τα τεχνητά νευρωνικά δίκτυα (Artificial Neural Networks) είναι ένα υποπεδίο της Βαθιάς Μάθησης, αποτελούνται από ένα σύνολο νευρώνων όπου η δομή τους χωρίζεται σε τρία στρώματα:

- Στρώμα εισόδου
- Κρυφό στρώμα, το οποίο μπορεί να αποτελείται από πολλά επιμέρους στρώματα ίδιου επιπέδου
- Στρώμα εξόδου

Κάθε τεχνητός νευρώνας συνδέεται με όλους τους νευρώνες του επόμενου στρώματος σχεδιάζοντας έτσι ένα τεχνητό νευρωνικό δίκτυο.



Σχήμα 1.2 : Τεχνητό Νευρωνικό Δίκτυο. Source

1.3 Κατηγορίες Chatbot

Τα συστήματα αυτόματου διαλόγου που γνωρίζουμε σήμερα βασίζονται σε τρείς κατηγορίες, σε γλωσσικά όπου λειτουργούν βάσει κανόνων (Rule - Based),σε Chatbots μηχανικής μάθησης και σε υβριδικά (Hybrid chatbots) [4].

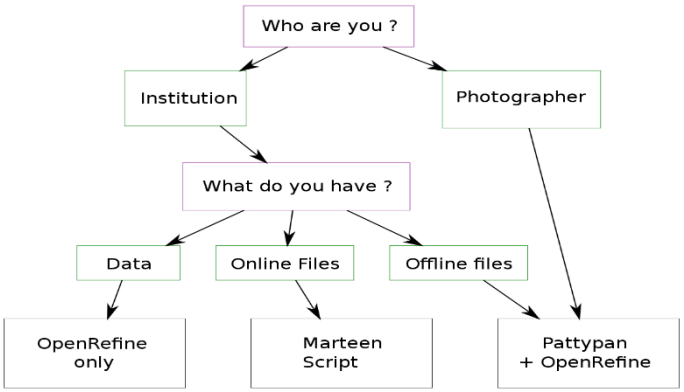
1.3.1 Chatbots βάσει κανόνων (Rule – based)

Η κατηγορία αυτή ανήκει στην πρώτη γενιά των Chatbots, όπου η δυνατότητα των συστημάτων να ανταποκριθούν και να εξυπηρετήσουν τους χρήστες βασίζεται εξ ολοκλήρου σε ένα σύνολο κανόνων οι οποίοι υποστηρίζονται από μια τεχνική αντιστοίχισης μοτίβων. Χρησιμοποιούν γλωσσικές συνθήκες για να επαληθεύσουν εάν τα δεδομένα που εκλαμβάνουν κατά την είσοδο πληρούν τα γλωσσικά κριτήρια τα οποία είναι ζωτικής σημασίας για το σύστημα [4,9].

Οι γλωσσικές συνθήκες βοηθούν στο να γίνουν οι απαραίτητοι έλεγχοι για λέξεις οι οποίες έχουν κοινή σημασία, τη θέση αυτών των λέξεων μέσα σε μία φράση ή και φράσεις ολόκληρες που μπορούν να ειπωθούν με διαφορετικό τρόπο. Με αυτόν τον τρόπο ο μηχανισμός ενός τέτοιου Chatbot μπορεί να διασφαλίσει ότι ερωτήσεις που έχουν το ίδιο νόημα, θα λάβουν την ίδια ακριβώς απάντηση. Στην περίπτωση που παρατηρηθεί κάποιο σφάλμα στην κατανόηση, είναι εφικτό για έναν άνθρωπο να ρυθμίσει λεπτομερώς όλες τις γλωσσικές συνθήκες, κάτι που δεν είναι εφικτό να γίνει σε συστήματα τα οποία βασίζονται σε AI [4,8].

Αυτή η τεχνική περιορίζει το εύρος της συνομιλίας σε μία συγκεκριμένη περίπτωση χρήσης, καθώς ο σχεδιαστής του Chatbot δημιουργεί μια απλή ροή συζήτησης λόγω του ότι θα πρέπει να εξαντλήσει όλες τις πιθανές περιπτώσεις μέσα σε ένα διάλογο [8,9].

Τα bots αυτής της κατηγορίας δεν είναι σε θέση να θυμούνται τις προηγούμενες αλληλεπιδράσεις με τους χρήστες, έχουν τη μορφή ενός δέντρου αποφάσεων το οποίο καθοδηγεί το χρήστη σύμφωνα με τις απαντήσεις που δίνει ο ίδιος [8,9].



Σχήμα 1.3 : Δέντρο αποφάσεων. Source

1.3.2 Chatbots μηχανικής μάθησης

Τα chatbots που υποστηρίζονται από AI είναι περισσότερο πολύπλοκα από αυτά που βασίζονται σε απλούς κανόνες και μπορούν να ανταπεξέλθουν σε μεγαλύτερες και πιο σύνθετες συνομιλίες [4].

Οι τύποι αυτής της κατηγορίας είναι πιο εξελιγμένοι, διαδραστικοί και εξατομικευμένοι γενικότερα από τα chatbots που είναι προσανατολισμένα σε κανόνες. Χρησιμοποιούν ένα στενό φάσμα σεναρίων μέσω στατιστικών αλγορίθμων και δίνουν εντυπωσιακά αποτελέσματα όταν το πρόβλημα που αντιμετωπίζουν ταιριάζει ακριβώς στις δυνατότητες τους, από τη φύση τους μαθαίνουν μέσα από μοτίβα και προηγούμενες εμπειρίες συνομιλιών. Για να καταφέρουν όμως να αποδώσουν έστω και τα στοιχειώδη μέσα σε μια συζήτηση, τα συστήματα αυτά χρησιμοποιούν ένα πολύ μεγάλο σύνολο δεδομένων κατάρτισης κατά την εκπαίδευση τους και εξειδικευμένο ανθρώπινο δυναμικό για τη σχεδίαση [4].

Ωστόσο, ένα τέτοιο chatbot θα μπορούσε να λειτουργήσει και ως μαύρο κουτί. Εάν κάτι πάει στραβά κατά τη σχεδίαση και την υλοποίηση του μοντέλου της μηχανικής μάθησης, τότε το αποτέλεσμα που θα πάρουμε δε θα είναι και το πιο επιθυμητό και το να παρέμβει κανείς για να βελτιστοποιήσει την απόδοση θα είναι αρκετά δύσκολο [4].

1.3.3 Υβριδικά μοντέλα AI

Ενώ τα μοντέλα γλωσσικής και μηχανικής μάθησης αντικατοπτρίζουν συγκεκριμένους τύπους συστημάτων αυτόματου διαλόγου, η υβριδική προσέγγιση προσφέρει τη χρυσή τομή ανάμεσα από αυτές τις δύο κατηγορίες παρέχοντας μας πιο σύνθετες λύσεις στα AI chatbots [4].

Μια υβριδική προσέγγιση έχει περισσότερα, βασικά πλεονεκτήματα έναντι των εναλλακτικών λύσεων. Λαμβάνοντας υπόψη τις μεθόδους της μηχανικής μάθησης μας επιτρέπει να δημιουργήσουμε συστήματα αυτόματου διαλόγου ακόμα και χωρίς τη χρήση δεδομένων, παρέχοντας μια διαφάνεια στον τρόπο λειτουργίας, επιτρέπει στους χρήστες να κατανοήσουν την εφαρμογή και διατηρεί μια σταθερή προσωπικότητα και συμπεριφορά ανταποκρινόμενα έτσι στις επιχειρηματικές προσδοκίες [4].

Με την ενσωμάτωση της μηχανικής μάθησης μπορεί και υπερβαίνει τα επίπεδα των γλωσσικών κανόνων προσδίδοντας έξυπνα και περίπλοκα συμπεράσματα σε τομείς όπου η γλωσσική προσέγγιση είναι δύσκολη ή ακόμη και αδύνατη. Με την υβριδική προσέγγιση σε φυσικό επίπεδο, οι στατιστικοί

αλγόριθμοι μπορούν να ενσωματωθούν παράλληλα με τις με τις γλωσσικές συνθήκες διατηρώντας τους σε κοινή οπτική διεπαφή [4].

Με την υιοθέτηση μιας υβριδικής προσέγγισης από μια επιχείρηση, η επιχείρηση αποκτά τη δύναμη, την ευελιξία και την ταχύτητα καθώς θα είναι σε θέση να αναπτύξει εφαρμογές AI που θα κάνουν τη διαφορά στη διαχείριση και την εμπειρία των πελατών της. Ενώ αντίστοιχα, η λύση που προσφέρουν τα εναλλακτικά μοντέλα είναι δαπανηρά καθώς απαιτούν πόρους και συνήθως η εκμάθηση τους είναι δύσκολη [4].

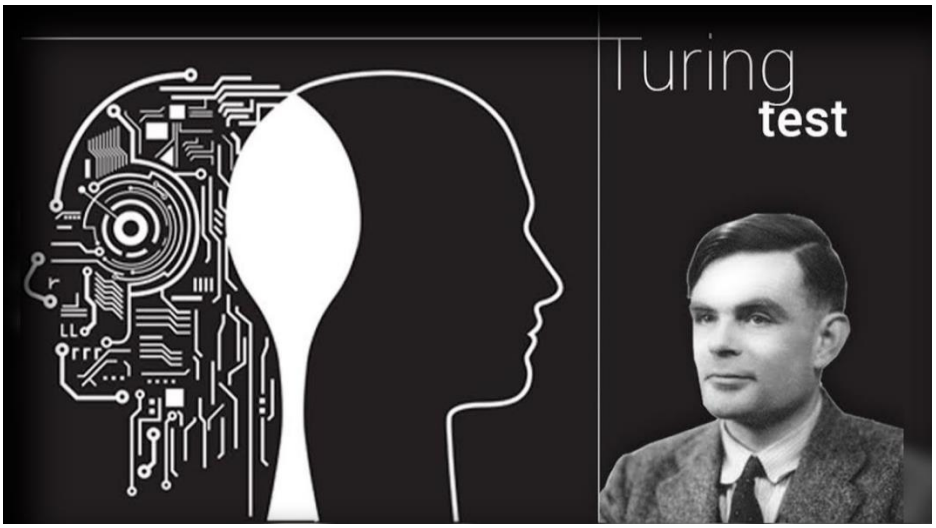
1.4 Ιστορική αναδρομή

1.4.1 Turing Test

Ένας Βρετανός μαθηματικός και επιστήμονας της πληροφορικής ο Alan Turing, δημοσιεύοντας το άρθρο “Computing Machinery and Intelligence” τοποθετεί το ερώτημα: «Μπορούν οι μηχανές να σκέφτονται;» [2].

Θέτοντας ως θεμελιώδη καθοριστικό γνώρισμα της ανθρώπινης νοημοσύνης τη γλώσσα, επινοεί ένα απλό τεστ γνωστό και ως «Το παιχνίδι μίμησης». Το τεστ αυτό στηρίζεται σε μια απλή υπόθεση : Αν ένας άνθρωπος μπορεί να συμμετέχει σε μια συζήτηση για μόλις 5 λεπτά χωρίς να καταλάβει ότι μιλάει σε μια μηχανή, τότε η μηχανή αυτή περνάει το τεστ [5,6].

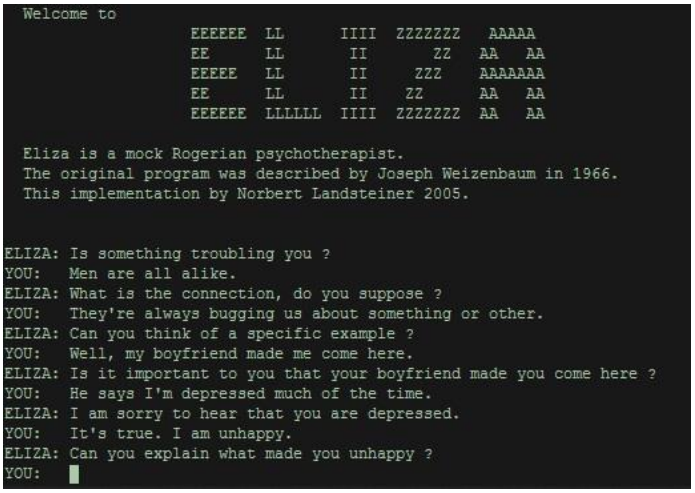
Το τεστ αυτό που εφευρέθηκε το 1950 από τον πρωτοπόρο επιστήμονα Turing, σκεπτόμενος για το αν μια μηχανή μπορεί να μιμηθεί την ανθρώπινη σκέψη [7], θέτει τις βάσεις της τεχνητής νοημοσύνης και καλεί το σύνολο των επιστημόνων και μηχανικών να εμπνευστούν τις απανταχού ομάδες σχεδίασης AI και να προσπαθήσουν να υλοποιήσουν υπολογιστικά συστήματα με διεπαφές χρήστη που τείνουν να αλληλεπιδρούν με πιο φυσικό τρόπο προς τον άνθρωπο [2,12].



Σχήμα 1.4 : Turing test. Source

1.4.2 ELIZA – Το πρώτο Chatbot

Το 1966 έρχεται το πρώτο chatbot από το πανεπιστήμιο του MIT με το όνομα ELIZA. Δημιουργήθηκε από τον Joseph Weizenbaum και χρησιμοποιεί μια μεθοδολογία αντιστοίχισης και αντικατάστασης προτύπων για την προσομοίωση συνομιλίας. Το chatbot ELIZA λειτουργεί περνώντας τις λέξεις των χρηστών σε έναν υπολογιστή και συνδέοντας τις με μια λίστα πιθανών απαντήσεων χρησιμοποιώντας ένα σενάριο που προσομοιώνει έναν ψυχοθεραπευτή [2,14].



Σχήμα 1.5 : ELIZA το πρώτο Chatbot. Source

1.4.3 Από την ELIZA μέχρι σήμερα

Μέσα στις επόμενες δεκαετίες, επιστήμονες και κατασκευαστές των Chatbots βασίζονται στο μοντέλο του Weizenbaum για να καταφέρουν να υλοποιήσουν πιο ευφυή πρότυπα όπου τείνουν να μοιάζουν περισσότερο με τον άνθρωπο [15].

Το **PARRY** έρχεται το 1972 από τον Αμερικανό ψυχίατρο Kenneth Colby, το πρόγραμμα προσπαθεί να προσομοιώσει την ασθένεια της σχιζοφρένειας [15].

Ο **Dr. Sbaitso** είναι ένα chatbot για το MS-Dos όπου δημιουργήθηκε το 1992 από την Creative Labs, είναι μια από τις πρώτες προσπάθειες να ενσωματωθεί AI και αναγνωρίζεται για το φωνητικό πρόγραμμα συνομιλίας του. Το πρόγραμμα συνομιλούσε με το χρήστη σαν να ήταν ψυχολόγος [15].

A.L.I.C.E, είναι ένα γενικό chatbot επεξεργασίας γλώσσας που χρησιμοποιεί ευρετική αντιστοίχιση προτύπων. Κάνει την εμφάνιση του το 1995 από τον Richard Wallace και λειτουργεί με το σχήμα XML γνωστό ως γλώσσα σήμανσης τεχνητής νοημοσύνης (AIML) [15].

Ευρέως γνωστή σήμερα η **Alexa,** είναι μία έξυπνη προσωπική βοηθός που αναπτύχθηκε από την Amazon και παρουσιάστηκε το 2014. Σήμερα έχει καταφέρει να ενσωματωθεί σε συσκευές όπως το Amazon Echo, το Echo Dot, το Echo Show και άλλα. Επίσης η Amazon παρέχει ένα API όπου κι επιτρέπει σε προγραμματιστές να ενσωματώσουν την Alexa σε εξωτερικές συσκευές [15].

1.5 Επίλογος

Έγινε μια αναφορά σε ορισμένες από τις βασικές έννοιες τις οποίες θα συναντήσουμε στην παρούσα Π.Ε, έπειτα αναλύθηκαν οι κατηγορίες στις οποίες κατατάσσονται τα Chatbots που κυριαρχούν στο χώρο σήμερα. Και τέλος αναφέρθηκε μια σύντομη ιστορική αναδρομή για την εξέλιξη και την πορεία που έχουν πάρει οι εφαρμογές αυτές από το πρώτο chatbot που υλοποιήθηκε μέχρι τα σημερινά chatbots.

Κεφάλαιο 2ο: Natural Language Processing

2.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα γίνει μια σύντομη αναφορά σε δύο από τις πιο γνωστές βιβλιοθήκες που χρησιμοποιούνται σήμερα για την επεξεργασία του φυσικού λόγου, έπειτα θα γίνει μια σύγκριση των δύο βιβλιοθηκών και θα αναφερθεί ο λόγος για τον οποίο επιλέχθηκε η μία από αυτές για την υλοποίηση της εφαρμογής. Στη συνέχεια θα αναλυθούν τρόποι και τεχνικές υλοποίησης των μεθόδων του NLP με τους αλγόριθμους που χρησιμοποιεί η βιβλιοθήκη αυτή.

2.2 Βασική ορολογία NLP

Ο όρος επεξεργασία φυσικής γλώσσας περιλαμβάνει ένα ευρύ σύνολο τεχνικών για αυτοματοποιημένη παραγωγή, χειρισμό και ανάλυση φυσικών γλωσσών. Αν και οι περισσότερες τεχνικές κληρονομούν σε μεγάλο βαθμό από τη γλωσσολογία και την τεχνητή νοημοσύνη, επηρεάζονται επίσης από τη μηχανική μάθηση, την υπολογιστική στατιστική και τη γνωστική επιστήμη [17,18].

Πριν δούμε μερικά παραδείγματα τεχνικών NLP παρακάτω, είναι χρήσιμο να εισάγουμε κάποια βασική ορολογία [17,18].

- **Clean** : Η διαδικασία καθαρισμού στοχεύει στην αφαίρεση άσχετων στοιχείων από τα δεδομένα.
- **Normalization** : Η κανονικοποίηση μετατρέπει ολόκληρο το κείμενο με πεζά γράμματα και αφαιρεί όλα τα σημεία στίξης.
- **Token** : Πριν μπορέσει να γίνει οποιαδήποτε πραγματική επεξεργασία στο κείμενο εισόδου, αυτό πρέπει να κατατμηθεί σε γλωσσικές μονάδες όπως λέξεις, σημεία στίξης, αριθμοί ή αλφαριθμητικά. Αυτές οι μονάδες είναι γνωστές ως tokens.
- **Sentence** : Μια διατεταγμένη ακολουθία από tokens (πρόταση).
- **Tokenization** : Η διαδικασία του διαχωρισμού μιας πρότασης σε tokens. Για τμηματικές γλώσσες όπως είναι τα ελληνικά ή τα αγγλικά η ύπαρξη του κενού διαστήματος σε μια πρόταση κάνει το tokenization σχετικά πιο εύκολο και χωρίς ιδιαίτερο ενδιαφέρον. Ωστόσο σε γλώσσες όπως τα αραβικά ή τα κινέζικα η διαδικασία είναι δυσκολότερη καθώς δεν υπάρχουν σαφή όρια. Επιπλέον, σχεδόν όλοι οι χαρακτήρες σε τέτοιες μη τμηματοποιημένες γλώσσες μπορούν να υπάρχουν ως λέξεις με ένα χαρακτήρα ή να ενωθούν για να σχηματίσουν λέξεις με πολλούς χαρακτήρες.
- **Stop words** : Είναι ένα σύνολο λέξεων χωρίς κάποια ιδιαίτερη σημασία για τις εφαρμογές του NLP και χρησιμοποιούνται για τον καθαρισμό των κειμένων από αυτές.
- **Text corpus** : Ένα σώμα κειμένου, που συνήθως περιέχει μεγάλο αριθμό προτάσεων.
- **Part of Speech (POS) Tag** : Μια λέξη «ετικέτα» για κάθε token η οποία μπορεί να ταξινομηθεί σε μία ή περισσότερες, ενός συνόλου λεξιλογικών ή τμηματικών κατηγοριών όπως ουσιαστικά, ρήματα, επίθετα και άρθρα.
- **Named entity Recognition (NER)** : Είναι μια εργασία εξαγωγής πληροφοριών με σκοπό τον εντοπισμό και την ταξινόμηση ατομικών στοιχείων από το κείμενο σε προκαθορισμένες

κατηγορίες, όπως ονόματα προσώπων, οργανισμών, τοποθεσιών, ποσοτήτων, εκφράσεων χρόνου κλπ.

- **Syntax Tree** : Ένα δέντρο που ορίζεται σε μια δεδομένη πρόταση όπου αντιπροσωπεύει τη συντακτική δομή της όπως ορίζεται από μια επίσημη γραμματική.

2.3 Natural Language Toolkit – NLTK

2.3.1 Εισαγωγή στο NLTK

Το NLTK είναι μια πλατφόρμα για τη δημιουργία python προγραμμάτων τα οποία λειτουργούν με δεδομένα της ανθρώπινης γλώσσας [17].

Χάρη σε έναν πρακτικό οδηγό που εισάγει τις βασικές αρχές του προγραμματισμού παράλληλα με θέματα της υπολογιστικής γλωσσολογίας, καθώς και με την ολοκληρωμένη τεκμηρίωση του API, το NLTK είναι κατάλληλο για γλωσσολόγους, μηχανικούς, φοιτητές, εκπαιδευτικούς, ερευνητές και χρήστες του κλάδου [17].

Το NLTK είναι διαθέσιμο για Windows, Mac OS X καθώς και για Linux. Η πλατφόρμα διανέμεται δωρεάν καθώς είναι ένα έργο ανοιχτού κώδικα και βασίζεται στη μεγάλη κοινότητα που το περικλείει. Δημιουργήθηκε το 2001 ως μέρος ενός μαθήματος υπολογιστικής γλωσσολογίας στο τμήμα επιστήμης υπολογιστών και πληροφορικής της Πενσυλβάνια. Έκτοτε έχει αναπτυχθεί και επεκταθεί με τη βοήθεια πολλών συντελεστών και πλέον χρησιμοποιείται σε μαθήματα πολλών πανεπιστημίων ως βάση ερευνητικών προγραμμάτων. Η σημερινή έκδοση της πλατφόρμας είναι η 3.6.7. η οποία δημοσιεύτηκε τον Δεκέμβριο του 2021 και βρίσκεται υπό συνεχή ανάπτυξη καθώς προστίθενται διαρκώς νέα modules και βελτιώνονται τα ήδη υπάρχοντα [17].

2.3.2 Τεχνικά χαρακτηριστικά

Το NLTK ορίζει μια υποδομή που μπορεί να χρησιμοποιηθεί για τη δημιουργία προγραμμάτων NLP με την Python. Παρέχει στους χρήστες βασικές τάξεις για την αναπαράσταση δεδομένων που σχετίζονται με το NLP, εύχρηστες διεπαφές για την εκτέλεση εργασιών όπως η προσθήκη ετικετών μέρους του λόγου, η συντακτική ανάλυση και η ταξινόμηση κειμένου, χαρακτηριστικές μεθόδους για την επεξεργασία κειμένου όπως tokenization, stemming, tagging, parsing, semantic reasoning και wrappers καθώς επίσης και εύχρηστες υλοποιήσεις για κάθε εργασία που μπορούν να συνδυαστούν για την επίλυση σύνθετων προβλημάτων [17].

Το NLTK σχεδιάστηκε με τέσσερις πρωταρχικούς στόχους.

- Την παροχή ενός διαισθητικού πλαισίου μαζί με σημαντικά δομικά στοιχεία, παρέχοντας στους χρήστες πρακτική γνώση του NLP με **απλότητα**.
- Με γνώμονα τη **συνέπεια** στην παροχή των διεπαφών και των δομών δεδομένων του πλαισίου με εύκολα και προβλέψιμα ονόματα στις μεθόδους.
- Την **επεκτασιμότητα** της δομής στην οποία μπορούν να ενσωματωθούν εύκολα νέες ενότητες λογισμικού συμπεριλαμβανομένων εναλλακτικών υλοποιήσεων και ανταγωνιστικών προσεγγίσεων για την ίδια εργασία.

- Και το **modularity**, όπου παρέχεται η δυνατότητα να χρησιμοποιηθούν πολλά ξεχωριστά modules χωρίς την απαραίτητη προϋπόθεση της κατανόησης ολόκληρης της εργαλειοθήκης [17].

2.4 SpaCy

2.4.1 Εισαγωγή στο spaCy

Το SpaCy είναι μια δωρεάν βιβλιοθήκη ανοιχτού κώδικα για προηγμένη επεξεργασία φυσικής γλώσσας στην Python [18].

Έχει σχεδιαστεί ειδικά για παραγωγική χρήση και βοηθάει τον προγραμματιστή να δημιουργεί εφαρμογές που επεξεργάζονται και «κατανοούν» μεγάλους όγκους κειμένων. Μπορεί να χρησιμοποιηθεί για τη δημιουργία συστημάτων εξαγωγής πληροφοριών, την κατανόηση της φυσικής γλώσσας ή για προεπεξεργασία κειμένου για βαθιά μάθηση [18].

Η βιβλιοθήκη είναι γραμμένη στη γλώσσα προγραμματισμού C, με δεσμεύσεις γλώσσας για Python. Η πρώτη έκδοση του SpaCy κυκλοφόρησε το 2015 περιλαμβάνοντας συγκεκριμένα χαρακτηριστικά επεξεργασίας κειμένου ενώ μέχρι σήμερα έχει καταφέρει να γίνει ένα από τα καλύτερα εργαλεία όσον αφορά το NLP για τους προγραμματιστές, τους φοιτητές καθώς και όλο το κοινό που το ενδιαφέρει η επεξεργασία φυσικού κειμένου. Η σημερινή έκδοση της βιβλιοθήκης είναι η v3.2 [18].

2.4.2 Τεχνικά χαρακτηριστικά

Το SpaCy είναι μια προηγμένη βιβλιοθήκη για την υλοποίηση τεχνικών NLP. Παρέχει γρήγορες και ακριβείς συντακτικές αναλύσεις καθώς υποστηρίζει μεγάλα διανύσματα λέξεων τα οποία καθιστούν τη διαδικασία γρηγορότερη και πιο αποτελεσματική [18].

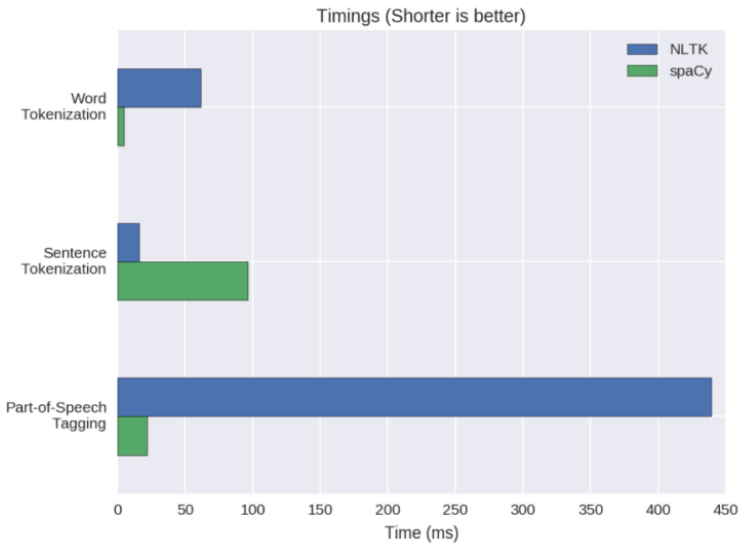
Διαθέτει μεθόδους για την επεξεργασία κειμένων, named entity recognition, υποστήριξη για περισσότερες από 60 ανθρώπινες γλώσσες, τουλάχιστον 46 στατιστικά μοντέλα για 16 γλώσσες, προεκπαιδευμένα word vectors, ενώ χαρακτηριστικό γνώρισμα του είναι η εύκολη ενσωμάτωση του σε δίκτυα βαθιάς μάθησης [18].

Διαθέτει επίσης πολλές ανεξάρτητες λειτουργίες όπως το tokenization, το dependency parsing, entity linking, similarity και πολλά ακόμη. Καθώς όμως κάποιες από τις λειτουργίες του είναι ανεξάρτητες, υπάρχουν κάποια κομμάτια τα οποία προϋποθέτουν τη φόρτωση εκπαιδευμένων pipelines που επιτρέπουν στο SpaCy να προβλέπει γλωσσικούς σχολιασμούς. Ένα εκπαιδευμένο pipeline μπορεί να αποτελείται από πολλαπλά στοιχεία που χρησιμοποιούν ένα στατιστικό μοντέλο εκπαιδευμένο σε δεδομένα με ετικέτα. Τα πακέτα των pipelines μπορεί να διαφέρουν ως προς το μέγεθος, την ταχύτητα στη χρήση μνήμης, την ακρίβεια και τα δεδομένα που περιλαμβάνουν [18].

2.5 Σύγκριση SpaCy – NLTK

Καθώς το SpaCy και το NLTK είναι δύο από τα πιο σημαντικά εργαλεία που χρησιμοποιούνται σήμερα για το NLP, υπάρχουν ουσιαστικές διαφορές μεταξύ τους οι οποίες είναι οι εξής [18-20] :

- Το NLTK παρέχει μια πληθώρα αλγορίθμων που μπορεί να διαλέξει ο χρήστης για ένα συγκεκριμένο πρόβλημα, κάτι που είναι αρκετά χρήσιμο για έναν ερευνητή αλλά κακό για έναν προγραμματιστή. Ενώ αντίθετα το spaCy διατηρεί τον καλύτερο αλγόριθμο για ένα πρόβλημα και τον ενημερώνει διαρκώς καθώς εξελίσσεται η τεχνολογία [19].
- Το NLTK υποστηρίζει διάφορες γλώσσες ενώ το spaCy διαθέτει στατιστικά μοντέλα για τουλάχιστον 16 γλώσσες καθώς επίσης και named entities για τις γλώσσες αυτές [19].
- Το NLTK είναι μια βιβλιοθήκη για την επεξεργασία συμβολοσειρών. Δέχεται σαν είσοδο συμβολοσειρές και επιστρέφει στην έξοδο συμβολοσειρές ή λίστες ολόκληρες συμβολοσειρών. Το spaCy από την άλλη, χρησιμοποιεί αντικειμενοστραφή προσέγγιση. Κατά την ανάλυση ενός κειμένου, το SpaCy επιστρέφει ένα document object του οποίου οι λέξεις και οι προτάσεις είναι από μόνες τους objects [19].
- Το spaCy υποστηρίζει διανύσματα λέξεων ενώ το NLTK όχι [18,19].
- Η απόδοση του spaCy σε σύγκριση με το NLTK είναι καλύτερη καθώς το spaCy χρησιμοποιεί πιο πρόσφατους και εξελιγμένους αλγόριθμους. Στην εικόνα που ακολουθεί μπορούμε να διαπιστώσουμε ότι, κατά τη μέθοδο του word tokenization και στην προσθήκη ετικετών POS, το spaCy αποδίδει καλύτερα ενώ στο tokenization ολόκληρων προτάσεων το NLTK υπερτερεί του spaCy. Αυτό οφείλεται στις διαφορετικές προσεγγίσεις των δύο εργαλείων. Ενώ το NLTK προσπαθεί να χωρίσει το κείμενο σε προτάσεις, το spaCy δημιουργεί ένα συντακτικό δέντρο για κάθε πρόταση στο οποίο παρέχει περισσότερη πληροφορία για το κείμενο [18-20].



Σχήμα 2.1 : Χρόνοι απόκρισης spaCy – NLTK. [19]

Με γνώμονα τα στατιστικά μοντέλα, τα διανύσματα λέξεων καθώς και την ευχρηστία που παρέχει το spaCy, επιλέχθηκε για την υλοποίηση της εφαρμογής την οποία θα αναλύσουμε αργότερα.

2.6 Ανάλυση τεχνικών με το spaCy

2.6.1 Language model

Καθώς το ATIS dataset που θα δούμε παρακάτω, περιέχει τα δεδομένα του στην αγγλική γλώσσα θα χρειαστεί να χρησιμοποιήσουμε ένα από τα μοντέλα της αγγλικής γλώσσας που παρέχει το spaCy.

Το spaCy διαθέτει τέσσερα μοντέλα τα οποία διαφέρουν στο μέγεθος των λεξικών που χρησιμοποιούν και των components τους. Για την παρούσα εργασία επιλέχθηκε το **en_core_web_lg** model το οποίο είναι ένα προεκπαιδευμένο μοντέλο και διαθέτει 685.000 μοναδικές λέξεις – κλειδιά ή αλλιώς μοναδικά διανύσματα (word vectors) μεγέθους 300 διαστάσεων [18].

Για να χρησιμοποιήσουμε το μοντέλο αυτό η διαδικασία είναι απλή. Εισάγουμε τη βιβλιοθήκη του spaCy και φορτώνουμε το μοντέλο όπως φαίνεται παρακάτω.

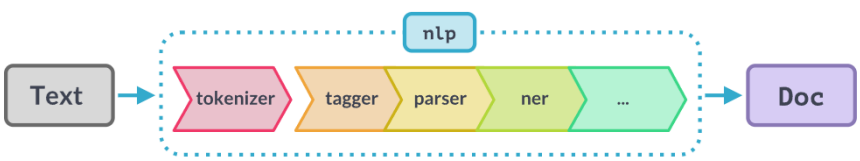
```
import spacy
nlp = spacy.load("en_core_web_lg")

doc = nlp("Apple is looking at buying U.K. startup for $1 billion")
```

Σχήμα 2.2 : Εισαγωγή spaCy [18].

Μέσω του nlp object μπορούμε πλέον να δημιουργούμε document objects τα οποία είναι αποτελέσματα μιας σειράς ενεργειών που ακολουθεί το μοντέλο μέσω του pipeline του (ή αλλιώς processing pipeline).

2.6.2 Pipeline

Το processing pipeline του spaCy το οποίο χρησιμοποιείται από τα εκπαιδευμένα μοντέλα τυπικά διαθέτει τα εξής : tagger, lemmatizer, parser, και entity recognizer. Κάθε component επεξεργάζεται σειριακά το εισαγόμενο κείμενο δημιουργώντας έτσι ένα document object στο οποίο προσθέτει τα δικά του χαρακτηριστικά και στη συνέχεια το περνάει στο επόμενο [18].



Σχήμα 2.3 : Spacy pipeline [18].

Οι δυνατότητες του processing pipeline εξαρτώνται πάντα από τα components, τα μοντέλα τους και τον τρόπο εκπαίδευσης [18].

2.6.3 Tokenization

Κατά τη διάρκεια της επεξεργασίας, το spaCy πρώτα διαμορφώνει και διαχωρίζει το κείμενο σε λέξεις, σημεία στίξης, προτάσεις κλπ εφαρμόζοντας τους ειδικούς κανόνες της αντίστοιχης γλώσσας. Κάθε document object που δημιουργείται από την επεξεργασία αποτελείται από μεμονωμένα tokens τα οποία μπορούμε να προσπελάσουμε [18]. Σύμφωνα με το κείμενο του σχήματος 2.2 το αποτέλεσμα είναι το εξής :

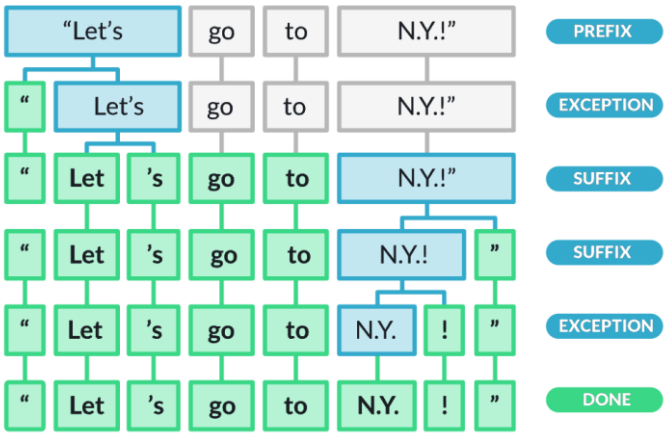
0	1	2	3	4	5	6	7	8	9	10
Apple	is	looking	at	buying	U.K.	startup	for	$	1	billion

Σχήμα 2.4 : Αποτέλεσμα tokenizer [18].

Το ακατέργαστο κείμενο στην αρχή χωρίζεται στους χαρακτήρες του κενού διαστήματος, με παρόμοιο τρόπο όπως το **text.split(‘ ’)** [18]. Στη συνέχεια ο tokenizer επεξεργάζεται το κείμενο από αριστερά προς τα δεξιά και σε κάθε token εκτελεί δύο ξεχωριστούς ελέγχους :

1. Αν το token ταιριάζει με έναν από τους κανόνες εξαίρεσης του, για παράδειγμα το «don’t» δεν περιέχει κενό αλλά θα πρέπει να χωριστεί σε δύο tokens, το «do» και το «n’t». Ενώ το «U.K» θα πρέπει να παραμένει πάντα ένα token [18].
2. Αν είναι εφικτό να διαχωριστεί ένα πρόθεμα, ένα επικάλυμμα ή ένα επίθεμα, για παράδειγμα σημεία στίξης όπως το κόμμα, οι τελείες, οι παύλες και τα εισαγωγικά [18].

Εάν βρεθεί αντιστοιχία με κάποιον κανόνα από τους παραπάνω τότε ο tokenizer τον εφαρμόζει και συνεχίζει το βρόγχο του ξεκινώντας με τα πρόσφατα διαχωρισμένα tokens. Με αυτόν τον τρόπο το spaCy μπορεί να χωρίσει σύνθετα και ένθετα tokens [18].



Σχήμα 2.5 : Παράδειγμα εφαρμογής κανόνων [18].

2.6.4 Part of Speech

Το spaCy μπορεί να αναλύσει και να προσθέσει ετικέτες μέσα στο document object. Σε αυτό το σημείο αναλαμβάνει το εκπαιδευμένο μοντέλο με το pipeline του και τα στατιστικά μοντέλα του, τα οποία επιτρέπουν στο spaCy να κάνει προβλέψεις σχετικά με την ετικέτα που εφαρμόζει καλύτερα στο συγκεκριμένο content [18].

Ένα εκπαιδευμένο component περιλαμβάνει δυαδικά δεδομένα που παράγονται δείχνοντας σε ένα σύστημα αρκετά παραδείγματα ώστε να κάνει προβλέψεις που γενικεύονται σε όλη τη γλώσσα. Για παράδειγμα μια λέξη που ακολουθεί το «the» στα αγγλικά είναι κατά πάσα πιθανότητα ουσιαστικό [18].

Οι γλωσσικοί σχολιασμοί με το spaCy διατίθενται ως χαρακτηριστικά των tokens. Επίσης, κωδικοποιεί όλες τις συμβολοσειρές σε hash values για τη μείωση της χρήσης της μνήμης και τη βελτίωση της αποτελεσματικότητας του [18].

Για να πάρουμε την αναγνώσιμη τιμή μιας συμβολοσειράς ενός χαρακτηριστικού χρειάζεται να προσθέσουμε ένα underscore (_) στο όνομα του [18] :

```
for token in doc:
    print(token.text, token.lemma_, token.pos_, token.tag_, token.dep_,
            token.shape_, token.is_alpha, token.is_stop)
```

Σχήμα 2.6 : παράδειγμα κώδικα [18].

Το αποτέλεσμα της προσπέλασης σύμφωνα με το document object του σχήματος 2.2 θα είναι της παρακάτω μορφής :

TEXT	LEMMA	POS	TAG	DEP	SHAPE	ALPHA	STOP
Apple	apple	PROPN	NNP	nsubj	Xxxxx	True	False
is	be	AUX	VBZ	aux	xx	True	True
looking	look	VERB	VBG	ROOT	xxxx	True	False
at	at	ADP	IN	prep	xx	True	True
buying	buy	VERB	VBG	pcomp	xxxx	True	False
U.K.	u.k.	PROPN	NNP	compound	X.X.	False	False
startup	startup	NOUN	NN	dobj	xxxx	True	False
for	for	ADP	IN	prep	xxx	True	True
$	$	SYM	$	quantmod	$	False	False
1	1	NUM	CD	compound	d	False	False
billion	billion	NUM	CD	pobj	xxxx	True	False

Σχήμα 2.7 : Χαρακτηριστικά labels [18].

2.6.5 Named Entities Recognition

Το spaCy έχει τη δυνατότητα να αναγνωρίζει διάφορους τύπους ονομασμένων οντοτήτων μέσα σε ένα document, ζητώντας από το μοντέλο μια πρόβλεψη. Τα named entities είναι διαθέσιμα μέσω της ιδιότητας **ents** των documents [18].

```
for ent in doc.ents:
    print(ent.text, ent.start_char, ent.end_char, ent.label_)
```

Σχήμα 2.8 : παράδειγμα κώδικα [18].

Το αποτέλεσμα που θα παίρναμε από τον παραπάνω κώδικα φαίνεται στο ακόλουθο σχήμα:

TEXT	START	END	LABEL	DESCRIPTION
Apple	0	5	ORG	Companies, agencies, institutions.
U.K.	27	31	GPE	Geopolitical entity, i.e. countries, cities, states.
$1 billion	44	54	MONEY	Monetary values, including unit.

Σχήμα 2.9 : Named Entities [18].

2.6.6 Word Vectors και ομοιότητες

Η ομοιότητα στο spaCy προσδιορίζεται συγκρίνοντας word vectors ή word embeddings, αναπαραστάσεις πολυδιάστατης σημασίας μιας λέξεως [18]. Τα word vectors δημιουργούνται με έναν αλγόριθμο παρόμοιο του word2vec.

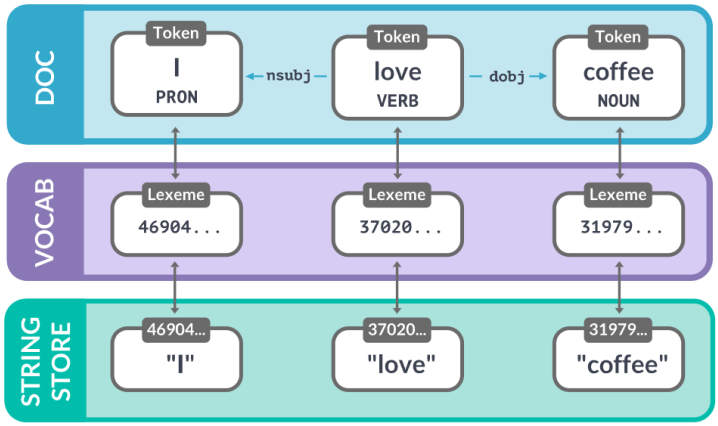
Το spaCy είναι σε θέση να συγκρίνει δύο αντικείμενα και να κάνει μια πρόβλεψη για το πόσο όμοια είναι. Η πρόβλεψη της ομοιότητας είναι χρήσιμη για τη δημιουργία συστημάτων προτάσεων ή την επισήμανση διπλότυπων [18].

2.6.7 Vocab, hashes και lexemes

Το spaCy έχει τη δυνατότητα να αποθηκεύει δεδομένα σε ένα λεξιλόγιο όποτε είναι δυνατό γι’ αυτό, καθώς το Vocab κοινοποιείται μέσα σε πολλά έγγραφα [18].

Για εξοικονόμηση μνήμης, το spaCy κωδικοποιεί και όλες τις συμβολοσειρές σε hashes, για παράδειγμα σε αυτήν την περίπτωση η λέξη «coffee» έχει το hash 3197928453018144401.

Με την ίδια λογική κωδικοποιεί και τις ετικέτες των οντοτήτων καθώς και του part of speech, εσωτερικά το spaCy επικοινωνεί μόνο με hashes [18].



Σχήμα 2.10 : Εσωτερικό σχήμα του String Store [18].

Οι λέξεις αποθηκεύονται εσωτερικά στο StringStore το οποίο μοιάζει με ένα look up table που λειτουργεί προς δύο κατευθύνσεις, για παράδειγμα, μπορούμε να αναζητήσουμε μια συγκεκριμένη λέξη για να πάρουμε το hash της ή διαφορετικά να αναζητήσουμε με βάση το hash για να πάρουμε τη λέξη [18].

```
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("I love coffee")
print(doc.vocab.strings["coffee"])  # 3197928453018144401
print(doc.vocab.strings[3197928453018144401])  # 'coffee'
```

Σχήμα 2.11 : Αναζήτηση στο StringStore [18].

2.6.8 Matchers

Οι Matchers μας επιτρέπουν να εξάγουμε λέξεις ή φράσεις μέσα από το εισαγόμενο κείμενο χρησιμοποιώντας κάποιους κανόνες που περιγράφουν τα χαρακτηριστικά των tokens τους [18].

Οι κανόνες μπορούν να αναφέρονται στους σχολιασμούς των tokens, για παράδειγμα το κείμενο ή οι ετικέτες του POS καθώς και το αν είναι σημεία στίξης. Εφαρμόζοντας ένα Matcher σε ένα document object μας δίνεται η πρόσβαση στα αντιστοιχισμένα tokens [18].

Στην παρακάτω εικόνα βλέπουμε ένα χαρακτηριστικό παράδειγμα ενός αντικειμένου Matcher.

```python
import spacy
from spacy.matcher import Matcher

nlp = spacy.load("en_core_web_lg")
matcher = Matcher(nlp.vocab)
# Add match ID "HelloWorld" with no callback and one pattern
pattern = [{"LOWER": "hello"}, {"IS_PUNCT": True}, {"LOWER": "world"}]
matcher.add("HelloWorld", [pattern])

doc = nlp("Hello, world! Hello world!")
matches = matcher(doc)
for match_id, start, end in matches:
    string_id = nlp.vocab.strings[match_id]  # Get string representation
    span = doc[start:end]  # The matched span
    print(match_id, string_id, start, end, span.text)
```

Σχήμα 2.12 : Υλοποίηση και εφαρμογή ενός Matcher [18].

Υλοποιούμε στην αρχή ένα pattern για το οποίο θέλουμε να εξάγουμε δεδομένα και το εφαρμόζουμε στο document. Στο συγκεκριμένο παράδειγμα το pattern ψάχνει tokens για τα οποία η ακόλουθη σειρά πρέπει να ισοδυναμεί με τη λέξη «hello» στη συνέχεια ένα σημείο στίξης (IS_PUNCT) και τη λέξη «world».

Το αποτέλεσμα του παραπάνω παραδείγματος είναι το εξής :

```
15578876784678163569 HelloWorld 0 3 Hello, world
```

Σχήμα 2.13 : Αποτέλεσμα παραδείγματος σχήματος 2.13 [18].

Όπου :

- Το match_id είναι το hash value του string ID “HelloWorld”.
- Το HelloWorld αναφέρεται στο όνομα του matcher που αντιστοιχεί το match.
- Οι αριθμοί 0-3 αναφέρονται στο εύρος μεταξύ της πρώτης και της τελευταίας θέσης των tokens μέσα στο document.
- Και η φράση “Hello, world” αντιστοιχεί στο κείμενο με το οποίο έγινε η αντιστοίχιση.

2.7 Επίλογος

Σε αυτό το κεφάλαιο είδαμε δύο από τις πιο γνωστές βιβλιοθήκες που χρησιμοποιούνται σήμερα σε προβλήματα NLP, έγινε μια σύγκριση μεταξύ αυτών των βιβλιοθηκών κι έπειτα μια ανάλυση του spaCy για τις τεχνικές που ακολουθεί σε αυτά τα προβλήματα καθώς είναι η βιβλιοθήκη που επιλέχθηκε για την υλοποίηση της παρούσας Π.Ε.

Κεφάλαιο 3ο: Μηχανική Μάθηση

3.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα αναφερθούν οι βιβλιοθήκες που χρησιμοποιήθηκαν για τα μοντέλα που παρέχουν, στατιστικοί αλγόριθμοι, μέθοδοι και τεχνικές καθώς και μοντέλα αλγορίθμων βαθιάς μάθησης που μελετήθηκαν κατά την έρευνα για την υλοποίηση της εφαρμογής.

3.2 Βιβλιοθήκες

Για την υλοποίηση και τη σχεδίαση της εφαρμογής, μελετήθηκαν και χρησιμοποιήθηκαν μοντέλα και αλγόριθμοι από δύο βιβλιοθήκες που είναι ευρέως γνωστές:

- **Scikit learn** : Η Scikit-learn είναι μια από τις πιο χρήσιμες βιβλιοθήκες για μηχανική μάθηση με Python. Περιέχει πολλά αποτελεσματικά εργαλεία για μηχανική μάθηση και στατιστική μοντελοποίηση, όπως ταξινόμηση, παλινδρόμηση, ομαδοποίηση και μείωση διαστάσεων [21].
- **TensorFlow** : Το TensorFlow είναι μια end-to-end πλατφόρμα ανοιχτού κώδικα για μηχανική μάθηση. Διαθέτει ένα ολοκληρωμένο, ευέλικτο οικοσύστημα εργαλείων, βιβλιοθηκών και κοινοτικών πόρων που επιτρέπει στους ερευνητές να προωθήσουν την τελευταία λέξη της τεχνολογίας στη μηχανική μάθηση και στους προγραμματιστές να δημιουργούν και να αναπτύσσουν εύκολα εφαρμογές που υποστηρίζονται από μηχανική μάθηση [22].

3.3 Βασική ορολογία

Οι αλγόριθμοι μηχανικής μάθησης χρησιμοποιούν στατιστικά στοιχεία για να βρουν μοτίβα μέσα από μεγάλες ποσότητες δεδομένων. Καθώς ο τομέας της μηχανικής μάθησης αποτελείται από ένα μεγάλο σύνολο τεχνικών, μεθόδων και μοντέλων, πριν προχωρήσουμε είναι σημαντικό να αναφερθεί μια βασική ορολογία.

- **Supervised learning** : Η εκπαίδευση ενός μοντέλου σε ένα σύνολο δεδομένων που περιέχουν ετικέτες [23,24].
- **Unsupervised learning** : Η εκπαίδευση ενός μοντέλου για την εύρεση μοτίβων σε ένα σύνολο δεδομένων που δεν περιέχουν ετικέτες [23,24].
- **Segmentation** : Είναι η διαδικασία κατάτμησης ενός συνόλου δεδομένων σε πολλαπλά διακριτά σύνολα. Αυτός ο διαχωρισμός γίνεται έτσι ώστε τα μέλη του ίδιου συνόλου να είναι παρόμοια μεταξύ τους και διαφορετικά από τα μέλη των υπόλοιπων συνόλων [23].
- **Train set** : Ένα υποσύνολο των δεδομένων τα οποία χρησιμοποιούνται για τη δημιουργία του μοντέλου [23].
- **Test set** : Ένα υποσύνολο των δεδομένων τα οποία χρησιμοποιούνται για την αξιολόγηση της απόδοσης του μοντέλου, ή Validation set [23].
- **Noise** : Οποιαδήποτε άσχετη πληροφορία ή τυχαιότητα σε ένα σύνολο δεδομένων επισκιάζει το υποκείμενο μοτίβο [23].
- **Overfitting** : Συμβαίνει όταν το μοντέλο που υλοποιήσαμε μαθαίνει πολύ καλά τα δεδομένα της εκπαίδευσης και ενσωματώνει λεπτομέρειες και θόρυβο ειδικά για το σύνολο δεδομένων.

Μπορούμε να πούμε ότι το overfitting συμβαίνει όταν έχει εξαιρετική απόδοση στο training set και κακή στο test σετ [23].

- **Underfitting** : Συμβαίνει όταν το μοντέλο που υλοποιήσαμε υπεργενικεύεται και αποτυγχάνει να ενσωματώσει σχετικές παραλλαγές στα δεδομένα που θα έδιναν στο μοντέλο περισσότερη προγνωστική ισχύ. Ουσιαστικά συμβαίνει όταν το μοντέλο έχει κακή απόδοση τόσο στο train set όσο και στο test set [23].

3.4 Feature Extraction

3.4.1 Εισαγωγή

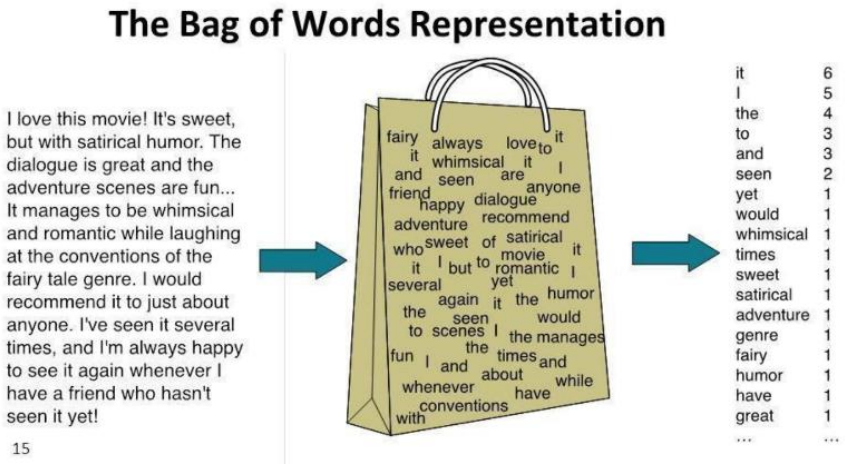
Τα κείμενα και γενικότερα τα έγγραφα στην αρχική τους μορφή είναι ακατέργαστα σύνολα δεδομένων. Ωστόσο, κατά τη χρήση της μαθηματικής μοντελοποίησης για έναν ταξινομητή, αυτές οι μη δομημένες ακολουθίες συμβολοσειρών πρέπει να μετατραπούν σε ένα δομημένο χώρο χαρακτηριστικών [25].

Θα πρέπει λοιπόν στην αρχή τα δεδομένα να καθαριστούν από το περιττό υλικό όπως είδαμε παραπάνω για να μπορούν να εφαρμοστούν τεχνικές εξαγωγής χαρακτηριστικών. Σε αυτήν την ενότητα θα αναφερθούν ορισμένες από αυτές τις τεχνικές που υπάρχουν και μελετήθηκαν.

3.4.2 Bag of Words

Ένα μοντέλο Bag – of – Words (BOW) αντιπροσωπεύει μια μειωμένη και απλοποιημένη αναπαράσταση ενός κειμένου, μιας πρότασης ή ενός συνόλου εγγράφων τα οποία επιλέχθηκαν βάση συγκεκριμένων κριτηρίων όπως η συχνότητα των λέξεων του [25,27].

Σε ένα BOW το σώμα του κειμένου θεωρείται σαν μια τσάντα λέξεων όπου, κατά τη διαδικασία δημιουργούνται λίστες λέξεων όπου περιέχουν τις ίδιες λέξεις του κειμένου σύμφωνα με τη συχνότητα που εμφανίζονται [26].



Σχήμα 3.1 : Αναπαράσταση BoW. Source

Οι λέξεις είναι συχνά αντιπροσωπευτικές του περιεχομένου μιας πρότασης. Ενώ η γραμματική και η σειρά εμφάνισης δε συμπεριλαμβάνονται στη διαδικασία, ο αριθμός των εμφανίσεων υπολογίζεται καθώς μπορεί να χρειαστεί αργότερα για τον καθορισμό του σημείου όπου εστιάζουν τα κείμενα [25].

3.4.3 Term Frequency – Inverse Document Frequency (TF – IDF)

Το TF-IDF λειτουργεί από τον προσδιορισμό της σχετικής συχνότητας των λέξεων σε ένα συγκεκριμένο έγγραφο σε σύγκριση με την αντίστροφη αναλογία της λέξης αυτής μέσα σε ολόκληρο το σώμα του εγγράφου [28].

Με αυτόν τον υπολογισμό μπορεί να καθοριστεί πόσο σχετική είναι μια δεδομένη λέξη μέσα στο συγκεκριμένο έγγραφο. Λέξεις που είναι κοινές σε ένα έγγραφο ή ένα σύνολο εγγράφων τείνουν να έχουν υψηλότερους αριθμούς TFIDF από κοινές λέξεις όπως τα άρθρα ή οι προθέσεις [28].

Η διαδικασία της εφαρμογής του TF-IDF είναι η εξής, δίνεται μια συλλογή εγγράφων **D**, μια λέξη **w** και ένα μεμονωμένο έγγραφο **d ϵ D**, και υπολογίζουμε

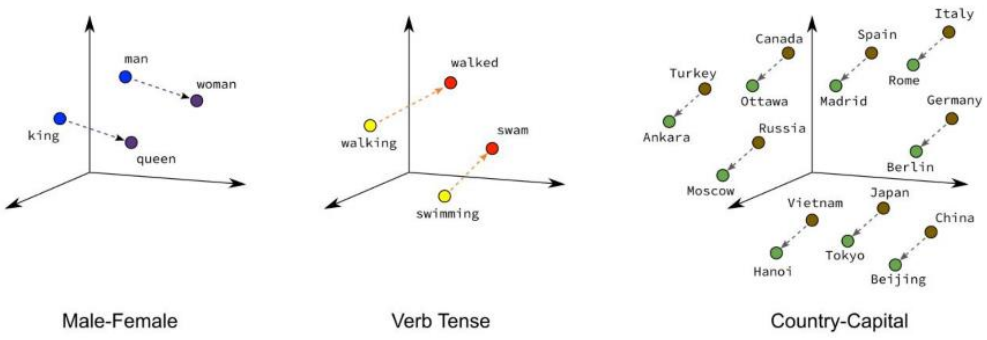
$$w_d = f_{w,\,d} * \log\left(|D|/f_{w,\,D}\right)$$

Σχήμα 3.2 : Εφαρμογή TF-IDF [28].

Όπου το **F*w,d*** ισούται με το πόσες φορές εμφανίζεται το **w** μέσα στο **d**, **|D|** είναι το μέγεθος του εγγράφου, και **F*w,d*** ισοδυναμεί με τον αριθμό των εγγράφων στα οποία εμφανίζεται το **w** στο **D** [28].

3.4.4 Word Embeddings

Τα Word Embeddings είναι μια τεχνική εκμάθησης χαρακτηριστικών στην οποία, κάθε λέξη ή φράση του λεξιλογίου αντιστοιχίζεται σε ένα διάνυσμα πραγματικών αριθμών N διαστάσεων [25]. Λέξεις που έχουν ίδια σημασία μεταξύ τους έχουν και παρόμοια αναπαράσταση [29].

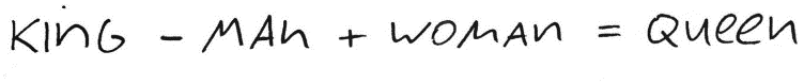


Εικόνα 3.3 : Word Embeddings. Source

Οι μέθοδοι των word embeddings μαθαίνουν μια διανυσματική αναπαράσταση για ένα προκαθορισμένο λεξιλόγιο σταθερού μεγέθους μέσα από ένα σώμα κειμένου. Η διαδικασία εκμάθησης συνδέεται είτε με το μοντέλο των νευρωνικών δικτύων σε κάποια εργασία όπως η ταξινόμηση εγγράφων είτε είναι μια διαδικασία χωρίς επίβλεψη χρησιμοποιώντας στατιστικά στοιχεία εγγράφων [29]. Μια τεχνική που χρησιμοποιείται για την εκμάθηση των word embeddings είναι το **Word2Vec**.

Το Word2Vec είναι μια στατιστική μέθοδος για την αποτελεσματική εκμάθηση ενός αυτόνομου word embedding μέσα από ένα text corpus. Αναπτύχθηκε από τον Tomas Mikovol το 2013 στην Google με στόχο να γίνει πιο αποτελεσματική η εκπαίδευση των embeddings που βασίζονται σε νευρωνικά δίκτυα κι από τότε κατάφερε να είναι ένα από τα πλέον πιο χρησιμοποιούμενα πρότυπα για την ανάπτυξη προ εκπαιδευμένων word embeddings [29].

Η εργασία περιλάμβανε επιπλέον την ανάλυση των διανυσμάτων που εκπαιδεύτηκαν και την εξερεύνηση των διανυσματικών μαθηματικών στις αναπαραστάσεις των λέξεων. Για παράδειγμα [30] :

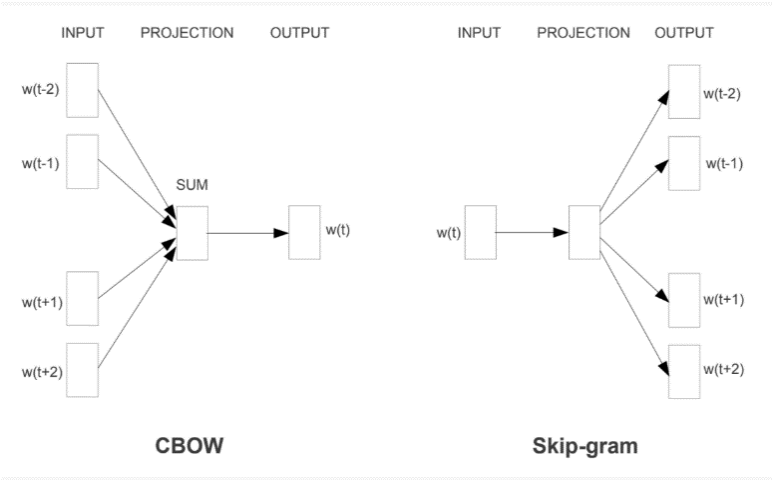


Σχήμα 3.4 : παράδειγμα διανυσματικών μαθηματικών. Source

Για την εκμάθηση των word embeddings με τη μέθοδο του word2vec εισήχθησαν δύο διαφορετικά μοντέλα [25,29] :

- **Continuous Bag-of-Words (CBOW)** : Όπου το μοντέλο αυτό μαθαίνει την ενσωμάτωση προβλέποντας την τρέχουσα λέξη με βάση τα συμφραζόμενα της [29].
- **Continuous Skip-Gram** : Στο μοντέλο αυτό δίνοντας του μία λέξη, μαθαίνει προβλέποντας τις γύρω λέξεις [29].

Και τα δύο μοντέλα επικεντρώνονται στην εκμάθηση λέξεων με βάση το τοπικό περιβάλλον χρήσης τους, όπου αυτό το πλαίσιο ορίζεται από ένα παράθυρο γειτονικών λέξεων. Αυτό το παράθυρο είναι η διαμορφώσιμη παράμετρος του μοντέλου [25,29].



Σχήμα 3.5 : Μοντέλα CBOW και Skip-Gram [29].

3.5 Text Classification

3.5.1 Εισαγωγή

Η έρευνα της ταξινόμησης κειμένων στοχεύει στη βελτίωση ποιότητας της αναπαράστασης των κειμένων καθώς και την ανάπτυξη ταξινομητών υψηλής ποιότητας. Η επεξεργασία της ταξινόμησης κειμένων περιλαμβάνει ένα ευρύ φάσμα μεθόδων, τεχνικών, μετρήσεων και αλγορίθμων ταξινόμησης με μηχανική μάθηση. Παρακάτω θα δούμε μερικούς από τους αλγόριθμους που μελετήθηκαν κατά την υλοποίηση της εργασίας.

3.5.2 Naive Bayes

Ο αλγόριθμος Naive Bayes είναι μια τεχνική ταξινόμησης με επίβλεψη όπου βασίζεται στο θεώρημα Bayes με μια υπόθεση ανεξαρτησίας μεταξύ των προβλέψεων. Δηλαδή, ο ταξινομητής υποθέτει ότι η παρουσία ενός συγκεκριμένου χαρακτηριστικού σε μια κλάση δε σχετίζεται με την παρουσία οποιουδήποτε άλλου χαρακτηριστικού [31-33].

Για παράδειγμα, αν ένα πρότυπο X σχετίζεται με παραπάνω από ένα χαρακτηριστικά για να το περιγράψουμε, τα οποία μπορεί να εξαρτώνται το ένα από το άλλο ή από την ύπαρξη άλλων χαρακτηριστικών, ο ταξινομητής λαμβάνει τις ιδιότητες αυτές ως ανεξάρτητα χαρακτηριστικά στην προσπάθεια του να ταξινομήσει το πρότυπο ως X. Το μοντέλο αυτό είναι εύκολο στην κατασκευή και ιδιαίτερα χρήσιμο σε μεγάλα σύνολα δεδομένων [31,32].

Το θεώρημα Bayes δηλώνει μια ακόλουθη σχέση, δεδομένης της μεταβλητής κλάσης y και του διανύσματος εξαρτημένων χαρακτηριστικών x_1 δια μέσου x_n :

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n \mid y)}{P(x_1, \ldots, x_n)}$$

Σχήμα 3.6 : Παράδειγμα NB [33].

Χρησιμοποιώντας την αφελή υπόθεση ανεξαρτησίας υπό όρους :

$$P(x_i|y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i|y),$$

Σχήμα 3.7 : Παράδειγμα NB [33].

Για κάθε i η σχέση απλοποιείται σε :

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)}$$

Σχήμα 3.8 : Παράδειγμα NB [33].

Όπου P(y| $x_1,\ldots,x_n$) είναι μια σταθερά δεδομένης της εισόδου, μπορεί να χρησιμοποιηθεί ο ακόλουθος κανόνας ταξινόμησης :

$$P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

$$\Downarrow$$

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y),$$

Σχήμα 3.9 : Παράδειγμα NB [33].

Και μπορεί να χρησιμοποιηθεί μια εκτίμηση Maximum A Posteriori (MAP) για να εκτιμηθεί το P(y) και P(x_i|y). Το αποτέλεσμα είναι η σχετική συχνότητα της κλάσης **y** στο training set [33].

Μερικά από τα πλεονεκτήματα ενός Naive Bayes αλγόριθμου είναι τα εξής [31]:

- Είναι εύκολο και γρήγορο να προβλέψει τις κατηγορίες του συνόλου των test data και επίσης αποδίδει καλά στην πρόβλεψη πολλαπλών κατηγοριών.
- Όταν ισχύει η υπόθεση της ανεξαρτησίας ο ταξινομητής αποδίδει καλύτερα σε σύγκριση με άλλα μοντέλα χρησιμοποιώντας λιγότερα δεδομένα εκπαίδευσης.
- Έχει καλή απόδοση στην περίπτωση κατηγορικών μεταβλητών εισόδου σε σύγκριση με αριθμητικές μεταβλητές [31].

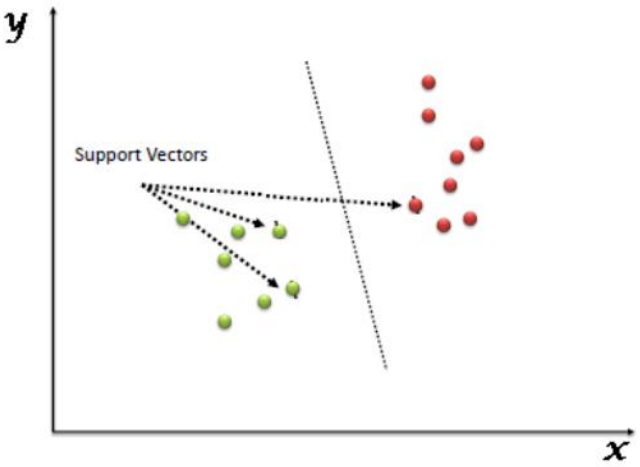
Και ορισμένα μειονεκτήματα [31]:

- Εάν η κατηγορική μεταβλητή έχει μια κατηγορία στο test dataset η οποία δεν παρατηρήθηκε στο σύνολο δεδομένων εκπαίδευσης, τότε το μοντέλο θα εκχωρήσει μια πιθανότητα μηδέν και θα αδυνατεί να κάνει μια πρόβλεψη. Αυτό είναι γνωστό ως «μηδενική συχνότητα». Για την επίλυση ενός τέτοιου ζητήματος μπορεί να χρησιμοποιηθεί μια τεχνική εξομάλυνσης.
- Επίσης, ο Naive Bayes είναι γνωστός ως κακός εκτιμητής, επομένως τα αποτελέσματα των πιθανοτήτων που δίνει δεν πρέπει να λαμβάνονται σοβαρά υπόψη.
- Ένα άλλο μειονέκτημα του είναι η υπόθεση των ανεξάρτητων προβλέψεων. Σε προβλήματα του πραγματικού κόσμου είναι σχεδόν αδύνατο να έχουμε ένα σύνολο προγνωστικών που είναι εντελώς ανεξάρτητοι [31].

3.5.3 Support Vector Machine

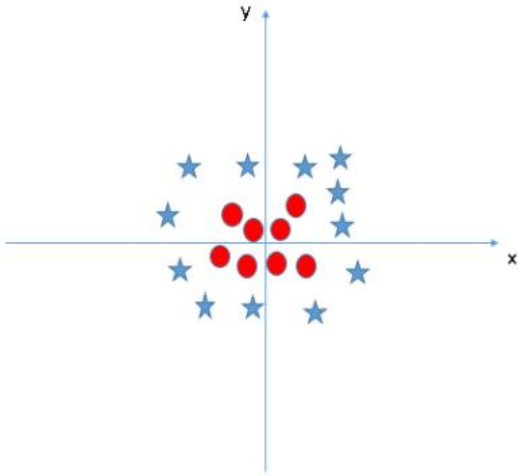
Οι μηχανές διανυσμάτων υποστήριξης (SVM) είναι ένα σύνολο μεθόδων εκμάθησης με επίβλεψη που χρησιμοποιούνται για ταξινόμηση, παλινδρόμηση και ανίχνευση ακραίων τιμών σε γραμμικά ή και μη γραμμικά διαχωρίσιμα δεδομένα [32,34].

Σε ένα γραμμικά διαχωρίσιμο αλγόριθμο SVM σχεδιάζουμε κάθε στοιχείο δεδομένων ως ένα σημείο σε 2 – διάστατο χώρο, όπου η τιμή κάθε χαρακτηριστικού είναι μια συγκεκριμένη συντεταγμένη. Και στη συνέχεια πραγματοποιείται η ταξινόμηση βρίσκοντας το υπερεπίπεδο που διαφοροποιεί τις δύο κατηγορίες όπως φαίνεται και παρακάτω [23,35]:



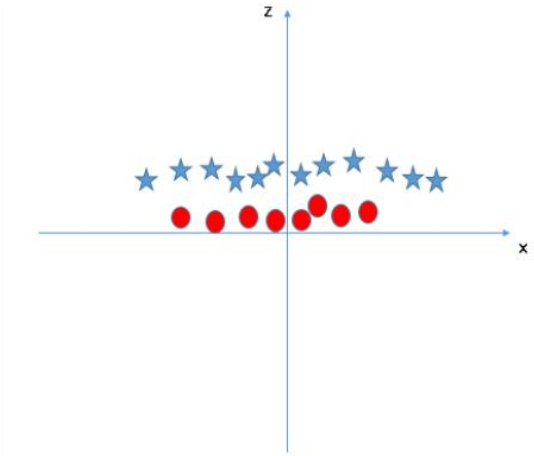
Σχήμα 3.10 : Παράδειγμα SVM [35].

Όταν τα δεδομένα που διαθέτουμε είναι γραμμικά διαχωρίσιμα τότε μπορούμε να τα διαχωρίσουμε εύκολα με μια ευθεία γραμμή. Ωστόσο υπάρχουν περιπτώσεις που τα δεδομένα δεν μπορούν να διαχωριστούν με μια γραμμή καθώς τα χαρακτηριστικά που διαθέτουν ανήκουν σε παραπάνω από μια κατηγορίες [23].



Σχήμα 3.11 : Παράδειγμα μη γραμμικού SVM [23].

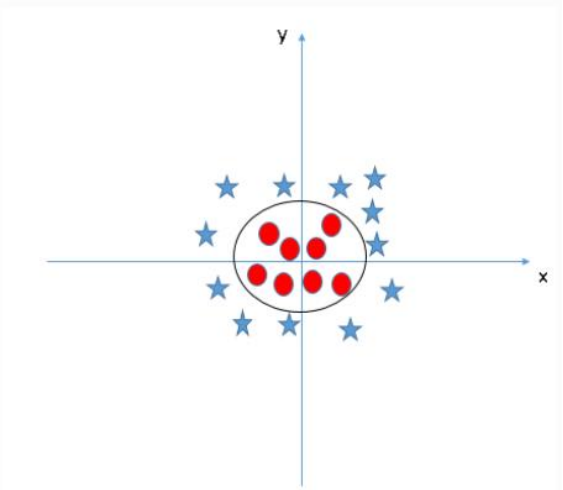
Στην παραπάνω εικόνα, για να διαχωρίσουμε τους κύκλους από τα αστέρια πρέπει να εισάγουμε ένα επιπλέον χαρακτηριστικό. Όπου στην περίπτωση που ήταν γραμμικά διαχωρίσιμα θα χρησιμοποιούσαμε απλά τα **x** και **y**. Ωστόσο, σε αυτά τα μη γραμμικά διαχωρίσιμα δεδομένα θα χρειαστεί να προσθέσουμε μια τρίτη διάσταση **z**. Το **z** ορίζεται ως **z=x2+y2**. Προσθέτοντας και το τρίτο χαρακτηριστικό, ο χώρος θα είναι ο εξής [23]:



Σχήμα 3.12 : Παράδειγμα μη γραμμικού SVM [23].

Στο παραπάνω σχήμα, όλες οι τιμές για το **z** θα είναι πάντα θετικές, επειδή το **z** είναι το τετραγωνικό άθροισμα των **x** και **y**. Σε αυτό το σημείο, ο ταξινομητής SVM διαιρεί το σύνολο των δεδομένων σε δύο διακριτές τάξεις βρίσκοντας ένα γραμμικό υπερεπίπεδο μεταξύ αυτών των δύο τάξεων [23].

Δεδομένου ότι ο χώρος που βρισκόμαστε τώρα είναι τρισδιάστατος, το υπερεπίπεδο μοιάζει με ένα επίπεδο παράλληλα με τον άξονα **x**. Αν μετατρέψουμε το χώρο σε δισδιάστατο με **z=1**, τότε σχήμα θα φαίνεται ως εξής [23] :



Εικόνα 3.13 : Παράδειγμα μη γραμμικού SVM [23].

Επομένως, σε περιπτώσεις μη γραμμικών δεδομένων, λαμβάνουμε μία περιφέρεια ακτίνας που ισοδυναμεί με 1.

Για να βρούμε το υπερεπίπεδο με τον SVM σε **n** διαστάσεις, ο αλγόριθμος χρησιμοποιεί μια τεχνική που ονομάζεται «kernel trick». Ο πυρήνας του είναι μια συνάρτηση που λαμβάνει μια είσοδο λιγότερων διαστάσεων και τη μετατρέπει σε χώρο περισσότερων. Με λίγα λόγια, μετατρέπει τα μη γραμμικά διαχωρίσιμα δεδομένα σε γραμμικά διαχωρίσιμα [23].

Μερικά από τα πλεονεκτήματα των SVM είναι :

- Είναι αποτελεσματικά σε χώρους υψηλών διαστάσεων [34].
- Είναι αποτελεσματικά σε περιπτώσεις όπου ο αριθμός των διαστάσεων είναι μεγαλύτερος από τον αριθμό των δειγμάτων [34].
- Χρησιμοποιούν ένα υποσύνολο σημείων εκπαίδευσης στη συνάρτηση απόφασης (που ονομάζεται διάνυσμα υποστήριξης) κι αυτό το καθορίζει αποδοτικό σε μνήμη [34].
- Είναι ευέλικτα. Μπορούν να καθοριστούν διαφορετικές συναρτήσεις στον πυρήνα για τη συνάρτηση αποφάσεων [34].

Και μερικά από τα μειονεκτήματα :

- Δεν αποδίδουν καλά όταν έχουμε μεγάλο σύνολο δεδομένων επειδή ο απαιτούμενος χρόνος εκπαίδευσης είναι μεγαλύτερος [35].
- Επίσης, δεν είναι τόσο αποδοτικοί όταν το σύνολο των δεδομένων έχει περισσότερο θόρυβο [35].
- Το SVM δεν παρέχει άμεσα εκτιμήσεις πιθανοτήτων, αυτές υπολογίζονται χρησιμοποιώντας cross-validation [35].

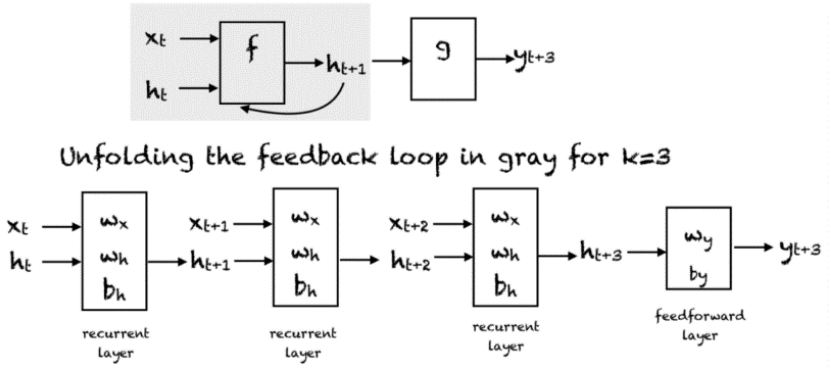
3.6 Recurrent Neural Network – RNN

3.6.1 Εισαγωγή

Ένα RNN είναι ένας ειδικός τύπος τεχνητού νευρωνικού δικτύου βαθιάς μάθησης, προσαρμοσμένος να λειτουργεί με δεδομένα χρονοσειρών ή δεδομένα που περιλαμβάνουν ακολουθίες [36,37].

Τα κλασικά feed forward νευρωνικά δίκτυα, αναφέρονται μόνο σε σημεία δεδομένων τα οποία είναι ανεξάρτητα μεταξύ τους. Ωστόσο, εάν τα δεδομένα που έχουμε είναι μια ακολουθία όπου η δομή της είναι στημένη έτσι ώστε, ένα σημείο δεδομένων να εξαρτάται από το προηγούμενο σημείο δεδομένων, θα πρέπει να τροποποιηθεί το νευρωνικό δίκτυο ώστε να ενσωματωθούν οι εξαρτήσεις μεταξύ αυτών τον σημείων [36,37].

Τα RNN δίκτυα έχουν την έννοια της «μνήμης» που τα βοηθά στο να αποθηκεύουν τις καταστάσεις ή τις πληροφορίες προηγούμενων εισόδων για να δημιουργήσουν την επόμενη έξοδο της ακολουθίας [36,37].



Σχήμα 3.14 : Γενική αρχιτεκτονική RNN [37].

Ένα απλό RNN διαθέτει ένα βρόγχο ανατροφοδότησης όπως φαίνεται και στην παραπάνω εικόνα. Σύμφωνα με το παράδειγμα της εικόνας, ο βρόγχος ανατροφοδότησης στο γκρι πλαίσιο μπορεί να ξετυλιχθεί σε **k=3** χρονικά βήματα (time steps) και δημιουργείται το δίκτυο που φαίνεται στο παραπάνω σχήμα. Οι συμβολισμοί που φαίνονται είναι οι εξής [37]:

- $\mathbf{x_t \in R}$ είναι η είσοδος στο χρονικό βήμα. Για την απλοποίηση του πράγματος, υποθέτουμε ότι το x_t είναι μια κλιμακωτή τιμή με ένα χαρακτηριστικό.
- $\mathbf{y_t \in R}$ είναι η έξοδος του δικτύου για το χρονικό βήμα t.
- $\mathbf{h_t \in R^m}$ το διάνυσμα αποθηκεύει τις τιμές των κρυφών μονάδων – καταστάσεων για τη χρονική στιγμή t. m είναι ο αριθμός των κρυφών μονάδων και, το διάνυσμα h_0 αρχικοποιείται στο μηδέν.
- $\mathbf{w_x \in R^m}$ είναι τα βάρη που σχετίζονται με τις εισόδους του επαναλαμβανόμενου επιπέδου.
- $\mathbf{w_h \in R^{mxm}}$ είναι τα βάρη που σχετίζονται με τις κρυφές μονάδες του επαναλαμβανόμενου επιπέδου.
- $\mathbf{w_y \in R^m}$ είναι τα βάρη που σχετίζονται με τις κρυφές μονάδες στην έξοδο.
- $\mathbf{b_h \in R^m}$ είναι το κατώφλι που σχετίζεται με το επαναλαμβανόμενο επίπεδο.
- $\mathbf{b_y \in R^m}$ είναι το κατώφλι που σχετίζεται με το επίπεδο της ανατροφοδότησης.

Σε κάθε χρονικό βήμα μπορούμε να ξεδιπλώσουμε το δίκτυο για **k** χρονικά βήματα για να πάρουμε το χρονικό βήμα **k+1**. Το ξεδιπλωμένο δίκτυο που φαίνεται παραπάνω δείχνει τη διαδικασία που λαμβάνει χώρα, έτσι με μια συνάρτηση ενεργοποίησης **f** έχουμε το εξής [37]:

$$h_{t+1} = f(x_t, h_t, w_x, w_h, b_h) = f(w_x x_t + w_h h_t + b_h)$$

Σχήμα 3.15 : Συνάρτηση ενεργοποίησης [37].

Η έξοδος **y** τη χρονική στιγμή **t** υπολογίζεται ως εξής :

$$y_t = f(h_t, w_y) = f(w_y \cdot h_t + b_y)$$

Σχήμα 3.16 : υπολογισμός εξόδου μιας χρονικής στιγμής [37].

Ως αποτέλεσμα, την στιγμή της ανατροφοδοσίας ενός RNN, το δίκτυο υπολογίζεις τις τιμές των κρυφών μονάδων του καθώς και την έξοδο του μετά από **k** χρονικές στιγμές. Τα βάρη που σχετίζονται με το δίκτυο διαμοιράζονται προσωρινά. Κάθε επαναλαμβανόμενο στρώμα έχει δύο σετ από βάρη, ένα για την είσοδο και ένα για την κρυφή μονάδα. Το τελευταίο επίπεδο προώθησης το οποίο υπολογίζει την τελική έξοδο της **k** χρονικής στιγμής είναι ακριβώς όπως ένα συνηθισμένο επίπεδο ενός παραδοσιακού δικτύου προώθησης [37].

3.6.2 Vanishing Gradient problem

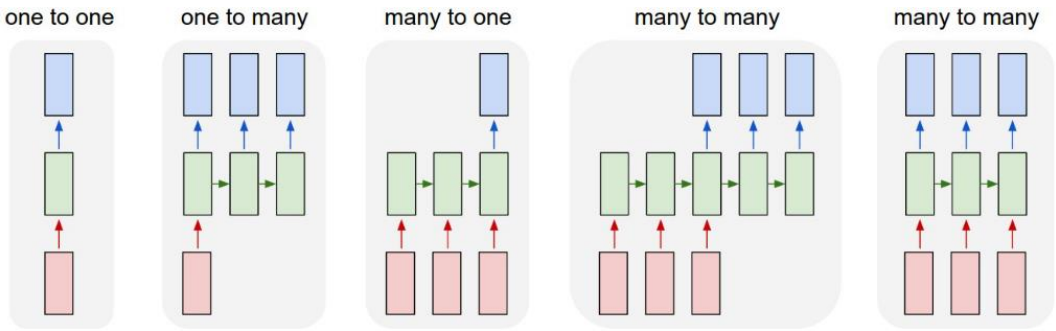
Τα RNN υποφέρουν από βραχυπρόθεσμη μνήμη, αν μια ακολουθία είναι αρκετά μεγάλη, θα δυσκολευτούν να μεταφέρουν πληροφορίες από προηγούμενες χρονικές στιγμές στις μεταγενέστερες [38].

Κατά τη διάρκεια του back propagation, τα RNN υποφέρουν από το πρόβλημα της διαβάθμισης που εξαφανίζεται (vanishing gradient [39]). Οι διαβαθμίσεις είναι τιμές που χρησιμοποιούνται για την ενημέρωση των συναπτικών βαρών ενός νευρωνικού δικτύου. Το πρόβλημα αυτό υφίσταται όταν οι τιμές αυτές συρρικνώνονται καθώς διαδίδονται ξανά στο χρόνο. Εάν μια τιμή διαβάθμισης συρρικνωθεί αρκετά, τότε συμβάλλει ελάχιστα στη μάθηση.

Με αυτόν τον τρόπο, τα επίπεδα των RNN που λαμβάνουν μικρή ενημέρωση κλίσης σταματούν να μαθαίνουν. Για το λόγο αυτό, τα RNN μπορούν να ξεχάσουν τις πληροφορίες που είδαν σε μεγάλες ακολουθίες έχοντας έτσι μια βραχυπρόθεσμη μνήμη. Η λύση σε αυτό το πρόβλημα έρχεται με τα δίκτυα LSTM [38].

3.6.3 Τύποι RNN

Υπάρχουν αρκετοί τύποι RNN δικτύων με διαφορετικές αρχιτεκτονικές, μερικοί από αυτούς φαίνονται στο παρακάτω σχήμα :



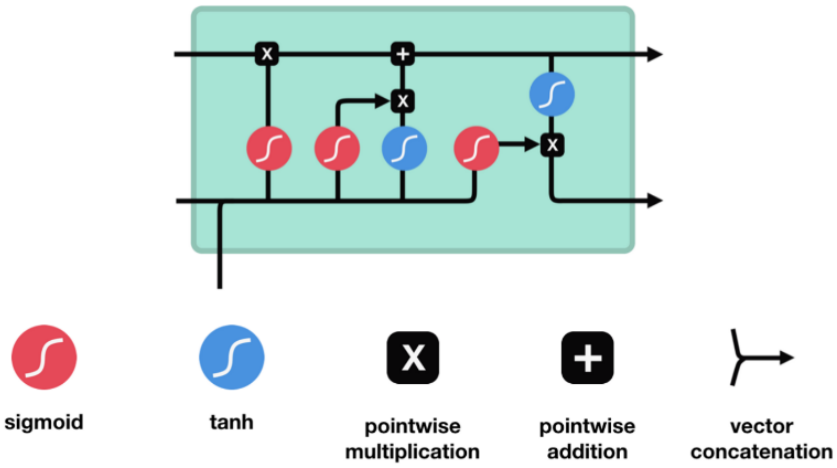
Εικόνα 3.17 : Τύποι RNN δικτύων. Source

- Ο τύπος **one-to-one** είναι ο απλός, τα παραδοσιακά νευρωνικά δίκτυα χρησιμοποιούν τον ίδιο [37].
- Ο τύπος **one-to-many**, μια μεμονωμένη είσοδος στο x_t μπορεί να παράγει πολλές εξόδους. Ένα παράδειγμα για αυτόν τον τύπο δικτύου είναι η παραγωγή της μουσικής [37].
- **Many-to-one** όπου σε αυτήν την περίπτωση, πολλές είσοδοι από διαφορετικές χρονικές στιγμές παράγουν μόνο μία έξοδο. Τέτοιοι τύποι δικτύων χρησιμοποιούνται σε sentiment analysis και emotion detectionόπου η ετικέτα κλάσης εξαρτάται από μια ακολουθία λέξεων [37].
- **Many-to-many** όπου πολλές είσοδοι μπορούν να παράγουν πολλές εξόδους. Για αυτούς τους τύπους υπάρχουν πολλές δυνατότητες, ένα χαρακτηριστικό παράδειγμα τους είναι οι μεταφραστές γλωσσών [37].

3.6.4 Αρχιτεκτονική LSTM

Τα δίκτυα LSTM (Long Short Term Memory) είναι μια ειδική κατηγορία των RNN, είναι ικανά να μαθαίνουν μακροπρόθεσμες εξαρτημένες ακολουθίες. Τα δίκτυα αυτά, σχεδιάστηκαν συγκεκριμένα για την αποφυγή των προβλημάτων μακροπρόθεσμης εξάρτησης [40].

Ένα LSTM έχει παρόμοια ροή ελέγχου με ένα RNN, επεξεργάζεται τα δεδομένα που διαβιβάζουν πληροφορίες καθώς αυτές διαδίδονται προς τα μπροστά. Οι διαφορές των λειτουργιών που υπάρχουν με τα απλά RNN βρίσκονται μέσα στα κελιά του LSTM [41].



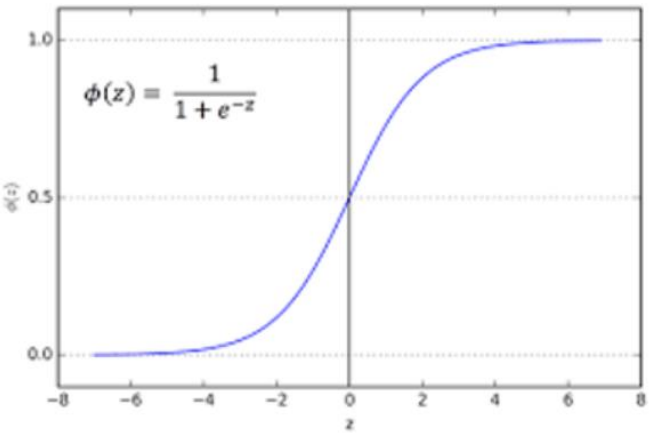
Σχήμα 3.18 : LSTM cell [41].

Οι λειτουργίες που βλέπουμε στην παραπάνω εικόνα, είναι αυτές που χρησιμοποιούνται για να επιτρέπουν στο LSTM να διατηρεί ή να ξεχνά πληροφορίες [41].

Η βασική ιδέα των LSTM είναι το **cell state** και οι εσωτερικές πύλες. Το cell state μεταφέρει τη σχετική πληροφορία ανάμεσα σε ολόκληρη τη διαδρομή της εξαρτημένης ακολουθίας. Θεωρητικά, μπορεί να μεταφέρει τις σχετικές πληροφορίες σε ολόκληρη την επεξεργασία της ακολουθίας λειτουργώντας κατ' αυτόν τον τρόπο ως η μνήμη του δικτύου. Έτσι, ακόμα και οι πληροφορίες από τις προηγούμενες χρονικές στιγμές μπορούν να οδηγήσουν σε μεταγενέστερες χρονικές στιγμές μειώνοντας τις επιπτώσεις της βραχυπρόθεσμης μνήμης [41].

Καθώς το cell state συνεχίζει τη διαδικασία, οι αποφάσεις για το αν προστεθεί η αφαιρεθεί η πληροφορία μέσα σε αυτό λαμβάνονται από τις εσωτερικές πύλες. Αυτές οι πύλες είναι διαφορετικά νευρωνικά δίκτυα όπου απώτερος σκοπός τους είναι οι αποφάσεις για το αν θα περάσει η πληροφορία ή όχι, μαθαίνοντας για το ποιες πληροφορίες είναι χρήσιμες κατά τη διάρκεια της υλοποίησης του μοντέλου [40-42].

Οι πύλες περιλαμβάνουν σιγμοειδή συνάρτηση ενεργοποίησης με τιμές μεταξύ του 0 και 1. Με αυτόν τον τρόπο μπορεί να ενημερώνει ή να «ξεχνάει» τα δεδομένα καθώς οποιοσδήποτε αριθμός πολλαπλασιαστεί με το 0 το αποτέλεσμα θα είναι 0, συνεπώς και η πληροφορία θα απορριφθεί, ενώ αν πολλαπλασιαστεί με μια τιμή μεγαλύτερη του 0, τότε η πληροφορία θα διατηρηθεί ή θα παραμείνει ίδια [41].

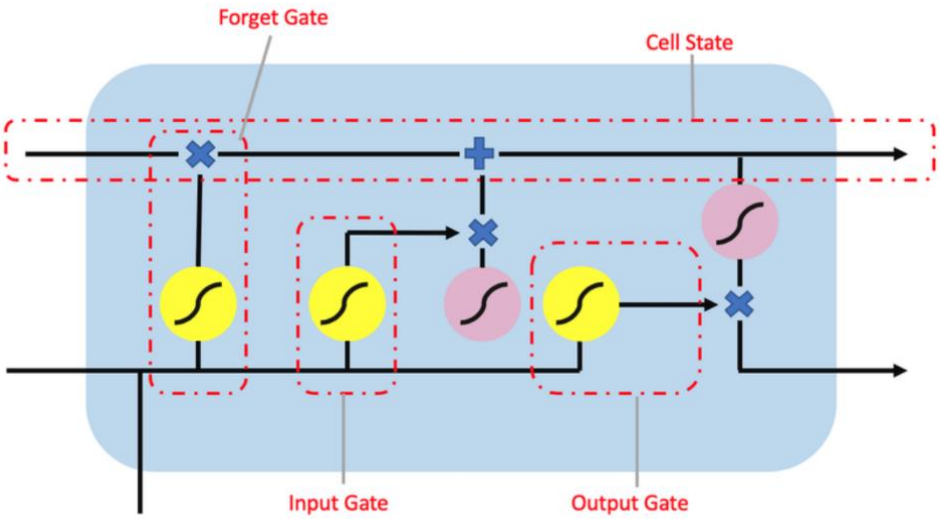


Σχήμα 3.19 : Sigmoid Function. Source

Το LSTM cell αποτελείται από τρείς διαφορετικές πύλες [41,42] :

- **Forget Gate** : Η πύλη αυτή αποφασίζει ποιες πληροφορίες θα πρέπει να πεταχτούν ή να διατηρηθούν. Πληροφορίες από την προηγούμενη κρυφή κατάσταση και πληροφορίες από την τρέχουσα είσοδο, διαβιβάζονται μέσω της σιγμοειδούς συνάρτησης με τις τιμές μεταξύ 0-1, όσο πιο κοντά είναι η τιμή στο 0 σημαίνει απέρριψε ενώ όσο πιο κοντά στο 1 σημαίνει κράτησε [41].
- **Input Gate** : Αυτή η πύλη υπάρχει για να ενημερωθεί η κατάσταση του cell. Αρχικά οι τιμές περνάνε την προηγούμενη κρυφή κατάσταση και την τρέχουσα είσοδο με τη σιγμοειδή συνάρτηση. Αυτή αποφασίζει ποιες τιμές θα ενημερωθούν μετατρέποντας τις μεταξύ 0 και 1, παράλληλα οι τιμές περνάνε από μια tanh συνάρτηση μεταξύ των τιμών -1 και 0 για να βοηθήσει τη ρύθμιση του δικτύου. Έπειτα πολλαπλασιάζονται τα αποτελέσματα των δύο συναρτήσεων που προκύπτουν. Τα αποτελέσματα της σιγμοειδούς συνάρτησης θα είναι αυτά όπου στην ουσία θα αποφασίσουν για το ποιες είναι οι σημαντικές πληροφορίες που θα παραμείνουν [41].
- **Output Gate** : Η πύλη εξόδου αποφασίζει ποια θα πρέπει να είναι η επόμενη κρυφή κατάσταση. Καθώς η κρυφή κατάσταση περιέχει πληροφορίες για προηγούμενες εισόδους, η ίδια χρησιμοποιείται και για προβλέψεις. Αρχικά, η προηγούμενη κρυφή κατάσταση μαζί με την τρέχουσα είσοδο περνάει από μια σιγμοειδή συνάρτηση, στη συνέχεια η νέα τροποποιημένη κατάσταση του cell περνάει από μια συνάρτηση tanh. Έπειτα πολλαπλασιάζονται η έξοδος της σιγμοειδούς με την έξοδο της tanh για να αποφασιστεί ποιες πληροφορίες θα πρέπει να εμπεριέχει η κρυφή κατάσταση. Η έξοδος που θα παραχθεί, είναι η κρυφή κατάσταση. Η νέα κατάσταση του cell και η νέα κρυφή κατάσταση περνάνε στη συνέχεια σε νέα χρονική στιγμή [41].

Το cell state πολλαπλασιάζεται αρχικά με το vector λήθης, αυτό έχει τη δυνατότητα να απορριφθούν οι άχρηστες τιμές από το cell state, στη συνέχεια προσμετρώνται τα αποτελέσματα εξόδου της πύλης εισόδου και ενημερώνεται το cell state με τιμές που το νευρωνικό δίκτυο θεωρεί σχετικές. Αυτό έχει ως αποτέλεσμα να παράγεται το νέο cell state [41].



Σχήμα 3.20 : LSTM cell gates. Source

3.7 Επίλογος

Στο κεφάλαιο αυτό αναφέρθηκαν δύο ευρέως γνωστές και χρησιμοποιούμενες βιβλιοθήκες οι οποίες προσφέρουν μοντέλα κι αλγόριθμους για την υλοποίηση εφαρμογών μηχανικής μάθησης ή και βαθιάς μάθησης. Στη συνέχεια αναφέρθηκαν περιπτώσεις μεθόδων εξαγωγής χαρακτηριστικών, αλγόριθμοι ταξινόμησης κειμένων καθώς και την κατηγορία αλγόριθμων βαθιάς μάθησης RNN με ορισμένες αρχιτεκτονικές που την ακολουθούν και το μοντέλο LSTM το οποίο είναι ένα από τα συχνά χρησιμοποιούμενα μοντέλα για τους τύπους RNN.

Κεφάλαιο 4ο: Υλοποίηση Μοντέλων

4.1 Εισαγωγή

Η δομή του συστήματος που θα αναλύσουμε αποτελείται από δύο ξεχωριστές εργασίες, την αναγνώριση της πρόθεσης του χρήστη και την εξαγωγή της σημαντικής πληροφορίας από το εισαγόμενο κείμενου που λαμβάνουμε.

Σε αυτό το κεφάλαιο θα μελετήσουμε τη βάση γνώσεων της εφαρμογής και θα αναλύσουμε αυτές τις δύο σημαντικές εργασίες.

4.2 Βάση γνώσεων

Για βάση γνώσεων της παρούσας πτυχιακής εργασίας επιλέχθηκε το **ATIS dataset (Airline Travel Information Systems)**. Αποτελείται από ένα σύνολο δεδομένων από ανθρώπους οι οποίοι αναζητούν πληροφορίες πτήσεων σε αυτοματοποιημένα συστήματα διερεύνησης ταξιδιών από αεροπορικές εταιρείες, το σύνολο των δεδομένων χωρίζεται σε 4978 εκφράσεις για training και 893 εκφράσεις για test, ταξινομημένα σε 26 κατηγορίες προθέσεων.

Τα δεδομένα που παρέχει το dataset είναι ομαδοποιημένες εκφράσεις οι οποίες είναι γραμμένες στην αγγλική γλώσσα, τις κατηγορίες των προθέσεων όπου ανήκει η κάθε έκφραση καθώς και την κωδικοποίηση των εκφράσεων. Παρέχει επίσης δύο λεξικά, ένα λεξικό με τα δεδομένα εισόδου και ένα λεξικό με τα named entities για τα δεδομένα εξόδου.

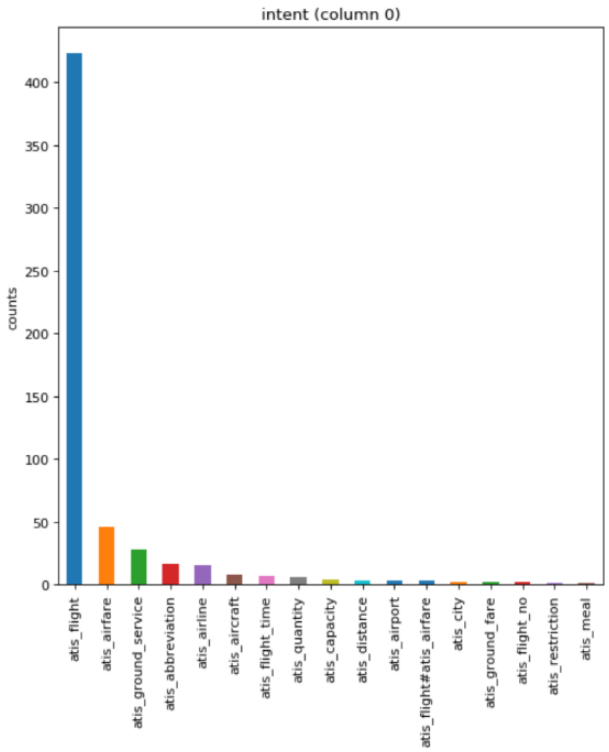
Πίνακας 4.1 : Ενδεικτικό παράδειγμα δεδομένων του ATIS dataset

Intent label	Flight
Query text	BOS show me the flights from Pittsburgh to Los Angeles on Thursday EOS
Query vector	[178 770 581 827 429 444 682 851 563 216 654 845 179]
Slot text	O O O O O O B-fromloc.city_name O B-toloc.city_name I-toloc.city_name O B-depart_date.day_name O
Slot vector	[128, 128, 128, 128, 128, 128, 48, 128, 78, 125, 128, 26, 128]

Στο παραπάνω σχήμα βλέπουμε ένα ενδεικτικό παράδειγμα των δεδομένων που περιέχει το dataset. Όπου, το **intent label** είναι η κατηγορία (η πρόθεση) στην οποία ανήκει η συγκεκριμένη έκφραση, το **query text** είναι η έκφραση του χρήστη, το **query vector** είναι το vector της συγκεκριμένης έκφρασης το οποίο μας το παρέχει το dataset μέσα από το λεξικό εισόδου του. Το **slot text** αποτελεί την κωδικοποίηση της έκφρασης του χρήστη σε μορφή IOB2 δείχνοντας μας με αυτόν τον τρόπο τη σημαντική πληροφορία που βρίσκεται μέσα σε μια έκφραση και τέλος, το **slot vector** το οποίο είναι το vector του αντίστοιχου slot text το οποίο παρέχει επίσης το dataset βάση του λεξικού εξόδου του.

Η μορφή IOB2 (Inside-Outside-Beginning) είναι μια κοινή μορφή προσθήκης ετικετών σε tokens που καθιστά σε ποια οντότητα (named entity) ανήκει το καθένα. Όπου, το πρόθεμα **I** πριν από μια

ετικέτα υποδεικνύει ότι το token έπεται ενός άλλου της ίδιας οντότητας με το οποίο συνδυάζεται. Μια ετικέτα **O** υποδηλώνει ότι το token δεν ανήκει σε κάποια οντότητα και, το πρόθεμα **B** πριν από κάποια ετικέτα υποδεικνύει ότι το token είναι το αρχικό μιας οντότητας κι ότι πιθανόν να ακολουθεί και δεύτερο της ίδιας οντότητας σε συνδυασμό με αυτό [43].



Εικόνα 4.1 : Μοναδικές κατηγορίες ATIS dataset. Source

Σύμφωνα με την παραπάνω εικόνα, μπορούμε να διαπιστώσουμε πως το σύνολο των δεδομένων που μας παρέχει το dataset δεν είναι σωστά ισορροπημένο δηλαδή, οι κατηγορίες που διαθέτει έχουν διαφορετικό μέγεθος εκφράσεων, πράγμα που επηρεάζει τα μοντέλα της μηχανικής μάθησης κατά την εκπαίδευση τους όπως θα δούμε παρακάτω.

4.3 Intent Recognition – IR

4.3.1 Εισαγωγή

Intent Recognition ή αλλιώς Intent Classification, είναι η διαδικασία της λήψης γραπτού ή προφορικού λόγου ως είσοδο σε ένα σύστημα και της ταξινόμησης του βάση της πρόθεσης που επιδιώκει ο χρήστης. Αποτελεί βασικό συστατικό για τη λειτουργία ενός Chatbot και η πρώτη διαδικασία που θα εκτελεστεί κατά την είσοδο των δεδομένων του χρήστη [44].

4.3.2 Υλοποίηση

Για την υλοποίηση της διαδικασίας αναγνώρισης της πρόθεσης, θα χρησιμοποιήσουμε τη βιβλιοθήκη του spaCy που είδαμε νωρίτερα για το κομμάτι της επεξεργασίας του φυσικού λόγου και, για το μοντέλο της μηχανικής μάθησης επιλέχθηκε η μέθοδος ταξινόμησης SVC των μεθόδων του SVM μέσα από τη βιβλιοθήκη scikit-learn. Ο λόγος που επιλέχθηκε το συγκεκριμένο μοντέλο είναι για την αποδοτικότητα του σε χώρους πολλών διαστάσεων.

Για την υλοποίηση του μοντέλου, αρχικά διαβάζουμε τα δεδομένα του dataset για train set και test set ξεχωριστά τα οποία είναι σε ακατέργαστη μορφή :

Type	Size	Value
str	80	i want to fly from boston at 838 am and arrive in denver at 1110 in t ...
str	76	what flights are available from pittsburgh to baltimore on thursday m ...
str	83	what is the arrival time in san francisco for the 755 am flight leavi ...
str	40	cheapest airfare from tacoma to orlando
str	68	round trip fares from pittsburgh to philadelphia under 1000 dollars
str	54	i need a flight tomorrow from columbus to minneapolis
str	67	what kind of aircraft is used on a flight from cleveland to dallas
str	63	show me the flights from pittsburgh to los angeles on thursday
str	38	all flights from boston to washington
str	58	what kind of ground transportation is available in denver
str	49	show me the flights from dallas to san francisco
str	63	show me the flights from san diego to newark by way of houston
str	47	what is the cheapest flight from boston to bwi
str	36	all flights to baltimore after 6 pm

Σχήμα 4.2 : Δεδομένα από το dataset.

Για να μπορέσουμε να εισάγουμε αυτά τα δεδομένα στο SVC θα πρέπει πρώτα να τα επεξεργαστούμε και να φτιάξουμε word vectors ίδιου καθώς είναι κείμενα και οι αλγόριθμοι μηχανικής μάθησης επικοινωνούν με αριθμούς.

Για την υλοποίηση των word vectors θα χρειαστεί να χρησιμοποιήσουμε το μέγεθος των vectors που χρησιμοποιεί το μοντέλο **en_core_web_lg** του spaCy. Παρατηρούμε στην παραπάνω εικόνα ότι οι εκφράσεις δεν είναι ίδιου μεγέθους λέξεων, θα πρέπει λοιπόν να φτιάξουμε όλα τα διανύσματα με ίδιο μέγεθος με μια μέθοδο που λέγεται **sequence padding** καθώς είναι απαραίτητο για την καλύτερη αποδοτικότητα του μοντέλου.

```python
#process raw data
def encode_sentences(sentences, spacy_embeddings):
    #number of sentences
    sentences_length = len(sentences)

    #create an array for sequence padding
    X_set = np.zeros((sentences_length,spacy_embeddings))

    #iterate sentences  to create word vectors with padding
    for index, sentence in enumerate(sentences):
        doc = nlp(sentence)
        X_set[index, :] = doc.vector
    return X_set
```

Σχήμα 4.3 : Επεξεργασία ακατέργαστων δεδομένων.

Στη μέθοδο της παραπάνω εικόνας γίνεται η επεξεργασία των δεδομένων και δημιουργούνται τα word vectors. Η μέθοδος δέχεται δύο παραμέτρους, τα δεδομένα του X set και το μέγεθος των vectors που χρησιμοποιεί το spacy. Έπειτα, δημιουργείται ένας πίνακας μεγέθους ***sentences_length * spacy_embeddings*** με αρχικοποιημένες τιμές το 0 σε ολόκληρο τον πίνακα. Όπου sentences_length είναι το σύνολο των γραμμών που περιέχει το X set και spacy_embeddings = 300 είναι το μέγεθος που χρησιμοποιεί το μοντέλο του spacy για τα διανύσματα του. Και στη συνέχεια γίνεται η προσπέλαση των δεδομένων για να υλοποιηθούν τα διανύσματα και να αντικατασταθούν με τα μηδενικά του πίνακα που υλοποιήσαμε προηγουμένως.

Μετά την παραπάνω διαδικασία τα δεδομένα των train και test set είναι έτοιμα για χρήση και μπορούμε να δημιουργήσουμε το μοντέλο, να το εκπαιδεύσουμε και να αξιολογήσουμε την απόδοση του πάνω στο test set των δεδομένων :

```python
from sklearn.svm import SVC
from sklearn.metrics import classification_report,accuracy_score

#create svc model
model = SVC()
model.fit(X_train, intents_train)

y_predicts = model.predict(X_test)

#count correct predictions
correct_predicts = 0
for i in range(len(intents_test)):
    if y_predicts[i] == intents_test[i]:
        correct_predicts += 1
```

Σχήμα 4.4 : Υλοποίηση μοντέλου.

Στο σχήμα 4.5 βλέπουμε την υλοποίηση του μοντέλου, την εκπαίδευση του πάνω στα training δεδομένα, έπειτα την αξιολόγηση του πάνω στα testing δεδομένα και στο τέλος δημιουργούμε έναν counter για να μετρήσουμε πόσα σωστά αποτελέσματα έχουμε βάση του συνόλου των test δεδομένων.

```
Test set Classification report
                    precision    recall  f1-score   support

  atis_abbreviation       1.00      1.00      1.00        33
      atis_aircraft       0.75      1.00      0.86         9
       atis_airfare       0.95      0.88      0.91        48
       atis_airline       1.00      0.76      0.87        38
        atis_flight       0.97      0.99      0.98       632
   atis_flight_time       0.00      0.00      0.00         1
atis_ground_service       1.00      1.00      1.00        36
      atis_quantity       0.67      0.67      0.67         3

           accuracy                           0.97       800
          macro avg       0.79      0.79      0.79       800
       weighted avg       0.97      0.97      0.97       800

accuracy score :  0.97125
correct predicts :  777
```

Σχήμα 4.5 : Αποτελέσματα αξιολόγησης του μοντέλου.

Στο παραπάνω σχήμα βλέπουμε το classification report από τα αποτελέσματα της αξιολόγησης του μοντέλου. Η ακρίβεια του μοντέλου είναι 97% και οι σωστές προβλέψεις που έγιναν στα δεδομένα είναι 777/800. Ωστόσο παρατηρούμε στην εικόνα πως σε ορισμένες κατηγορίες προθέσεων τα αποτελέσματα των μετρήσεων δεν είναι καλά, αυτό οφείλεται στο γεγονός ότι τα δεδομένα του dataset δεν είναι σωστά ισορροπημένα.

4.3.3 Παραδείγματα μοντέλου

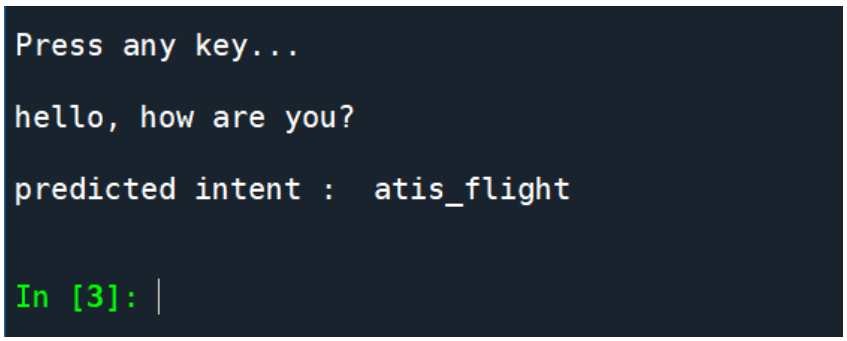
Ένα απλό παράδειγμα των αποτελεσμάτων που θα μας δώσει το μοντέλο δίνοντας του ως είσοδο τη δεδομένη έκφραση «Can you find me a flight to Boston please» φαίνεται στο παρακάτω σχήμα :

```
Press any key...

Can you find me a flight to Boston please

predicted intent :  atis_flight


In [2]:
```

Σχήμα 4.6 : Παράδειγμα 1ο IR.

Η πρόβλεψη για τα δεδομένα της συγκεκριμένης έκφρασης είναι **atis_flight**. Το αποτέλεσμα είναι σωστό.

Ένα δεύτερο παράδειγμα για το συγκεκριμένο μοντέλο είναι το ακόλουθο, δίνεται ως είσοδος η δεδομένη έκφραση «hello, how are you?» :



Σχήμα 4.7 : Παράδειγμα 2ο IR.

Το αποτέλεσμα που φαίνεται να μας δίνει το μοντέλο είναι και πάλι το **atis_flight**, πράγμα που σε αυτήν την περίπτωση δεν είναι σωστό. Αυτό όμως οφείλεται στο γεγονός ότι πρώτον, τα δεδομένα δεν είναι σωστά ισορροπημένα, και δεύτερον στο ότι δεν υπάρχει στο dataset κατηγορία greetings. Σε αυτήν την περίπτωση το μοντέλο θα λάβει υπόψιν του τις περιπτώσεις της κατηγορίας που έχει μάθει καλύτερα θεωρώντας πως αυτό είναι το σωστό.

Για την αποφυγή αυτού του προβλήματος θα μιλήσουμε στο επόμενο κεφάλαιο όπου θα γίνει η υλοποίηση της τελικής εφαρμογής.

4.4 Slot Filling – SF

4.4.1 Εισαγωγή

Slot Filling είναι η διαδικασία του εντοπισμού συνεχόμενων περιοχών λέξεων σε μια έκφραση που αντιστοιχεί σε ορισμένες παραμέτρους (slots), μέσα από ένα ερώτημα του χρήστη.

Το Slot Filling είναι μια από τις πιο σημαντικές προκλήσεις στα σύγχρονα chatbots, καθώς είναι απαραίτητο για τη διαδικασία εξαγωγής της σημαντικής πληροφορίας μέσα από τα δεδομένα που εισάγει ο χρήστης, τα μοντέλα που χρησιμοποιούνται για αυτήν τη διαδικασία λειτουργούν με εποπτευόμενη μάθηση και είναι αρκετά αποτελεσματικά χρησιμοποιώντας μεγάλα σύνολα δεδομένων κατά την εκπαίδευση τους.

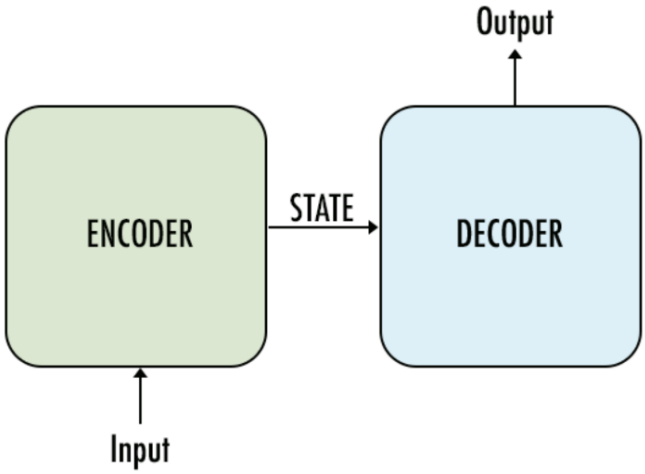
Βάση της εικόνας 4.1, ο σκοπός της διαδικασίας slot filling είναι το μοντέλο που χρησιμοποιεί να προβλέπει μια ετικέτα της μορφής IOB2 για κάθε token της εισαγόμενης έκφρασης του χρήστη.

Για την υλοποίηση αυτής της εργασίας, δημιουργήθηκε ένα νευρωνικό δίκτυο RNN με αρχιτεκτονική μοντέλων Seq2seq.

4.4.2 Sequence to Sequence

Τα μοντέλα Sequence to Sequence (Seq2seq) είναι μια ειδική κατηγορία αρχιτεκτονικών RNN και βρίσκουν εφαρμογή σε περιπτώσεις όπως τη μηχανική μετάφραση, τη σύνοψη κειμένων, την υλοποίηση chatbots και πολλών άλλων [45-47].

Η πιο κοινή αρχιτεκτονική που χρησιμοποιείται για τη δημιουργία μοντέλων seq2seq είναι η αρχιτεκτονική Encoder – Decoder. Ένα μοντέλο Encoder – Decoder παρέχει ένα μοτίβο για την αντιμετώπιση προβλημάτων πρόβλεψης αλληλουχίας σε ακολουθίες [47].

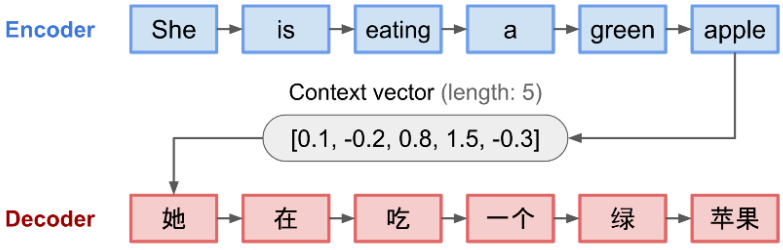


Σχήμα 4.8 : Encoder – Decoder [46].

Στην πραγματικότητα το νευρωνικό δίκτυο αποτελείται από δύο μοντέλα, έναν Encoder και έναν Decoder όπου και τα δύο είναι RNN μοντέλα, εσωτερικά αυτά τα δύο μοντέλα αποτελούνται από μοντέλα LSTM ή GRU (στη δική μας περίπτωση LSTM)[46].

Ο Encoder λαμβάνει μια ακολουθία εισόδου και επεξεργάζεται κάθε token της στην προσπάθεια του να ενσωματώσει όλη την πληροφορία της ακολουθίας σε ένα διάνυσμα σταθερού μήκους το οποίο ονομάζεται και διάνυσμα εσωτερικής κατάστασης ή διάνυσμα πλαισίου και αποτελεί τα δεδομένα εισόδου του μοντέλου Decoder [48].

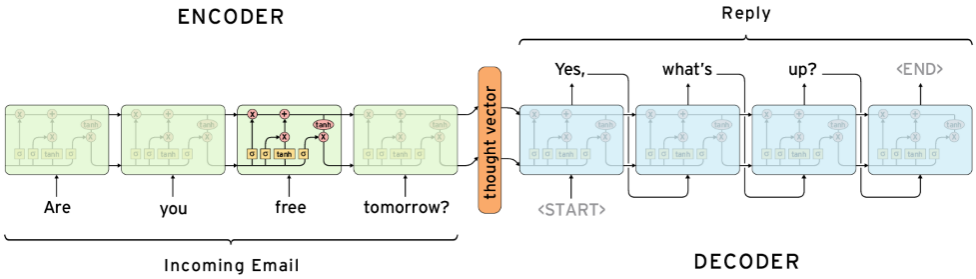
Το διάνυσμα πλαισίου αποτελείται από αριθμούς και το μέγεθος του είναι ο αριθμός των κρυφών μονάδων που εμπεριέχει ο encoder.



Σχήμα 4.9 : context vector. Source

Ο Decoder αποτελείται από το LSTM του οποίου οι αρχικές καταστάσεις αρχικοποιούνται στις τελικές καταστάσεις του LSTM που διαθέτει ο Encoder. Χρησιμοποιώντας αυτές τις καταστάσεις, ο Decoder ξεκινά να δημιουργεί τη διαδικασία εξόδου. Ο Decoder κατά τη διάρκεια της εκπαίδευσης του συμπεριφέρεται λίγο διαφορετικά από τον Encoder καθώς απαιτείται να χρησιμοποιήσουμε δύο tensors καθοδηγητές, έναν για να τον καθοδηγεί έτσι ώστε να ακολουθεί το σωστό αποτέλεσμα, κι έναν όπου καθορίζει τα true targets βάση του πρώτου tensor. Αυτό έχει ως αποτέλεσμα και την ταχύτερη εκπαίδευση του [48].

Κατά την εξαγωγή των συμπερασμάτων, η είσοδος του Decoder σε κάθε χρονικό βήμα αποτελεί την έξοδο του από την προηγούμενη χρονική στιγμή [48].



Σχήμα 4.10 : Παράδειγμα Encoder-Decoder. Source

4.4.3 Υλοποίηση

Για την υλοποίηση του μοντέλου θα χρησιμοποιηθεί ένα seq2seq νευρωνικό δίκτυο με μοντέλα encoder-decoder της βιβλιοθήκης του tensorflow. Επίσης, για λεξικό εισόδου θα χρησιμοποιηθεί το λεξικό του spacy με την αριθμητική μορφή των vectors που χρησιμοποιεί το ATIS και για λεξικό εξόδου θα χρησιμοποιηθεί το λεξικό εξόδου του ATIS με τα slots.

Καθώς το dataset μας παρέχει τα query vectors έτοιμα βάση του λεξικού εισόδου που διαθέτει, το πρώτο πράγμα που χρειάζεται είναι να γίνει η μετατροπή με το λεξικό του spacy. Για να το πετύχουμε αυτό χρησιμοποιείται η μέθοδος που φαίνεται στο παρακάτω σχήμα :

```
# recreate query vectors with spacy
def create_spacy_tensors(queries, t2ispacy, i2t, nlp):
    input_tensor_spacy = []

    for i in range(len(queries)):
        input_tensor = []

        #input tensor
        for x in queries[i]:
            input_tensor.append(t2ispacy[nlp.vocab.strings[i2t[x]]])

        inp = np.array(input_tensor, dtype=np.int64)
        input_tensor_spacy.append(inp)

    return input_tensor_spacy
```

Σχήμα 4.11 : Δημιουργία νέων vectors.

Όπου δέχεται ως παραμέτρους τη λίστα με τα query vectors του dataset, δύο dictionaries τα t2ispacy και i2t για τις αντιστοιχήσεις των τιμών των vectors με τις πραγματικές λέξεις και τελευταίο το object του spaCy. Και στη συνέχεια μετατρέπει το κάθε query vector ξεχωριστά σύμφωνα με τα δεδομένα που χρειαζόμαστε.

Κάθε ακέραιος αριθμός που εμφανίζεται μέσα σε ένα query vector αντιπροσωπεύει τον αριθμό της θέσης της λέξης (token) μέσα στο λεξικό του spaCy. Ο λόγος για τον οποίο χρειάζεται αυτή η διαδικασία είναι για να επεκταθεί το αρχικό λεξικό καθώς το μοντέλο του spaCy που χρησιμοποιείται διαθέτει 685.000 μοναδικές λέξεις κλειδιά. Με αυτόν τον τρόπο θα μπορούμε αργότερα να εκμεταλλευτούμε περισσότερες πόλεις στην εφαρμογή καθώς το ATIS dataset στις εκφράσεις του εμπεριέχει συγκεκριμένες περιοχές της Αμερικής.

Μετά την παραπάνω διαδικασία, το επόμενο κομμάτι είναι η υλοποίηση των tensors. Θα χρειαστούμε δύο tensors, input και output. Όπως με το προηγούμενο μοντέλο για το intent recognition χρειάστηκε η διαδικασία του sequence padding, το ίδιο θα χρειαστεί να γίνει και σε αυτό το μοντέλο. Για την υλοποίηση των tensors θα πρέπει να βρεθεί η μεγαλύτερη έκφραση σε μέγεθος, tokens μέσα από τα δεδομένα του dataset. Όπου, στην περίπτωση μας η μεγαλύτερη έκφραση του ATIS περιέχει 48 tokens οπότε το σχήμα των tensors θα είναι (X,48).

Έχοντας και τους tensors μπορεί να γίνει πλέον η σχεδίαση του νευρωνικού δικτύου και των μοντέλων.

```
def encoder_layer(len_input, vocab_size, embedding_dim, units):
    encoder_inputs = Input(shape=(len_input,))
    encoder_emb = Embedding(input_dim=vocab_size, output_dim=embedding_dim)
    encoder_lstm = LSTM(units=units, return_sequences=True, return_state=True)
    encoder_outputs, state_h, state_c = encoder_lstm(encoder_emb(encoder_inputs))
    encoder_states = [state_h, state_c]

    encoder = Model(encoder_inputs, [encoder_outputs, state_h, state_c])
    return encoder_inputs, encoder_states, encoder
```

Σχήμα 4.12 : Encoder Model.

Στο παραπάνω σχήμα φαίνεται το μοντέλο του Encoder, η μέθοδος δέχεται σαν παραμέτρους το μέγεθος της μεγαλύτερης έκφρασης, το μέγεθος του λεξικού εισόδου, το embedding dimensionality = 256, και τον αριθμό των hidden states για το LSTM (1024).

Το αντικείμενο Input μας βοηθά να σχεδιάσουμε ένα tensor για το μοντέλο καθώς γνωρίζουμε μόνο την είσοδο και την έξοδο μέχρι στιγμής. Στη συνέχεια το embedding layer όπου δημιουργεί τα embedding vectors που θα χρειαστούμε. Έπειτα το LSTM layer που του δίνουμε ως παραμέτρους τις μονάδες που θα περιέχει, return_state=True για να επιστρέφει ολόκληρη την ακολουθία της εξόδου. Και στη συνέχεια υλοποιούμε το μοντέλο encoder με τα hidden state, cell state και το output που πήραμε από το LSTM καθώς και το input shape. Και τέλος, η μέθοδος επιστρέφει το σχήμα εισόδου, τις καταστάσεις του LSTM και το μοντέλο Encoder.

```python
def decoder_layer(vocab_size, embedding_dim, units, encoder_states):
    decoder_inputs = Input(shape=(None,))
    decoder_emb = Embedding(input_dim=vocab_size, output_dim=embedding_dim)
    decoder_lstm = LSTM(units=units, return_sequences=True, return_state=True)
    decoder_output, _, _ = decoder_lstm(decoder_emb(decoder_inputs), initial_state=encoder_states)

    decoder_d1 = Dense(units, activation='relu')
    decoder_d2 = Dense(vocab_size, activation='softmax')
    decoder_out = decoder_d2(Dropout(rate=0.2)(decoder_d1(Dropout(rate=0.2)(decoder_output))))

    #testing
    dec_state_h = Input(shape=(units,))
    dec_state_c = Input(shape=(units,))
    decoder_res, decoder_h, decoder_c = decoder_lstm(decoder_emb(decoder_inputs), initial_state=[dec_state_h,dec_state_c])
    decoder_res_out = decoder_d2(decoder_d1(decoder_res))

    decoder = Model(inputs=[decoder_inputs, dec_state_h, dec_state_c], outputs=[decoder_res_out, decoder_h, decoder_c])
    return decoder_inputs,decoder_out, decoder
```

Σχήμα 4.13 : Decoder Model.

Στη συνέχεια, στο παραπάνω σχήμα υλοποιείται το μοντέλο Decoder. Η μέθοδος δέχεται ως παραμέτρους το μέγεθος του λεξικού εξόδου, το embedding dimensionality = 256, τα units = 1024 και τις αρχικοποιημένες καταστάσεις του encoder.

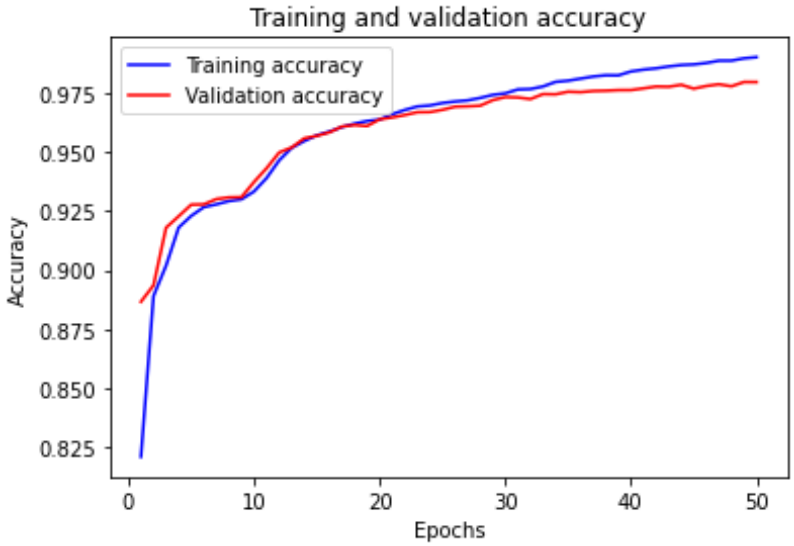
Αντίστοιχα υλοποιούμε το input shape, μόνο που σε αυτήν την περίπτωση δε μας ενδιαφέρει το σχήμα των διαστάσεων που θα πάρουμε εφόσον δε γνωρίζουμε το μέγεθος. Στη συνέχεια, αντίστοιχη δουλειά θα κάνει το embedding layer με τη μόνη διαφορά ότι εδώ του δίνουμε το μέγεθος του λεξικού εξόδου. Έπειτα υλοποιείται το LSTM layer που του δίνουμε τα states του encoder για να το κατευθύνουμε. Και στη συνέχεια υλοποιούνται δύο Dense layers όπου, το πρώτο layer συνδέεται με τα αποτελέσματα του LSTM layer και βοηθά με την αλλαγή των διαστάσεων των δεδομένων ενώ το δεύτερο συνδέεται με το πρώτο για να αντιστοιχίσει τα δεδομένα αυτά με το λεξικό. Και στη συνέχεια υλοποιείται το μοντέλο του decoder. Το τελικό μοντέλο φαίνεται παρακάτω :

```python
def create_model(len_input, vocab_input, vocab_output, embedding_dim, units):
    # encoder_layer
    encoder_inputs , encoder_states, encoder = encoder_layer(len_input, vocab_input, embedding_dim, units)
    #decoder_layer
    decoder_inputs , decoder_out, decoder = decoder_layer(vocab_output, embedding_dim, units, encoder_states)

    model = Model([encoder_inputs, decoder_inputs], decoder_out)
    model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['sparse_categorical_accuracy'])

    return model, encoder, decoder
```

Σχήμα 4.14 : Υλοποίηση μοντέλου.

Κατά την υλοποίηση του μοντέλου χρησιμοποιείται adam optimizer σε 50 εποχές και sparse categorical crossentropy ως loss function καθώς τα δεδομένα μας αποτελούνται από ακέραιους αριθμούς, αυτή η μέτρηση έχει ως αποτέλεσμα την εξοικονόμηση χρόνου από τη μνήμη.

Κατά την προσαρμογή του μοντέλου σε 50 εποχές παρατηρείται overfitting καθώς το validation accuracy φαίνεται να μένει σταθερό γύρω στην $35^{η}$ εποχή ενώ το training accuracy συνεχίζει με μια κλίση προς τα πάνω, το training accuracy είναι 97,8% και το validation accuracy 96,6% :



Σχήμα 4.15 : Εκπαίδευση του μοντέλου.

Στη συνέχεια, εφόσον το αρχικό μας μοντέλο έχει σχεδιαστεί και μπορεί να χρησιμοποιηθεί για να κάνει προβλέψεις και τα βάρη του έχουν εκπαιδευτεί για αυτή τη λειτουργία, δεν μπορούμε να χρησιμοποιήσουμε το συγκεκριμένο μοντέλο καθώς η συγκεκριμένη αρχιτεκτονική δεν έχει σχεδιαστεί για να λειτουργεί στη συνέχεια αναδρομικά. Αυτό θα ήταν πρόβλημα κάθε φορά που θα περιμέναμε μια απάντηση από το νευρωνικό δίκτυο. Αντίθετα, θα πρέπει να υλοποιήσουμε δύο καινούρια μοντέλα, ένα για την κωδικοποίηση των εκφράσεων όπου θα χρησιμοποιήσουμε τους tensors που υλοποιήθηκαν νωρίτερα για τον encoder και, το inference model το οποίο θα δέχεται τα δεδομένα του encoder και θα προβλέπει τα slots όπου, σε αυτό το μοντέλο θα δημιουργήσουμε καινούριους tensors.

```
def create_inference_encoder(encoder_inputs, encoder_outputs, state_h, state_c):
    #inference encoder
    return Model(encoder_inputs, [encoder_outputs, state_h, state_c])

def create_inference_model(units, decoder_lstm, decoder_emb, decoder_d1, decoder_d2):
    decoder_inputs = Input(shape=(None,))
    state_h = Input(shape=(units,),)
    sate_c = Input(shape=(units,),)


    decoder_res, decoder_h, decoder_c = decoder_lstm(decoder_emb(decoder_inputs), initial_state=[state_h, sate_c])
    decoder_out = decoder_d2(decoder_d1(decoder_res))
    inference_model = Model(inputs=[decoder_inputs, state_h, sate_c], outputs=[decoder_out, decoder_h, decoder_c])

    return inference_model
```

Σχήμα 4.16 : Inference models.

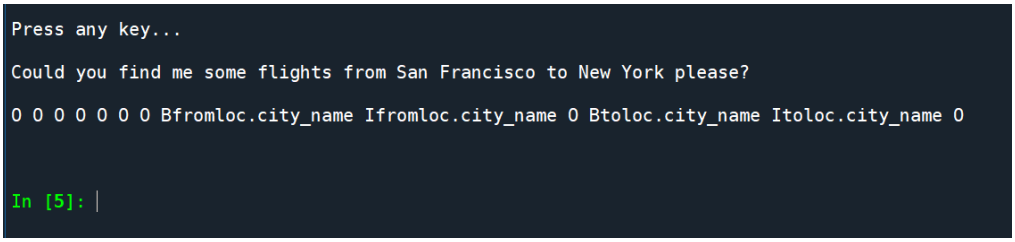
Στο παραπάνω σχήμα, στην πρώτη μέθοδο υλοποιείται ο inference encoder με τους tensors του πρώτου μοντέλου.

Στη δεύτερη μέθοδο υλοποιείται το inference model όπου, χρησιμοποιούνται τα layers του προηγούμενου decoder πλην τους tensors καθώς το μοντέλο απαιτεί την κωδικοποίηση τους ως αρχικές καταστάσεις του inference encoder. Και, επειδή το inference model θα λειτουργεί αυτόνομα σε σχέση

με τον προηγούμενο decoder, αυτές οι καταστάσεις θα παρέχονται ως είσοδοι στο μοντέλο γι' αυτό και ορίζονται ως inputs.

4.4.4 Παραδείγματα μοντέλου

Ας δούμε μια πρώτη εκτίμηση στο μοντέλο του SF με ένα παράδειγμα. Δίνουμε ως είσοδο στο μοντέλο τη δεδομένη πρόταση «Could you find me some flights from San Francisco to New York please?», το αποτέλεσμα που θα λάβουμε θα είναι το εξής :



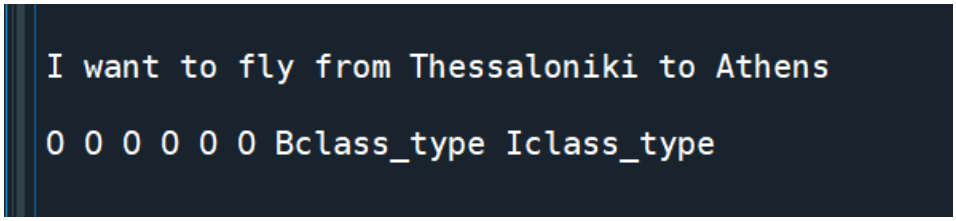
Σχήμα 4.17 : Παράδειγμα 1ο SF.

Στην απάντηση που παίρνουμε στο παραπάνω σχήμα, βλέπουμε ότι η αντιστοίχιση με τα slots στα πρώτα 7 tokens (could you .. from) ισοδυναμεί με O, όπου αυτό σημαίνει πως δεν αναγνωρίζει σημαντική πληροφορία.

Ενώ στα tokens «San Francisco» αντιστοιχούν τα slots **Bfromloc.city_name** και **Ifromloc.cityname** τα οποία υποδηλώνουν πόλη αφετηρίας και πως τα δύο αυτά tokens είναι συνδυασμός (Beginning - Inside).

Στη συνέχεια το token «to» αντιστοιχίζεται με O ενώ τα tokens «New York» με τα slots **Btoloc.city_name** και **Itoloc.city_name** όπου υποδηλώνουν την πόλη προορισμού κι επίσης πως είναι συνδυασμός. Και τέλος, το τελευταίο token αντιστοιχεί με O. Επομένως, τα αποτελέσματα στο συγκεκριμένο παράδειγμα είναι σωστά.

Ένα δεύτερο παράδειγμα του μοντέλου φαίνεται στο παρακάτω σχήμα, δίνεται ως είσοδος η δεδομένη έκφραση «I want to fly from Thessaloniki to Athens» :



Σχήμα 4.18 : Παράδειγμα 2ο SF.

Σε αυτό το σημείο τα αποτελέσματα που λαμβάνουμε από την έξοδο του μοντέλου είναι λάθος καθώς αντιλαμβάνεται πώς τα πρώτα 6 tokens αντιστοιχούν με άχρηστη πληροφορία, ενώ το 6ο token εμπεριέχει τη χρήσιμη πληροφορία της πόλης αφετηρίας. Επίσης στα τελευταία δύο tokens αντιστοιχίζει

δύο λάθος slot labels που πραγματικά υποδεικνύουν τον τύπο της κλάσης του εισιτηρίου ενώ στο δικό μας παράδειγμα το token εμπεριέχει πληροφορία πόλεως.

Αυτό το πρόβλημα οφείλεται στο ότι συνολικά στα δεδομένα του atis περιλαμβάνονται ορισμένες πόλεις σε όλα του τα παραδείγματα σε συνδυασμό με τις λέξεις κλειδιά «from» και «to». Παρόλο που χρησιμοποιείται το λεξικό του spaCy για αυτό το πρόβλημα, θα δούμε στη συνέχεια τη λύση του προβλήματος. Οι δύο συγκεκριμένες πόλεις του παραδείγματος προφανώς δεν υπάρχουν μέσα στο σύνολο των εκφράσεων των δεδομένων, είναι δεδομένα όπου το μοντέλο δεν έχει ξανά δει.

Για την περίπτωση δεδομένων όπως του παραπάνω παραδείγματος, καθώς τα δεδομένα υλοποίησης δεν ήταν αρκετά ενδεχομένως και η δομή του μοντέλου, θα δούμε στο επόμενο κεφάλαιο μια λύση για το συγκεκριμένο πρόβλημα χρησιμοποιώντας αντικείμενα Matchers της βιβλιοθήκης του spaCy.

4.5 Επίλογος

Στο κεφάλαιο αυτό αναφέρθηκαν τα δεδομένα του ATIS dataset καθώς και οι μορφές των δεδομένων που εμπεριέχει, στη συνέχεια αναπτύχθηκαν και αναλύθηκαν οι δύο από τις σημαντικές εργασίες που χρησιμοποιούν τα chatbots μαζί με την υλοποίηση μοντέλων για τους σκοπούς της παρούσας Π.Ε.

Κεφάλαιο 5ο: Τελική Εφαρμογή

5.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα αναφερθούν και θα αναλυθούν οι λειτουργίες και οι τεχνικές που υλοποιήθηκαν καθώς και το framework που χρησιμοποιήθηκε για να δημιουργηθεί η τελική εφαρμογή.

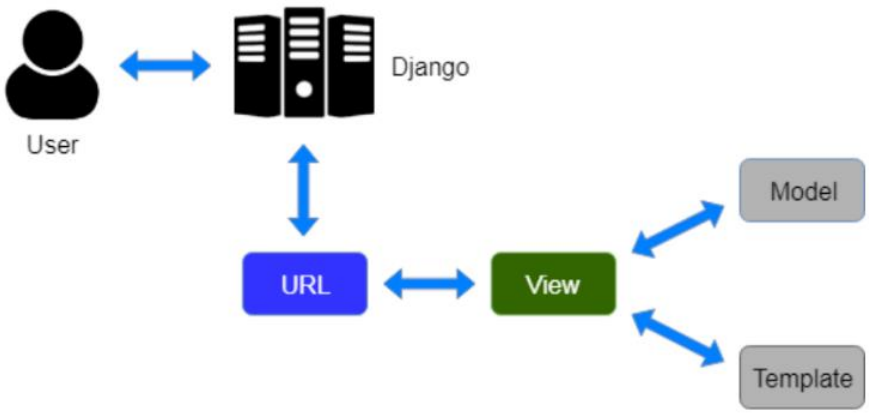
5.2 Django Framework

5.2.1 Αρχιτεκτονική

Το Django είναι ένα open source web framework και λειτουργει με τη γλώσσα προγραμματισμού Python. Επιτρέπει την ταχεία ανάπτυξη ασφαλών και διατηρήσιμων ιστότοπων. Το Django ακολουθεί την αρχιτεκτονική MVT (Model – View – Template)[49].

Το MVT είναι ένα design pattern το οποίο αποτελείται από τρία σημαντικά στοιχεία, το model, το view και το template[49].

- Το model βοηθά στο χειρισμό της βάσης δεδομένων, είναι ένα επίπεδο πρόσβασης το οποίο χειρίζεται τα δεδομένα.
- Το view χρησιμοποιείται για την εκτέλεση της επιχειρηματικής λογικής και την αλληλεπίδραση με το model για τη μεταφορά δεδομένων και την προσκόμιση τους στο template.
- Το template είναι ένα επίπεδο παρουσίασης που χειρίζεται πλήρως το κομμάτι διεπαφής του χρήστη.

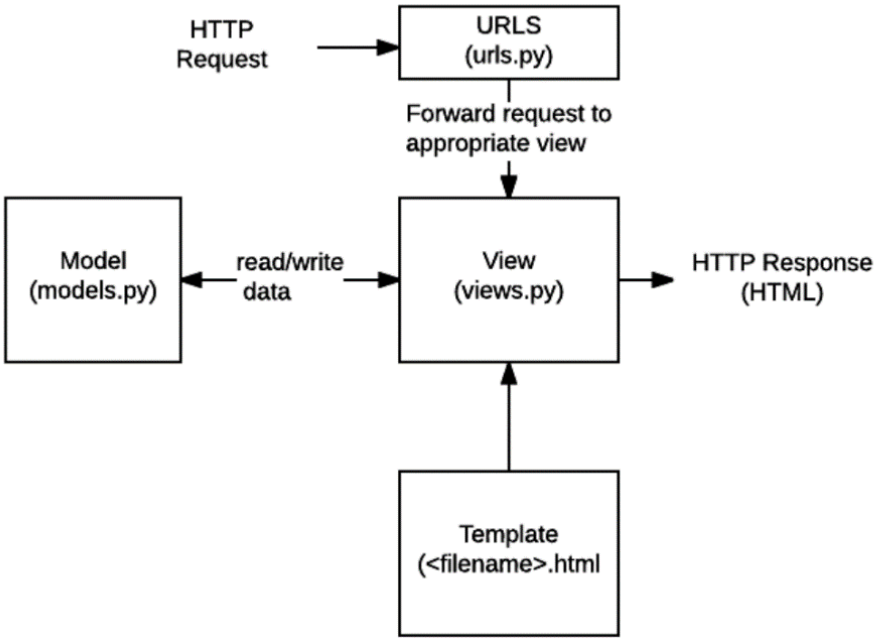


Σχήμα 5.1 : παράδειγμα αιτήματος από χρήστη. Source

Στην παραπάνω εικόνα φαίνεται ένα παράδειγμα request από το χρήστη. Ο χρήστης ζητά έναν πόρο στο Django, το Django λειτουργεί ως ελεγκτής και ελέγχει το διαθέσιμο πόρο στη διεύθυνση URL, εάν αντιστοιχεί η διεύθυνση αυτή με κάποιο πόρο τότε καλείται ένα view το οποίο αλληλεπιδρά με το model

και το template ετοιμάζει ένα πρότυπο, το Django απαντά στο χρήστη και παρουσιάζει τα αποτελέσματα.

Οι διευθύνσεις URL χρησιμοποιούνται για την ανακατεύθυνση των αιτημάτων HTTP στις κατάλληλες προβολές.



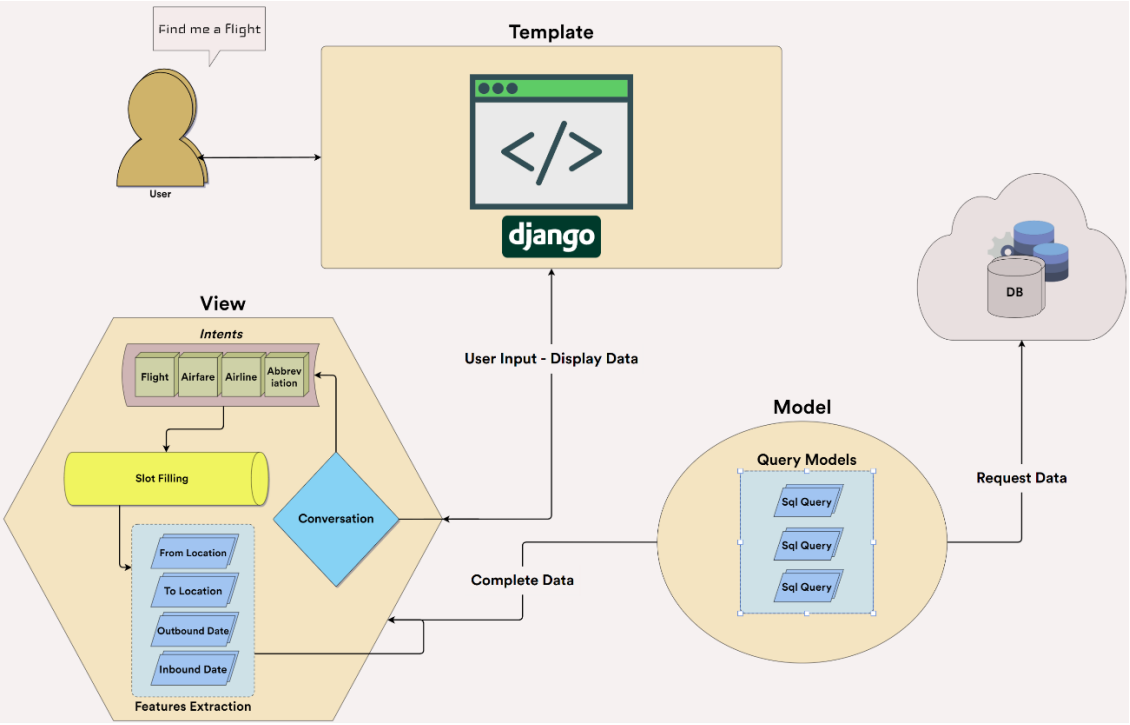
Σχήμα 5.2 : Αρχιτεκτονική Django. Source

5.3 Δομή Εφαρμογής

5.3.1 Δομή

Η εφαρμογή είναι γραμμένη με το Django framework και χρησιμοποιεί τη λογική του MVT. Το framework αυτό, από τη φύση του χρησιμοποιεί ως κύρια πλατφόρμα για την ανάπτυξη εφαρμογών το **WSGI (Web Server Gateway Interface)**, το πρότυπο αυτό χρησιμοποιείται για να μεταβιβάζει τα σύγχρονα αιτήματα (synchronous requests) των χρηστών προς τη web εφαρμογή.

Στην περίπτωση αυτής της εφαρμογής, εφόσον το chatbot απαιτεί ένα κανάλι επικοινωνίας χρησιμοποιείται το πρότυπο **ASGI (Asynchronous Server Gateway Interface)** το οποίο σχεδιάστηκε για να εκτελείται με ασύγχρονο τρόπο κι επίσης να υποστηρίζει πολλά πρωτόκολλα καθώς και κανάλια επικοινωνίας.



Σχήμα 5.3 : Δομή Εφαρμογής. Created on Terrastruct.

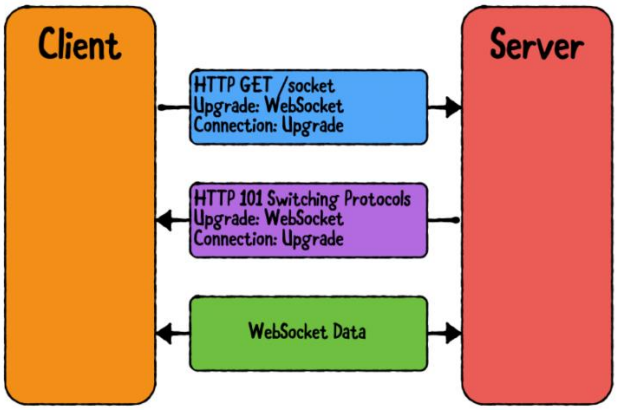
Στο παραπάνω σχήμα φαίνεται η δομή της εφαρμογής. Ο χρήστης στέλνει ένα αίτημα στο server, το url έρχεται για να κάνει την κατάλληλη σύνδεση με το view και ο server απαντάει στο αίτημα ανοίγοντας ένα κανάλι επικοινωνίας. Έπειτα ο χρήστης μπορεί να ξεκινήσει τη συζήτηση ως εξής :

- Ο χρήστης γράφει την ερώτηση στο chat και την στέλνει
- Ο server ακούει στην ερώτηση, παίρνει το input text και το περνάει πρώτα στο μοντέλο του IR για να καταλάβει την πρόθεση του χρήστη. Στη συνέχεια,
- Το αποτέλεσμα του μοντέλου IR θα δημιουργήσει ένα object της αντίστοιχης Intent class κι έπειτα τα δεδομένα θα περάσουν στο μοντέλο SF.
- Τα αποτελέσματα του μοντέλου SF θα χρησιμοποιηθούν από τα φίλτρα εξαγωγής της πληροφορίας. Αν η πληροφορία που εξήγαγαν καλύπτει τα απαιτούμενα δεδομένα των ζητούμενων αποτελεσμάτων του χρήστη τότε
    - ετοιμάζεται ένα αίτημα προς τη βάση δεδομένων και,
    - ο server απαντάει σύμφωνα με τα δεδομένα που βρήκε.
- Αν οι πληροφορίες που εξήχθησαν δεν είναι αρκετές, τότε ο server απαντάει κατάλληλα ζητώντας περισσότερες πληροφορίες από το χρήστη.

και στη συνέχεια διεξάγετε η ίδια διαδικασία από την αρχή.

5.3.2 WebSocket API

Καθώς το πρωτόκολλο HTTP είναι ακαταστατιτό, αυτό σημαίνει πως δεν μπορεί να κρατήσει ανοιχτεί μια σύνδεση με το server πράγμα που το καθιστά ακατάλληλο για τη συγκεκριμένη εφαρμογή, θα χρειαστεί να χρησιμοποιήσουμε ένα διαφορετικό πρωτόκολλο, το websocket.



Σχήμα 5.4 : Διαδικασία διεκπεραίωσης WebSocket. Source

Το WebSocket API, είναι μια προηγμένη τεχνολογία που καθιστά δυνατό το άνοιγμα μιας αμφίδρομης διαδραστικής επικοινωνίας μεταξύ του προγράμματος περιήγησης του χρήστη και του διακομιστή της εφαρμογής[50]. Το API αυτό θα μας επιτρέπει να στέλνουμε μηνύματα στο server πάνω στην ίδια σύνδεση. Ο κώδικας υλοποίησης του WebSocket είναι σε Javascript.

```
// WebSocket
var loc = window.location;
var prtc = 'ws://';
if(loc.protocol == 'https:'){
  prtc = 'wss://';
}
var endpoint = prtc + window.location.host + window.location.pathname;
var socket = new WebSocket(endpoint);
// WebSocket end
```

Σχήμα 5.5 : Υλοποίηση WebSocket.

Το object ενός WebSocket μας παρέχει μεθόδους για τη διεξαγωγή της επικοινωνίας, ορισμένες από αυτές είναι οι εξής :

- **socket.onopen** : Με την οποία ανοίγει μια σύνδεση
- **socket.send** : Η οποία χρησιμοποιείται για την αποστολή των μηνυμάτων επικοινωνίας
- **socket.onmessage** : Όπου χρησιμοποιείται κατά τη διάρκεια που λαμβάνουμε ένα μήνυμα από το server
- **socket.close** : Που χρησιμοποιείται για το κλείσιμο μιας επικοινωνίας
- **socket.onerror** : Για την περίπτωση που συμβεί κάποιο σφάλμα κατά τη διάρκεια της επικοινωνίας.

```
//receive message
socket.onmessage = function(e){
    console.log("message", e);
};
//on error
socket.onerror = function(e){
    onsole.log("error", e);
};
// create connection
socket.onopen = function(e){
    console.log("open", e);
}
// send a message
socket.send("Message");
// close connection
socket.close();
```

Σχήμα 5.6 : Μέθοδοι υλοποίησης WebSocket.

5.3.3 URLs

Καθώς η εφαρμογή περιέχει μόνο το chatbot, θα χρειαστεί να δημιουργηθεί μόνο ένα url ανακατεύθυνσης αιτημάτων. Στο παρακάτω σχήμα φαίνεται η δήλωση του στο αρχείο **urls.py** :

```
urls.py 1
main > urls.py > ...
1  from django.urls import path
2  from . import views
3
4  urlpatterns = [
5      path("", views.index, name="index"),
6  ]
```

Σχήμα 5.7 : Αρχείο urls.py.

5.3.4 Views

Το αρχείο views.py περιλαμβάνει τα πρότυπα των υποσέλιδων. Είναι στην ουσία ο τρόπος για τη διεξαγωγή HTML κώδικα δυναμικά. Στην περίπτωση της παρούσας Π.Ε το πρότυπο θα είναι ένα :

views.py 2

main > views.py > ...

```python
from django.shortcuts import render
from .models import Greetings
from django.http import JsonResponse

from .apps import MainConfig

import random


def index(request):
    #get random greeting
    r = random.randint(1,5)
    greeting = Greetings.objects.get(id=r)

    my_dict = {
        "greeting" : greeting,
    }
    return render(request, 'main/base.html', my_dict)

```

Σχήμα 5.8 : Αρχείο views.py.

Στο σχήμα φαίνεται η μέθοδος index, η οποία περιλαμβάνει το κύριο πρότυπο της εφαρμογής. Μέσω της μεθόδου render αναπαράγει τον κώδικα του base.html αρχείου.

5.3.5 Models

Το αρχείο models.py περιέχει τα μοντέλα που χρησιμοποιούμε για την επικοινωνία με τη βάση δεδομένων μας. Κάθε μοντέλο αντιστοιχεί σε μια κλάση ενός πίνακα της βάση δεδομένων.

```python
class AirAskInbound(models.Model):
    id = models.AutoField(primary_key=True)
    response = models.TextField()

    def __str__(self):
        return self.id

class AirInbound(models.Model):
    id = models.AutoField(primary_key=True)
    response = models.TextField()

    def __str__(self):
        return self.id

# ===== Airport Codes - Cities - Mainlands

#main_airport_codes
class AirportCodes(models.Model):
    id = models.AutoField(primary_key=True)

    airport_code = models.TextField()
    airport_cityname = models.TextField()

    airport_world_area_code = models.IntegerField()

    def __str__(self):
        return self.id
```

Σχήμα 5.9 : Ενδεικτικό παράδειγμα μοντέλων.

Τα μοντέλα που υπάρχουν είναι για την αναζήτηση response απαντήσεων μέσα από lookup tables καθώς επίσης και για τη διεξαγωγή σημαντικών πληροφοριών όπως iata κωδικοί, χώρες προέλευσης, πόλεις κλπ.

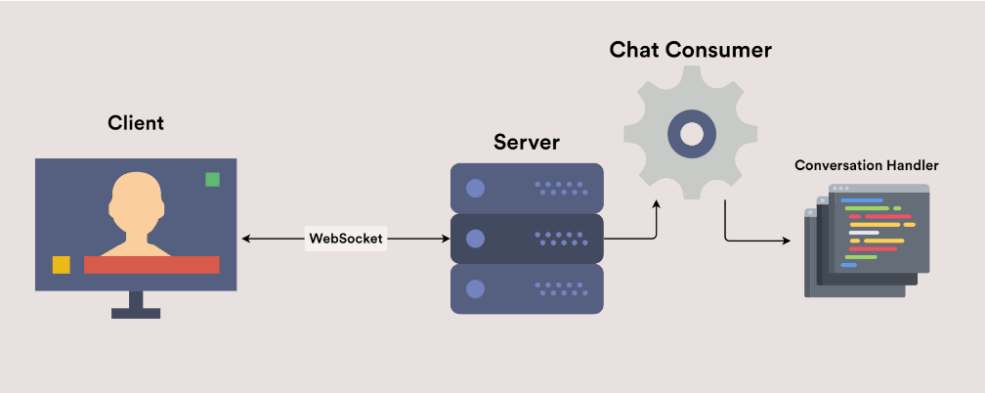
Το Django μεταδίδει τις πληροφορίες των δεδομένων μέσω των **migrations**. Τα migrations είναι ο τρόπος με τον οποίο μπορούν να γίνουν αλλαγές στα δεδομένα, αλλαγές όπως η προσθήκη πεδίων, διαγραφή μοντέλου, δημιουργία καινούριου μοντέλου κλπ.

5.4 Chat Consumer

Από τη μεριά του server, για να μπορέσουμε να διαχειριστούμε τα μηνύματα του χρήστη και το κανάλι επικοινωνίας χρειάζεται η υλοποίηση ενός consumer. Ο consumer είναι αυτός που θα αναλάβει τη διαδικασία της επικοινωνίας και λειτουργεί όπως το view διαχειρίζεται τα request μέσω του WSGI. Καθώς τα channels του Django έχουν σχεδιαστεί περισσότερο για διαλειτουργικότητα, ο consumer είναι ο άμεσος τρόπος που παρέχουν τα κανάλια με μια αφαιρετικότητα για την υλοποίηση ASGI εφαρμογών.

Για την παρούσα εφαρμογή υλοποιήθηκε η κλάση **ChatConsumer** όπου κληρονομεί από την κλάση **AsyncConsumer**. Η κλάση αυτή υλοποιεί τις παρακάτω μεθόδους οι οποίες είναι υπεύθυνες για την αποστολή και τη λήψη των μηνυμάτων του χρήστη.

- **websocket_connect** : η οποία στέλνει πίσω στον client ένα μήνυμα για την εδραίωση της σύνδεσης της επικοινωνίας.
- **websocket_receive** : η οποία φροντίζει να λάβει το μήνυμα από τον client, να το παραδώσει στη μέθοδο **getConversation** για να λάβει την απάντηση του bot και στη συνέχεια να το παραδώσει πίσω στον client.
- **websocket_disconect** : η οποία επιστρέφει στον client το μήνυμα πως η σύνδεση διακόπηκε.
- **getConversation** : η οποία λαμβάνει το μήνυμα μέσω της **websocket_receive** και φροντίζει να το παραδώσει στο σύστημα για να λάβει την απάντηση του bot, στη συνέχεια το παραδίδει πίσω στην **websocket_receive**.

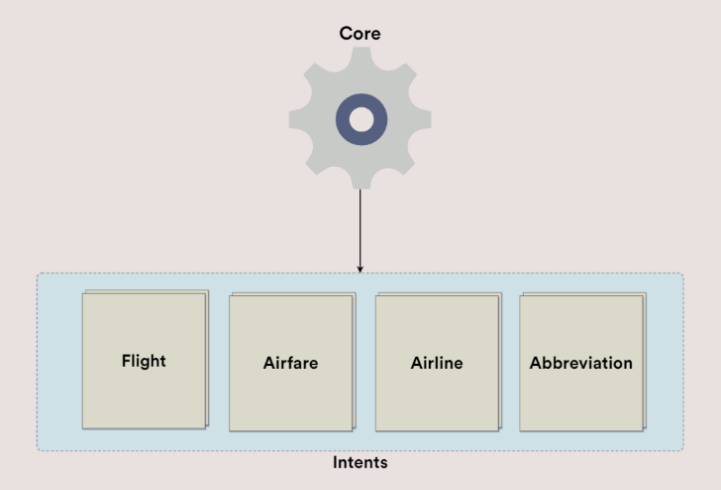


Σχήμα 5.10 : Chat Consumer. Created on Terrastruct.

5.5 Conversation

Η μέθοδος getConversation του consumer που είδαμε παραπάνω, είναι ο διαμεσολαβητής του καναλιού της εφαρμογής με την εσωτερική διαδικασία της.

Για να ξεκινήσει η διαδικασία του εντοπισμού της πρόθεσης του χρήστη καθώς και οι υπόλοιπες λειτουργίες, υλοποιήθηκε η κλάση **Conversation**. Η κλάση αυτή περιλαμβάνει τη μέθοδο **core** η οποία είναι υπεύθυνη για την υλοποίηση της αντίστοιχης πρόθεσης αλλά και τη διατήρηση της μέχρι το τέλος του διαλόγου με το χρήστη.



Σχήμα 5.11 : Core μέθοδος. Created on Terrastruct.

Η μέθοδος **core** λαμβάνει τα δεδομένα από τη μέθοδο getConversation που είδαμε παραπάνω και, το πρώτο πράγμα που κάνει είναι να δει αν υπάρχει ήδη μια ροή συζήτησης.

Αν δεν υπάρχει, τότε τρέχει τον αλγόριθμο IR δίνοντας του τα δεδομένα εισόδου που κατέχει και σύμφωνα με το αποτέλεσμα που θα πάρει δημιουργεί ένα object του αντίστοιχου intent το οποίο θα το διατηρήσει μέχρι να εξυπηρετηθεί το αίτημα της συγκεκριμένης πρόθεσης του χρήστη. Το object αυτό θα είναι υπεύθυνο για τη διαδικασία του του διαλόγου.

Στην περίπτωση που υπάρχει ήδη μια ροή, τότε η μέθοδος απλά «ακολουθεί» τη διαδικασία με το ήδη υπάρχον object.

5.6 Υλοποίηση Intents

Όπως είδαμε νωρίτερα, το ATIS dataset δεν περιέχει σωστά ισορροπημένα δεδομένα με αποτέλεσμα να επηρεάσει το μοντέλο ταξινόμησης για το intent recognition.

Για το λόγο αυτό, επιλέχθηκαν να υλοποιηθούν τέσσερα συγκεκριμένα intents :

- **Flight** : όπου ο χρήστης θα μπορεί να αναζητεί πληροφορίες για τις φθηνότερες πτήσεις.
- **Airfare** : στο οποίο ο χρήστης θα αναζητεί πληροφορίες για τα φθηνότερα αεροπορικά εισιτήρια.

- **Airline** : που ο χρήστης θα αναζητεί πληροφορίες για τις εταιρείες που εξυπηρετούν τις συγκεκριμένες γραμμές.
- **Abbreviation** : όπου το bot θα αντιλαμβάνεται συντομογραφίες, καθώς το dataset δεν περιλαμβάνει greetings αυτό το intent θα χρησιμοποιηθεί για greetings.

Για τις πρώτες τρεις περιπτώσεις έχουν δημιουργηθεί κλάσεις οι οποίες υλοποιούν ένα σύνολο μεθόδων. Κάθε κλάση υλοποιεί αυτές τις μεθόδους και κάνει ένα request για να βρει τα διαφορετικά αποτελέσματα ανά την περίπτωση. Εφόσον δρομολογηθεί το intent, το bot ακολουθεί μια ροή διαλόγου μέχρι να συλλέξει όλη την απαραίτητη πληροφορία που χρειάζεται για να καταφέρει να εξυπηρετήσει το αίτημα του χρήστη.

5.6.1 Flight Class

Η κλάση Flight περιγράφει την πρόθεση της πτήσης. Μέσα από τις μεθόδους που υλοποιεί, συλλέγει τα σημαντικά δεδομένα του χρήστη με σκοπό την εύρεση πτήσεων :

- **scanMessage** : Η οποία είναι υπεύθυνη για τη διαχείριση της εξαγωγής των δεδομένων.
- **getText** : όπου, διατηρεί μια συγκεκριμένη ροή για τη συζήτηση βάση των πληροφοριών που λείπουν για την ολοκλήρωση του αιτήματος.
- **getFromLoc** : η οποία χρησιμοποιεί φίλτρα για να εξάγει συγκεκριμένα τα δεδομένα της τοποθεσίας αφετηρίας.
- **getToLoc** : όπου χρησιμοποιεί φίλτρα για να εξάγει την πληροφορία τοποθεσίας προορισμού.
- **getOutboundDate** : όπου χρησιμοποιεί φίλτρα για την εξαγωγή της ημερομηνίας αναχώρησης.
- **getInboundDate** : η οποία είναι για την εξαγωγή της ημερομηνίας επιστροφής σε περίπτωση που υπάρχει.
- **requestAPI** : όπου με την οποία θα σχηματιστεί ένα request για να λάβουμε τα αποτελέσματα της αναζήτησης των πτήσεων.

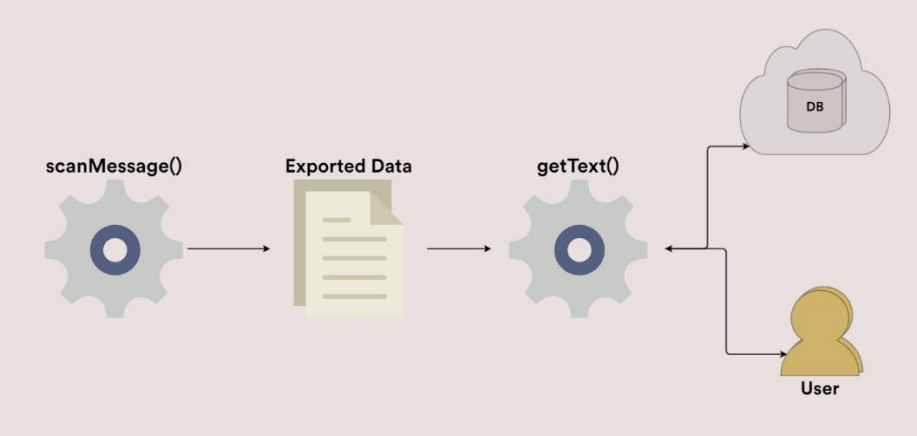
Οι πληροφορίες που θα πρέπει να διεξαχθούν για να ολοκληρωθεί η αναζήτηση είναι η χώρα προέλευσης η οποία αναζητείται βάση της πόλης αναχώρησης. Ο κωδικός iata του αεροδρομίου της πόλης αναχώρησης, ο κωδικός iata του αεροδρομίου της πόλης προορισμού αντίστοιχα, η ημερομηνία αναχώρησης και η ενδεχόμενη ημερομηνία επιστροφής εφόσον υπάρχει.

Η διαδικασία έχει ως εξής, πρώτα εκτελείται η μέθοδος scanMessage η οποία περνάει το κείμενο του χρήστη μέσα από τον αλγόριθμο SF κι έπειτα ελέγχει μέσα από φίλτρα για όλες τις σημαντικές πληροφορίες καθώς ο χρήστης θα μπορούσε να εισάγει όλη την πληροφορία σε μία πρόταση. Στη συνέχεια, γίνεται ο έλεγχος μέσω της μεθόδου getText η οποία ελέγχει για το μήνυμα που θα επιστρέψει βάση της πληροφορίας που διαθέτει εκείνη την στιγμή.

Στην περίπτωση που διαθέτει όλη την απαραίτητη πληροφορία τότε εκτελείται η μέθοδος requestAPI η οποία επιστρέφει τα δεδομένα που θα βρεί.

Στην περίπτωση που λείπει συγκεκριμένη πληροφορία, τότε θα γίνει έλεγχος για να βρεθεί ποια πληροφορία λείπει και θα διεξαχθεί αντίστοιχο μήνυμα συγκεκριμένα για το κομμάτι που λείπει. Στην

περίπτωση που λείπουν όλες οι σημαντικές πληροφορίες τότε το bot θα ζητήσει κάθε πληροφορία ξεχωριστά δίνοντας προτεραιότητα στις πόλεις αναχώρησης και προορισμού, κι έπειτα στις ημερομηνίες.



Σχήμα 5.12 : Διαδικασία μεθόδων. Created on Terrastruct.

5.6.2 Airfare Class

Η κλάση Airfare περιγράφει την πρόθεση της εύρεσης αεροπορικών εισιτηρίων. Η διαδικασία που χρησιμοποιεί είναι αντίστοιχη της κλάσης Flight.

Υλοποιεί τις μεθόδους **scanMessage, getText, getFromLoc, getToLoc, getOutboundDate, getInboundDate** και **requestAPI** με τη διαφορά ότι στη μέθοδο requestAPI σχηματίζει το request να αναζητεί για τα φθηνότερα εισιτήρια.

5.6.3 Airline Class

Η Airline κλάση περιγράφει την πρόθεση του χρήστη να μάθει πληροφορίες σχετικά με τις αεροπορικές εταιρείες που τις εκτελούν.

Αντίστοιχα με τις παραπάνω κλάσεις, και σε αυτήν την κλάση υλοποιούνται οι μέθοδοι **scanMessage, getText, getFromLoc, getToLoc, getOutboundDate, getInboundDate** και **requestAPI** με την ίδια λογική που περιγράφεται παραπάνω. Η διαφορά είναι πάλι στη διαμόρφωση του request καθώς αναζητά πληροφορίες για τις αεροπορικές εταιρείες.

5.6.4 Abbreviation

Για το Abbreviation intent, καθώς δεν απαιτεί ιδιαίτερη διαδικασία εξαγωγής χαρακτηριστικών, έχει υλοποιηθεί μία συνθήκη μέσα στη μέθοδο **core** της κλάσης conversation.

Η διαδικασία σε αυτήν την περίπτωση είναι απλή. Εφόσον η πρόβλεψη του IR μοντέλου μας οδηγήσει σε αυτήν τη συνθήκη, γίνεται έλεγχος με φίλτρα πάνω στο εισαγόμενο κείμενο και,

- Στην περίπτωση που προσδιοριστεί κάποιο token με σημασία της λέξης hello, τότε το μήνυμα που θα επιστραφεί στον χρήστη θα είναι χαιρετισμός, διαφορετικά
- Στην περίπτωση που δεν προσδιοριστεί κανένα token τότε το μήνυμα που θα επιστραφεί στο χρήστη θα ανήκει στην κατηγορία των άγνωστων προθέσεων, δηλαδή ότι το bot δεν καταλαβαίνει τι εννοεί ο χρήστης.

Στην περίπτωση αυτού του intent, η μέθοδος **core** δε φροντίζει να κρατήσει κάποια περεταίρω πληροφορία με αποτέλεσμα στην επόμενη αποστολή μηνύματος του χρήστη να γίνεται ξανά διαδικασία αναγνώρισης intent.

Η πρόθεση Abbreviation μας βοηθά να λύσουμε ορισμένες περιπτώσεις συντομογραφιών που καταλαβαίνει το IR μοντέλο ωστόσο, δεν αποτελεί την πραγματική λύση σε περιπτώσεις όπως το δεύτερο παράδειγμα της ενότητας 4.3.3 που είδαμε.

5.7 API

Κατά την υλοποίηση της εφαρμογής, έγιναν πειράματα πάνω σε πραγματικά δεδομένα μέσω του API που παρέχει η εταιρεία Skyscanner. Το API παρέχει δεδομένα για φθηνούς προορισμούς και για τα φθηνότερα εισιτήρια, παρέχει επίσης ένα ενημερωμένο documentation για την εύκολη ενσωμάτωση, και του σχεδιασμού των requests προς αυτό. Ο σύνδεσμος για το API είναι εδώ.

Στα πλαίσια της υλοποίησης της εφαρμογής, σχεδιάστηκαν και χρησιμοποιήθηκαν οι εξής μέθοδοι

- **browseFlights** : η οποία υλοποιεί το endpoint των Routes του API και χρησιμοποιήθηκε για το intent flight και airline.
- **browseAirfare** : όπου υλοποιεί το endpoint των Quotes του API και χρησιμοποιήθηκε στο airfare intent.

Αντίστοιχα οι μέθοδοι για την εξαγωγή των χαρακτηριστικών δεδομένων που επιστρέφουν οι παραπάνω μέθοδοι :

- getRoutes
- getQoutes
- getPlaces
- getCarriers
- getCurrencies
- getFlightDataList
- getAirfareDatalist

Για τη δημιουργία ενός request στα endpoints του API χρειάστηκαν οι πληροφορίες της γλώσσας των δεδομένων που θα επιστραφούν, του νομίσματος, των iata codes για τις πόλεις αφετηρίας και προορισμού, η χώρα προέλευσης και οι ημερομηνίες αναχώρησης και ενδεχόμενης επιστροφής.

```
#browse quotes
self.conn.request("GET", "/apiservices/browsequotes/v1.0/"
            +data["mainland"]+"/"
            +'EUR'+"/"
            +'el-GR'+"/"
            +data['fromloc']+"/"
            +data['toloc']+"/"
            +data['outbound']+"/"
            +data['inbound']
            , headers=self.headers)
```

Σχήμα 5.13 : Ενδεικτικό παράδειγμα request.

Τα δεδομένα που επιστρέφει το API είναι σε μορφή json, ένα ενδεικτικό παράδειγμα φαίνεται στο παρακάτω σχήμα.


```
"Quotes": [
  {
    "QuoteId": 1,
    "MinPrice": 381,
    "Direct": true,
    "OutboundLeg": {
      "CarrierIds": [
        470
      ],
      "OriginId": 68033,
      "DestinationId": 42833,
      "DepartureDate": "2017-02-03T00:00:00"
    },
    "InboundLeg": {
      "CarrierIds": [
        470
      ],
      "OriginId": 42833,
      "DestinationId": 68033,
      "DepartureDate": "2017-02-06T00:00:00"
    },
    "QuoteDateTime": "2016-11-09T21:20:00"
  },
...
],
```


Σχήμα 5.14 : Ενδεικτικό παράδειγμα response API. Source

5.8 Database

Για βάση δεδομένων χρησιμοποιήθηκε η MySQL. Το σχήμα της βάσης δεδομένων περιλαμβάνει 19 πίνακες για τα δεδομένα της εφαρμογής εκ των οποίων οι παρακάτω περιέχουν τις ερωτήσεις που θα κάνει το bot στους χρήστες :

- **airaskinbound** : περιέχει ερωτήσεις για το αν ο χρήστης θέλει εισιτήριο επιστροφής.
- **airfromloc** : περιέχει ερωτήσεις για την τοποθεσία αφετηρίας για τα intents Airfare και Airline.

- **airinbound** : περιέχει ερωτήσεις για την ημερομηνία επιστροφής για τα intents Airfare και Airline.
- **airoutbound** : περιέχει ερωτήσεις για την ημερομηνία αναχώρησης στα intents Airfare/Airline.
- **airtoloc** : περιέχει ερωτήσεις για την τοποθεσία προορισμού στα intents Airfare/Airline
- **apifound** : περιέχει εκφράσεις για τα αποτελέσματα του request που βρέθηκαν
- **apinoutfound** : περιέχει εκφράσεις για την περίπτωση που δε βρεθούν αποτελέσματα από το request.
- **apirequest** : περιέχει εκφράσεις για το σημείο που γίνεται το request.
- **askinbound** : περιέχει ερωτήσεις για την περίπτωση εισιτηρίων επιστροφής
- **fromloc** : περιέχει ερωτήσεις για την περιοχή αφετηρίας της περίπτωσης Flight intent
- **greetings** : περιέχει εκφράσεις χαιρετισμού
- **inbound** : περιέχει ερωτήσεις για την ημερομηνία επιστροφής του Flight intent
- **outbound** : περιέχει ερωτήσεις για την ημερομηνία αναχώρησης του Flight intent
- **toloc** : περιέχει ερωτήσεις για την τοποθεσία προορισμού του Flight intent
- **uknown** : περιέχει εκφράσεις για την περίπτωση που το bot δεν καταλαβαίνει την ερώτηση του χρήστη.

Κάθε πίνακας περιέχει από 5 εκφράσεις – ερωτήσεις με σκοπό το ζητούμενο αποτέλεσμα να γίνει πιο αληθοφανές προς τον τελικό χρήστη.

Στη συνέχεια, οι παρακάτω πίνακες :

- **citiesmainland** : που περιέχει 26517 πόλεις και την αντίστοιχη χώρα στην οποία ανήκει κάθε πόλη υπό τη μορφή iso2
- **airportcodes** : όπου περιέχει 9134 iata κωδικούς αεροδρομίων, την αντίστοιχη πόλη στην οποία βρίσκεται το αεροδρόμιο και τον αντίστοιχο wac κωδικό.

Οι οποίοι χρησιμοποιούνται για την αντιστοίχιση των πόλεων που εισάγουν οι χρήστες, με τους κωδικούς iata καθώς και τις χώρες που ανήκουν.

Στη συνέχεια, οι πίνακες **ticket** που περιέχει τα εισιτήρια και **carriers** που περιέχουν τις εταιρείες που εξυπηρετούν τα δρομολόγια. Οι πίνακες αυτοί περιέχουν ψεύτικα δεδομένα τα οποία χρησιμοποιούνται στην εφαρμογή, καθώς η χρήση του API ήταν για περιορισμένο χρόνο.

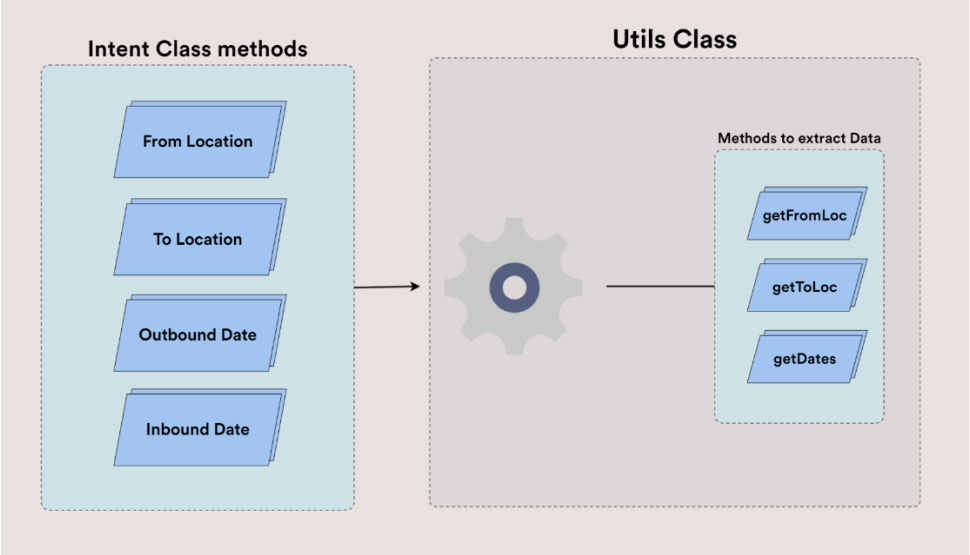
5.9 Εξαγωγή χαρακτηριστικών

5.9.1 Utils Class

Για τη διαχείριση των μεθόδων εξαγωγής χαρακτηριστικών δημιουργήθηκε η κλάση Utils. Η κλάση αυτή υλοποιεί τρείς μεθόδους

- **getFromLoc** : Η μέθοδος αυτή χρησιμοποιεί τρείς τακτικές για τη διεξαγωγή της χρήσιμης πληροφορίας της τοποθεσίας αναχώρησης μέσα από φίλτρα.
- **getToLoc** : Παρόμοια με την προηγούμενη μέθοδο, χρησιμοποιεί κι αυτή τρεις τακτικές για να εξάγει τη χρήσιμη πληροφορία της τοποθεσίας προορισμού.
- **getDate** : Η μέθοδος με την οποία γίνεται η εξαγωγή των ημερομηνιών, είτε αυτή είναι ημερομηνία αναχώρησης είτε επιστροφής. Χρησιμοποιεί επίσης τρεις τακτικές.

Τη χρήση των μεθόδων της κλάσης Utils την εκμεταλλεύονται οι μέθοδοι **getFromLoc**, **getToLoc**, **getOutboundDate** και **getInboundDate** τις οποίες υλοποιούν οι intent κλάσεις Flight, Airfare και Airline.



Σχήμα 5.15 : Utils methods. Created on Terrastruct.

5.9.2 Locations Handler

Για την εξαγωγή των πληροφοριών των τοποθεσιών αφετηρίας και προορισμού υλοποιήθηκε η κλάση LocationFilter.

Η κλάση αυτή υλοποιεί τρεις μεθόδους με διαφορετικές τακτικές για να εξάγει τοποθεσίες αφετηρίας :

- **extractFromLoc** : Όπου σε αυτή τη μέθοδο γίνεται η διαδικασία εξαγωγής μέσα από τα slots που παρέχει το μοντέλο SF.
- **findFromLoc** : Η οποία εξάγει δεδομένα από ορισμένα αντικείμενα Matchers του spaCy. Με αυτόν τον τρόπο αποφεύγουμε το πρόβλημα που περιγράφεται στην ενότητα 4.4.4 του μοντέλου SF, επίσης μας επιτρέπει να ενσωματώσουμε περιοχές εκτός της Αμερικής.
- **findLoc** : Αυτή η μέθοδος είναι πιο απλή σε σχέση με τις παραπάνω και στοχεύει στις περιπτώσεις όπου ο χρήστης θα εισάγει μόνο το όνομα της πόλης σαν κείμενο εισόδου. Σε αυτήν την περίπτωση ο έλεγχος είναι συνδυαστικός. Χρησιμοποιείται και το SF μοντέλο καθώς και το spaCy για επαλήθευση.

Αυτές οι μέθοδοι εκτελούνται σειριακά μέχρι να βρεθεί το ζητούμενο αποτέλεσμα. Κάθε επόμενη μέθοδος εκτελείται στις αστοχίες τις προηγούμενης.

Το πρόβλημα της **extractFromLoc** περιγράφεται στα παραδείγματα της ενότητας 4.4.4, τα προβλήματα που θα μπορούσαν να προκύψουν με τη μέθοδο **findFromLoc** που χρησιμοποιεί Matchers είναι να δώσει ο χρήστης σαν είσοδο σκέτα ονόματα πόλεων ή περιπτώσεις χωρίς τη λέξη from καθώς

οι Matchers στοχεύουν στη λέξη αυτή. Τη λύση σε αυτό το πρόβλημα έρχεται να δώσει η μέθοδος **findLoc.**

Αντίστοιχα για την εξαγωγή πληροφοριών των τοποθεσιών προορισμού η κλάση υλοποιεί τρείς επιπλέον μεθόδους – τακτικές :

- **extractToLoc** : Η οποία χρησιμοποιεί τα slots του μοντέλου SF.
- **findToLoc** : όπου αντίστοιχα η μέθοδος χρησιμοποιεί spaCy Matchers για να καλύψει τις αστοχίες της προηγούμενης μεθόδου και,
- **findLoc** : η οποία είναι η μέθοδος που χρησιμοποιείται και παραπάνω.

Με την ίδια λογική χρησιμοποιούνται και αυτές οι μέθοδοι.

```
fromloc1 = [{'LOWER': 'from'}, {'ENT_TYPE' : 'GPE'}, {'ENT_TYPE' : 'GPE'}]
fromloc2 = [{'LOWER': 'from'}, {'ENT_TYPE' : 'GPE'}]
fromloc3 = [{'LOWER': 'from'}, {'LOWER': 'anywhere'}]
matcher.add('FromLocComplex', [fromloc1])
matcher.add('FromLocSingle', [fromloc2])
matcher.add('FromLocAnywhere', [fromloc3])

toloc1 = [{'LOWER': 'to'}, {'ENT_TYPE' : 'GPE'}, {'ENT_TYPE' : 'GPE'}]
toloc2 = [{'LOWER': 'to'}, {'ENT_TYPE' : 'GPE'}]
toloc3 = [{'LOWER': 'to'}, {'LOWER' : 'anywhere'}]
matcher.add('ToLocComplex', [toloc1])
matcher.add('ToLocSingle', [toloc2])
matcher.add('ToLocAnywhere', [toloc3])
```

Σχήμα 5.16 : spaCy Matchers για τοποθεσίες.

Τις παραπάνω μεθόδους τις εκμεταλλεύονται οι μέθοδοι **getFromLoc** και **getToLoc** της κλάσης Utils.

5.9.3 Dates Handler

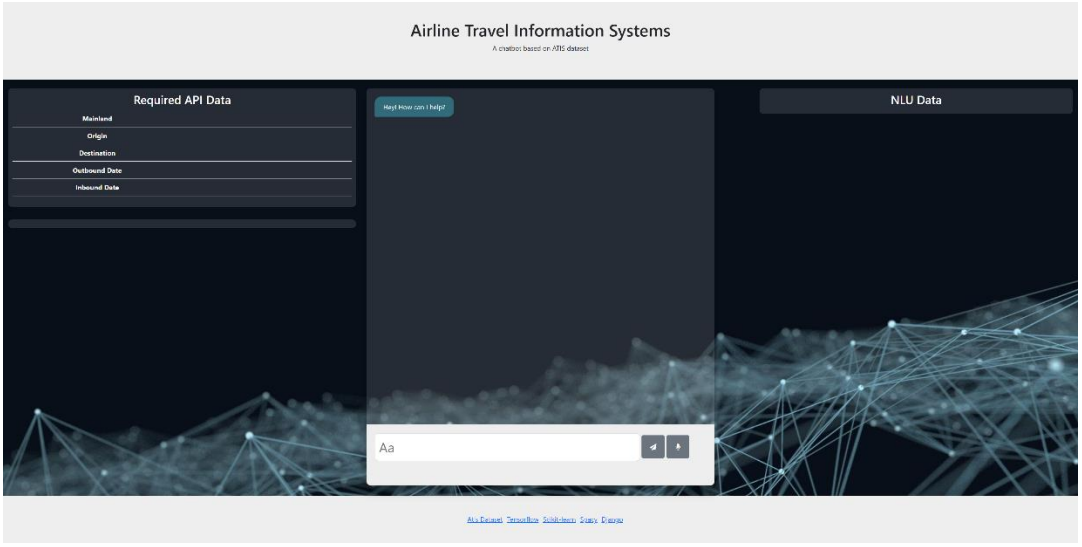
Για την εξαγωγή των ημερομηνιών, υλοποιήθηκε η κλάση DateFilter. Η κλάση αυτή, μέσα από ένα σύνολο μεθόδων φτάνει στο αποτέλεσμα της διαχείρισης τριών διαφορετικών περιπτώσεων ημερομηνιών, και της μετατροπής αυτών των ημερομηνιών σε μια συγκεκριμένη μορφή.

- **catch_dates** : Αυτή η μέθοδος διαχειρίζεται ημερομηνίες που περιλαμβάνουν λέξεις και αριθμούς μαζί, για παράδειγμα **5th of July** ή **July 10th**. Το αποτέλεσμα της εξόδου θα είναι της μορφής YYY-MM-DD.
- **catch_dates_with_days** : Αυτή η μέθοδος στοχεύει στις κοντινές ημερομηνίες της παρούσας εβδομάδας ή της επόμενης με λέξεις καθώς αντίστοιχα και στον τωρινό ή επόμενο μήνα, για παράδειγμα **this Monday, next Friday** ή αντίστοιχα **this Month, next month** με αποτέλεσμα πάλι ημερομηνία της μορφής YYY-MM-DD ή YYY-MM σε περιπτώσεις μήνα.
- **catch_datetimes** : Για τις περιπτώσεις ημερομηνιών σε μορφή YYY-MM-DD.

Η διαδικασία εκτέλεσης των μεθόδων είναι σειριακά μέχρι να βρεθεί ο κατάλληλος τύπος ημερομηνίας, στην περίπτωση που δε βρεθεί τότε η ημερομηνία ξανά ζητείται από το χρήστη. Οι μέθοδοι αυτοί χρησιμοποιούνται από τη μέθοδο getDate της κλάσης Utils.

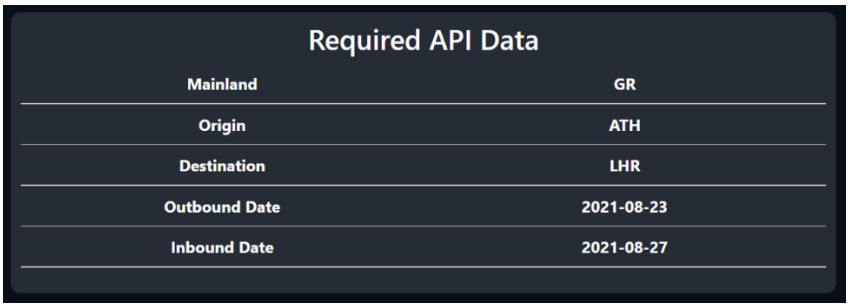
5.10 Τελική εφαρμογή

Μετά την υλοποίηση των παραπάνω, δημιουργήθηκαν τα γραφικά με HTML, CSS και Javascript. Το τελικό αποτέλεσμα φαίνεται στο παρακάτω σχήμα :



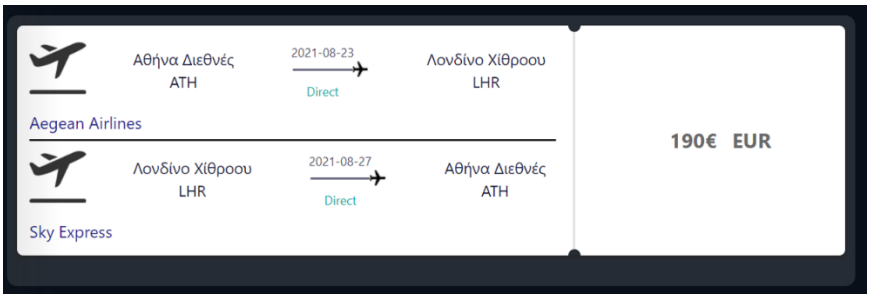
Σχήμα 5.17 : Τελική εφαρμογή.

Στο πλαίσιο αριστερά πάνω φαίνονται τα δεδομένα που συλλέγει η εφαρμογή κατά τη διάρκεια της συζήτησης.



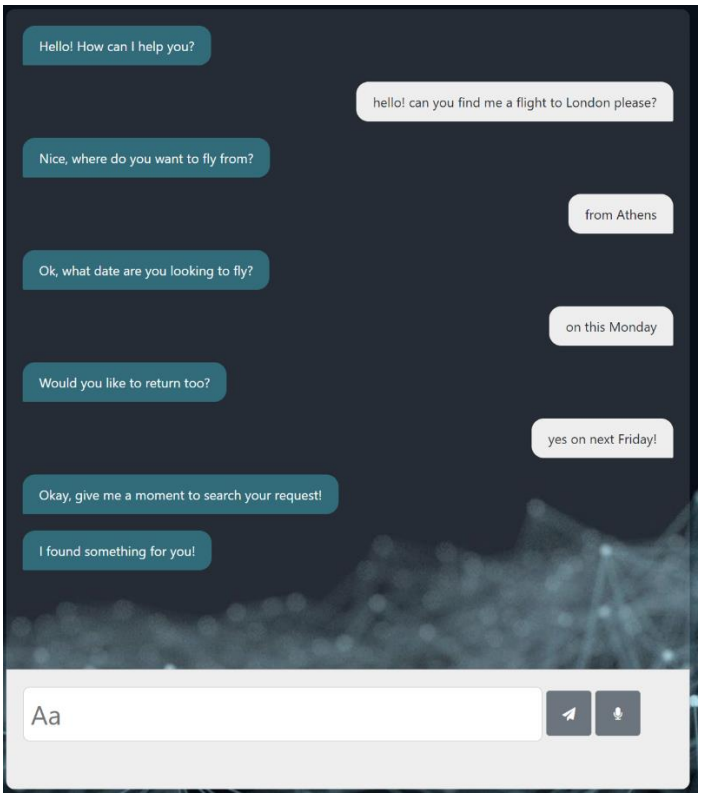
Σχήμα 5.18 : Συλλογή δεδομένων.

Κάτω από αυτό το σημείο εμφανίζονται τα αποτελέσματα της αναζήτησης.



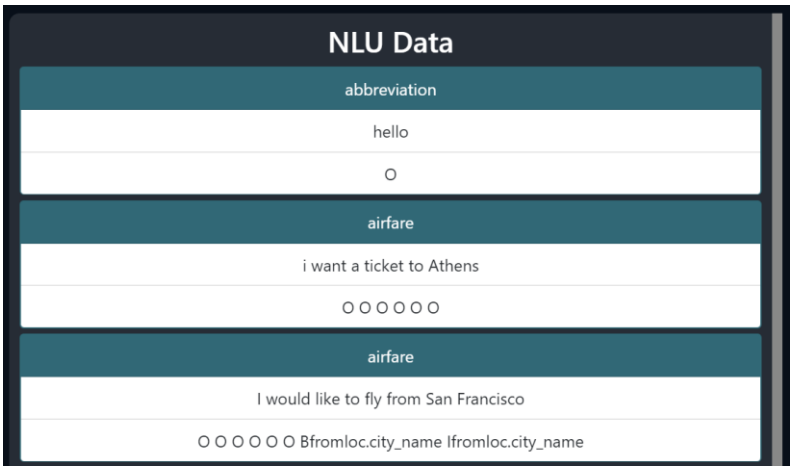
Σχήμα 5.19 : Αποτελέσματα αναζήτησης.

Το πλαίσιο στη μέση της σελίδας χρησιμοποιείται για το κανάλι της επικοινωνίας.



Σχήμα 5.20 : Κανάλι επικοινωνίας.

Και, στο πλαίσιο δεξιά του καναλιού εμφανίζονται τα αποτελέσματα των μοντέλων IR και SF.



Σχήμα 5.21 : Αποτελέσματα NLU.

Κεφάλαιο 6ο: Συμπεράσματα και προτάσεις βελτίωσης

6.1 Συμπεράσματα

Στο πλαίσιο της παρούσας Π.Ε μελετήθηκαν αρχιτεκτονικές αλγόριθμων τεχνιτών νευρωνικών δικτύων με τεχνικές που χρησιμοποιούνται στο πεδίο της επεξεργασίας της φυσικής γλώσσας.

Μέσα από την υλοποίηση αυτών των αλγορίθμων έγιναν κατανοητά σημαντικά προβλήματα που υφίστανται στο NLP καθώς και περιπτώσεις αντιμετώπισης τους, συγκεκριμένα το πρόβλημα των δεδομένων της βάσης γνώσεων που είδαμε για την εφαρμογή, το οποίο έγινε περισσότερο εμφανές κατά την υλοποίηση της καθώς τα αποτελέσματα από το μοντέλο IR οδηγούσαν σε σοβαρό πρόβλημα στο σημείο της διεξαγωγής του διαλόγου με το bot. Αυτό είχε ως αποτέλεσμα τη διατήρηση μιας συγκεκριμένης ροής διαλόγου τη φορά η οποία δεν αφήνει το χρήστη να την αλλάξει πριν η συζήτηση φτάσει στο τέλος.

Επίσης, το πρόβλημα που συναντήσαμε με το μοντέλο SF το οποίο οδήγησε στην ανάγκη της ενσωμάτωσης του μοντέλου του spaCy στο συγκεκριμένο κομμάτι του slot filling. Στη συνέχεια, κατά τη διάρκεια της υλοποίησης της εφαρμογής, με το σημείο της ενσωμάτωσης του εξωτερικού API συμπεραίνεται η πραγματική χρησιμότητα των chatbots και η απλότητα που προσφέρουν για την εξυπηρέτηση των τελικών χρηστών καθώς και τη σημαντικότητα του σε μια επιχείρηση.

6.2 Προτάσεις βελτίωσης

Στην περίπτωση που συνεχιζόταν η ανάπτυξη της συγκεκριμένης εφαρμογής θα πρότεινα για αρχή να αναδομηθεί το ATIS dataset επιλέγοντας πρώτα τις σημαντικές κατηγορίες που θα καθιστούσαν περισσότερο χρήσιμη την εφαρμογή στοχεύοντας σε τελικούς χρήστες οι οποίοι ενδιαφέρονται να αναζητήσουν πληροφορίες συγκεκριμένα για να ταξιδέψουν κι έπειτα, θα χρησιμοποιούσα περισσότερες εκφράσεις σε κάθε κατηγορία ισορροπώντας πρώτον το σύνολο των δεδομένων, και δεύτερον δίνοντας τη δυνατότητα στα μοντέλα να εξάγουν περισσότερη πληροφορία από το χρήστη όπως το πόσα εισιτήρια αναζητά, την κλάση του εισιτηρίου κλπ.

Έπειτα, η δοκιμή των δεδομένων και στα δύο νευρωνικά δίκτυα για να γίνει σύγκριση των αποτελεσμάτων εκ νέου, και πιθανόν σε μια παραμετροποίηση του νευρωνικού δικτύου για το SF task.

Όσον αφορά τη δομή της εφαρμογής, θα βελτίωνα τον πυρήνα υλοποιώντας τη δυνατότητα να παρέχει περισσότερες προθέσεις να τρέχουν παράλληλα, δίνοντας έτσι στον τελικό χρήστη την ελευθερία να συζητήσει για περισσότερα ενδεχομένως ζητήματα που τον απασχολούν ταυτόχρονα. Και επίσης, στην περίπτωση που βρισκόταν ένα API αντίστοιχο της Skyscanner πιθανόν μια υλοποίηση ενός συστήματος κρατήσεων θέσεων.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Apoorva Verma, "The Rise of Chatbots and the Analytics behind them", Analytics India Magazine, April 2016.

[2] Manisha Salecha, "Story of Eliza, the first chatbot developed in 1966", Analytics India Magazine, Oct. 2016.

[3] Srishti Deoras, "10 Things adopting Bots can do to your Business", Analytics India Magazine, Jul 2016.

[4] Artificial Solutions, "Chatbots: The Definitive Guide", 2021. Available: https://www.artificial-solutions.com/chatbots#1.

[5] V-Soft Consulting, "Chatbots and the Turing Test", [Online], Available : https://blog.vsoftconsulting.com/blog/chatbots-and-the-turing-test

[6] John Powell, "Trust Me, I'm a Chatbot: How Artificial Intelligence in Health Care Fails the Turing Test" University of Oxford, Oxford UK, 2019.

[7] Lauern Kunze, "We're thinking about the Turing Test all wrong", 2018, [Online], Available: https://qz.com/1487101/the-turing-test-shows-how-chatbots-ultimate-goal-isnt-intelligence-its-language/

[8] Kaila Krayewski, "How NLP Text-Based Chatbots Work", [Online], Available : https://www.ultimate.ai/blog/ai-automation/how-nlp-text-based-chatbots-work.

[9] Jim Diaz, "The 3 Different Generations Of Chatbot Technology", Dec 2017, [Online], Available : https://rul.ai/blog/3-different-generations-chatbot-technology/.

[10] Daria Zaboj, "The Essential Chatbot Terminology for Beginners", Aug 2020, [Online], Available : https://www.chatbot.com/blog/chatbot-terminology/.

[11] Jason Brownlee, "What is Natural Language Processing?", Aug 2019, [Online], Available : https://machinelearningmastery.com/natural-language-processing/.

[12] Bayan Abu Shawar, Eric Atwell, "Different measurements metrics to evaluate a chatbot system", Academic and Industrial Research in Dialog Technologies, Jan. 2007.

[13] IBM Cloud Education, "What is Machine Learning?", Jul 2020, [Online], Available : https://www.ibm.com/cloud/learn/machine-learning.

[14] Wikipedia, "ELIZA", [Online], Available : https://en.wikipedia.org/wiki/ELIZA.

[15] Ina, "The History Of Chatbots – From ELIZA to ALEXA", Oct 2017, [Online], Available : https://onlim.com/en/the-history-of-chatbots/.

[16] Reshmi S., Kannan Balakrishnan, "Empowering Chatbots with Business Intelligence by big data integration", Jan – Feb 2018, [Online], Available : http://www.ijarcs.info

[17] Bird, Steven, Edward Loper and Ewan Klein, "Natural Language Processing with Python", 2009, O'Reilly Media Inc, [Online], Available : https://www.nltk.org/.

[18] spacy.io, [Online], Available : https://spacy.io/

[19] medium.com, "Introduction to libraries of NLP in Python – NLTK vs spaCy", Jun 2018, [Online], Available : https://medium.com/@akankshamalhotra24/introduction-to-libraries-of-nlp-in-python-nltk-vs-spacy-42d7b2f128f2.

[20] Pema Grg, "NLTK or SPACY?", Sep 2018, [Online], Available : https://blog.ekbana.com/private-nltk-vs-spacy-3926b3674ee4.

[21] scikit-learn, [Online], Available : https://scikit-learn.org/stable/.

[22] tensorflow, [Online], Available : https://www.tensorflow.org/.

[23] ML Glossary, [Online], Available: https://ml-cheatsheet.readthedocs.io/en/latest/glossary.html.

[24] technology review, Karen Hao, "What is machine learning?", Nov 2018, [Online], Available : https://www.technologyreview.com/2018/11/17/103781/what-is-machine-learning-we-drew-you-another-flowchart/.

[25] Kamran Kowsar, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, Donald Brown, "Text Classification Algorithms: A Survay", April 2019.

[26] Armand Joulin, Edouard Grave, Piotr Bojanowski, Tomas Mikolov, "Bag of Tricks for Efficient Text Classification", Cornell University, Aug 2016.

[27] Imran Sheikh, Irina Illina, Dominique Fohr, Georges Linares, "Learning Word Importance with the Neural Bag-of-Words Model", University of Lorraine, Aug 2016.

[28] Juan Ramos, "Using TF-IDF to Determine Word Relevance in Document Queries", Rutgers University.

[29] Jason Brownlee, "What are Word Embeddings for Text?", Oct 2017, [Online], Available : https://machinelearningmastery.com/what-are-word-embeddings/.

[30] Tomas Mikovol, "Linguistic Regularities in Continuous Space Word Representations", Microsoft Research.

[31] Sunil Ray, "6 Easy Steps to Learn Naive Bayes Algorithm with codes in Python and R", Sep 2017, [Online], Available : https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/.

[32] Anuradha Patra, Divakar Singh, "A Survey Report on Text Classification with Different Term Weighing Methods and Comparison between Classification Algorithms", International Journal of Computer Application (0975 - 8887), Aug 2013, [Online].

[33] scikit-learn, Naive Bayesm, [Online], Available : https://scikit-learn.org/stable/modules/naive_bayes.html.

[34] scikit-learn, Support Vector Machine, [Online], Available : https://scikit-learn.org/stable/modules/svm.html#classification.

[35] Sunil Ray, "Understanding Support Vector Machine (SVM) algorithm from examples (along with code)", Sep 2017, [Online], Available : https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/.

[36] Pengfei Liu, Xipeng Qiu, Xuanjing Huang, "Recurrent Neural Networks for Text Classification with Multi-Task Learning", Fudan University, May 2016.

[37] Mehreen Saeed, “An Introduction to Recurrent Neural Networks and the Math that Powers them”, Sep 2021, [Online], Available : https://machinelearningmastery.com/an-introduction-to-recurrent-neural-networks-and-the-math-that-powers-them/.

[38] Phone Le, Willem Zuidema, “Quantifying the vanishing gradient and long distance dependency problem in recursive neural networks and recursive LSTMs”, University of Amsterdam, Mar 2016.

[39] SuperDataScience Team, “Recurrent Neural Networks (RNN) – The Vanishing Gradient Problem”, Aug 2018, [Online], Available : https://www.superdatascience.com/blogs/recurrent-neural-networks-rnn-the-vanishing-gradient-problem.

[40] Sepp Hochreiter, Jurgen Schimidhuber, “LONG SHORT-TERM MEMORY”, 1997.

[41] Michael Phi, “Illustrated Guide to LSTM’s and GRU’s : A step by step explanation”, Sep 2018, [Online], Available : https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21.

[42] Wojciech Zaremba, Oriol Vinyals, “Recurrent Neural Network Regularization”, Feb 2015.

[43] Wikipedia, “Inside-Ooutside-Beginning”, [Online], Available : https://en.wikipedia.org/wiki/Inside%E2%80%93outside%E2%80%93beginning_(tagging).

[44] Christopher Marshall, “What is intent recognition and how can I use it?”, Aug 2020, [Online], Available : https://medium.com/mysuperai/what-is-intent-recognition-and-how-can-i-use-it-9ceb35055c4f.

[45] Ilya Sutskever, Oriol Vinyals, Quoc V. Le, “Sequence to Sequence Learning with Neural Networks”, Dec 2014.

[46] Manish Chablani, “Sequence to sequence model : Introduction and concepts”, Jun 2017, [Online], Available : https://towardsdatascience.com/sequence-to-sequence-model-introduction-and-concepts-44d9b41cd42d.

[47] Jason Brownlee, “How to Develop an Encoder-Decoder for Sequence-to-Sequence Prediction”, Aug 2020, [Online], Available : https://machinelearningmastery.com/develop-encoder-decoder-model-sequence-sequence-prediction-keras/.

[48] Harshall Lamba, “Word Level English to Marathi Neural Machine Translation using Encoder-Decoder Model”, Feb 2019, [Online], Available : https://towardsdatascience.com/word-level-english-to-marathi-neural-machine-translation-using-seq2seq-encoder-decoder-lstm-model-1a913f2dc4a7.

[49] Django framework, [Online], Available : https://www.djangoproject.com/.

[50] WebSocket API, [Online], Available : https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API.