ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«ΔΙΑΔΙΚΤΥΑΚΗ ΕΦΑΡΜΟΓΗ ΓΙΑ ΤΗΝ ΑΝΑΛΥΣΗ ΕΛΛΗΝΙΚΟΥ ΚΕΙΜΕΝΟΥ»

Γραφῶμετρο κύριο όνομα

Της φοιτήτριας
Άνι Χατσατριάν
Αρ. Μητρώου: 113811

Επιβλέπων
Παναγιώτης Αδαμίδης
Καθηγητής

Θεσσαλονίκη, Ιούνιος 2021

Τίτλος Π.Ε. Διαδικτυακή εφαρμογή για την ανάλυση ελληνικού κειμένου
Κωδικός Π.Ε. 20182
Ονοματεπώνυμο φοιτήτριας Άνι Χατσατριάν
Ονοματεπώνυμο εισηγητή Παναγιώτης Αδαμίδης
Ημερομηνία ανάληψης Π.Ε. 31/08/2020
Ημερομηνία περάτωσης Π.Ε. 11/06/2021

Βεβαιώνω ότι είμαι η 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 πτυχιακή εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

«Για τους γονείς μου»

Πρόλογος

Ο λόγος που επέλεξα αυτή την εφαρμογή είναι επειδή η γλωσσολογία είναι ένας τομέας που με ενδιαφέρει και παρατήρησα ότι δεν υπάρχουν πολλές εφαρμογές στο Διαδίκτυο που να έχουν να κάνουν με την ανάλυση του γραπτού ελληνικού λόγου και να διαθέτουν ένα χρηστικό περιβάλλον. Επίσης, με αυτή την πτυχιακή θέλω να αποπειραθώ τη δημιουργία δικών μου αλγορίθμων για την ανάλυση κειμένου, οι οποίοι θα βασίζονται στους κανόνες της ελληνικής γραμματικής, και να συγκρίνω το αποτέλεσμα με τη χρήση μιας εξειδικευμένης βιβλιοθήκης για την επεξεργασία φυσικής γλώσσας.

Περίληψη

Ο σκοπός της παρούσας πτυχιακής είναι η ανάπτυξη μιας διαδικτυακής εφαρμογής η οποία μπορεί να χρησιμοποιηθεί για την ανάλυση ελληνικού κειμένου. Η ανάλυση αφορά συστατικά στοιχεία του κειμένου, όπως αριθμός λέξεων και προτάσεων, αναγνώριση μερών του λόγου και άλλες πληροφορίες από το κείμενο που παρέχεται από το χρήστη. Πραγματοποιήθηκε με δύο τρόπους: με τη χρήση της βιβλιοθήκης spaCy και με την ανάπτυξη αλγορίθμων βασιζόμενων στους κανόνες της ελληνικής γραμματικής. Σκοπός της πτυχιακής, πέρα από την ολοκλήρωση και δημοσίευση της εφαρμογής, είναι η σύγκριση ανάμεσα στους δύο διαφορετικούς τρόπους ανάλυσης. Για την δημιουργία της εφαρμογής χρησιμοποιήθηκαν η γλώσσα προγραμματισμού Python για το τμήμα του εξυπηρετητή και η JavaScript για τη δημιουργία της διεπαφής του χρήστη. Για την διευκόλυνση της ανάπτυξης και δημοσίευσης της εφαρμογής χρησιμοποιήθηκαν βιβλιοθήκες όπως η React.js και FastAPI. Το αποτέλεσμα της πτυχιακής είναι η δημοσιευμένη εφαρμογή που επιτρέπει την εναλλαγή ανάμεσα στους δύο τρόπους ανάλυσης και παρουσιάζει τα αποτελέσματα της. Παράλληλα συμπεραίνεται από τη σύγκριση ανάμεσα στους τρόπους ανάλυσης, ότι η βιβλιοθήκη spaCy προσφέρει αρκετά ακριβή αποτελέσματα αλλά απαιτεί παραπάνω υπολογιστικούς πόρους και χρόνο για την ανάλυση, ενώ οι προσαρμοσμένοι αλγόριθμοι, αν και πιο γρήγοροι στην εκτέλεση, για απλές προτάσεις λειτουργούν με την ίδια ακρίβεια, σε πιο περιπλοκές προτάσεις η αντιστοίχηση είναι λανθασμένη ή και σε κάποιες περιπτώσεις αδύνατη. Παρόλα αυτά με την επέκταση των προσαρμοσμένων αλγόριθμων, είναι δυνατόν να επιτευχθούν πιο ακριβή αποτελέσματα.

Web application for Greek text analysis

Ani Chatsatrian

Abstract

Nowadays most of the writing takes place on a computer rather than on paper. This fact, alongside the advancement and open access to text mining techniques and modern web technologies makes it easier to create a tool to analyze any kind of text. There are plenty of applications that do this for languages like English or French. However, there are not many that provide text analysis for the Greek language. The purpose of this thesis is to create a web application that analyzes Greek text using two different approaches. First following a rule-based approach, by translating Greek grammatical rules to algorithms and using them to analyze the text. The second approach is to analyze the Greek text by using the spaCy library. A comparison of the two approaches is presented at the end of this thesis, as well as recommendations for further research.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω αρχικά τον κ. Αδαμίδη για την υπομονή και που μου έδωσε τη δυνατότητα να πραγματοποιήσω τη συγκεκριμένη πτυχιακή, ακόμη και μετά από τις αποτυχημένες προσπάθειές μου.
Επίσης, τη φίλη και συγκάτοικο μου Αμαλία για τη κατανόηση και ψυχολογική υποστήριξη κατά τη διάρκεια εκπόνησης της πτυχιακής. Όπως και την Αντιγόνη και τον Αργύρη για τις σημαντικές συμβουλές και διορθώσεις τους.
Τέλος, όλους τους φίλους, φίλες και συγγενείς για τα ενθαρρυντικά τους σχόλια.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Συντομογραφίες

ΕΦΓ	Επεξεργασία Φυσικής Γλώσσας
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία
API	Application Programming Interface
BEM	Block Element Modifier
CRF	Conditional Random Field
CSS	Cascading Style Sheets
ECMAScript	European Computer Manufacturers Association Script
EL	Entity Linking
HMM	Hidden Markov Model
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
NER	Named Entity Recognition
NLP	Natural Language Processing
NLU	Natural Language Understanding
POS	Parts Of Speech
REST	Representational State Transfer
SBD	Sentence Boundary Detection
SPA	Single-Page Application
UPOS	Universal Parts of Speech

Κεφάλαιο 1ο: Εισαγωγή

Ο σκοπός αυτής της πτυχιακής είναι η ανάπτυξη μιας εφαρμογής που παρέχει τη δυνατότητα στους χρήστες να εισάγουν ένα οποιοδήποτε κείμενο που είναι γραμμένο στα ελληνικά, να αιτηθούν την ανάλυση του και να λάβουν τα αποτελέσματα της ανάλυσης μέσω της εφαρμογής.

Παράλληλα, ο χρήστης έχει την επιλογή να διαλέξει ανάμεσα σε δύο διαφορετικές στρατηγικές ανάλυσης του κειμένου. Η πρώτη επιλογή είναι με τη χρήση μιας βιβλιοθήκης, η οποία είναι εξειδικευμένη στην ανάλυση και επεξεργασία της φυσικής γλώσσας. Η δεύτερη επιλογή χρησιμοποιεί προσαρμοσμένους κανόνες, βασιζόμενους σε κανόνες της γραμματικής που αναπτύχθηκαν κατά της διάρκεια της πτυχιακής.

Πέρα από την ανάπτυξη της εφαρμογής, η πτυχιακή περιλαμβάνει ένα ερευνητικό μέρος που καλύπτει την ανάλυση και επεξεργασίας φυσικής γλώσσας, ο οποίος είναι ένας τομέας που περιλαμβάνει κλάδους όπως η τεχνητή νοημοσύνη και η υπολογιστική γλωσσολογία και αφορά την αλληλεπίδραση των υπολογιστών με τις ανθρώπινες γλώσσες σε οποιαδήποτε μορφή, είτε γραπτή, είτε προφορική.

Επειδή ο τομέας της επεξεργασίας μιας φυσικής γλώσσας είναι ιδιαίτερα ευρύς, στα πλαίσια της πτυχιακής θα ασχοληθούμε με την ανάλυση γραπτού λόγου, και συγκεκριμένα με την επισήμανση μερών του λόγου. Η πτυχιακή είναι ένα πείραμα σύγκρισης ανάμεσα στους διαφορετικούς τρόπους διεξαγωγής της ανάλυσης και έχει ως σκοπό την ενημέρωση και παρουσίαση των διαφορών ανάμεσα τους.

Η επισήμανση μερών του λόγου αφορά τη διαδικασία ταξινόμησης των λέξεων ενός κειμένου στα διάφορα μέρη του λόγου που υπάρχουν σε μια συγκεκριμένη γλώσσα. Μπορεί να πραγματοποιηθεί με δύο τρόπους: α) με τη χρήση κανόνων, μια σειρά ελέγχων οι οποίοι οδηγούν στον καθορισμό του μέρους του λόγου στο οποίο ανήκει μια λέξη, και β) με τη στοχαστική αναγνώριση η οποία βασίζεται στον προσδιορισμό της πιθανότητας εμφάνισης μιας λέξης σε μια πρόταση. Οι τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής μπορούν να χωριστούν σε δύο τμήματα:

- Το πρώτο τμήμα αφορά την ανάπτυξη και δημοσίευση της σελίδας την οποία χρησιμοποιεί ο χρήστης για την εισαγωγή του κειμένου προς ανάλυση και την εμφάνιση των αποτελεσμάτων. Σε αυτό το τμήμα επιλέχθηκε η χρήση των καθιερωμένων τεχνολογιών ανάπτυξης διαδικτυακών εφαρμογών, HTML, CSS και JavaScript, καθώς και η βιβλιοθήκη react.js για την καλύτερη οργάνωση του κώδικα. Για τη δημοσίευση της σελίδας έγινε η χρήση της εφαρμογής Github pages, η οποία δίνει τη δυνατότητα εύκολης δημοσίευσης εφαρμογών που χρησιμοποιούν τη βιβλιοθήκη react.js.
- Το δεύτερο τμήμα περιλαμβάνει όλη τη λογική της ανάλυσης του κειμένου και επειδή η ανάλυση απαιτεί αρκετούς υπολογιστικούς πόρους, είναι επόμενο να μεταφερθεί σε έναν ξεχωριστό διακομιστή που θα επιτρέπει την επικοινωνία με το πρώτο τμήμα. Για αυτό το κομμάτι χρησιμοποιήθηκε η γλώσσα προγραμματισμού python, καθώς η βιβλιοθήκη spaCy, η οποία χρησιμοποιείται για τη στοχαστική αναγνώριση του κειμένου, είναι γραμμένη σε αυτή τη γλώσσα. Για την επίτευξη της επικοινωνίας μεταξύ της διαδικτυακής σελίδας με το διακομιστή επιλέχθηκε η τεχνολογία FastAPI η οποία δίνει τη δυνατότητα δημοσίευσης του διακομιστή με την υπηρεσία Deta.

Συγκεκριμένα, στο κεφάλαιο 2, γίνεται μια εισαγωγή στο θεωρητικό υπόβαθρο της πτυχιακής, όπου αναφέρεται ο ορισμός της Επεξεργασίας Φυσικής Γλώσσας (ΕΦΓ) και οι εφαρμογές της. Μεγαλύτερο

βάρος δίνεται στο κομμάτι της αναγνώρισης των μερών του λόγου, με μια μικρή αναφορά στα μέρη του λόγου της ελληνικής γλώσσας και την ανάλυση των τρόπων που γίνεται η αναγνώριση.

Το κεφάλαιο 3 είναι αφιερωμένο στις τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη της διαδικτυακής εφαρμογής. Για κάθε τεχνολογία που χρησιμοποιήθηκε αναφέρεται ο ρόλος, ο λόγος και ο τρόπος με τον οποίο χρησιμοποιήθηκαν στην υλοποίηση της εφαρμογής.

Στο κεφάλαιο 4 πραγματοποιείται η εξήγηση των δύο διαφορετικών στρατηγικών που ακολουθούνται για την ανάλυση του κειμένου, καθώς και ο κοινός τρόπος απεικόνισης των αποτελεσμάτων. Σε αυτό το κεφάλαιο παρουσιάζεται η λογική των προσαρμοσμένων αλγορίθμων και αναλύονται οι λειτουργίες της βιβλιοθήκης spaCy.

Στο κεφάλαιο 5 παρουσιάζονται τα αποτελέσματα από την ανάλυση τόσο με τη χρήση των προσαρμοσμένων αλγορίθμων όσο και με τη χρήση της βιβλιοθήκης spaCy. Στο τέλος πραγματοποιείται η σύγκριση των αποτελεσμάτων με βάση την εγκυρότητα της αυτόματης αντιστοίχησης κάθε λέξης στο εκάστοτε μέρος του λόγου και το χρόνο εκτέλεσης της ανάλυσης. Στο τελευταίο κεφάλαιο παρουσιάζονται τα συμπεράσματα που προέκυψαν από την όλη διαδικασία και αναλύονται διάφορες προτάσεις βελτίωσης και επέκτασής της.

Συνοψίζοντας, η πτυχιακή αυτή θα φανεί χρήσιμη σε όσους ασχολούνται ή επιθυμούν να ξεκινήσουν την ενασχόληση με τον τομέα της επεξεργασίας φυσικών γλωσσών. Επίσης, μπορεί να χρησιμοποιηθεί και ως παράδειγμα για τη δημιουργία οποιουδήποτε είδους διαδικτυακής εφαρμογής που απαιτεί μια σελίδα ως διεπαφή, μια υποδομή backend και τη χρήση δευτερευόντων βιβλιοθηκών

Κεφάλαιο 2ο: Επεξεργασία Φυσικής Γλώσσας

2.1 Εισαγωγή

Για τη καλύτερη κατανόηση των αρχών και μεθόδων που χρησιμοποιήθηκαν στη πτυχιακή, είναι σημαντικό να αναφερθούν κάποια βασικά θέματα για τον ευρύ τομέα της επεξεργασίας φυσικής γλώσσας.

Η επικοινωνία μεταξύ υπολογιστή και ανθρώπου έχει γίνει καθημερινότητά μας, καθώς χρησιμοποιούμε ευρέως εφαρμογές όπως η μετάφραση κειμένων από τη μία γλώσσα στην άλλη ή δίνουμε εντολές σε συσκευές που διεκπεραιώνουν τις μικρές εργασίες που τις αναθέτουμε. Ο τομέας της επεξεργασίας φυσικής γλώσσας (ΕΦΓ) έχει ως στόχο να διευκολύνει την επικοινωνία αυτή, μετατρέποντας τη φυσική γλώσσα σε κάτι το οποίο μπορεί να επεξεργαστεί ένας υπολογιστής και το αποτέλεσμα της επεξεργασίας να βγάζει νόημα και να έχει κάποια αξία [1].

Είναι σαφές λοιπόν, ότι η ΕΦΓ είναι αρκετά σημαντικός τομέας και πιθανότατα ένας από τους πιο συχνά εφαρμοσμένους τομείς της τεχνητής νοημοσύνης, καθώς δίνει στους υπολογιστές τη δυνατότητα να κατανοούν πως επικοινωνούμε εμείς οι άνθρωποι.

2.1.1 Προκλήσεις της Επεξεργασίας Φυσικής Γλώσσας

Η ασάφεια και τα ανακριβή χαρακτηριστικά των φυσικών γλωσσών είναι μερικά από τα στοιχεία που καθιστούν την ΕΦΓ δύσκολη. Η επεξεργασία φυσικής γλώσσας είναι μια προκλητική διαδικασία στη επιστήμη των υπολογιστών και αυτό συμβαίνει κατά κύριο λόγο εξαιτίας της φύσης των ανθρώπινων γλωσσών. Για την πλήρη κατανόηση της ανθρώπινης γλώσσας απαιτείται η κατανόηση τόσο των λέξεων όσο και του τρόπου με τον οποίο οι έννοιες συνδέονται μεταξύ τους, έτσι ώστε να μεταφερθεί ένα μήνυμα με επιτυχία.

Έχοντας πλήρες και συνεκτικές προτάσεις και παραγράφους, εμείς οι άνθρωποι αναζητάμε τις έννοιες των λέξεων και ενσωματώνουμε τις εμπειρίες και γνώσεις μας για να κατανοήσουμε το περιεχόμενο ενός κειμένου ή συνομιλίας. Μπορούμε να διαχωρίσουμε αβίαστα τις προτάσεις, να προσδιορίσουμε τις σχέσεις και να συνάγουμε το νόημα σχεδόν αμέσως. Είμαστε πάντα ενήμεροι όταν κάτι λείπει από μια πρόταση, παράγραφο ή έγγραφο συνολικά. Ως κοινωνικά όντα αναγνωρίζουμε και προσαρμόζουμε άμεσα τον τόνο και τα συναισθήματα μας για να μεταφέρουμε σκέψεις για θέματα που κυμαίνονται από τον καιρό ως τη πιθανότητα εξωγήινης ύπαρξης στο σύμπαν. Αν και συχνά θεωρούμε αυτές τις δεξιότητες ως δεδομένες, πρέπει να θυμόμαστε ότι έχουν εξελιχθεί μέσα από πολλά χρόνια συνομιλίας, εκπαίδευσης και συζητήσεων με άλλους, για να μην αναφέρουμε όλες τις γνώσεις που προέρχονται από τους προγόνους μας.

Την ίδια στιγμή, η επιστήμη των υπολογιστών μαζί με το τομέα της επεξεργασίας φυσικής γλώσσας είναι σχετικά πρόσφατα θέματα τα οποία εξερευνούμε. Για αυτό το λόγο, όταν προσπαθούμε να υλοποιήσουμε την ίδια έμφυτη ανάλυση που κάνουμε στο μυαλό μας σε μια εφαρμογή ανάλυσης κειμένων, ερχόμαστε αντιμέτωποι με πολλές πολλές τεχνικές και μη προκλήσεις. Μερικές από αυτές τις προκλήσεις, χωρισμένες σε πεδία και με σειρά δυσκολίας, είναι οι παρακάτω [2]:

Πεδίο χαρακτήρων

- Κωδικοποιήσεις χαρακτήρων, όπως ASCII, Latin-1, UTF-8, UTF-16
- Κεφαλαίοι ή μη χαρακτήρες, σημεία στίξης, τόνου και αριθμοί.

Πεδίο λέξεων

- Χωρισμός του κειμένου σε λέξεις, ιδιαίτερα σε γλώσσες που δεν χρησιμοποιούν κενά διαστήματα ανάμεσα σε λέξεις.
- Αναγνώριση μερών του λόγου.
- Αναγνώριση συνωνύμων λέξεων.
- Αφαίρεση καταλήξεων και εύρεσης της ρίζας μιας λέξης.
- Συντομογραφίες, ακρωνύμιο και η ορθογραφία παίζουν μεγάλο ρόλο στην κατανόηση λέξεων.

Πεδίο πολλαπλών λέξεων και προτάσεων

- Ανίχνευση περιγραφών, όπως το μεγάλο καφέ παπούτσι.
- Διάσπαση των προτάσεων σε υποκείμενο-ρήμα-αντικείμενο και άλλες σχέσεις.
- Αναγνώριση αρχής και τέλους των προτάσεων.
- Αναγνώριση του κεντρικού σημείου μιας πρότασης, όπως στη πρόταση «Στην Άννα αρέσουν τα σκυλιά, αλλά δε θα αγοράσει ποτέ ένα.», στην οποία το κεντρικό σημείο είναι η Άννα.
- Εννοιολογικές οριοθετήσεις χρησιμοποιώντας το πλαίσιο μιας πρότασης.
- Συνδυασμός των ορισμών και σχέσεων των λέξεων σε μια πρόταση για το προσδιορισμό της έννοιας της.

Πεδίο πολλαπλών προτάσεων και παραγράφων

Σε αυτό το πεδίο η ανάλυση γίνεται πιο περίπλοκη, καθώς ο στόχος της ανάλυσης είναι η εύρεση της πρόθεσης ενός συγγραφέα. Οι αλγόριθμοι περίληψης συνήθως απαιτούν την αναγνώριση προτάσεων οι οποίες είναι πιο σημαντικές από άλλες.

Πεδίο ολόκληρου κειμένου

Όπως και το προηγούμενο πεδίο, η εύρεση της έννοιας ενός εγγράφου συχνά απαιτεί γνώση που υπερβαίνει αυτό που περιέχεται στο ίδιο το κείμενο. Οι συγγραφείς συχνά θεωρούν ότι οι αναγνώστες τους έχουν ένα υπόβαθρο κατανόησης. Για παράδειγμα το μεγαλύτερο μέρος αυτής της πτυχιακής δεν θα έχει νόημα εάν ο αναγνώστης δεν έχει χρησιμοποιήσει ποτέ υπολογιστή.

Πεδίο πολλαπλών κειμένων

Σε αυτό το πεδίο ερχόμαστε αντιμέτωποι με τη διαδικασία εύρεσης κειμένων ή εγγράφων που έχουν να κάνουν με ένα συγκεκριμένο πεδίο ενδιαφέροντος. Πέρα από αυτές τις προκλήσεις, οι ανθρώπινοι παράγοντες παίζουν επίσης ρόλο στην ανάλυση κειμένου. Διαφορετικοί πολιτισμοί, γλώσσες και ερμηνείες της ίδιας φράσης είναι μερικοί από αυτούς.

2.1.2 Εφαρμογές Επεξεργασίας Φυσικής Γλώσσας

Παρόλες τις προκλήσεις, ο τομέας της ΕΦΓ συνεχίζει να εξελίσσεται και να μας βοηθά στην καθημερινότητα μας. Μερικές από τις εφαρμογές της ΕΦΓ είναι οι εξής [3]:

- Υπηρεσίες ηλεκτρονικού ταχυδρομείου οι οποίες παρέχουν λειτουργίες όπως η αναγνώριση και κατηγοριοποίηση της ανεπιθύμητης αλληλογραφίας.
- Προσωπικοί βοηθοί που βασίζονται σε ένα εύρος τεχνικών ΕΦΓ, ώστε να μπορούν να αλληλεπιδρούν με τον χρήστη, να κατανοούν τις ηχητικές εντολές του, και να ανταποκρίνονται ανάλογα.
- Σύγχρονες μηχανές αναζήτησης που αποτελούν τον ακρογωνιαίο λίθο του Διαδικτύου σήμερα, χρησιμοποιούν την ΕΦΓ σε μεγάλο βαθμό για λόγους όπως η καλύτερη κατανόηση των ερωτημάτων, η αυτόματη απάντηση των ερωτήσεων, η ανάκτηση πληροφοριών και η ομαδοποίηση των αποτελεσμάτων.
- Οι υπηρεσίες αυτόματης μετάφρασης, όπως προαναφέρθηκαν, είναι άμεσες εφαρμογές της ΕΦΓ.
- Πολλοί οργανισμοί και εταιρείες αναλύουν τις ροές των κοινωνικών τους μέσων για να κατανοήσουν καλύτερα τους πελάτες τους.
- Η ΕΦΓ χρησιμοποιείται ευρέως στο ηλεκτρονικό εμπόριο για να διευκολύνει την εξαγωγή σχετικών πληροφοριών ξεκινώντας από τις περιγραφές προϊόντων και φτάνοντας μέχρι και την ανάλυση και κατανόηση των κριτικών που γράφουν οι χρήστες για τα προϊόντα.
- Η ΕΦΓ αποτελεί τη βάση των εργαλείων διόρθωσης ορθογραφίας και γραμματικής.
- Βοηθητικά bots που μπορούν να ανταποκριθούν σε ένα σύστημα ερωτο-απαντήσεων και χρησιμοποιούνται σε online υπηρεσίες για να βοηθήσουν τους χρήστες με οποιοδήποτε πρόβλημα έχουν.
- Η ΕΦΓ χρησιμοποιείται στον ακαδημαϊκό τομέα για την ανάπτυξη εργαλείων όπως η ανίχνευση λογοκλοπής, η αυτόματη διόρθωση και βαθμολογία εξετάσεων.

2.1.3 Λειτουργίες Επεξεργασίας Φυσικής Γλώσσας

Υπάρχει μια συλλογή από βασικές διαδικασίες που εμφανίζονται συχνά σε εφαρμογές ΕΦΓ [3]. Λόγω της θεμελιώδους φύσης και της ευρύτερης χρήσης τους, είναι σημαντικό να γίνει μια μικρή αναφορά σε κάθε μία από αυτές.

Γλωσσική μοντελοποίηση (Language modeling)

Σε αυτή τη διαδικασία πραγματοποιείται η πρόβλεψη της λέξης που ακολουθεί σε μια πρόταση, αναλύοντας τις προηγούμενες λέξεις της πρότασης. Ο στόχος της είναι ο υπολογισμός της πιθανότητας εμφάνισης μιας σειράς λέξεων σε μια γλώσσα. Χρησιμοποιείται στην αναγνώριση ομιλίας, οπτικών χαρακτήρων και γραφής, καθώς και στην αυτόματη μετάφραση και στον ορθογραφικό έλεγχο.

Ταξινόμηση κειμένου (Text classification)

Η ταξινόμηση κειμένου αφορά τη διαδικασία ομαδοποίησης του σε ένα γνωστό σύνολο κατηγοριών με βάση το περιεχόμενο του. Αποτελεί μια από τις πιο δημοφιλείς διαδικασίες ΕΦΓ και χρησιμοποιείται σε πολλά εργαλεία όπως η αναγνώριση ανεπιθύμητης αλληλογραφίας και η ανάλυση συναισθημάτων.

Εξαγωγή πληροφοριών (Information extraction)

Σε αυτή τη λειτουργία, ο στόχος είναι η εξαγωγή συγκεκριμένων πληροφοριών όπως ημερομηνίες, ώρες συνάντησης και συμβάντα από ηλεκτρονικά μηνύματα, έτσι ώστε να δοθεί η δυνατότητα στους χρήστες να εισάγουν αυτόματα την πληροφορία αυτή στο ηλεκτρονικό ημερολόγιο τους. Το ίδιο ισχύει για πληροφορίες όπως ονόματα, ηλεκτρονικές διευθύνσεις και τηλεφωνικοί αριθμοί.

Ανάκτηση πληροφορίας (Information retrieval)

Το έργο αυτής της διαδικασίας είναι η εύρεση εγγράφων που σχετίζονται με το ερώτημα που έχει θέσει ένας χρήστης και συνήθως χρησιμοποιείται από μηχανές αναζήτησης.

Διαλογικοί πράκτορες (Conversational agents)

Το καθήκον της λειτουργίας αυτής είναι η δημιουργία συστημάτων διαλόγου που έχουν τη δυνατότητα να συνομιλούν σε φυσική γλώσσα.

Σύνοψη κειμένου (Text summarization)

Σε αυτή τη λειτουργία, ο στόχος είναι η δημιουργία σύντομων περιλήψεων από εκτενή έγγραφα, χωρίς όμως να χάνεται το βασικό περιεχόμενο και η συνολική σημασία του.

Αυτόματη ερωταπόκριση (Question answering)

Στόχος αυτής της λειτουργίας είναι η δημιουργία ενός συστήματος που είναι σε θέση να απαντήσει αυτόματα ερωτήσεις που τίθενται σε φυσική γλώσσα.

Σε αυτή την πτυχιακή θα ασχοληθούμε με την ταξινόμηση κειμένου και συγκεκριμένα θα επικεντρωθούμε στην αναγνώριση μερών του λόγου.

2.2 Αναγνώριση μερών του λόγου

Τα μέρη του λόγου είναι χρήσιμες ενδείξεις για τη δομή και το νόημα των προτάσεων [3]. Γνωρίζοντας ότι μια λέξη είναι ουσιαστικό ή ρήμα, έχουμε περισσότερες πληροφορίες για τις πιθανές γειτονικές λέξεις και τη συντακτική δομή μιας πρότασης. Αυτό καθιστά την αναγνώριση μερών του λόγου μια βασική πτυχή της ανάλυσης κειμένων.

Η αναγνώριση μερών του λόγου (POS) ανήκει στη λειτουργία της ταξινόμησης ενός κειμένου, καθώς αυτό που κάνουμε είναι να αναθέτουμε ετικέτες στις λέξεις ανάλογα με το μέρος του λόγου στο οποίο ανήκουν.

Αφορά τη διαδικασία αντιστοίχησης ενός μέρους του λόγου σε κάθε λέξη ενός κειμένου. Η αναγνώριση είναι μια διαδικασία αποσαφήνισης, αφού οι λέξεις μπορεί να είναι διφορούμενες, δηλαδή να υπάρχουν περιπτώσεις που μια λέξη αντιστοιχεί σε περισσότερα από ένα μέρη του λόγου, και ο στόχος της είναι να βρεθεί η σωστή ετικέτα για την κάθε κατάσταση [4].

Πριν συνεχίσουμε με τη λειτουργία της αναγνώρισης μερών του λόγου είναι σημαντικό να αναφέρουμε αρχικά τα μέρη του λόγου στην ελληνική γλώσσα.

2.2.1 Μέρη του λόγου στην ελληνική γλώσσα

Με βάση την ελληνική γραμματική [5] υπάρχουν δέκα είδη λέξεων στην ελληνική γλώσσα τα οποία ονομάζονται μέρη του λόγου. Αυτά είναι τα ακόλουθα:

- άρθρο,
- ουσιαστικό,
- επίθετο,
- αντωνυμία,
- ρήμα,
- μετοχή,
- επίρρημα,
- πρόθεση,
- σύνδεσμος,
- επιφώνημα.

Το ουσιαστικό και το επίθετο λέγονται και ονόματα.

Αφού γνωρίζουμε τώρα τα μέρη του λόγου αξίζει να αναφερθεί ο τρόπος που αντιστοιχίζονται στις δύο στρατηγικές που ακολουθούνται σε αυτή την εφαρμογή.

Πίνακας 2.1: Ετικέτες με τα μέρη του λόγου κάθε στρατηγικής

Ετικέτα	Μέρος του λόγου	Βιβλιοθήκη spaCy	Προσαρμοσμένη υλοποίηση
ABBR	συντομογραφία	✗	✓
ADJ	επίθετο	✓	✓
ADP	πρόθεση	✓	✓
ADV	επίρρημα	✓	✓
AUX	βοηθητικό ρήμα	✓	✗
CONJ	σύνδεσμος	✓	✓
DET	άρθρο	✓	✓
INTJ	επιφώνημα	✓	✗
NOUN	ουσιαστικό	✓	✓
NUM	αριθμός	✓	✓
PART	μόρια	✓	✗
PARTCP	μετοχή	✗	✓
PRON	αντωνυμία	✓	✓
PROPN	κύριο όνομα	✓	✓
PUNCT	στίξη	✓	✓
SCONJ	δευτερεύον σύνδεσμος	✓	✗
SYM	σύμβολο	✓	✗
VERB	ρήμα	✓	✓
X	άλλο	✓	✓

2.3 Ετικέτες μερών του λόγου

Για την αντιστοίχηση των μερών του λόγου χρησιμοποιούνται συγκεκριμένες ετικέτες αντί των ορισμών τους, έτσι ώστε να είναι ευκολότερος ο προσδιορισμός τους στον κώδικα.

Η βιβλιοθήκη spaCy χρησιμοποιεί τις καθολικές ετικέτες μερών του λόγου (UPOS) [15]. Κατά συνέπεια το ίδιο πρότυπο χρησιμοποιήθηκε και για τη στρατηγική με τους προσαρμοσμένους αλγορίθμους.

Οι μόνες διαφορές όσον αφορά τις ετικέτες ανάμεσα στις δύο υλοποιήσεις, οι οποίες απεικονίζονται και στο πίνακα 2.1, είναι:

- στη προσαρμοσμένη υλοποίηση προστέθηκε η ετικέτα «PARTCP» για την αποσαφήνιση αντωνυμιών και η ετικέτα «ABBR» για την αντιπροσώπευση συντομογραφιών,
- η προσαρμοσμένη υλοποίηση δεν υποστηρίζει τις ετικέτες «AUX», «INTJ», «PART», «SCONJ» και «SYM», οι οποίες αντιστοιχούν σε βοηθητικά ρήματα, επιφωνήματα, μόρια, δευτερεύοντες σύνδεσμοι και σύμβολα.

2.3.1 Τρόποι αναγνώρισης μερών του λόγου

Γενικά μπορούμε να χωρίσουμε τους τρόπους αναγνώρισης μερών του λόγου σε δύο κατηγορίες: α) στην αναγνώριση βάσει κανόνων και β) στη στοχαστική αναγνώριση.

Στοχαστική αναγνώριση

Η στοχαστική αναγνώριση αναφέρεται σε οποιαδήποτε προσέγγιση που ενσωματώνει κατά κάποιο τρόπο τη συχνότητα ή την πιθανότητα εμφάνισης μιας λέξης σε μια πρόταση ή κείμενο. Σε αυτή τη προσέγγιση εφαρμόζονται πολλές τεχνικές από τομείς όπως η μηχανική μάθηση και η στατιστική.

Οποιαδήποτε στοχαστική αναγνώριση μερών του λόγου εφαρμόζει μία από τις παρακάτω προσεγγίσεις:

- Προσέγγιση συχνότητας λέξεων: η αποσαφήνιση γίνεται με βάση τη πιθανότητα εμφάνισης μιας λέξης με μια συγκεκριμένη ετικέτα.
- Προσέγγιση πιθανότητας ακολουθίας ετικετών: η αποσαφήνιση πραγματοποιείται υπολογίζοντας την πιθανότητα εμφάνισης μιας δεδομένης ακολουθίας ετικετών. Τα υπό συνθήκη τυχαία πεδία (CRF) και τα κρυφά Μαρκοβιανά μοντέλα (HMM) είναι πιθανολογικές προσεγγίσεις για την εκχώρηση ετικέτας [4].

Η στοχαστική αναγνώριση περιλαμβάνει αλγορίθμους οι οποίοι εκπαιδεύονται πάνω σε ένα συγκεκριμένο κείμενο (corpus).

Ένα τέτοιο σύστημα επεξεργασίας φυσικής γλώσσας χρησιμοποιεί στατιστικά μοντέλα για την κατανόηση του κειμένου και παράγει το προβλεπόμενο αποτέλεσμα, σε αντίθεση με την ανάλυση βάσει κανόνων στην οποία ακολουθείται μια σειρά από αυστηρά καθορισμένους κανόνες.

Στατιστικά μοντέλα

Στην επεξεργασία φυσικής γλώσσας, ένα στατιστικό μοντέλο περιέχει εκτιμήσεις με τη κατανομή πιθανότητας γλωσσικών τμημάτων, όπως οι λέξεις και φράσεις, δίνοντας τη δυνατότητα ανάθεσης γλωσσολογικών χαρακτηρισμών σε αυτά.

Στη θεωρία πιθανοτήτων και τη στατιστική, μια κατανομή πιθανότητας για μια συγκεκριμένη μεταβλητή είναι ένας πίνακας τιμών που περιέχει όλα τα πιθανά αποτελέσματα αυτής της μεταβλητής στις πιθανότητες εμφάνισής της σε ένα πείραμα [6].

Η στατιστική μοντελοποίηση των γλωσσών είναι είναι ζωτικής σημασίας για πολλές διαδικασίες επεξεργασίας φυσικών γλωσσών, όπως η δημιουργία κειμένου και η κατανόηση γραπτής ή προφορικής γλώσσας. Για το λόγο αυτό, ένα στατιστικό μοντέλο βρίσκεται στην καρδιά σχεδόν κάθε εφαρμογής ΕΦΓ [6].

Μοντέλα Νευρωνικών Δικτύων

Τα στατιστικά μοντέλα που χρησιμοποιούνται σε εργαλεία ΕΦΓ για την αναγνώριση μερών του λόγου είναι μοντέλα νευρωνικών δικτύων. Ένα νευρωνικό δίκτυο είναι ένα σύνολο από αλγορίθμους πρόβλεψης [6]. Αποτελείται από μεγάλο αριθμό απλών στοιχείων επεξεργασίας, όπως νευρώνες στον εγκέφαλο, που αλληλεπιδρούν στέλνοντας και λαμβάνοντας σήματα προς και από γειτονικούς κόμβους.

Συνήθως, οι κόμβοι σε ένα νευρωνικό δίκτυο ομαδοποιούνται σε στρώματα, που περιλαμβάνουν ένα επίπεδο εισόδου, ένα επίπεδο εξόδου και ένα ή περισσότερα κρυμμένα στρώματα μεταξύ τους. Κάθε κόμβος σε ένα επίπεδο (εκτός από το επίπεδο εξόδου) συνδέεται με κάθε κόμβο στο διαδοχικό επίπεδο μέσω σύνδεσης. Μια σύνδεση έχει μια τιμή βάρους που σχετίζεται με αυτήν. Κατά τη διάρκεια της εκπαίδευσης, ο αλγόριθμος προσαρμόζει τα βάρη για να ελαχιστοποιήσει το σφάλμα που κάνει στις προβλέψεις του. Αυτή η αρχιτεκτονική επιτρέπει σε ένα νευρωνικό δίκτυο να αναγνωρίζει μοτίβα, ακόμη και σε πολύπλοκες εισόδους δεδομένων.

Όταν εισέρχεται ένα σήμα, πολλαπλασιάζεται με μια τιμή βάρους, που είναι ένας πραγματικός αριθμός. Οι τιμές εισαγωγής και βάρους που μεταφέρονται σε ένα νευρωνικό δίκτυο προέρχονται γενικά από τα διανύσματα λέξεων που δημιουργούνται κατά τη διάρκεια της εκπαίδευσης του δικτύου.

Το νευρωνικό δίκτυο προσθέτει τα αποτελέσματα των πολλαπλασιασμών μαζί για κάθε κόμβο, μεταδίδει το άθροισμα σε μια συνάρτηση ενεργοποίησης. Η συνάρτηση ενεργοποίησης δημιουργεί ένα αποτέλεσμα που κυμαίνεται συνήθως από το 0 έως το 1, παράγοντας έτσι ένα νέο σήμα που μεταδίδεται σε κάθε κόμβο στο διαδοχικό στρώμα ή, στην περίπτωση του επιπέδου εξόδου, μια παράμετρο εξόδου. Συνήθως, το επίπεδο εξόδου έχει τόσους κόμβους όσο και τον αριθμό πιθανών ξεχωριστών εξόδων για τον δεδομένο αλγόριθμο.

Για παράδειγμα, ένα νευρωνικό δίκτυο που έχει δημιουργηθεί για την αναγνώριση μερών του λόγου, θα πρέπει να περιέχει στο επίπεδο εξόδου τόσες εξόδους όσες και ο αριθμός των μερών του λόγου που υποστηρίζει το σύστημα. Στη συνέχεια το σύστημα αντιστοίχησης παράγει τη κατανομή της πιθανότητας όλων των πιθανών μερών του λόγου που αντιστοιχούν σε μια δεδομένη λέξη και σε ένα δεδομένο πλαίσιο.

Κεφάλαιο 3ο: Ανάλυση Τεχνολογιών

Έχοντας καλύψει το θεωρητικό υπόβαθρο της πτυχιακής, μπορούμε να συνεχίσουμε με την υλοποίηση.

3.1 Ανάπτυξη κώδικα

Για τη συγγραφή του κώδικα χρησιμοποιήθηκε αποκλειστικά η εφαρμογή Visual Studio Code. Δημιουργήθηκαν δύο ξεχωριστά repositories, ένα για τη σελίδα (frontend) και ένα για το διακομιστή της διεπαφής (API backend server) και διατηρούνται στην πλατφόρμα GitHub:

- frontend: https://github.com/sortingbubbles/write-o-meter
- backend: https://github.com/sortingbubbles/write-o-meter-backend

3.2 Frontend

Υπάρχουν πολλοί τρόποι στις ημέρες μας για να δημιουργήσει κανείς μια διαδικτυακή εφαρμογή. Στην προκειμένη περίπτωση οι ανάγκες της εφαρμογής δεν είναι ιδιαίτερα περίπλοκες. Το μόνο που χρειαζόμαστε είναι μια μέθοδος για να παραλάβουμε το κείμενο που θα υποστεί την ανάλυση και ένα τρόπο για την εμφάνιση των αποτελεσμάτων της ανάλυσης.

Επίσης, θέλουμε να έχουμε τη δυνατότητα να επιλέγουμε ανάμεσα στις δύο στρατηγικές ανάλυσης, με χρήση α) των προσαρμοσμένων αλγορίθμων ή β) της βιβλιοθήκης spaCy.

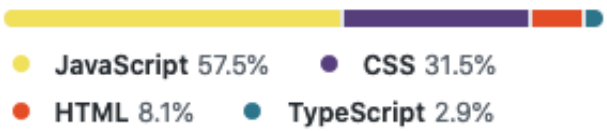
Όποτε η διεπαφή της εφαρμογής περιέχει:

- μια φόρμα με ένα πλαίσιο κειμένου,
- ένα κουμπί για την ολοκλήρωση της και
- ένα διακόπτη για την εναλλαγή από τη μία μέθοδο ανάλυσης στην άλλη.

Για την υλοποίηση της διαδικτυακής διεπαφής, όπως εμφανίζονται στην εικόνα 3.1, επιλέχθηκαν οι κλασικές τεχνολογίες: HTML, CSS, JavaScript, όπως και η βιβλιοθήκη React.js και η γλώσσα προγραμματισμού TypeScript. Για τη δημοσίευση της εφαρμογής έγινε χρήση της πλατφόρμας GitHub.

3.2.1 Κλασικές τεχνολογίες ανάπτυξης διαδικτυακών εφαρμογών

Οι τεχνολογίες HTML, CSS και JavaScript είναι καθιερωμένες για την ανάπτυξη ιστοσελίδων που δεν έχουν πολλές απαιτήσεις. Ειδικά χρησιμοποιώντας τις ανανεωμένες εκδόσεις της καθεμιάς τεχνολογί-



Σχήμα 3.1: Frontend τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής

ας, HTML5, CSS3 και ECMAScript 5, μπορούν να αναπτυχθούν και πολύπλοκες σελίδες με πλούσιο περιεχόμενο.

Ο συνδυασμός αυτών των τριών τεχνολογιών επιτρέπει τη δημιουργία απλών αλλά και περίπλοκων εφαρμογών, οι οποίες μπορούν εύκολα να δημοσιευτούν και να προβληθούν από κάθε είδους συσκευή που είναι συνδεδεμένη στο Διαδίκτυο.

HTML

Είναι σχεδόν σίγουρο στις ημέρες μας ότι η ανάπτυξη μιας διαδικτυακής εφαρμογής, με τον ένα ή τον άλλο τρόπο θα περιλαμβάνει στοιχεία της HTML εφόσον είναι η γλώσσα σήμανσης που χρησιμοποιούμε στο Διαδίκτυο.

Σε αυτή την εφαρμογή ακολουθήθηκαν οι κανονισμοί της HTML5 για τη διατήρηση της σημασιολογικής φύσης της γλώσσας και την εύκολη μετάβαση με χρήση μόνο του πληκτρολογίου ή ενός αναγνώστη οθόνης (e-reader).

Από την HTML τα βασικά στοιχεία που χρησιμοποιήθηκαν είναι τα εξής:

- <form>: για τη δημιουργία της φόρμας που εισάγει ο χρήστης το κείμενο
- <label>: για την καλύτερη σήμανση των στοιχείων της φόρμας
- <textarea>: για την εισαγωγή του κειμένου προς ανάλυση
- <input type=”submit”>: για την αποστολή της φόρμας
- <input type=”radio”>: για την εναλλαγή από τη μία στρατηγική ανάλυσης στην άλλη

CSS

Για την διαμόρφωση της εμφάνισης της σελίδας δε χρειάστηκε τίποτα παραπάνω από απλούς κανόνες CSS σε ένα κοινό αρχείο (App.css).

Η τοποθέτηση των στοιχείων στη σελίδα έγινε με τη χρήση των κανόνων flexbox. Για τη διαμόρφωση των στοιχείων <input type=”radio”> ακολουθήθηκαν οι οδηγίες από τη διαδικτυακή σελίδα του w3schools [7], όπως και η βοηθητική ένδειξη για τη φόρτωση των αποτελεσμάτων η οποία πραγματοποιήθηκε με τη βοήθεια της σελίδας loading.io [8].

Για τη δημιουργία της βασικής χρωματικής παλέτας της σελίδας χρησιμοποιήθηκε η ιστοσελίδα Coolors [9] και το εικονίδιο της εφαρμογής έγινε με τη χρήσης των σελίδων Photopea [10] και ConvertICO [11].

Το κάθε μέρος του λόγου εμφανίζεται στην εφαρμογή με συγκεκριμένο χρώμα και επιγραφή, έτσι ώστε ο χρήστης να έχει τη δυνατότητα να δει με αποτελεσματικό τρόπο την αντιστοίχηση των μερών του λόγου. Το σχεδιαστικό κομμάτι αυτής της παρουσίασης πραγματοποιήθηκε με τη χρήση κλάσεων.

Για την ονομασία των κλάσεων χρησιμοποιήθηκε η μέθοδος BEM. Σε αυτή τη μέθοδο η ονομασία των κλάσεων CSS ακολουθεί συγκεκριμένους κανόνες ανάλογα με το που βρίσκεται στην ιεραρχία το στοιχείο στο οποίο προσθέτουμε τη κλάση.

JavaScript

Για τη δυναμική αντιστοίχηση των κλάσεων στα αντίστοιχα στοιχεία και όλο το λειτουργικό κομμάτι της εφαρμογής, χρησιμοποιήθηκε η γλώσσα προγραμματισμού JavaScript, ως η πλέον διαδεδομένη τεχνολογία για ανάπτυξη διαδικτυακών σελίδων. Η έκδοση που χρησιμοποιήθηκε είναι η ECMAScript 5, καθώς υποστηρίζεται ευρέως από όλους τους κοινούς φυλλομετρητές.

Οι δύο ξεχωριστές κλήσεις στο backend server (μία για την κάθε μέθοδο ανάλυσης) έγιναν με τη χρήση της διεπαφής εφαρμογών fetch [12] [13].

Είναι αλήθεια πως για την ανάπτυξη μιας τόσο απλής διεπαφής ίσως θα ήταν αρκετό να χρησιμοποιήσει κανείς μόνο τις παραπάνω βασικές τεχνολογίες: HTML, CSS και JavaScript. Παρόλα αυτά προτίμησα να προσθέσω και δύο πιο νέες τεχνολογίες στο frontend κομμάτι της εργασίας, οι οποίες έμμεσα χρησιμοποιούν ή παράγουν κώδικα JavaScript. Οι τεχνολογίες αυτές είναι η βιβλιοθήκη React.js και η γλώσσα προγραμματισμού TypeScript και αναλύονται στη συνέχεια.

3.2.2 Νεότερες τεχνολογίες ανάπτυξης διαδικτυακών εφαρμογών

React.js

Για την ανάπτυξη της διεπαφής έγινε προσθήκη της βιβλιοθήκης React.js, η οποία είναι μια αρκετά δημοφιλής βιβλιοθήκη για τη δημιουργία διαδικτυακών εφαρμογών μιας σελίδας (SPA). Οι εφαρμογές μιας σελίδας ανανεώνουν δυναμικά το περιεχόμενο ενός ιστότοπου χωρίς να χρειαστεί η επαναφόρτωση της. Ουσιαστικά η εφαρμογή μιας σελίδας είναι κώδικας γραμμένος στη γλώσσα προγραμματισμού JavaScript, ο οποίος είναι τακτοποιημένος σε φακέλους και αρχεία τα οποία μπορεί να αναπτύξει και οργανώσει κανείς με εργαλεία που παρέχονται από την ίδια την εφαρμογή.

Όταν ένας χρήστης επισκέπτεται τη διεύθυνση URL που αντιστοιχεί στην διαδικτυακή εφαρμογή, όλοι οι πόροι, όπως ο κώδικας, οι εικόνες και τα αρχεία ήχου, καθώς και η αντιμετώπιση των αλληλεπιδράσεων του χρήστη γίνονται μέσω της εφαρμογής μιας σελίδας. Στην προκειμένη περίπτωση η βιβλιοθήκη React είναι αυτή που αναλαμβάνει αυτό το ρόλο.

Πέρα από τον ευκολότερο χειρισμό μιας εφαρμογής, με τη βιβλιοθήκη React έγινε δημοφιλή η ιδέα των συστατικών (components). Κάθε component είναι ένα μικρό τμήμα της εφαρμογής που περιλαμβάνει ξεχωριστά το κώδικα HTML, CSS και JavaScript που αντιστοιχεί σε αυτό το τμήμα. Μόλις οριστεί ένα component, μπορεί να χρησιμοποιηθεί από άλλα components, βοηθώντας έτσι τη δημιουργία ολόκληρης της εφαρμογής.

Στη συγκεκριμένη περίπτωση δημιουργήθηκε ένα component, με το όνομα «WriteOMeterForm» που περιέχει όλα τα στοιχεία εισόδου, ελεύθερο κείμενο, επιλογή για το είδος της ανάλυσης και το κουμπί για την αποστολή των στοιχείων της φόρμας. Στο ίδιο component τοποθετήθηκε και το πλαίσιο των αποτελεσμάτων, καθώς το αρχείο δε ξεπερνά τις 200 γραμμές κώδικα. Μπορείτε να δείτε τη σύσταση του component στην εικόνα 3.2 και αναλυτικά όλα τα κομμάτια του κώδικα στο παράρτημα Β.

Με τη χρήση μιας τέτοιας βιβλιοθήκης η διαδικασία δημιουργίας της φόρμας αποστολής για το κείμενο έγινε πιο απλή. Επίσης ο χειρισμός των διαφορετικών σταδίων (αρχική εμφάνιση, αποστολή κειμένου,

αναμονή αποτελεσμάτων, εμφάνιση αποτελεσμάτων) της εφαρμογής έγινε πολύ πιο εύκολος.

TypeScript

Τέλος, θέλοντας να ακολουθήσω τα πρότυπα που χρησιμοποιούνται σήμερα, πρόσθεσα και κομμάτια κώδικα που είναι γραμμένα με τη γλώσσα προγραμματισμού TypeScript, η οποία αποτελεί ένα υπερσύνολο της γλώσσας προγραμματισμού JavaScript. Η σημαντική διαφορά και το μεγαλύτερο πλεονέκτημα που προσφέρει η γλώσσα προγραμματισμού TypeScript έχει να κάνει με τον καθορισμό των τύπων κάθε μεταβλητής και παραμέτρων. Έτσι για κάθε μεταβλητή είναι αναγκαίο να οριστεί το είδος της τιμής του, είτε αυτό είναι μια συμβολοσειρά, είτε αριθμός, είτε άλλα είδη πολύπλοκων αντικειμένων τα οποία ορίζονται στο πλαίσιο κάθε εφαρμογής.

Ο κώδικας που είναι γραμμένος σε TypeScript περνά από στατικό έλεγχο και στη συνέχεια μεταγλωττίζεται σε κώδικα JavaScript. Στη περίπτωση που βρεθεί μεταβλητή με τιμή διαφορετική από το είδος στο οποίο έχει καθοριστεί, τότε ο μηχανισμός της TypeScript εμφανίζει το λάθος κατά τη διάρκεια της

```
1   import React from 'react'
2   import { analyzeWithCustomAlgorithms, analyzeWithSpacy } from './api.ts'
3   import { partsOfSpeech } from './parts-of-speech.ts'
4
5   class WriteOMeterForm extends React.Component {
6     constructor(props) {···
20    }
21
22    handleChange(event) {···
24    }
25
26    handleAnalysisMethodChange(event) {···
28    }
29
30    getClassByPartOfSpeech(pos) {···
32    }
33
34    showStatisticResults() {···
44    }
45    showResults() {···
64    }
65
66    showLoadingState() {···
75    }
76
77    async handleSubmit(event) {···
94    }
95
96    render() {
97      const hasResults = !!this.state.results
98      const isLoading = this.state.isLoading
99      return (
100       <form className="wom-form" onSubmit={this.handleSubmit}>
101         <section className="wom-form__controls">
102           <label className="wom-form__text-area">···
105           <section>
106             <section className="wom-form__controls__method-options">···
133             {hasResults && this.showStatisticResults()}
134           </section>
135         </section>
136         <input disabled={this.state.isLoading} className="wom-form__input" type="submit" value="Ανάλυση" />
137         {isLoading && this.showLoadingState()}
138         {hasResults && this.showResults()}
139       </form>
140     )
141   }
142 }
143 export default WriteOMeterForm
```

Σχήμα 3.2: Το βασικό component της εφαρμογής

μεταγλώττισης. Ο κώδικας JavaScript που δημιουργείται μετά τη διαδικασία μεταγλώττισης δεν περιλαμβάνει κανένα έλεγχο προς το τύπο των αρχείων ή κάποια παραπάνω λειτουργία.

Η αυστηρότητα της γλώσσας προγραμματισμού TypeScript μπορεί να προσαρμοστεί μέσω του αρχείου tsconfig.json [14]. Με βάση αυτό ρυθμίζονται οι κανόνες τους οποίους θα πρέπει να ακολουθεί ο κώδικα. Παράλληλα προσδιορίζονται ακριβώς ποια αρχεία και φάκελοι πρέπει να περάσουν από τη διαδικασία στατικού ελέγχου και μεταγλώττισης.

Οι ρυθμίσεις που επιλέχθηκαν για τη συγκεκριμένη εφαρμογή, όπως εμφανίζονται στην εικόνα 3.3, προσδιορίζουν ότι η εφαρμογή περιέχει αρχεία TypeScript αλλά και αρχεία JavaScript. Επίσης με τη προσθήκη ελέγχου στα αρχεία jsx, προσδιορίζεται ότι χρησιμοποιείται και η βιβλιοθήκη react. Ακόμη προσδιορίζεται και η στοχευμένη έκδοση της JavaScript που δημιουργείται με τα τη μεταγλώττιση, η οποία είναι η ECMAScript 5.

3.2.3 Δημοσίευση σελίδας

Εφόσον ο κώδικας της εφαρμογής είναι δημοσιευμένος στη πλατφόρμα GitHub, για τη δημοσίευση της σελίδας έγινε η χρήση της λειτουργίας GitHub pages [15], η οποία δουλεύει αυτόματα με εφαρμογές React. Η υπηρεσία αυτή συγκετρώνει το στατικό περιεχόμενο που παράγεται από τη βιβλιοθήκη React, το οποίο βρίσκεται μέσα στο τοπικό φάκελο dist, και το δημοσιεύει σε μια διεύθυνση που βασίζεται στο όνομα χρήστη και το όνομα του έργου στη σελίδα του GitHub.

Οπότε με μια απλή εντολή yarn deploy η σελίδα δημοσιεύτηκε μέσω του προσωπικού μου λογαριασμού στο GitHub στη διεύθυνση https://sortingbubbles.github.io/write-o-meter.

Έχοντας αναλύσει το frontend κομμάτι της εφαρμογής, ήρθε η στιγμή να αναλύσουμε το backend κομμάτι.

tsconfig.json > ...

```
{
  "compilerOptions": {
    "target": "es5",
    "lib": [
      "dom",
      "dom.iterable",
      "esnext"
    ],
    "allowJs": true,
    "skipLibCheck": true,
    "esModuleInterop": true,
    "allowSyntheticDefaultImports": true,
    "strict": true,
    "forceConsistentCasingInFileNames": true,
    "module": "esnext",
    "moduleResolution": "node",
    "resolveJsonModule": true,
    "isolatedModules": true,
    "noEmit": true,
    "jsx": "react"
  },
  "include": [
    "src"
  ]
}
```

Σχήμα 3.3: Αρχείο tsconfig της εφαρμογής

3.3 Backend

Για την τέλεση της ανάλυσης δημιουργήθηκε μια backend εφαρμογή η οποία λειτουργεί ως εξυπηρετητής και επικοινωνεί με τη frontend εφαρμογή για να παραλάβει το κείμενο προς ανάλυση και να επιστρέψει τα αποτελέσματα.

Ο λόγος που χρειαζόμαστε τον εξυπηρετητή είναι επειδή η βιβλιοθήκη spaCy, δεν παρέχεται μέσω μιας διεπαφής για την διεξαγωγή της ανάλυσης του κειμένου, οπότε δεν άμεσος υπάρχει τρόπος επικοινωνίας ανάμεσα στην frontend εφαρμογή με τη βιβλιοθήκη.

Επιπλέον, ήταν θεμιτό η προσαρμοσμένη ανάλυση με τους αλγορίθμους γραμματικής να γίνει σε έναν server, καθώς απαιτεί αρκετούς υπολογισμούς και ελέγχους. Σε αντίθετη περίπτωση, δηλαδή στη περίπτωση που η προσαρμοσμένη ανάλυση πραγματοποιούνταν στο frontend κομμάτι της εφαρμογής, η απόδοση της σελίδας θα βασιζόταν στενά στην υπολογιστή ισχύ του υπολογιστή του κάθε χρήστη. Ενώ τώρα η ανάλυση γίνεται από έναν απομακρυσμένο υπολογιστή ο οποίος έχει τους δικούς του πόρους.

3.3.1 Αρχιτεκτονική

Αρχικά, χρειάστηκε να δημιουργηθεί και να δημοσιευθεί μια διεπαφή προγραμματισμού εφαρμογών (REST API) με δύο σημεία επαφής (endpoints) τα οποία αντιστοιχούν στις δύο διαφορετικές στρατηγικές ανάλυσης. Με αυτό το τρόπο έγινε δυνατή δυνατή η κλήση των αιτημάτων (HTTP requests) από τη δημοσιευμένη σελίδα του frontend κομματιού της εφαρμογής.

Τα σημεία επαφής είναι τα ακόλουθα:

- /analyze-custom: αυτό το endpoint καλείται για την ανάλυση με τους αλγορίθμους γραμματικής και τελεί όλες τους κανόνες που θα αναλύσουμε στη συνέχεια. Η μορφή του αποτέσματος που επιστρέφεται μέσω αυτής της κλήσης απεικονίζεται στην εικόνα 3.4.
- /analyze-spacy: αυτό το endpoint καλείται για την ανάλυση με τη χρήση της βιβλιοθήκης spaCy, όπως προδιαθέτει και το όνομα. Αντίστοιχα η μορφή του αποτελέσματα που επιστρέφεται από τον εξυπηρετητή σε αυτή τη περίπτωση απεικονίζεται στην εικόνα 3.5

Για κάθε σημείο επαφής, η πληροφορία που απαιτείται είναι το κείμενο που θα τεθεί σε ανάλυση στο σώμα (body) με τη παρακάτω μορφή: {text: 'Κείμενο προς ανάλυση'}.

Στη συνέχεια, στη διεπαφή έγινε η εισαγωγή της βιβλιοθήκης spaCy όπως και ένα από τα τρία διαθέσιμα μοντέλα για την ελληνική γλώσσα. Επιλέχθηκε το πιο ελαφρύ λόγω του μικρού μεγέθους της εφαρμογής.

3.3.2 Γλώσσα προγραμματισμού και βιβλιοθήκες

Python

Για την backend υποδομή της εφαρμογής επιλέχθηκε η γλώσσα προγραμματισμού Python και αυτό επειδή είναι απλή, ισχύρη και αρκετά εδραιωμένη στο τομέα της εξόρυξης δεδομένων και ανάλυσης κειμένων. Σημαντικό ρόλο στην επιλογή έπτο γεγονός ότι η βιβλιοθήκη spaCy είναι γραμμένη με αυτή τη γλώσσα.

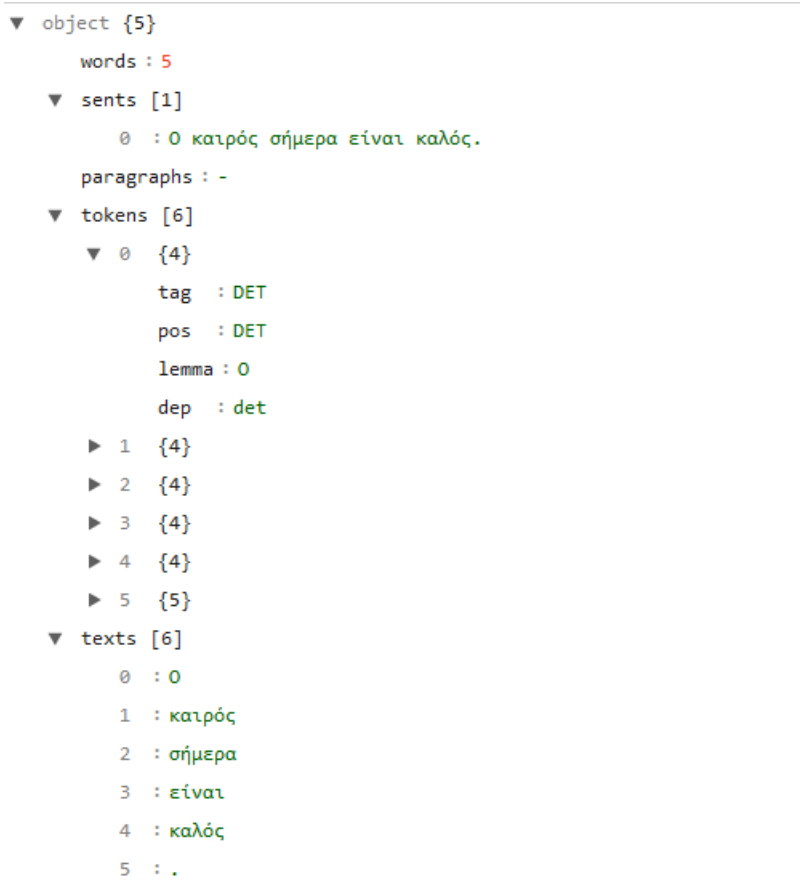
Επίσης η Python είναι μια γλώσσα προγραμματισμού ενσωματωμένη με πολλές και διαφορετικές μεθόδους επεξεργασίας συμβολοσειρών, όπως οι μέθοδοι startsWith και endsWith, πράγμα που βοήθησε στην υλοποίηση των ελέγχων και την αντιστοίχιση στην προσαρμοσμένη ανάλυση.

Βιβλιοθήκη re (regular expressions)

Για την ανάλυση με τους προσαρμοσμένους αλγορίθμους γραμματικής χρειάστηκαν να γίνουν πολλές πράξεις και έλεγχοι ανάμεσα σε συμβολοσειρές. Για αυτό το λόγο χρειάστηκε η βιβλιοθήκη re της python, η οποία έκανε εύκολη την αντικατάσταση χαρακτήρων και τον διαχωρισμό συμβολοσειρών σε λέξεις και προτάσεις.

Βιβλιοθήκη pydantic

Με αυτή τη βιβλιοθήκη κάνουμε χρήση του πεδίου BaseModel που επιτρέπει να ορίσουμε μορφές δεδομένων, έτσι ώστε να υπάρχει μεγαλύτερη ασφάλεια και μικρότερα περιθώρια λάθους στις πληροφορίες που δεχόμαστε και επιστρέφουμε.



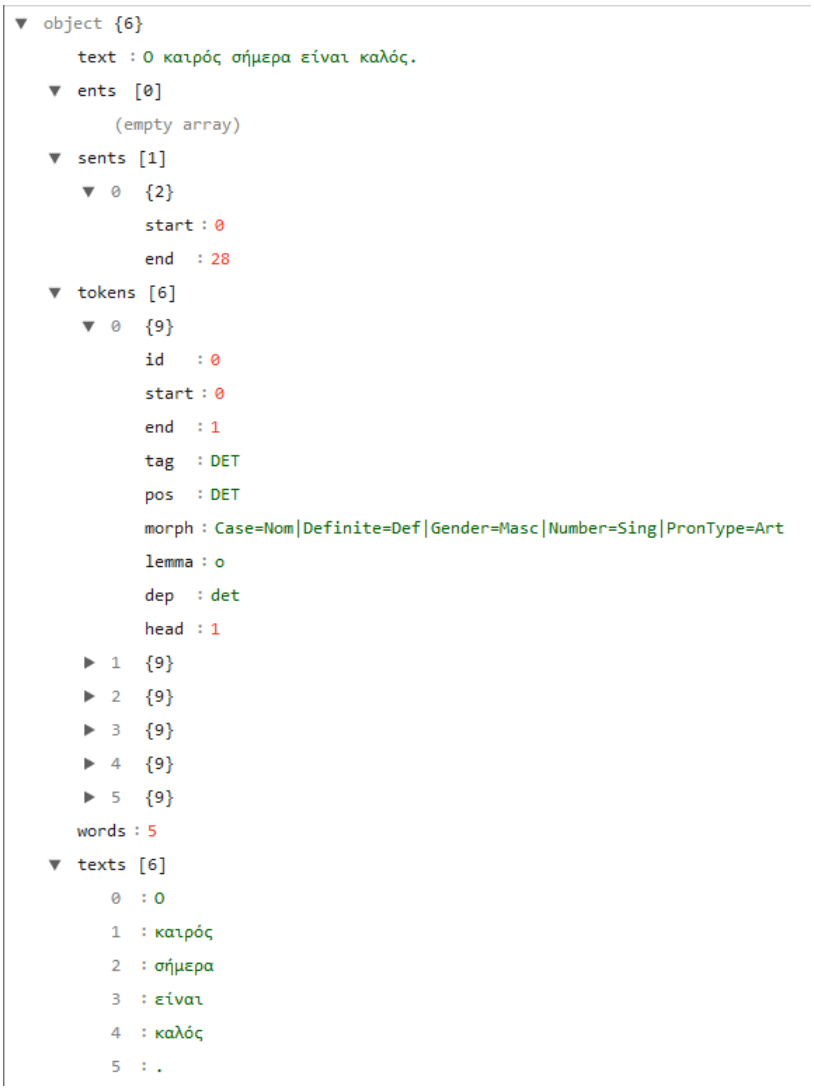
Σχήμα 3.4: Απάντηση εξυπηρετητή με τους προσαρμοσμένους αλγορίθμους

Βιβλιοθήκη spaCy

Για την εφαρμογή της στοχαστικής ανάλυσης ενός κειμένου επιλέχθηκε η βιβλιοθήκη spaCy, καθώς είναι μια αξιόπιστη βιβλιοθήκη ανοιχτού κώδικα που χρησιμοποιείται σε πολλές εφαρμογές ΕΦΓ. Παρότι δημιουργήθηκε πριν πέντε χρόνια, διατηρείται και ανανεώνεται συνεχώς. Για την πραγματοποίηση της ανάλυσης με τη βιβλιοθήκη spaCy ακολουθήθηκε η εξής διαδικασία:

1. φόρτωση της βιβλιοθήκης spaCy με το μοντέλο της ελληνικής γλώσσας,
2. παραλαβή του κειμένου που στάλθηκε από τη σελίδα μέσω της διεπαφής,
3. κάλεσμα της μεθόδου ανάλυσης που παρέχει η βιβλιοθήκη,
4. διαμόρφωση και επιστροφή του αποτελέσματος της ανάλυσης.

Οι λειτουργίες και δυνατότητες που παρέχει αυτή η βιβλιοθήκη θα ακολουθήσουν σε επόμενο κεφάλαιο.



Σχήμα 3.5: Απάντηση εξυπηρετητή με τη βιβλιοθήκη spaCy

3.3.3 Τεχνολογίες για τη δημιουργία και τη δημοσίευση API

FastAPI

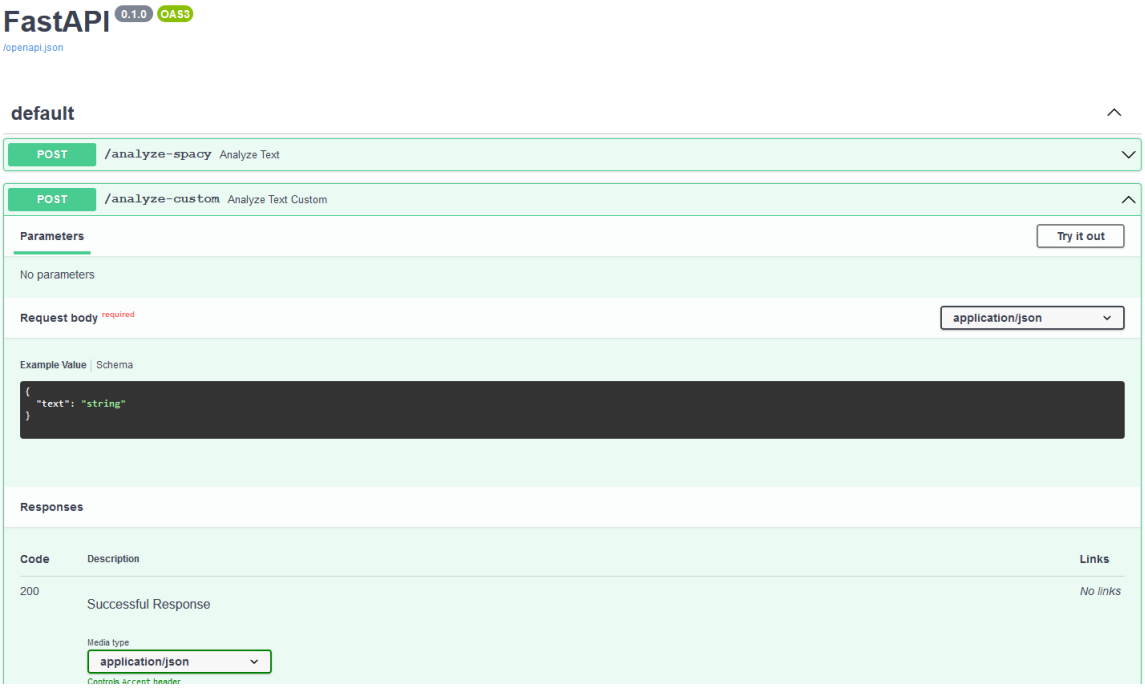
Από τις διάφορες επιλογές που υπάρχουν για τη δημιουργία REST API με τη γλώσσα προγραμματισμού Python, επέλεξα την τεχνολογία FastAPI. Όπως προδιαθέτει το όνομα, είναι γρήγορη στη δημιουργία αλλά και στη λειτουργία, και έκανε τη διαδικασία της δημοσίευσης πιο απλή.

Επιπρόσθετα, η βιβλιοθήκη αυτή παρέχει τη λειτουργία CORSMiddleware, με την οποία αποφεύγουμε προβλήματα σύνδεσης ανάμεσα στη δημοσιευμένη σελίδα στο GitHub και τη backend διεπαφή, καθώς υπάρχουν κανόνες που δεν επιτρέπουν συνδέσεις από διαφορετικά πεδία. Με αυτή τη λειτουργία μπορούμε με ασφάλεια να επιτρέψουμε συνδέσεις από τη σελίδα στον server.

Deta

Η δημοσίευση της διεπαφής έγινε με τη χρήση της πλατφόρμας Deta, η οποία συνεργάζεται άψογα με το FastAPI. Η πλατφόρμα αυτή, όχι μόνο δίνει τη δυνατότητα δημοσίευσης της εφαρμογής στο Διαδίκτυο, αλλά παράλληλα παρέχει αρχεία καταγραφής όλων των κλήσεων που γίνονται στον εξυπηρετητή. Αυτό βοήθησε σημαντικά τον εντοπισμό σφαλμάτων, ειδικά στην αρχή κατά τον πειραματισμό με τις ρυθμίσεις της διεπαφής.

Επίσης, δημιουργώντας τη διεπαφή με τα δύο σημεία επικοινωνίας (API endpoints) δημοσιεύτηκε αυτόματα και μια σελίδα που περιγράφει κάθε σημείο, τι δεδομένα δέχεται και τι επιστρέφει. Η σελίδα είναι προσβάσιμη στην εξής διεύθυνση https://vzjega.deta.dev/docs και απεικονίζεται στην εικόνα 3.6.



Σχήμα 3.6: Σελίδα με τις πληροφορίες των endpoints

Κεφάλαιο 4ο: Στρατηγικές υλοποίησης

Στην εφαρμογή, εμφανίζονται δύο διαφορετικές επιλογές στο χρήστη:

- ανάλυση με τους αλγορίθμους γραμματικής και
- ανάλυση με τη βιβλιοθήκη spaCy

Η κάθε επιλογή αντιπροσωπεύει τη διαφορετική στρατηγική ανάλυσης που θα υποστεί το κείμενο που θέτει ο χρήστης προς ανάλυση. Σε αυτό το κεφάλαιο θα αναλύσουμε αυτές τις δύο διαφορετικές στρατηγικές που χρησιμοποιήθηκαν, δίνοντας περισσότερη έμφαση στην αναγνώριση μερών του λόγου.

4.1 Ανάλυση με τους αλγορίθμους γραμματικής

Σε αυτή τη στρατηγική όλος ο κώδικας και η λογική αναπτύχθηκε από εμένα. Εξαίρεση αποτελεί η χρήση της βιβλιοθήκης re η οποία δίνει τη δυνατότητα εκτέλεσης κανονικών εκφράσεων (regular expressions).

Για την αναγνώριση των μερών του λόγου σε αυτή τη στρατηγική, ακολουθείται η λειτουργία βάσει κανόνων τους οποίους ανέπτυξα μεταφράζοντας μερικούς κανόνες της ελληνικής γραμματικής σε αλγορίθμους. Παράλληλα έφτιαξα μια συλλογή από λεξικά που περιέχουν για παράδειγμα όλα τα άρθρα, τα επιρρήματα ή κοινές καταλήξεις που υποδεικνύουν συγκεκριμένο μέρος του λόγου.

4.1.1 Βήματα ανάλυσης

Μετά τη παραλαβή του κειμένου προς ανάλυση από το χρήστη, στο backend κομμάτι ξεκινά η διαδικασία της ανάλυσης. Ο κώδικας που χρησιμοποιείται σε αυτή τη φάση εμφανίζεται στην εικόνα 4.1.

Τα βήματα που εκτελούνται σε αυτή την ανάλυση είναι τα εξής:

1. Αρχικά δημιουργούμε μια μεταβλητή που θα χρησιμοποιήσουμε για να επιστρέψουμε τα αποτελέσματα της ανάλυσης.
2. Στη συνέχεια αφαιρούνται τα κενά που είναι συνεχόμενα από το κείμενο έτσι ώστε να μπορέσουμε να το χωρίσουμε αποδοτικότερα.
3. Το κείμενο χωρίζεται σε προτάσεις και αποθηκεύουμε το αποτέλεσμα στη μεταβλητή που δημιουργήσαμε στην αρχή.
4. Σε αυτό το βήμα πραγματοποιείται η αντιστοίχηση των ετικετών σε κάθε λέξη, καθώς καλείται η μέθοδος που πραγματοποιεί την αποσαφήνιση. Θα αναλύσουμε περισσότερο αυτή το βήμα στη συνέχεια καθώς είναι το πιο σημαντικό.
5. Μετά την ανάλυση και αποθήκευση των ετικετών για κάθε λέξη, γίνεται η ένωση και καταχώρηση τους στο τελικό αποτέλεσμα για να μπορέσουμε να δείξουμε τις λέξεις με τις ετικέτες στη σελίδα.
6. Καλούμε τη μέθοδο για την καταμέτρηση των λέξεων, αποθηκεύοντας το αποτέλεσμα στη μεταβλητή με τα αποτελέσματα.

7. Τέλος, επιστρέφουμε τη μεταβλητή με τα αποτελέσματα.

Πριν αρχίσουμε τους ελέγχους και τις συγκρίσεις, αφαιρούμε τα πολλαπλά κενά σε όλο το κείμενο, έτσι ώστε να ξεκινήσουμε την ανάλυση χωρίζοντας το κείμενο σε προτάσεις.

4.1.2 Διαχωρισμός προτάσεων

Ο χωρισμός προτάσεων πραγματοποιείται ψάχνοντας για τα σημεία στίξης που υποδηλώνουν το τέλος μια πρότασης και ακολουθούνται από ένα κενό.

Αυτά τα σημεία στίξης είναι η μονή τελεία (.), τα αποσιωπητικά (...), το ερωτηματικό (;) και το θαυμαστικό (!). Η πράξη αυτή πραγματοποιείται με τη βοήθεια κανονικών εκφράσεων και το αποτέλεσμα του χωρισμού αποθηκεύεται στην τελική μεταβλητή ως ένας πίνακας από συμβολοσειρές.

Για να αναδείξουμε τον αριθμό των προτάσεων στη σελίδα, το μόνο που έχουμε να κάνουμε είναι να δείξουμε το μήκος του πίνακα που καταχωρήθηκε στη μεταβλητή «sents».

4.1.3 Αποσαφήνιση μερών του λόγου

Εφόσον έχουμε αφαιρέσει ήδη τα πολλαπλά κενά από το κείμενο, ο διαχωρισμός των λέξεων γίνεται με βάση τα κενά διαστήματα μέσα στο κείμενο.

Όπως προαναφέρθηκε στην εισαγωγή αυτού του κεφαλαίου, η αναγνώριση των μερών του λόγου σε αυτή την περίπτωση γίνεται βάσει κανόνων. Οι κανόνες στη προκειμένη περίπτωση είναι μια σειρά από έλεγχοι οι οποίοι αφορούν συγκεκριμένα χαρακτηριστικά που μπορεί να έχει ένα μέρος του λόγου.

Συγκεκριμένα, για την επίτευξη της αναγνώρισης δημιουργήθηκαν πίνακες με όλα τα άρθρα, αντωνυμίες, επιρρήματα, προθέσεις και συνδέσμους που βρίσκονται στην ελληνική γραμματική [5], και χρησιμοποιούνται στην αναγνώριση ως λεξικά (dictionaries).

```
40  async def analyze_text_custom(request: AnalysisRequestBody):
41      res = {
42          'words': '-',
43          'sents': '-',
44          'paragraphs': '-'
45      }
46      text = re.sub(' +', ' ', request.text)
47      res['sents'] = re.split('\\. |\\.\\.\\. |; | ! ', text)
48      res['tokens'] = tokenize_sentences(text.strip().split(' '))
49      res['texts'] = split_texts(res['tokens'])
50      res['words'] = find_num_of_words(res['tokens'])
51      return res
```

Σχήμα 4.1: Κώδικας με τα βασικά βήματα

Πέρα από τα λεξικά, έλεγχοι γίνονται και στις καταλήξεις των λέξεων. Για παράδειγμα, μια λέξη που τελειώνει σε '-ένος', '-ένη', '-ένο', '-ώντας', '-όντας' αντιστοιχίζεται ως μετοχή.

Η χρήση των λεξικών και των καταλήξεων στις περισσότερες περιπτώσεις οδηγούν σε σωστή αντιστοίχηση του μέρος του λόγου που ανήκει η λέξη, παρόλα αυτά είναι αδύνατο να καλύψουν όλες τις λέξεις της ελληνικής γλώσσας. Ειδικά εάν αναλογιστεί κανείς τον αριθμό των πτώσεων, τους χρόνους των ρημάτων και όλες τις υπόλοιπες ιδιομορφίες της ελληνικής γλώσσας. Για αυτό το λόγο, μαζί με τη χρήση των λεξικών, συμπεριλαμβάνονται και έλεγχοι που βασίζονται στη προηγούμενη και επόμενη λέξη από αυτή που προσπαθούμε να προσδιορίσουμε.

Για να λειτουργήσουν αυτοί οι έλεγχοι σημαίνει ότι κάποιες λέξεις αντιστοιχίζονται πριν από κάποιες άλλες και βοηθούν στην αντιστοίχηση των υπόλοιπων. Με λίγα λόγια δηλαδή, η αντιστοίχηση χωρίζεται σε διαφορετικές φάσεις, οι οποίες πραγματοποιούνται η μία μετά την άλλη ολοκληρώνοντας σταδιακά την ανάλυση του κειμένου.

Στη συγκεκριμένη περίπτωση η ανάλυση μπορεί να χωριστεί σε τρεις φάσεις:

1. Στην πρώτη φάση συγκρίνοντας την κάθε λέξη με τα λεξικά και τις καταλήξεις χωρίζουμε τις λέξεις από τα σημεία στίξης και κάνουμε την αντιστοίχηση για τις λέξεις οι οποίες ταιριάζουν σε κάποιο μέρος του λόγου από τους ελέγχους που κάνουμε.

2. Στη δεύτερη φάση για κάθε λέξη που δεν έχει γίνει αντιστοίχηση, παίρνουμε την προηγούμενη και την επόμενη λέξη (εάν υπάρχουν) και πραγματοποιούμε ελέγχους για να διακρίνουμε εάν η πρώτη λέξη είναι κύριο όνομα, ουσιαστικό ή επίθετο. Για παράδειγμα, θεωρούμε πως μια λέξη είναι κύριο όνομα όταν το πρώτο γράμμα είναι κεφαλαίο και η προηγούμενη ετικέτα δεν είναι σημείο στίξης.

3. Στην τρίτη και τελική φάση επαναλαμβάνουμε για δεύτερη φορά τους ίδιους ελέγχους που κάναμε στη δεύτερη φάση αλλά μόνο για τα στοιχεία που δεν έχουν ετικέτα.

4. Εάν ακόμη και μετά τη τελική φάση βρεθούν λέξεις που δεν έχουν ετικέτα, τότε σε αυτές τις λέξεις εκχωρούμε την ετικέτα X η οποία αντιστοιχεί και εμφανίζεται ως «άλλο» στην εφαρμογή.

Σε αυτό το σημείο θα ήθελα να προσθέσω ότι η συγκεκριμένη διαδικασία μπορεί να επεκταθεί με τη προσθήκη παραπάνω ελέγχων και φάσεων, καθώς όσες περισσότερες φορές σκανάρεται το κείμενο τόσο

```python
def find_num_of_words(tokens):
    counter = 0
    for token in tokens:
        if token['pos'] == 'ABBR':
            counter = counter + token['numOfWords']
        if token['pos'] != 'PUNCT' and token['pos'] != 'ABBR':
            counter = counter + 1
    return counter
```

Σχήμα 4.2: Κώδικας με τον αλγόριθμο καταμέτρησης λέξεων

πιο πιθανό είναι να βρεθούν περισσότερες αντιστοιχίσεις που βασίζονται στην αποτελεσματικότητα της προηγούμενης φάσης.

Στη συγκεκριμένη περίπτωση αποφάσισα ότι τρεις επαναλήψεις είναι αρκετές με αυτούς τους ελέγχους, επειδή δεν ήθελα να αυξήσω κατά πολύ τη πολυπλοκότητα των αλγορίθμων και δεν παρατήρησα μεγάλη διαφορά όταν επανέλαβα τους ελέγχους για μια τέταρτη φορά.

4.1.4 Καταμέτρηση λέξεων

Η καταμέτρηση των λέξεων, όπως εμφανίζεται στην εικόνα 4.2, πραγματοποιείται μετά την αποσαφήνιση των μερών του λόγου γιατί πρέπει να αφαιρέσουμε τα σημεία στίξης από το σύνολο των λέξεων πριν ξεκινήσουμε το μέτρημα.

Επίσης κατά αυτόν τον τρόπο είμαστε σε θέση να συμπεριλάβουμε στο σύνολο των λέξεων και τις λέξεις που προκύπτουν από τις συντομογραφίες, τις οποίες έχουμε ήδη αναγνωρίσει μετρώντας τον αριθμό των τελειών (.) που εμφανίζονται σε μια διαχωρισμένη λέξη.

4.2 Ανάλυση με τη βιβλιοθήκη spaCy

Η δεύτερη στρατηγική που χρησιμοποιήθηκε για την ανάλυση του κειμένου ακολουθεί τη στοχαστική ανάλυση και πραγματοποιήθηκε με τη χρήση της βιβλιοθήκης spaCy.

4.2.1 Η βιβλιοθήκη spaCy

Η βιβλιοθήκη spaCy είναι μια ευρέως γνωστή βιβλιοθήκη ανοιχτού κώδικα της γλώσσας προγραμματισμού Python, η οποία περιλαμβάνει γλωσσολογικά δεδομένα και αλγορίθμους που μπορεί να χρησιμοποιήσει κανείς για να επεξεργαστεί μια φυσική γλώσσα [16].

Οι λόγοι για τον οποίους η βιβλιοθήκη spaCy είναι τόσο γνωστή αφορούν το γεγονός ότι είναι εύκολη στη χρήση, προσφέρει εκπαιδευμένα μοντέλα στις εξής γλώσσες: Αγγλικά, Γαλλικά, Γερμανικά, Ελληνικά, Ισπανικά, Ιταλικά, Λιθουανικά, Νορβηγικά, Ολλανδικά, Πορτογαλικά, καθώς και τον μερικό συνδυασμό διαφορετικών γλωσσών.

Δυνατότητες της βιβλιοθήκης spaCy

Αυτο-χαρακτηρίζοντας τον εαυτό της ως έτοιμη για χρήση σε περιβάλλον παραγωγής [16] παρέχει τις εξής δυνατότητες:

- Διαχωρισμός λεξικογραφικών μονάδων (tokenization): διαχωρίζει κείμενα σε λέξεις, σημεία στίξης, κτλ
- Σήμανση μερών του λόγου (part-of-speech tagging): διαχωρίζει στοιχεία και λέξεις σε μέρη του λόγου.
- Ανάλυση εξαρτήσεων (dependency parsing): αναθέτει ετικέτες συντακτικής εξάρτησης που περιγράφουν τις σχέσεις μεταξύ λέξεων.

- Λημματοποίηση (lemmatization): εκχωρεί τις βασικές μορφές των λέξεων. Για παράδειγμα, η βασική μορφή της λέξης "έκλαψε" είναι το ρήμα "κλαίω".
- Ανίχνευση ορίων προτάσεων (sentence boundary detection): βρίσκει και χωρίζει μεμονωμένες προτάσεις από ένα κείμενο.
- Εντοπισμός ονοματικών οντοτήτων (named entity recognition): αναγνωρίζει και επισημάνει λέξεις του "πραγματικού κόσμου" όπως εταιρίες, τοποθεσίες κτλ.
- Συνδέσεις οντοτήτων (entinty linking): αναθέτει μοναδικά αναγνωριστικά σε οντότητες από μια βάση γνώσεων.
- Ομοιότητες (similarity): συγκρίνει λέξεις, κείμενα και έγγραφα για να βρει πόσο όμοια είναι μεταξύ τους.
- Ταξινόμηση κειμένου (text classification): αναθέτει κατηγορίες ή ετικέτες σε ένα ολόκληρο έγγραφο ή κομμάτια ενός εγγράφου.
- Αντιστοίχιση βάσει κανόνα (rule-based matching): βρίσκει ακολουθίες από λέξεις βασιζόμενη σε κείμενα και γλωσσολογικές σημειώσεις, παρόμοια με τη λειτουργία των κανονικών εκφράσεων.
- Εκπαίδευση (training): ενημερώνει και βελτιώνει τις προβλέψεις στατιστικών μοντέλων.
- Σειριοποίηση (serialization): αποθηκεύει αντικείμενα σε αρχεία ή bytes συμβολοσειρών.

Από όλες αυτές τις δυνατότητες για την εφαρμογή της πτυχιακής χρησιμοποιήθηκαν πρώτα ο διαχωρισμός λεξικογραφικών μονάδων και μετά η σήμανση μερών του λόγου.

Διαχωρισμός λεξικογραφικών μονάδων

Κατά τη διάρκεια της ανάλυσης, η βιβλιοθήκη spaCy διαχωρίζει το κείμενο σε μικρές μονάδες, δηλαδή χωρίζει κομμάτια κειμένου σε λέξεις, σημεία στίξης κτλ. Αυτό πραγματοποιείται με την χρήση κανόνων που βασίζονται στη κάθε γλώσσα. Σημαντικό είναι να σημειωθεί ότι για την βιβλιοθήκη spaCy οι συντομογραφίες αποτελούν μια λεξικογραφική μονάδα, σε αντίθεση με την προσαρμοσμένη υλοποίηση κατά την οποία κάθε λέξη της συντομογραφίες προσμετράτε.

Η εγκατάσταση της βιβλιοθήκης spaCy δε περιέχει στατιστικά μοντέλα. Για την αναγνώριση των μερών του λόγου όμως είναι αναγκαία, καθώς περιέχουν όλη τη γνώση που έχει συλλεχθεί για μια συγκεκριμένη γλώσσα από μια σειρά πηγών. Για την προσθήκη τους χρειάζεται να γίνει ξεχωριστή εγκατάσταση.

Αρχικά ολόκληρο το κείμενο διαχωρίζεται με βάση τα κενά διαστήματα που περιέχει. Στη συνέχεια γίνεται επεξεργασία του κειμένου από τα αριστερά στα δεξιά. Σε κάθε τμήμα συμβολοσειρών πραγματοποιεί δύο ελέγχους:

1. Ελέγχει εάν η συμβολοσειρά αντιστοιχεί σε κάποιο κανόνα εξαίρεσης.
2. Υπάρχει η δυνατότητα διάσπασης της συμβολοσειράς περισσότερο; Ειδικά για τις περιπτώσεις που υπάρχουν κόμματα (,), τελείες (.), παύλες (-), απόστροφοι (') και εισαγωγικά («,»).

Εάν βρεθεί αντιστοίχηση, τότε εφαρμόζεται ο κανόνας και η διαδικασία συνεχίζεται για το νέα διαχωρισμένη συμβολοσειρά. Με αυτό το τρόπο η βιβλιοθήκη spaCy καταφέρνει τον διαχωρισμό πολύπλοκων λεξικογραφικών μονάδων όπως τα πολλαπλά σημεία στίξης.

Σήμανση μερών του λόγου

Μετά το διαχωρισμό των λεξικογραφικών μονάδων, η βιβλιοθήκη spaCy έχει τη δυνατότητα πια να επεξεργαστεί και να αναθέσει ετικέτες σε ένα κείμενο. Σε αυτή τη φάση γίνεται η χρήση των εκπαιδευμένων μοντέλων, γιατί δίνουν τη δυνατότητα πρόβλεψης της ετικέτας που αντιστοιχεί στο συγκεκριμένο πλαίσιο.

Στατιστικά μοντέλα

Η εγκατάσταση της βιβλιοθήκης spaCy δε περιέχει στατιστικά μοντέλα. Για την αναγνώριση των μερών του λόγου όμως είναι αναγκαία, καθώς περιέχουν όλη τη γνώση που έχει συλλεχθεί για μια συγκεκριμένη γλώσσα από μια σειρά πηγών. Για την προσθήκη τους χρειάζεται να γίνει ξεχωριστή εγκατάσταση.

Ανάλυση ελληνικού κειμένου με τη βιβλιοθήκη spaCy

Υπάρχουν πολλά προ-εκπαιδευμένα στατιστικά μοντέλα που είναι διαθέσιμα για διαφορετικές γλώσσες στη σελίδα της βιβλιοθήκης spaCy.

Για την ανάλυση ελληνικού κειμένου συγκεκριμένα, τα διαθέσιμα στατιστικά μοντέλα είναι τρίς και βασίζονται σε ειδησεολογικές πηγές [17]:

- el_core_new_sm: η πιο ελαφριά έκδοση του μοντέλου, μόλις 12MB, με 94% ακρίβεια εντοπισμού των μερών του λόγου.
- el_core_new_md: η μεσαία έκδοση του μοντέλου, με μέγεθος 42MB και 96% ακρίβεια εντοπισμού των μερών του λόγου.
- el_core_new_lg: η πιο ολοκληρωμένη έκδοση του μοντέλου, με μέγεθος 554MB και 97% ακρίβεια στην αντιστοίχιση μερών του λόγου.

Για τη σύγκριση με τους προσαρμοσμένους αλγορίθμους γραμματικής, χρησιμοποιήθηκε η πιο ελαφριά έκδοση του μοντέλου.

Κεφάλαιο 5ο: Αποτελέσματα και σύγκριση

5.1 Μέτρα σύγκρισης αποτελεσμάτων

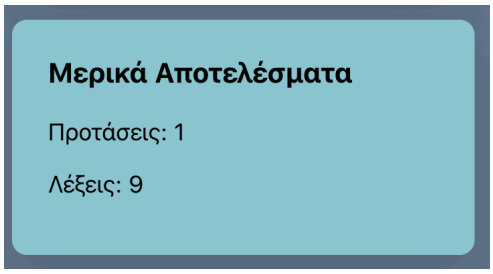
Πριν ξεκινήσουμε με τα αποτελέσματα των δύο στρατηγικών, είναι αναγκαίο να ορίσουμε τα μέτρα σύγκρισης της αποτελεσματικότητας των στρατηγικών.

Το πρώτο μέτρο σύγκρισης αφορά τον ορισμό της ακρίβειας της κάθε στρατηγικής. Ο όρος της ακρίβειας στη προκειμένη περίπτωση συνδέεται με την ορθή αντιστοίχηση των λέξεων του κειμένου με το μέρος του λόγου που ανήκουν στην ελληνική γλώσσα. Η καταμέτρηση της γίνεται μετρώντας τα στοιχεία του κείμενου που αντιστοιχήθηκαν σωστά και συγκρίνοντάς τα με σύνολο των στοιχείων που συμπεριλαμβάνονται σε όλο το κείμενο. Τα στοιχεία μπορεί να είναι λέξεις, σύμβολα ή σημεία στίξης. Η ακρίβεια στη συνέχεια μετατρέπεται σε ποσοστό για την ευκολότερη ανάγνωση της.

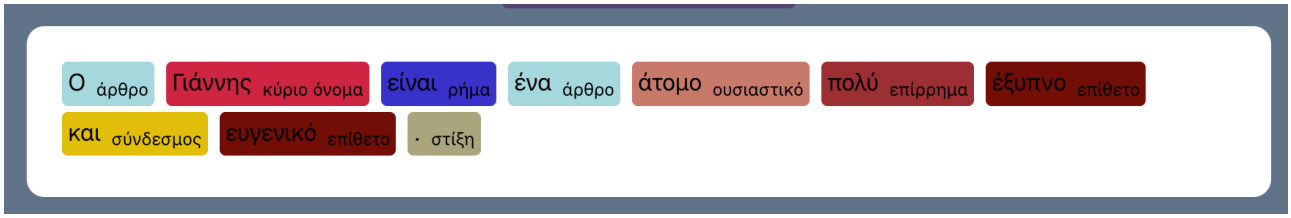
Το δεύτερο μέτρο σύγκρισης των αποτελεσμάτων αφορά τη ταχύτητα με την οποία ολοκληρώνεται η ανάλυση. Συγκεκριμένα αφορά το χρόνο που περνά από τη στιγμή που ο χρήστης κάνει κλικ στη φόρμα για την αποστολή του κειμένου μέχρι τη στιγμή που εμφανίζονται τα αποτελέσματα της ανάλυσης στην οθόνη του.

5.2 Τρόπος εμφάνισης αποτελεσμάτων

Τα αποτελέσματα από κάθε ανάλυση εμφανίζονται σε δύο περιοχές της σελίδας. Στην μία περιοχή εμφανίζονται τα στατιστικά στοιχεία όπως φαίνονται στην εικόνα , δηλαδή ο αριθμός λέξεων και παραγράφων και στην άλλη εμφανίζεται το κείμενο που εισήχθηκε από τον χρήστη με τις ετικέτες που αντιστοιχούν στα μέρη του λόγου. Η κάθε ετικέτα έχει διαφορετικό χρώμα και φέρει το όνομα του μέρους του λόγου στο δεξιά χαμηλότερο σημείο της.



Σχήμα 5.1: Πλαίσιο εμφάνισης στατιστικών αποτελεσμάτων



Σχήμα 5.2: Εμφάνιση αποτελεσμάτων αντιστοίχησης των μερών του λόγου

5.3 Αποτελέσματα ανάλυσης με τους προσαρμοσμένους αλγορίθμους

Η συγκεκριμένη υλοποίηση, όσον αφορά το χρόνο εκτέλεσης, είναι αρκετά αποδοτική και επιστρέφει γρήγορα τα αποτελέσματα της ανάλυσης, μέσα σε χρόνο λιγότερο από ένα δευτερόλεπτο, καθώς η διαδικασία εύρεσής τους είναι απλή και η αλγοριθμική πολυπλοκότητα που εφαρμόζεται είναι γραμμική. Τα αποτελέσματα της καταμέτρησης των προτάσεων και λέξεων είναι σωστά και στις περιπτώσεις που οι προτάσεις συμπεριλαμβάνουν συντομογραφίες, τότε καταμετρούνται και αυτές ως ξεχωριστές λέξεις.

Όσον αφορά την αποσαφήνιση των μερών του λόγου, για απλές προτάσεις φαίνεται να είναι ικανοποιητική εφόσον κατά μέσο όρο πάνω από τις μισές λέξεις του κειμένου αντιστοιχίζονται στο σωστό μέρος του λόγου. Παρόλα αυτά, η μέθοδος δε φαίνεται τόσο αξιόπιστη για πολύπλοκες προτάσεις που περιέχουν πολλά επίθετα στη σειρά ή μεγάλες εκφράσεις.

Σε μεγάλα κείμενα υπάρχουν συχνά λέξεις που δεν ήταν δυνατός ο προσδιορισμός της ετικέτας τους (εμφανίζονται με την ετικέτα “άλλο”). Κατόπιν παρατήρησης, φαίνεται ότι αυτές οι λέξεις είναι ουσιαστικά που προηγούνται από πολλά επίθετα ή πολλά επίθετα στη σειρά. Επιπλέον, κάποια ρήματα συχνά εμφανίζονται με λάθος ετικέτα, καθώς οι χρόνοι και οι καταλήξεις είναι πάρα πολλές.

Εφαρμόζοντας πολλαπλές αναλύσεις σε 5 διαφορετικές προτάσεις, όπως φαίνεται στο πίνακα 5.1, ο μέσος όρος ακρίβειας κυμάνθηκε στο 64%. Υπήρχαν περιπτώσεις που η ακρίβεια έφτανε μέχρι και το 77%, αλλά ταυτόχρονα υπήρχαν προτάσεις που η ακρίβεια δε ξεπερνούσε το 60%.

Η συνεργασία με γλωσσολόγους είναι δυνατό να οδηγήσει στη δημιουργία περισσότερων και αποδοτικότερων κανόνων που θα βοηθήσουν στην ακριβέστερη αποσαφήνιση. Ειδικότερα η αύξηση των λεξικών σίγουρα θα λύσει κάποια από τα προβλήματα που προκύπτουν στα μεγάλα κείμενα. Για τις μεγάλες φράσεις, η αύξηση των φάσεων σε συνδυασμό με την αύξηση των λεξικών μπορούν να αυξήσουν την αποτελεσματικότητα αυτής της στρατηγικής.

Σχήμα 5.3: Αποτελέσματα ανάλυσης με τη χρήση των προσαρμοσμένων αλγορίθμων

Πίνακας 5.1: Κείμενα με την ακρίβεια αποτελεσμάτων κάθε στρατηγικής

Κείμενο	Ακρίβεια Βιβλιοθήκης spaCy	Ακρίβεια προσαρμοσμένης υλοποίησης
1	96%	64%
2	83%	57%
3	92%	63%
4	88%	77%
5	87%	60%

5.4 Αποτελέσματα ανάλυσης με τη βιβλιοθήκη spaCy

Όπως είναι αναμενόμενο η χρήση μιας εξειδικευμένης βιβλιοθήκης που βασίζεται σε εκπαιδευμένα μοντέλα έχει υψηλά ποσοστά αποτελεσματικότητας. Από τις δοκιμές που έγιναν, όλες οι λέξεις αντιστοιχήθηκαν σε κάποιο μέρος του λόγου. Οι μόνες εξαιρέσεις στις οποίες εμφανίστηκε η ετικέτα «άλλο», ήτανε κύρια ονόματα. Σε όλες τις υπόλοιπες περιπτώσεις, οι λέξεις είχαν αντίστοιχο μέρος του λόγου και ελάχιστες φορές η ετικέτα ήταν λανθασμένη. Αυτές οι περιπτώσεις αφορούν προτάσεις που ξεκινούν με ένα ρήμα ή επίρρημα, δηλαδή είναι προτάσεις με μια όχι τόσο συνηθισμένη σειρά των λέξεων.

Παρόλα αυτά, όπως είναι γνωστό η ελληνική γλώσσα είναι μια γλώσσα με μεγάλη ποικιλομορφία, πλούσια σε πτώσεις και χρόνους, και συνδυαστική όσον αφορά την σειρά των λέξεων. Οπότε συχνά κάποιες λέξεις δεν αντιστοιχίζονται στο σωστό μέρος του λόγου. Πράγμα που δείχνει ότι υπάρχει ακόμη χώρος για έρευνα και πειραματισμούς σε αυτόν τον τομέα για την ελληνική γλώσσα.

Συγκεκριμένα, με τα κείμενα που χρησιμοποιήθηκαν για τον έλεγχο των αποτελεσμάτων, ο μέσος όρος ακρίβειας της αντιστοίχησης κυμαίνεται στο 84%, το οποίο είναι 10 μονάδες λιγότερο από την ακρίβεια που αναγράφεται στην σελίδα της βιβλιοθήκης spaCy [17]. Αυτή η διαφορά μπορεί να αποδοθεί στο γεγονός ότι οι προτάσεις που χρησιμοποιήθηκαν για τις δοκιμές συμπεριλάμβαναν πολλές ξένες λέξεις και κύρια ονόματα, τα οποία δεν αναγνωρίζει η βιβλιοθήκη spaCy με το στατιστικό μοντέλο που χρησιμοποιήθηκε για την ανάλυση.

Σχήμα 5.4: Αποτελέσματα ανάλυσης με τη χρήση της βιβλιοθήκης spaCy

5.5 Σύγκριση αποτελεσμάτων

Συγκρίνοντας τα αποτελέσματα από τις δύο αυτές προσεγγίσεις, το πρώτο που είναι εμφανές είναι ότι η ανάλυση των μερών του λόγου με την βιβλιοθήκη spaCy έχει μεγαλύτερη ακρίβεια από την ανάλυση με τους προσαρμοσμένους αλγορίθμους, με διαφορά που φτάνει το 32% σε κάποιες περιπτώσεις.

Θεωρώ ότι με μερικές προσθήκες και καλύτερη μελέτη από ειδικευμένους θα μπορούσε μια προσέγγιση βάσει κανόνων να φτάσει την ακρίβεια της βιβλιοθήκης spaCy. Έχοντας αναπτύξει το κομμάτι κώδικα που κάνει την αποσαφήνιση, πιστεύω ότι μια τέτοια λύση δε θα είναι διατηρήσιμη καθώς ο κώδικας περιέχει πολλούς ελέγχους που κάνουν το κώδικα δύσκολο στην ανάγνωση, αλλά στη προκειμένη περίπτωση είναι δύσκολο να αποφευχθούν.

Τα κοινά λάθη και στις δύο προσεγγίσεις αφορούσαν περιπτώσεις όπως, λέξεις οι οποίες συνοδεύονται από ένα άρθρο και δεν ήταν επίθετα. Σε αυτές τις περιπτώσεις και οι δύο προσεγγίσεις έδιναν λανθασμένα την ετικέτα του επιθέτου, ενώ η λέξη ήταν επίρρημα ή ρήμα. Επίσης, λανθασμένες ετικέτες προέκυπταν συχνά σε ελλείπεις προτάσεις ή προτάσεις όπου η σειρά των λέξεων ήταν λίγο πιο διαφορετική από τη καθιερωμένη σειρά υποκείμενο-ρήμα-αντικείμενο.

Η ανάλυση με τους προσαρμοσμένους αλγορίθμους, αν και αναγνωρίζει αποτελεσματικότερα τα κύρια ονόματα, πολλές φορές δε καταφέρνει να αντιστοιχίσει ουσιαστικά και επίθετα που είναι στη σειρά. Επίσης υπάρχει μια δυσκολία στο προσδιορισμό των ρημάτων και αυτό προκύπτει από το μικρό λεξικό καταλήξεων που χρησιμοποιείται στην ανάλυση.

Ο χρόνος εκτέλεσης της ανάλυσης είναι ένα άλλο στοιχείο που έχει διαφορά ανάμεσα στις δύο υλοποιήσεις. Θα προσέξει κανείς ότι η ανάλυση με τη βιβλιοθήκη spaCy συνήθως χρειάζεται περισσότερο χρόνο για την επιστροφή των αποτελεσμάτων από την ανάλυση με τους προσαρμοσμένους αλγορίθμους. Προφανώς οι απλοί υπό όρους έλεγχοι είναι πιο γρήγοροι για έναν απλό υπολογιστή από την εκτέλεση ολόκληρων μοντέλων μηχανικής μάθησης. Οπότε θα μπορούσε να πει κανείς ότι η αποτελεσματικότητα στη προκειμένη περίπτωση, και ακόμη και γενικά στη στοχαστική μέθοδο, έρχεται με ένα μικρό κόστος σε χρόνο.

Από τη σκοπιά ενός προγραμματιστή που θέλει να δημιουργήσει μια εφαρμογή που παρέχει την υπηρεσία ανάλυσης ελληνικών κειμένων, προτείνεται με μεγαλύτερη σιγουριά η υλοποίηση με τη βιβλιοθήκη spaCy, καθώς προσφέρει ποιοτικά αποτελέσματα και με την ίδια βιβλιοθήκη μπορεί κανείς πολύ εύκολα να επεκτείνει την εφαρμογή και για άλλες γλώσσες.

Κεφάλαιο 6ο: Συμπεράσματα και προτάσεις βελτίωσης

Αυτή η εφαρμογή πραγματοποιήθηκε στα πλαίσια της πτυχιακής με περιορισμένους πόρους όσον αφορά τον γλωσσολογικό τομέα και τις γνώσεις που είχα για την επεξεργασία φυσικής γλώσσας.

Με την εκπόνηση της αναπτύχθηκε και δημοσιεύτηκε η εφαρμογή για τη ανάλυση ελληνικού κειμένου. Η ανάλυση που πραγματοποιήθηκε αφορά τον αριθμό των λέξεων, τον αριθμό των παραγράφων και την αντιστοίχηση των λέξεων του κειμένου στα μέρη του λόγου της ελληνικής γλώσσας.

Για την ανάλυση του κειμένου αναπτύχθηκαν δύο διαφορετικοί μέθοδοι: μία με τη βιβλιοθήκη spaCy και μία με τη συγγραφή προσαρμοσμένων αλγορίθμων που βασίζονται στην ελληνική γραμματική. Οι δύο αυτοί μέθοδοι αντιστοιχούν στην εύρεση μερών του λόγου ακολουθώντας τη στοχαστική μέθοδο και την ανάλυση κατά βάσει κανόνων.

Είναι φανερό ότι μια εφαρμογή με τη χρήση προσαρμοσμένων αλγορίθμων έχει αξία, ειδικά στον ερευνητικό τομέα γιατί μεταφράζοντας την φυσική γλώσσα στη γλώσσα του υπολογιστή πραγματικά αρχίζει κανείς να μελετά τη γλώσσα και ακόμη περισσότερο συνειδητοποιεί ότι οι κανόνες πολλές φορές είναι λιγότεροι από τις εξαιρέσεις. Ως προγραμματίστρια βρήκα τη διαδικασία της αντιστοίχησης αρκετά διασκεδαστική και ενδιαφέρουσα, αλλά και ιδιαίτερα εκπαιδευτική.

Γνωρίζοντας πλέον περισσότερα για τον τομέα και τις δυνατότητες που υπάρχουν ως προς την επεξεργασία φυσικής γλώσσας, μαζί με την αναγνώριση μερών του λόγου, μπορούν να προστεθούν και άλλες δυνατότητες, όπως ο ορθογραφικός έλεγχος, η λημματοποίηση, η ανάλυση εξαρτήσεων.

Επίσης, μπορούν να προστεθούν και άλλα στατιστικά στοιχεία στο κομμάτι των μερικών αποτελεσμάτων, όπως η συχνότητα εμφάνισης των λέξεων και η αναγνώριση του ύφους και συναισθημάτων ενός κειμένου.

Όσον αφορά την υλοποίηση της αναγνώρισης των μερών του λόγου με τους προσαρμοσμένους αλγορίθμους, επαναλαμβάνω ότι μια τέτοια λύση μπορεί να επεκταθεί σε συνεργασία με ειδικούς γλωσσολόγους και να έχει ως στόχο την επίτευξη μεγαλύτερης αποτελεσματικότητας.

Για το κομμάτι της ανάλυσης με τη βιβλιοθήκη spaCy, πέρα από την περαιτέρω εκμετάλλευση των λειτουργιών που διαθέτει και την εμφάνισή τους στην εφαρμογή, αρκετά ενδιαφέρουσα θα είναι η σύγκριση μεταξύ αναλύσεων που προκύπτουν από διαφορετικά μοντέλα. Στην πτυχιακή αυτή έγινε χρήση του πιο απλού μοντέλου που συνοδεύεται με τη βιβλιοθήκη spaCy. Υπάρχουν και άλλα μοντέλα που μπορούν να χρησιμοποιηθούν με τη βιβλιοθήκη, αλλά αυτά είναι πέρα από το εύρος μιας πτυχιακής εργασίας.

Ως τελευταία πρόταση βελτίωσης, θα είχε ενδιαφέρον η σύγκριση της ανάλυσης αυτών των δύο στρατηγικών με μια τρίτη, όπως η ΕΦΓ μέσω των εφαρμογών της Google (Cloud Nature Language) [18].

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Κ. Γεωργούλη, *Τεχνητή Νοημοσύνη: Μια εισαγωγική προσέγγιση*. Σύνδεσμος Ελληνικων Ακαδημαϊκων Βιβλιοθηκών, 2015.

[2] G. S. Ingersoll, T. S. Morton, and A. L. Farris, *Taming Text*. 2013.

[3] S. Vajjala, B. Majumder, A. Gupta, and H. Surana", *Practical Natural Language Processing: A Comprehensive Guide to Building Real-World NLP Systems*. 2020.

[4] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 2020.

[5] Μ. Τριανταφυλλίδης, *Νεοελληνική Γραμματική*. Οργανισμός Εκδόσεως Διδακτικών Βιβλίων.

[6] Y. Vasiliev, *Natural Language Processing with Python and spaCy*. 2020.

[7] W3Schools, "How to create a custom checkbox and radio buttons." `https://www.w3schools.com/howto/howto_css_custom_checkbox.asp`.

[8] P. Ltd, "Pure css loaders." `https://loading.io/css/`.

[9] Coolors, "The super fast color schemes generator!." `https://coolors.co`.

[10] I. Kutskir, "Photopea." `https://www.photopea.com`.

[11] ConvertICO, "Png to ico conversion." `https://convertico.com/`.

[12] MDN, "Fetch api." `https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API`.

[13] MDN, "Using fetch." `https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch`.

[14] B. A. Syed, "Typescript docs: tsconfig.json." `https://basarat.gitbook.io/typescript/project/compilation-context/tsconfig#compileroptions`.

[15] GitHub, "About github pages." `https://docs.github.com/en/pages/getting-started-with-github-pages/about-github-pages`.

[16] SpacyDocs, "spacy 101: Everything you need to know." `https://spacy.io/usage/spacy-101`.

[17] SpacyDocs, "Available trained pipelines for greek." `https://spacy.io/models/el`.

[18] GoogleCloud, "Cloud natural language." `https://cloud.google.com/natural-language`.

ΠΑΡΑΡΤΗΜΑ Α: ΒΑΣΙΚΟΣ ΚΩΔΙΚΑΣ BACKEND

```
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware
from pydantic import BaseModel
import spacy
import re

nlp = spacy.load("el_core_news_sm")
class AnalysisRequestBody(BaseModel):
    text: str

app = FastAPI()

origins = [
    "https://sortingbubbles.github.io",
    "http://localhost:3000",
]

app.add_middleware(
    CORSMiddleware,
    allow_origins=origins,
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

@app.post("/analyze-spacy")
async def analyze_text(request: AnalysisRequestBody):
    doc = nlp(request.text)
    print(doc)
    for token in doc:
        print(token.text, token.pos_, token.dep_)
    words = [token.text for token in doc if token.is_punct != True]
    texts = [token.text for token in doc]
    res = doc.to_json()
    res['words'] = len(words)
    res['texts'] = texts
    return res

@app.post("/analyze-custom")
async def analyze_text_custom(request: AnalysisRequestBody):
    res = {
```

```
        'words': '-',
        'sents': '-',
        'paragraphs': '-'
    }
    text = re.sub(' +', ' ', request.text)
    res['sents'] = re.split('\\. |\\.\\.\\. |; | ! ', text)
    res['tokens'] = tokenize_sentences(text.strip().split(' '))
    res['texts'] = split_texts(res['tokens'])
    res['words'] = find_num_of_words(res['tokens'])
    return res

def split_texts(tokens):
    texts = []
    for token in tokens:
        texts.append(token['lemma'])
    return texts

def find_num_of_words(tokens):
    counter = 0
    for token in tokens:
        if token['pos'] == 'ABBR':
            counter = counter + token['numOfWords']
        if token['pos'] != 'PUNCT' and token['pos'] != 'ABBR':
            counter = counter + 1
    return counter


def tokenize_sentences(words):
    simeia_stiksis_telika_mona = [';', '!', ',', '-', ':', ')', '»', '·', '.']
    simeia_stiksis_arxi_leksis = ['(', '«']
    tokens = []
    for word in words:
        tok = False
        symbol = False
        if word.endswith(('...')):
            symbol = punct(word[-3:], True)
            tok = preliminary_check(word[0:-3])
        elif word.endswith(('..')):
            symbol = punct(word[-1], True)
            tok = abbreviation(word[:-1], findOccurrences(word[:-1], '.'))
        elif findOccurrences(word, '.') > 1:
            tok = abbreviation(word, findOccurrences(word, '.'))
        elif word.endswith(tuple(simeia_stiksis_telika_mona)):
```

```
            tok = preliminary_check(word[0:-1])
            symbol = punct(word[-1], True)
        if word.startswith(('--')):
            symbol = punct(word[0:2], False)
            tok = preliminary_check(word[2:])
        elif word.startswith(tuple(simeia_stiksis_arxi_leksis)):
            symbol = punct(word[0], False)
            tok = preliminary_check(word[1:])

        if symbol and symbol['isAtEnd'] == False:
            tokens.append(symbol)

        if tok == False:
            tokens.append(preliminary_check(word))
        else:
            tokens.append(tok)

        if symbol and symbol['isAtEnd'] == True:
            tokens.append(symbol)
    for (index, token) in enumerate(tokens):
        if token['pos'] == False:
            previousToken = False
            nextToken = False
            if (index > 0):
                previousToken = tokens[index-1]
            if (index < len(tokens) - 1):
                nextToken = tokens[index + 1]
            tokens[index] = secondary_check(
                    token['word'],
                    previousToken,
                    nextToken
            )
    for (index, token) in enumerate(tokens):
        if token['pos'] == False:
            previousToken = False
            nextToken = False
            if (index > 0):
                previousToken = tokens[index-1]
            if (index < len(tokens) - 1):
                nextToken = tokens[index + 1]
            tokens[index] = last_check(
                token['word'],
                previousToken,
```

```
                nextToken
            )
    return tokens


def punct(symbol, einai_teliko):
    punct = {
        'tag': 'PUNCT',
        'pos': 'PUNCT',
        'lemma': symbol,
        'dep': 'punct',
        'isAtEnd': einai_teliko
    }
    return punct

def last_check(word, previousToken, nextToken):
    print('ast', word)
    res = secondary_check(word, previousToken, nextToken)
    if res['pos'] == False:
        return {
            'tag': 'OTHER',
            'pos': 'OTHER',
            'lemma': word,
            'dep': 'other'
        }
    else:
        return res


def secondary_check(word, previousToken, nextToken):
    if previousToken and previousToken['pos'] == 'ADV':
        return {
            'tag': 'ADJ',
            'pos': 'ADJ',
            'lemma': word,
            'dep': 'adj'
        }
    if word[0].isupper():
        if (
                previousToken
                and
                (
                        previousToken['pos'] != 'PUNCT'
```

```
                    or
                    previousToken['pos'] == 'DET'
                )
        ):
            return {
                'tag': 'PROPN',
                'pos': 'PROPN',
                'lemma': word,
                'dep': 'propn'
            }
    if previousToken and previousToken['pos'] == 'DET':
        if (
                nextToken
                and
                (nextToken['pos'] == False or nextToken['pos'] == 'NOUN')
        ):
            return {
                'tag': 'ADJ',
                'pos': 'ADJ',
                'lemma': word,
                'dep': 'adj'
            }
        if (
                nextToken
                and
                (nextToken['pos'] != False or nextToken['pos'] != 'ADJ')
        ):
            return {
                'tag': 'NOUN',
                'pos': 'NOUN',
                'lemma': word,
                'dep': 'noun'
            }
    if previousToken and previousToken['pos'] == 'ADP':
        if nextToken and (nextToken['pos'] == False):
            return {
                'tag': 'ADJ',
                'pos': 'ADJ',
                'lemma': word,
                'dep': 'adj'
            }
        return {
            'tag': 'NOUN',
```

```
            'pos': 'NOUN',
            'lemma': word,
            'dep': 'noun'
        }
    if previousToken and previousToken['pos'] == 'VERB':
        if nextToken and (nextToken['pos'] == False):
            return {
                'tag': 'ADJ',
                'pos': 'ADJ',
                'lemma': word,
                'dep': 'adj'
            }
        return {
            'tag': 'NOUN',
            'pos': 'NOUN',
            'lemma': word,
            'dep': 'noun'
        }
    if previousToken and previousToken['pos'] == 'NUM':
        return {
            'tag': 'NOUN',
            'pos': 'NOUN',
            'lemma': word,
            'dep': 'noun'
        }
    if previousToken and previousToken['pos'] == 'ADJ':
        return {
            'tag': 'NOUN',
            'pos': 'NOUN',
            'lemma': word,
            'dep': 'noun'
        }
    return {
        'tag': 'OTHER',
        'pos': 'OTHER',
        'lemma': word,
        'dep': 'other'
    }

def preliminary_check(word):
    arthra = ο['', η'', το'', του'', της'', του'', το'', τον'', τη'', την'', το'', οι'
    antonimies = εγώ['', εσύ'', εμένα'', μου'', εσένα'', σου'21', εσύ'', εμείς'', εσείς'
    epirimata = πού['', γύρω'', πιο'', πότε'', ήδη'', κάπου'', σήμερα'', αύριο'', χθες'
```

```
protheseis = από['', εξαιτίας'', προς'', ως'', για'', με'', σε'', κατά'', μετά'', απ
syndesmoi = και['', ή'', μα'', ώστε'', δηλαδή'', ότι'', όταν'', αν'', γιατί'', παρά'
if word.lower() in arthra:
    return {
        'tag': 'DET',
        'pos': 'DET',
        'lemma': word,
        'dep': 'det'
    }
if word.lower() in syndesmoi:
    return {
    'tag': 'CONJ',
    'pos': 'CONJ',
    'lemma': word,
    'dep': 'conj'
}
if word.lower() in protheseis:
    return {
        'tag': 'ADP',
        'pos': 'ADP',
        'lemma': word,
        'dep': 'adp'
}
if word.lower() in antonimies:
    return {
        'tag': 'PRON',
        'pos': 'PRON',
        'lemma': word,
        'dep': 'pron'
    }
if word.endswithίως(('', ώς'', όν'', ως'', όλου'')) or word in epirimata:
    return {
        'tag': 'ADV',
        'pos': 'ADV',
        'lemma': word,
        'dep': 'adv'
    }
if word.endswithικός(('', ικό'', ική'')):
    return {
        'tag': 'ADJ',
        'pos': 'ADJ',
        'lemma': word,
        'dep': 'adj'
```

```
        }
    if word.startswith(('0', '1', '2', '3', '4', '5', '6','7','8', '9')):
        return {
            'tag': 'NUM',
            'pos': 'NUM',
            'lemma': word,
            'dep': 'num'
        }
    if word.endswithώνω(('', εί'', ξε'', ουμε'', ήκει'', είτε'', ώνει'', αίνει'', αι'', n
        return {
            'tag': 'VERB',
            'pos': 'VERB',
            'lemma': word,
            'dep': 'verb'
        }
    if word.endswithένος(('', ένη'', ένο'', ώντας'', όντας'')):
        return {
            'tag': 'PARTCP',
            'pos': 'PARTCP',
            'lemma': word,
            'dep': 'prtcp'
        }
    return {
        'pos': False,
        'word': word
    }


def findOccurrences(s, ch):
    return len([i for i, letter in enumerate(s) if letter == ch])


def abbreviation(word, numOfAbbreviatedWords):
    return {
        'tag': 'ABBR',
        'numOfWords': numOfAbbreviatedWords,
        'pos': 'ABBR',
        'lemma': word,
        'dep': 'abbr'
    }
```

ΠΑΡΑΡΤΗΜΑ Β: ΒΑΣΙΚΟΣ ΚΩΔΙΚΑΣ FRONTEND

```
import React from 'react'
import { analyzeWithCustomAlgorithms, analyzeWithSpacy } from './api.ts'


const partsOfSpeech = {
  ADJ: επίθετο'',
  ADP: πρόθεση'',
  ADV: επίρρημα'',
  AUX: βοηθητικό' ρήμα',
  CONJ: σύνδεσμος'',
  CCONJ: σύνδεσμος'',
  PARTCP: μετοχή'',
  ABBR: συντομογραφία'',
  DET: άρθρο'',
  INTJ: επιφώνημα'',
  NOUN: ουσιαστικό'',
  NUM: αριθμός'',
  PART: σωματίδιο'',
  PRON: αντωνυμία'',
  PROPN: κύριο' όνομα',
  PUNCT: στίξη'',
  SCONJ: δευτερεύουσα' σύνδεση',
  SYM: σύμβολο'',
  OTHER: άλλο'',
  VERB: ρήμα'',
  X: άλλο'',
}

class WriteOMeterForm extends React.Component {
  constructor(props) {
    super(props)
    this.state = {
      value: Γράψτε' εδώτοκείμενοπροςανάλυση     .',
      method: 'spacy-lib',
      results: null,
      isLoading: false
    }
    this.handleChange = this.handleChange.bind(this)
    this.handleAnalysisMethodChange = this.handleAnalysisMethodChange.bind(this)
    this.handleSubmit = this.handleSubmit.bind(this)
    this.getClassByPartOfSpeech = this.getClassByPartOfSpeech.bind(this)
```

```
  this.showResults = this.showResults.bind(this)
  this.showLoadingState = this.showLoadingState.bind(this)
}

handleChange(event) {
  this.setState({ value: event.target.value })
}

handleAnalysisMethodChange(event) {
  this.setState({ method: event.target.value })
}

getClassByPartOfSpeech(pos) {
  return `wom-text__${pos.toLowerCase()}`
}

showStatisticResults() {
  if (this.state.results) {
    return (
      <section className="wom-form__controls__method-options">
        <h3>Μερικά Αποτελέσματα</h3>
        <p>Προτάσεις: {this.state.results.sents.length}</p>
        <p>Λέξεις: {this.state.results.words}</p>
      </section>
    )
  }
}
showResults() {
  if (this.state.results && this.state.results.tokens) {
    return (
      <p class="text-results">
          {this.state.results.tokens.map((token, index) => {
            const classForToken = this.getClassByPartOfSpeech(token.pos)
            return (
              <span
                key={token.start}
                  className={`wom-text ${classForToken}`}
              >
                {this.state.results.texts[index]}
                <sub> {partsOfSpeech[token.pos]}</sub>
              </span>
            )
          })}
```

```
        </p>
      )
    }
  }

  showLoadingState() {
    return (
      <div class="lds-ellipsis">
        <div></div>
        <div></div>
        <div></div>
        <div></div>
      </div>
    )
  }

  async handleSubmit(event) {
    event.preventDefault()
    try {
      let response
      this.setState({isLoading: true})
      if (this.state.method === 'spacy-lib') {
        response = await analyzeWithSpacy(this.state.value)
      } else {
        response = await analyzeWithCustomAlgorithms(this.state.value)
      }
      const results = await response.json()
      this.setState({ results })
    } catch (e) {
      console.error(e)
    } finally {
      this.setState({ isLoading: false })
    }
  }

  render() {
    const hasResults = !!this.state.results
    const isLoading = this.state.isLoading
    return (
      <form className="wom-form" onSubmit={this.handleSubmit}>
        <section className="wom-form__controls">
          <label className="wom-form__text-area">
            <textarea
```

```
            value={this.state.value}
            onChange={this.handleChange}
        />
    </label>
    <section>
      <section className="wom-form__controls__method-options">
        <h3>Ανάλυση με</h3>
        <p>
          <label>τηβιβλιοθήκη
              SpaCy
            <input
              type="radio"
              value="spacy-lib"
              checked={this.state.method === 'spacy-lib'}
              onChange={this.handleAnalysisMethodChange}
            />
            <span className="wom-form__controls__method-options__radio">
            </span>
          </label>
        </p>
        <p>
          <label>τουςαλγορίθμουςγραμματικής

            <input
              type="radio"
              value="grammatical-algorithms"
              checked={this.state.method === 'grammatical-algorithms'}
              onChange={this.handleAnalysisMethodChange}
            />
            <span className="wom-form__controls__method-options__radio"
            </span>
          </label>
        </p>
      </section>
      {hasResults && this.showStatisticResults()}
    </section>
</section>
<input
      disabled={this.state.isLoading}
      className="wom-form__input"
      type="submit"
      value="νάλυσηΑ"
/>
```

```
          {isLoading && this.showLoadingState()}
          {hasResults && this.showResults()}
        </form>
      )
    }
}
export default WriteOMeterForm
```

ΠΑΡΑΡΤΗΜΑ Γ: ΚΕΙΜΕΝΑ ΣΥΓΚΡΙΣΗΣ

Σε αυτό το παράρτημα παρατίθενται όλες οι προτάσεις που χρησιμοποιήθηκαν για την παρουσίαση και σύγκριση των αποτελεσμάτων.

Κείμενο 1

Ανατροπές στις πανελλαδικές εξετάσεις, αναμένεται να φέρει η εφαρμογή του νέου συστήματος που δεσμεύει θέσεις σε δημόσια ΙΕΚ, καθώς και η επαναφορά της βάσης εισαγωγής.

Κείμενο 2

Ο όρος ουσιαστικά σημαίνει «πολυτελής διαμονή στη φύση», δηλαδή σε σκηνή με πολυτέλεια, αλλά για να το καταλάβετε καλύτερα φανταστείτε αυτό: να πηγαίνετε για πεζοπορία, ποδηλασία, καγιάκ, ιππασία, ψάρεμα ή ό,τι άλλο θέλει η καρδιά σας και στη συνέχεια να επιστρέφετε σε ένα «σπίτι» με ένα άνετο κρεβάτι, γκουρμέ γεύματα και έτοιμη φωτιά στη μέση για να καθίσετε περιμετρικά της.

Κείμενο 3

Η Νάξος, το μεγαλύτερο νησί των Κυκλάδων δεν θέλει ιδιαίτερες συστάσεις. Με πλούσια ιστορία, ενετικά κάστρα και καστρόσπιτα, χωριά που σφύζουν από ζωή ένα ιδιαίτερο ανάγλυφο, κάμπους και ορεινούς όγκους, αρχαία μνημεία και φυσικά υπέροχες παραλίες η Νάξος είναι ένα νησί που απευθύνεται σε όλες τις «φυλές» των τουριστών.

Κείμενο 4

Θετικά προσκείμενες στην υιοθέτηση του ελάχιστου φόρου 15% στις πολυεθνικές επιχειρήσεις εμφανίζονται οι Facebook και Google μέσα από τις επίσημες τοποθετήσεις τους, που διένειμαν στα διεθνή media. Ειδικότερα, οι δύο κολοσσοί του κλάδου της τεχνολογίας σημειώνουν ότι «υποστηρίζουν το έργο που γίνεται για την ενημέρωση των διεθνών φορολογικών κανόνων».

Κείμενο 5

Η Βικιπαίδεια έχει, ελεύθερου περιεχομένου, αδελφικά εγχειρήματα που εκπληρώνουν μη-εγκυκλοπαιδικούς ρόλους. Αυτά περιλαμβάνουν: το Βικιλεξικό, ένα πρόγραμμα λεξικού ελεύθερης πρόσβασης, τα Βικιβιβλία, ένα πρόγραμμα με βιβλία και εγχειρίδια ελεύθερης πρόσβασης, τη Βικιθήκη, μια βιβλιοθήκη ελεύθερης πρόσβασης, τα Βικιφθέγματα, συλλογή αποφθεγμάτων, και τα Βικινέα, μια πηγή ειδήσεων ελεύθερης πρόσβασης.