ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΘΕΜΑ: Εμπλουτισμός της βιβλιοθήκης spaCy για ανάλυση συναισθήματος σε ελληνικά κείμενα

Εκπόνηση διπλωματικής εργασίας: Γεώργιος Σούρπης (Α.Μ. 141112)

Επιβλέπουσα Καθηγήτρια: Παναγιώτα Τσελέντη

Αθήνα, Μαρτίου 2023

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ/ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Γεώργιος Σούρπης του Ιωάννη, με αριθμό μητρώου 141112, φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής Μηχανικών του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών, δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της πτυχιακής/διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολο τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από εμένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος. Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου»

«Με επιφύλαξη παντός δικαιώματος δηλώνω υπεύθυνα και γνωρίζοντας τις κυρώσεις του Ν. 2121/1993 περί Πνευματικής Ιδιοκτησίας, ότι η παρούσα πτυχιακή εργασία είναι εξ ολοκλήρου αποτέλεσμα δικής μου ερευνητικής εργασίας, δεν αποτελεί προϊόν αντιγραφής ούτε προέρχεται από ανάθεση σε τρίτους. Όλες οι πηγές που χρησιμοποιήθηκαν (κάθε είδους, μορφής και προέλευσης) για τη συγγραφή της περιλαμβάνεται στη βιβλιογραφία».

Ο Δηλών

Σούρπης Γεώργιος

Υπογραφή

Εγκρίθηκε από τριμελή εξεταστική επιτροπή

Αθήνα, Μαρτίου 2023

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

Επιβλέπουσα καθηγήτρια:
Παναγιώτα Τσελέντη

Panagiota Tselenti
Digitally signed by Panagiota Tselenti
Date: 2023.03.07 18:39:10 +02'00'

Μέλος επιτροπής:
Γεώργιος Μπάρδης

Georgios Bardis
Digitally signed by Georgios Bardis
Date: 2023.03.15 14:50:02 +02'00'

Μέλος επιτροπής:
Χρήστος Τρούσσας

Christos Troussas
Digitally signed by Christos Troussas
Date: 2023.03.15 12:22:00 +02'00'

ΠΕΡΙΛΗΨΗ

Όντας μάρτυρες των τρεχουσών εξελίξεων που πραγματοποιούνται με ταχύτατους ρυθμούς και παρακολουθώντας την ανεξέλικτη αύξηση των διαδικτυακών κοινοτήτων στον ψηφιακό κόσμο, καθίσταται επιτακτική η ανάγκη, θωράκισης αυτών με τρόπο αποτελεσματικό και ασφαλή. Συνεπώς, το πρόβλημα που καλούμαστε να μελετήσουμε και να αντιμετωπίσουμε είναι η ανάλυση συναισθήματος σε κείμενα που είναι γραμμένα στην ελληνική γλώσσα , ακριβώς γιατί οι ψηφιακοί ελληνικοί γλωσσικοί πόροι είναι πολύ λιγότεροι από τους αγγλικούς. Ο σκοπός της διπλωματικής εργασίας είναι ο εντοπισμός και ο προσδιορισμός της προσβλητικής γλώσσας (cyberbullying, ρητορική μίσους, επιθετικότητα) σε ελληνικά κείμενα. Πιο συγκεκριμένα, μελετήσαμε την βιβλιοθήκη επεξεργασίας φυσικής γλώσσας, spaCy (που είναι γραμμένη στη γλώσσα Python), ακριβώς λόγω των ελλείψεων της σε ό,τι αφορά την ελληνική γλώσσα, όπως το πεδίο αναγνώρισης ονοματισμένων οντοτήτων (NER – Named Entity Recognition) το όποιο δεν είναι ολοκληρωμένο σε σχέση με εκείνο της αγγλικής γλώσσας. Στη συνέχεια, διερευνήσαμε δύο μοντέλα μηχανικής μάθησης, το CNN και το LSTM (σε γλώσσα Python), χρησιμοποιώντας σαν οδηγό ένα συναισθηματικό λεξιλόγιο. Τέλος, για την εν λόγω μελέτη σχεδιάστηκε και αναπτύχθηκε μια βοηθητική διαδικτυακή εφαρμογή με στόχο την αναγνώριση του συναισθήματος σε ελληνικά κείμενα και τον εμπλουτισμού του πεδίου NER.

Λέξεις-κλειδιά: Συναισθηματική Ανάλυση, NER, spaCy, Python, LSTM, CNN

ABSTRACT

Witnessing the current developments that are taking place at a rapid pace and watching the inexorable growth of online communities in the digital world, the need to shield them in an effective and secure manner becomes imperative. Therefore, the problem we are called to study and deal with is the analysis of sentiment in texts written in the Greek language, because the digital Greek language resources are much less than the English ones. The purpose of the thesis is to identify offensive language (cyberbullying, hate speech, aggression) in Greek texts. More specifically, we studied the natural language processing library, spaCy (which is written in the Python, due to its shortcomings regarding the Greek language, such as the field of Named Entity Recognition (NER – Named Entity Recognition) which it is not completed in relation to that of the English language. Next, we investigated two machine learning models, CNN and LSTM (in Python), using an emotional vocabulary as a guide. Finally, an auxiliary web application was designed and developed with the aim of recognizing emotion in Greek texts and enriching the field of NER.

Keywords: Sentiment Analysis, NER, spaCy, Python, LSTM, CNN

Πίνακας περιεχομένων

ΚΕΦΑΛΑΙΟ 1: Εισαγωγή

Σε ένα τόσο γρήγορα τεχνολογικά εξελισσόμενο κόσμο, η επεξεργασία φυσική γλώσσας κερδίζει όλο και περισσότερο έδαφος. Όσον αφορά την επεξεργασία φυσική γλώσσας στα ελληνικά κείμενα, δύο είναι τα βασικότερα πακέτα σε γλώσσα Python. Το πρώτο και βασικότερο πακέτο είναι το Natural Language Toolkit (NLTK), όπου περιέχει μηχανισμούς σε περισσότερα από 50 σώματα και λεξιλογικούς πόρους, όπως το WordNet, καθώς και μια γκάμα βιβλιοθηκών για την επεξεργασία κειμένου, ταξινόμηση, stemming, ανάλυση και σημασιολογική συλλογιστική. Παρ' όλα αυτά το NLTK, δεν καλύπτει τις ανάγκες της ελληνικής γλώσσας, αφού οι γλωσσικοί πόροι είναι πολύ περιορισμένοι. Το δεύτερο πακέτο είναι το spaCy. Μεταξύ άλλων, υπάρχουν και άλλες βιβλιοθήκες, σε άλλες γλώσσες προγραμματισμού, όπως για παράδειγμα η OpenNLP που πρόκειται για μια εργαλειοθήκη που βασίζεται στη μηχανική μάθηση για εργασίες NLP, ή το MALLET για την στατική επεξεργασία φυσική γλώσσας, ταξινόμηση εγγράφων, ομαδοποίηση κ.α. που είναι βασισμένες σε γλώσσα Java.

Στην παρούσα διπλωματική θα μελετήσουμε κυρίως το Spacy, καθώς είναι η πιο ολοκληρωμένη βιβλιοθήκη σε σχέση με την NLTK. Στη συνέχεια, θα γίνει ένας επιπλέον εμπλουτισμός της βιβλιοθήκης όσο αφορά το πεδίο αναγνώρισης ονομαστικών οντοτήτων (NER), αφού είναι ελλιπές σε σχέση με εκείνο της αγγλικής γλώσσας, δημιουργώντας έτσι ένα νέο εκπαιδευόμενο πακέτο, που χρησιμοποιείται σε όλη την εργασία.

Αμέσως μετά, θα μιλήσουμε για την ανάλυση συναισθήματος (Sentiment analysis) ή εξόρυξη γνώμης (Opinion mining) όπου ονομάζεται το γνωστικό αντικείμενο που αναλύει τις γνώμες, τα συναισθήματα, τις αξιολογήσεις, τις εκτιμήσεις και τις συμπεριφορές των ανθρώπων προς οντότητες όπως: προϊόντα, υπηρεσίες, οργανισμούς, άτομα, ζητήματα, γεγονότα, θέματα και χαρακτηριστικά τους. Γενικά, τρία είναι τα κυρίως επίπεδα στα όποια έχει μελετηθεί η εξόρυξη γνώμης: η ανάλυση σε επίπεδο εγγράφου, η ανάλυση σε επίπεδο πρότασης και τέλος η ανάλυση με βάση την όψη / τα χαρακτηριστικά. Επιπλέον, η εξόρυξη γνώμης μπορεί να πραγματοποιηθεί με τρεις τεχνικές, την τεχνική της μηχανικής μάθησης, την τεχνική βασισμένη σε λεξικό και τέλος την υβριδική τεχνική, η ανάλυση των οποίων θα υλοποιηθεί παρακάτω.

Η τεχνική της μηχανικής μάθησης, περιλαμβάνει την τεχνική της επιβλεπόμενης μάθησης (όταν το εκπαιδευτικό σύνολο περιλαμβάνει μόνο επισημασμένα δεδομένα), την μη επιβλεπόμενη μάθηση (όταν το σύνολο εκπαίδευσης περιλαμβάνει μόνο δεδομένα που είναι μη επισημασμένα), και τέλος το συνδυασμό των δύο προηγούμενων τεχνικών (όταν το εκπαιδευτικό σύνολο περιλαμβάνει επισημασμένα και μη επισημασμένα δεδομένα). Κάποια από τα γνωστά μοντέλα επιβλεπόμενης μάθησης είναι το Naïve Bayes, οι Μηχανές

Διανυσμάτων Υποστήριξης (Support Vector Machines - SVM), και τα Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Network-CNN). Ένα μοντέλο μη επιβλεπόμενης μάθησης είναι το K-means συσταδοποίηση. Το πλεονέκτημα της μηχανικής μάθησης είναι ότι η ανάπτυξη των μοντέλων να μπορεί να εφαρμοστεί σε οποιονδήποτε τομέα και γλώσσα. Όσον αφορά τα μειονεκτήματα της, ενώ έχει αυξημένη ακρίβεια μπορεί να αποδώσει λάθος συναίσθημα στις λέξεις και η χρήση της σε τομείς εκτός του εκπαιδευτικού συνόλου τους μπορεί να μειώσει σημαντικά την απόδοση της.

Αναφορικά με την τεχνική βασισμένη σε λεξικό συναισθήματος, πρόκειται για μια τεχνική που κάνει χρήση ενός συνόλου λέξεων ή φράσεων, όπου για κάθε μια λέξη/φράση έχει ένα προκαθορισμένο συναίσθημα και με την βοήθεια κανόνων καθορίζουν το συναισθηματικό προσανατολισμό του κειμένου. Η δημιουργία του λεξικού γνώμης, μπορεί να γίνει βασισμένη σε λεξικό (dictionary-based) ή σε συλλογή (corpus-based). Αυτές οι προσεγγίσεις αφορούν την επέκταση ενός αρχικού συνόλου με λέξεις που είναι γνωστός ο συναισθηματικός προσανατολισμός τους. Πιο συγκεκριμένα, η λειτουργία της πρώτης προσέγγισης (dictionary-based), ως επί τον πλείστον, αφορά την πραγματοποίηση μιας αναζήτησης σε κάποιο λεξικό ώστε να βρει συνώνυμα και αντώνυμα μιας λέξης και να δώσει μέσω συνώνυμου ή αντώνυμου τον ίδιο ή τον αντίθετο συναισθηματικό προσανατολισμό. Το πλεονέκτημα της τεχνικής αυτής είναι η δημιουργία ενός λεξικού γνώμης το όποιο δεν εξαρτάται από το σύνολο των δεδομένων στο όποιο θα γίνει η εφαρμογή του αλλά έχει την δυνατότητα να χρησιμοποιηθεί από διαφορετικούς τομείς. Η δεύτερη προσέγγιση(corpus-based), επικεντρώνεται στον εντοπισμό του συναισθηματικού προσδιορισμού των λέξεων σε ένα σύνολο ενός τομέα. Συνεπώς, το μόνο κοινό τους μειονέκτημα είναι η αδυναμία συναισθηματικού προσδιορισμού εκείνων των λέξεων που αλλάζουν με τα συμφραζόμενα.

Ωστόσο, στο σημείο αυτό δεν θα μπορούσε να παραληφθεί η αναφορά της 3ης τεχνικής, της λεγόμενης υβριδικής. Ουσιαστικά, η τεχνική αυτή χρησιμοποιεί τον συνδυασμό των ήδη προαναφερόμενων προσεγγίσεων, με απώτερο σκοπό την εξάλειψη των μειονεκτημάτων τους.

Σχετικά με το συναισθηματικό λεξιλόγιο για τα ελληνικά κείμενα, αυτό που είναι το πιο διαδεδομένο αυτή την στιγμή είναι το «greek_sentiment_lexicon» αλλά και πάλι θεωρείται περιορισμένης έκτασης με αποτέλεσμα να μην επιτυγχάνεται η αναγνώριση του συναισθήματος σε όλα τα κείμενα. Παρ' όλα αυτά, δεν μπορεί να αμφισβητηθεί η προσπάθεια εμπλουτισμού που έγινε με περισσότερους όρους ώστε να είναι εφικτό και ικανοποιητικό το αποτέλεσμα αναγνώρισης συναισθήματος σε ελληνικά κείμενα.

Τέλος, έχοντας ως δεδομένο το παραπάνω λεξιλόγιο, προχωρήσαμε στην υλοποίηση ενός νευρωνικού δικτύου με χρήση της γλώσσας Python και με απώτερο στόχο την εκπαίδευση και την αναγνώριση του συναισθήματος σε όλο και περισσότερα ελληνικά κείμενα.

ΚΕΦΑΛΑΙΟ 2: Θεωρητικό Υπόβαθρο

Σε αυτό το κεφάλαιο, θα δούμε την Αναγνώριση και Ταξινόμηση Ονομαστικών Οντοτήτων (Named Entity Recognition and Classification- NERC) αλλά και τον εντοπισμό του συναισθήματος. Επιπλέον, θα κάνουμε μια αναφορά για την μηχανική μάθηση και για τα πιο διαδομένα μοντέλα που χρησιμοποιούνται στο κλάδο της γλωσσικής ανάλυσης.

2.1 Αναγνώριση και Ταξινόμηση Ονομαστικών Οντοτήτων (Named Entity Recognition and Classification- NERC)

Η αναγνώριση ονομαστικών οντοτήτων και η ταξινόμηση τους, αποτελεί ένα βασικό στοιχείο για την μετατροπή του μη δομημένου κειμένου σε δομημένα δεδομένα, στην διαδικασία εξαγωγής πληροφορίας. Η NERC περιλαμβάνει τον προσδιορισμό των κατάλληλων ονομάτων σε κείμενα (NER) και την ταξινόμηση τους σε ένα σύνολο προκαθορισμένων κατηγοριών ενδιαφέροντος (NEC). Σε αντίθεση με τα εργαλεία προ-επεξεργασίας που ασχολούνται με την συντακτική ανάλυση, το NERC αφορά την αυτόματη εξαγωγή σημασιολογίας από το περιεχόμενο ενός κειμένου. Το παραδοσιακό σύνολο ονομαστικών οντοτήτων, το όποιο αναπτύχθηκε για την εργασία NERC στο MUC-6 [1], αποτελείται από Πρόσωπο, Οργανισμό, Τοποθεσία και εκφράσεις ημερομηνίας και ώρας, όπως Ανδρέας Παπανδρέου, Apple, Λονδίνο, 24 Απριλίου, 2020 κ.λπ.

Το NERC είναι γενικά μια εργασία σχολιασμού αλλά μπορεί να περιλαμβάνει απλώς την παραγωγή μιας λίστας από Nes. Αποτελείται από δύο βασικές εργασίες, εκείνη της αναγνώρισης όπου γίνεται προσδιορισμό των ορίων ενός NE (συνήθως αναφέρεται ως NER) και εκείνη της ονομαστικής ταξινόμησης οντοτήτων (NEC) που περιλαμβάνει την ανίχνευση της τάξης ή του τύπου NE[1]. Πολλές φορές, κυρίως σε παλαιότερες εργασίες, θα παρατηρήσει κανείς ότι χρησιμοποιούσαν τον όρο NER για τον συνδυασμό και των δύο παραπάνω εργασιών.

Η βασική πρόκληση για το NEC είναι ότι τα NE μπορεί να είναι αρκετά ασαφή (π.χ. «Ιούλιος» μπορεί να είναι το όνομα κάποιου προσώπου ή ο μήνας του έτους). Για αυτό τον λόγο, οι δύο εργασίες του NER και του NEC συνήθως επιλύονται ως μια ενιαία εργασία. Μια επιπλέον εργασία σχετικά με τις ονομαστικές οντότητες είναι η ονομασία σύνδεσης οντοτήτων (NEL). Η εργασία NEL είναι να αναγνωρίζει εάν μια ονομαστική οντότητα που αναφέρεται σε ένα κείμενο αντιστοιχεί σε οποιαδήποτε NE που βρίσκεται σε μία βάση γνώσεων αναφοράς. Μια έκφραση στο κείμενο που αναφέρεται σε μια ονομαστική οντότητα μπορεί να είναι υπό διαφορετικές μορφές(π.χ. «κ. Παπαδόπουλος» και «Γεώργιος Παπαδόπουλος» είναι και οι δύο

αναφορές της ίδιας οντότητας του πραγματικού κόσμου, που εφαρμόζονται από ελαφρώς διαφορετικές γλωσσικές πραγματοποιήσεις).

Η βάση γνώσεων αναφοράς που χρησιμοποιείται είναι συνήθως η Wikipedia. Το NEL είναι αρκετά πιο δύσκολο από το NEC γιατί οι διακρίσεις δεν πρέπει να γίνονται μόνο σε επίπεδο τάξης, αλλά και εντός τάξεων. Για παράδειγμα, υπάρχουν πολλά άτομα με το όνομα «Γιώργος Παπαδόπουλος». Όσο πιο δημοφιλές είναι το όνομα, τόσο πιο δύσκολο γίνεται και το έργο της NEL. Ένα ακόμα επιπλέον πρόβλημα είναι ότι οι βάσεις γνώσεων είναι ελλιπείς. Για παράδειγμα μπορεί να περιέχουν μονό διάσημους με το όνομα «Γιώργος Παπαδόπουλος». Αυτό το καθιστά ιδιαίτερα δύσκολο σε εργασίες που αφορούν πρόσφατα γεγονότα, καθώς υπάρχει μια χρονική καθυστέρηση μεταξύ των νεοεμφανιζόμενων οντοτήτων που εμφανίζονται στις ειδήσεις ή στα μέσα κοινωνικής δικτύωσης και την ενημέρωση των γνωστικών βάσεων με τις πληροφορίες τους.

2.2 Ονομαστικές Οντότητες

Οι πιο γνωστές ονομαστικές οντότητες είναι το πρόσωπο, ο οργανισμός, η τοποθεσία, η ημερομηνία και η ώρα. Γνωστοποιήθηκαν το 1995 μέσω της εργασίας αναγνώρισης και ταξινόμησης ονομαστικών οντοτήτων η όποια και οδήγησε στην ανάπτυξη πολλών συστημάτων που υπάρχουν ακόμη και σήμερα. Ύστερα λόγω ανάγκης(χρήση των εργαλείων NERC σε πραγματικές εφαρμογές) έγινε επέκταση των άνω ονομαστικών οντοτήτων ανάλογα με το καθήκον. Σταδιακά άρχισαν επίσης να θεωρούνται ως ονομαστικές οντότητες εφημερίδες, χρηματικά ποσά, ακόμα και πιο λεπτομερείς ταξινομήσεις των παραπάνω, όπως συγγραφείς, μουσικά συγκροτήματα, ποδοσφαιρικές ομάδες, τηλεοπτικά προγράμματα και ούτω καθεξής.

Το NERC επίσης αποτελεί σημείο αναφορά για πολλές άλλες σύνθετες εφαρμογές και εργασίες όπως η οικοδόμηση οντολογίας, εξαγωγή σχέσεων, η απάντηση ερωτήσεων, η εξαγωγή πληροφοριών, η ανάκτηση πληροφοριών, η μηχανική μετάφραση και ο σημασιολογικός σχολιασμός. Με την έλευση των ανοιχτών σεναρίων εξαγωγής πληροφοριών που εστιάζουν σε ολόκληρο τον Ιστό και της ανάλυσης των μέσων κοινωνικής δικτύωσης, όπου εμφανίζονται συνεχώς νέες οντότητες και ονομασμένες εργασίες σύνδεσης οντοτήτων, το φάσμα των οντοτήτων που εξήχθησαν διευρύνθηκαν δραματικά, γεγονός που έφερε πολλές νέες προκλήσεις.

2.3 Αξιολόγηση Ονομαστικής Οντότητας

Η πρώτη σημαντική σειρά αξιολόγησης για το NERC ήταν το MUC, το οποίο ασχολήθηκε για πρώτη φορά το 1996 με την ονομαστική οντότητα. Ο στόχος αυτού ήταν να αναγνωρίσει ονομαστικές οντότητες στα ενημερωτικά κείμενα (newswire) και οδήγησε όχι μόνο στην ανάπτυξη του συστήματος αλλά στην πρώτη πραγματική παραγωγή προτύπου NE-

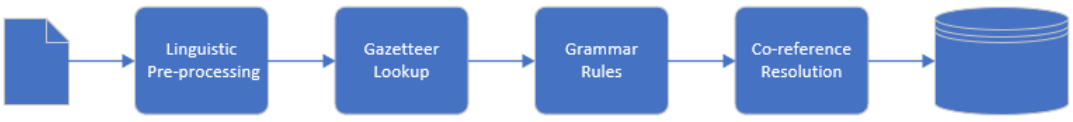
σχολιασμένων σωμάτων για την εκπαίδευση και δοκιμή. Το 2003 ακολουθεί η ConLL[2], μια άλλη σημαντική εταιρεία αξιολόγησης, η οποία παρέχει δεδομένα χρυσού προτύπου για το newswire όχι μόνο στα αγγλικά αλλά επίσης στα ισπανικά, ολλανδικά και γερμανικά. Το σώμα αυτό που δημιουργήθηκε από αυτήν την προσπάθεια αξιολόγησης είναι πλέον ένα από τα πιο δημοφιλή πρότυπα για το NERC.

Άλλες εκστρατείες αξιολόγησης άρχισαν αργότερα να απευθύνονται στο NERC για είδη εκτός από το newswire, όπως το ACE [3] και OntoNotes [4] τα οποία εισήγαγαν νέα είδη ονομαστικών οντοτήτων. Και τα δύο σώματα αποτελούν εμπλουτισμό του newswire και περιλαμβάνουν επίσης εκπομπή ειδήσεων, μετάδοση συνομιλίας, ιστολόγια και τηλεφωνική ομιλία. Το ACE περιέχει επιπλέον ένα υπό-σώμα με ομαδικές συζητήσεις UseNet, και απευθυνόταν όχι μόνο στα αγγλικά αλλά και στα αραβικά και οι μεταγενέστερες εκδόσεις και στα κινέζικα.

2.4 Προσεγγίζοντας το NERC

Οι προσεγγίσεις στο NERC μπορούν να χωριστούν σε δύο βασικές τεχνικές, αυτή βασισμένη σε κανόνες ή μοτίβα και σε εκείνη της μηχανικής μάθησης ή των στατιστικών μεθόδων εξαγωγής [40]. Βέβαια, αρκετά συχνά θα δούμε τις δυο τεχνικές αυτές να αναμειγνύονται. Οι τεχνικές που βασίζονται στην μάθηση βασίζονται σε κάποια μορφή ανθρώπινης επίβλεψης, με εξαίρεση τις καθαρά δομικές τεχνικές IE που εκτελούν μη επιβλεπόμενη μηχανική εκμάθηση με χρήση μη σχολιασμένων εγγράφων [5].

Πολλές πλατφόρμες μηχανικής γλώσσας, όπως GATE, Stanford CoreNLP και NLTK, επιτρέπουν στην υλοποίηση τους τεχνικές και αλγόριθμους για την εξαγωγή πληροφοριών, με την εισαγωγή διαφορετικών προ-επεξεργαστών και με το NERC να εμπλουτίζει τον αγωγό. Έτσι επιτρέπουν ένα επαναλαμβανόμενο μοτίβο πειραματισμού – αξιολόγησης των αποτελεσμάτων τους. Ένα παράδειγμα τυπικής επεξεργασίας για το NERC είναι το παρακάτω:



Εικόνα 2.1:Τυπικό NERC αγωγού.

Natural Language Processing for the Semantic Web

2.4.1 Προσεγγίσεις με κανόνες στο NERC

Συνήθως οι γλωσσικοί μέθοδοι βασισμένες σε κανόνες για το NERC περιλαμβάνουν ένα συνδυασμό κειμένων και κανόνων αντιστοίχισης προτύπων κωδικοποιημένων με το χέρι.

Οι κανόνες χρησιμοποιούν πληροφορίες για τα συμφραζόμενα για να προσδιορίσουν εάν οι υποψήφιες οντότητες από τα ενημερωτικά κείμενα είναι έγκυρες ή για να επεκτείνουν το σύνολο των υποψηφίων. Τα κείμενα λειτουργούν ως ένα σημείο εκκίνησης από το οποίο θα καθιερωθεί, θα απορριφθεί ή θα τελειοποιηθεί η τελική οντότητα που θα εξαχθεί.

Ένας τυπικός αγωγός επεξεργασίας NERC αποτελείται από την γλωσσική προ-επεξεργασία (tokenization διαχωρισμός προτάσεων, προσθήκη ετικετών POS), ακολουθούμενη από εύρεση οντοτήτων με χρήση gazetteers και γραμματικές, και στην συνέχεια το ψήφισμα συνεισφοράς.

Το σύνολο gazetteers για το NERC μπορεί να περιέχει εκατοντάδες ή και χιλιάδες καταχωρήσεις. Η χρήση όμως μόνο του gazetteers, θεωρείται ανεπαρκής τόσο για την αναγνώριση, όσο και για την ταξινόμηση οντοτήτων, και αυτό γιατί αρκετές ονομαστικές οντότητες είναι διφορούμενες (π.χ. το «Paris» θα μπορούσε να είναι μέρος ενός ονόματος οργανισμού, ενός ονόματος ατόμου, ποδοσφαιρικής ομάδας ή απλώς τοποθεσία) ή και ακόμα δεν μπορούν να προσδιορίσουν κάθε οντότητα. Όμως, όταν τα gazetteers συνδυάζονται με άλλους γλωσσικούς σχολιαστές προ-επεξεργασίας (οι αναφορές (part-of-speech tagging), χρήση κεφαλαίων, αλλά και συμφραζόμενα στοιχεία), μπορεί και το καθιστούν πολύ ισχυρούς.

Όσο για την χρήση της αντιστοίχισης προτύπων για το NERC απαιτεί την ανάπτυξη μοτίβων σε πολύπλευρες δομές που λαμβάνουν υπόψη πολλές διαφορετικές ιδιότητες των λέξεων, όπως η ορθογραφία (κεφαλαία), μορφολογία, πληροφορίες μέρους του λόγου κτλ. Όμως, οι παραδοσιακές γλώσσες αντιστοίχισης προτύπων, όπως το PERL, γίνονται γρήγορα μη διαχειρήσιμα λόγω πολυπλοκότητας όταν χρησιμοποιούνται για τέτοιες εργασίες. Για τον λόγο αυτόν συνήθως χρησιμοποιούνται σημειώσεις χαρακτηριστικών τιμών που επιτρέπουν στις συνθήκες να αναφέρονται σε διακριτά χαρακτηριστικά τα οποία προκύπτουν από πολλαπλά επίπεδα ανάλυσης. Ένα τέτοιο παράδειγμα είναι το JAPE το οποίο είναι ένα μοτίβο βασισμένο σε Java[6?]. Το JAPE κάνει χρήση μιας δηλωτικής σημείωσης που επιτρέπει τη σύνταξη κανόνων με ευαισθησία στο πλαίσιο και τη μη ντετερμινιστική αντιστοίχιση προτύπων που πρέπει να εκτελεστούν.

Οι κανόνες χωρίζονται σε φάσεις, οι όποιες εκτελούνται διαδοχικά. Κάθε φάση συνήθως αποτελείται από κανόνες για τον ίδιο τύπο οντότητας (π.χ. Οργανισμός) ή κανόνες που έχουν τις ίδιες συγκεκριμένες απαιτήσεις για την λειτουργία τους. Επιπλέον, για τον χειρισμό της ασάφειας, μπορεί κανείς να προτιμήσει τα μοτίβα να εμφανίζονται σε ένα συγκεκριμένο πλαίσιο ή τύπο οντότητας έναντι ενός άλλου σε μια δεδομένη στιγμή. Αυτό καθίστανται δυνατό από μια ποικιλία μηχανισμών προτεραιότητας που επιτρέπουν την αντιμετώπιση του ανταγωνισμού στους κανόνες. Ένας απλός κανόνας αντιστοίχισης προτύπων μπορεί να προσπαθήσει να ταιριάξει τα ονόματα όλων των πανεπιστημίων, π.χ. University of

Sheffield, University of Bristol, όπου το μοτίβο αποτελείται από τις συγκεκριμένες λέξεις University και ακολουθούμενο από το όνομα μιας πόλης. Με το gazetteer, μπορούμε να ελέγξουμε την αναφορά ενός ονόματος πόλης, όπως το Sheffield ή το Bristol. Ένας πιο περίπλοκος κανόνας μπορεί να προσπαθήσει να αναγνωρίσει το όνομα οποιοδήποτε οργανισμού αναζητώντας μια λέξη- κλειδί από ένα ενημερωτικό κείμενο, όπως Εταιρεία, Οργανισμός, Σχολείο κ.λπ. που εμφανίζονται μαζί με ένα ή περισσότερα ουσιαστικά (όπως αυτά αναγνωρίστηκαν από την μέθοδο POS Tagger), και ενδεχομένως να περιέχει και κάποιες συναρτησιακές λέξεις. Αυτοί του τύπου οι κανόνες μπορεί να είναι καλοί στην αντιστοίχιση τυπικών μοτίβων, αλλά μπορεί να είναι πολύ διφορούμενοι.

Όπως αναφέρθηκε παραπάνω, τα συστήματα βασισμένα σε κανόνες αναπτύσσονται με βάση τα γλωσσικά χαρακτηριστικά, όπως αναφορές (ετικέτες POS) ή πληροφορίες του περιβάλλοντος. Αντί όμως να αναπτύσσονται χειροκίνητα τέτοιοι κανόνες, τα συστήματα αυτά καθιστούν δυνατή την παραγωγή αυτόματων συνόλων κανόνων από επισημασμένα παραδείγματα εκπαίδευσης χρησιμοποιώντας εποπτευόμενη μάθηση.

2.4.2 Μέθοδοι Επιβλεπόμενης Μάθησης για το NERC

Η επιβλεπόμενη μάθηση για το NERC σε γενική προσέγγιση αποτελείται από πέντε στάδια:

- Γλωσσική προ-επεξεργασία,
- Εξαγωγή χαρακτηριστικών,
- Μοντέλα εκπαίδευσης σε δεδομένα εκπαίδευσης,
- Εφαρμογή μοντέλων σε δεδομένα δοκιμής,
- Εκ των υστέρων επεξεργασία των αποτελεσμάτων για την προσθήκη ετικετών στα έγγραφα. [7]

Η γλωσσική προ-επεξεργασία στο χαμηλότερο επίπεδο περιλαμβάνει συμβολισμό και διαχωρισμό προτάσεων. Επίσης ανάλογα με τα χαρακτηριστικά που χρησιμοποιούνται μπορεί να περιλαμβάνει μορφολογική ανάλυση, προσθήκη ετικετών με μέρη του λόγου(αναφορών). Όμως, τα πιο γνωστά χαρακτηριστικά περιλαμβάνουν, μορφολογικά χαρακτηριστικά (π.χ. χρήση κεφαλαίων, εμφάνιση ειδικών χαρακτήρων), χαρακτηριστικά μέρη του λόγου (π.χ. ετικέτες του συμβάντος), χαρακτηριστικά περιβάλλοντος (π.χ. λέξεις και POS λέξεων σε ένα παράθυρο γύρων από την εμφάνιση, συνήθως 1-3 λέξεις), χαρακτηριστικά Gazetteer (π.χ. εμφάνιση σε NE gazetteers), συντακτικά χαρακτηριστικά (π.χ. χαρακτηριστικά που βασίζονται στην ανάλυση της πρότασης) και τέλος χαρακτηριστικά αναπαράστασης λέξεων (λειτουργίες που βασίζονται σε εκπαίδευση χωρίς επίβλεψη σε κείμενο χωρίς ετικέτα κάνοντας χρήση π.χ. Brown ομαδοποίηση ή ενσωματώσεις λέξεων).

Όσο για τις στατιστικές προσεγγίσεις NERC χρησιμοποιούν μια ποικιλία μοντέλων όπως τα Hidden Markov Models (HMM) , Μοντέλα μέγιστης εντροπίας, Υποστήριξη διανυσμάτων μηχανών (SVM), Perceptron's, Conditional Random Fields (CRFs) ή νευρωνικά δίκτυα [7]. Από τις πιο επιτυχημένες προσεγγίσεις για το NERC είναι εκείνα τα μοντέλα που βασίζονται σε CRF και πιο πρόσφατα εκείνα με τα πολύ-επίπεδα νευρωνικά δίκτυα.

Τα CRF μοντελοποιούν το NERC με την προσέγγιση της επισήμανσης της ακολουθίας, δηλ. μοντελοποιείται η ετικέτα έτσι ώστε να εξαρτάται από ετικέτες των προηγούμενων και των επόμενων διακριτών οντοτήτων σε ένα συγκεκριμένο παράθυρο. Κάποια παραδείγματα τέτοιων πλαισίων που είναι διαθέσιμα για το NERC και βασίζονται σε CRF είναι το Stanford NER3 και το CRFSuite.4.

Όσο για τις προσεγγίσεις των πολύ-επίπεδων νευρωνικών δικτύων έχουν δύο πλεονεκτήματα. Ένα από αυτά είναι ότι μαθαίνουν λανθάνοντα χαρακτηριστικά, που αυτό σημαίνει ότι δεν απαιτούν κάποια γλωσσική επεξεργασία πέρα από τον διαχωρισμό προτάσεων. Αυτό έχει σαν αποτέλεσμα να τα καθιστά πιο ισχυρά σε τομείς που οι αρχιτεκτονικές τους βασίζονται σε ρητά χαρακτηριστικά, καθώς δεν απαιτούν να αντισταθμίζουν τα λάθη που έγιναν κατά την προ-επεξεργασία. Το επόμενο πλεονέκτημα είναι ότι μπορούν εύκολα να ενσωματώσουν κείμενο χωρίς ετικέτα, στο οποίο μπορούν να χρησιμοποιηθούν μέθοδοι εξαγωγής εκπαιδευμένων χαρακτηριστικών αναπαράστασης. Ένα τέτοιο σύστημα, που αποτελείται από ένα πολύ-επίπεδο νευρωνικό δίκτυο με προ-εκπαίδευση χωρίς επίβλεψη, είναι για το SENNA [7]. Όπως και τα προηγούμενα πλαίσια, έτσι και αυτό διανέμεται με δυνατότητα εξαγωγής και προσφέρει λειτουργικότητα για μοντέλα εκπαίδευσης σε νέα δεδομένα.

Σε μια μάθηση με επίβλεψη για το NERC υπάρχουν πλεονεκτήματα και μειονεκτήματα σε σχέση με μια προσέγγιση μηχανικής γνώσης που βασίζεται σε κανόνες. Και οι δύο μέθοδοι απαιτούν την χειροκίνητη παρέμβαση. Αυτό συμβαίνει γιατί οι προσεγγίσεις που βασίζονται σε κανόνες απαιτούν από ειδικούς μηχανικούς της γλώσσας να παράγουν χειροκίνητα κωδικοποιημένους κανόνες (οι προσεγγίσεις που βασίζονται στην μάθηση με επίβλεψη απαιτούν σχολιασμένα δεδομένα εκπαίδευσης αλλά όχι την παρουσία μηχανικών της γλώσσας). Όσο για την επιλογή της καταλληλότερης προσέγγισης σε μια εφαρμογή, εξαρτάται καθαρά από την εκάστοτε εφαρμογή και τον τομέα της [7].

ΚΕΦΑΛΑΙΟ 3: Συναισθηματική Ανάλυση

Τα βασικά εργαλεία στην κατανόηση του κειμένου είναι ο εντοπισμός και η ταξινόμηση της γνώμης και των συναισθημάτων [7]. Σε αυτό το κεφάλαιο θα μιλήσουμε για τα βασικά στοιχεία ενός τυπικού εργαλείου ανάλυσης συναισθημάτων, εισάγοντας μια ποικιλία από διαφορετικές πιθανές μεθόδους και δίνοντας παραδείγματα πραγματικών εφαρμογών σε διάφορους τομείς. Η ανάλυση συναισθήματος αφορά την ανάλυση κειμένου προκειμένου να κατανοήσουμε τις απόψεις των ανθρώπων. Το συναίσθημα μπορεί να είναι στην απλούστερη μορφή που σημαίνει να κατανοήσουμε εάν ένα άτομο μιλάει με θετικό ή αρνητικό τρόπο για κάτι, αλλά μπορεί να είναι και πιο λεπτής κατάστασης όπως ότι είναι φοβισμένος, σοκαρισμένος, θυμωμένος, έκπληκτος, χαρούμενος κτλ.

Τα εργαλεία εξόρυξης γνώμης λαμβάνουν ένα κομμάτι κειμένου ως είσοδο και εξάγουν πληροφορίες όπως αν ένα τμήμα του κειμένου είναι γνωμικό, τι είδους γνώμη εκφράζει (θετική ή αρνητική), το βαθμό ισχύος την γνώμης και πιθανών επιπρόσθετες πληροφορίες (πχ. ποια είναι η γνώμη, ποιος έχει τη γνώμη)[7]. Θα μπορούσε κανείς να πει ότι το έργο της εξόρυξης είναι αρκετά ξεκάθαρο, κάτι που δεν ισχύει όμως στην πράξη. Η εργασία είναι πολύ πιο περίπλοκη από μια απλή αναγνώριση θετικών και αρνητικών λέξεων και αυτό συμβαίνει επειδή η φυσική γλώσσα είναι απίστευτα περίπλοκη και διφορούμενη. Κυρίως η περιπλοκότητα αυτή διακρίνεται στο γραπτό λόγο αφού αρκετοί άνθρωποι χρησιμοποιούν ασυνήθιστους όρους για να περιγράψουν τα συναισθήματα τους, χαρακτηρίζουν τις δηλώσεις αρνητικά, δεν κάνουν σωστή χρήση της γραμματικής ή της ορθογραφίας, μπορεί να είναι σαρκαστικοί ή να υποθέτουν ότι ο αναγνώστης έχει πρόσθετη παγκόσμια γνώση ώστε να αποκρυπτογραφήσουν το νόημα χωρίς ρητή αναφορά (π.χ. οι αναφορές στον Βολντερμορτ ή τον Χίτλερ είναι γενικά αρνητικές). Αυτό έχει σαν αποτέλεσμα την απαίτηση μιας πολύπλοκης γλωσσικής ανάλυσης για να μπορέσει να αποκρυπτογραφηθεί σωστά το νόημα.

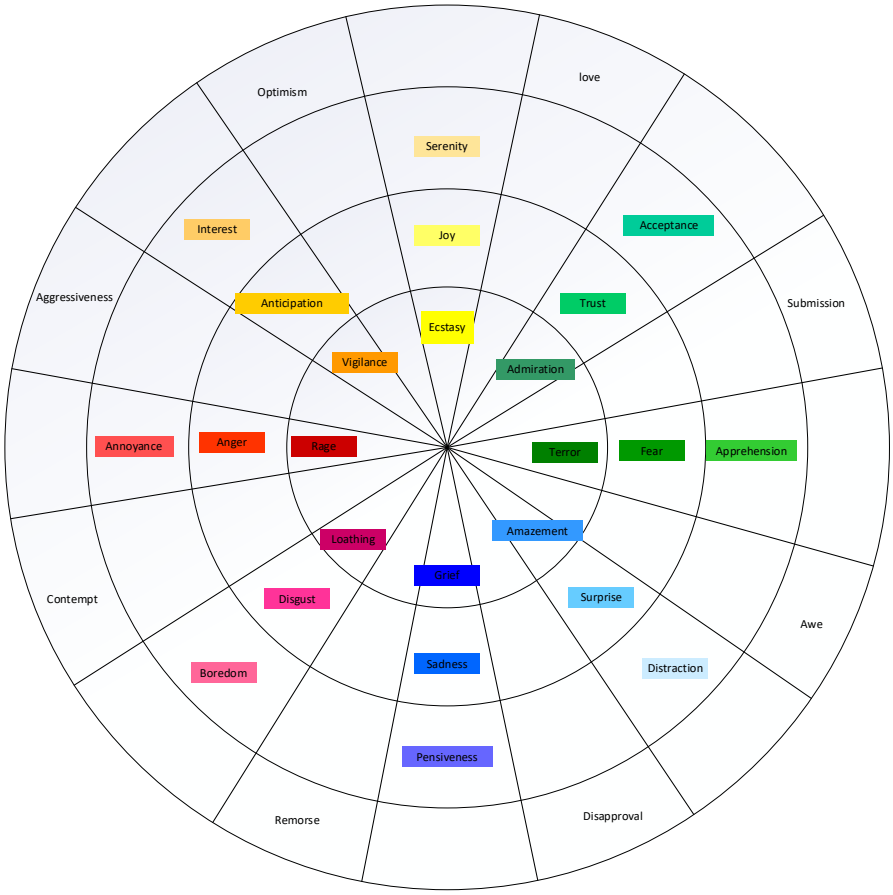
Με βάση την θεωρία όπου οι απόψεις και το συναίσθημα είναι δύο διαφορετικά πράγματα, το ίδιο ισχύει και για την εξόρυξη γνώμης με την συναισθηματική ανάλυση. Τα συναισθήματα εκφράζουν συνήθως μια συγκεκριμένη πολικότητα (θετική, αρνητική ή ουδέτερη). Για παράδειγμα “Νομίζω ότι το παντελόνι σου είναι όμορφο ” είναι ένα θετικό συναίσθημα που εκφράστηκε από εμένα. Οι απόψεις μπορεί να εκφράζουν κάτι πιο γενικό, το όποιο να δηλώνει μια άποψη για κάτι, χωρίς όμως να δηλώνει κάποιο ιδιαίτερο συναίσθημα.

Από τις πρώτες μέρες της έρευνας για την εξόρυξη γνώμης, ο όρος «εξόρυξη γνώμης» χρησιμοποιήθηκε για κάτι αρκετά περιεκτικό, ενώ η ανάλυση συναισθήματος χρησιμοποιήθηκε ειδικά για το έργο της ανίχνευσης πολικότητας. Παρακάτω, θα δούμε μεθόδους και εργαλεία σχετικά με την ανίχνευση συναισθήματος.

3.1 Ανίχνευση Συναισθήματος

Για εργασίες του πραγματικού κόσμου, τα εργαλεία της εξόρυξης γνώμης εμπλουτίζονται με περισσότερα σχήματα πέρα από αυτό της ανίχνευσης θετικού/αρνητικού συναισθήματος (γίνεται μια προσέγγιση βασισμένη στο συναίσθημα η οποία ταξινομεί τα κείμενα με γνώμη με περισσότερο έμφαση στα συναισθήματα που εκφράζονται). Αυτό είναι χρησιμότερο για πρακτικούς σκοπούς όπως για παράδειγμα αν έχουμε μια εταιρεία η οποία θέλει να γνωρίζει συγκεκριμένα εάν οι άνθρωποι φοβούνται για ένα προϊόν ή είναι θυμωμένοι με αυτό και όχι απλώς αν είναι αρνητικοί για αυτό.

Τα συναισθήματα μπορεί να έχουν εκφρασμένες λεπτές τιμές πολικότητας ως προς τις καθορισμένες οντολογίες. Ωστόσο, καθίστανται δύσκολο να οριστεί ένας πλήρης και σαφής οδηγός συναισθημάτων. Παρά τις πολλές προσπάθειες για τον καθορισμό προτύπων, δεν υπάρχει ακόμη συναίνεση για ένα βασικό σύνολο συναισθημάτων. Η πιο συχνά διαδεδομένη αναπαράσταση είναι αυτή του τροχού των συναισθημάτων του Plutchjk, όπως απεικονίζεται παρακάτω. Προσπαθεί να δείξει πως διαφορετικά συναισθήματα σχετίζονται, αλλά είναι ίσως πολύ περίπλοκο για αναπαράσταση ανίχνευσης συναισθημάτων.



Εικόνα 3.1: Τροχός Συναισθημάτων Plutchjk
(https://en.wikipedia.org/wiki/Robert_Plutchik)

Ο τροχός προσπαθεί να αναπαραστήσει τα οκτώ κύρια διπολικά συναισθήματα, τα όποια διακρίνουμε στο 2ο κύκλο από το κέντρο του τροχού. Ο στόχος του τροχού είναι η συγχώνευση των πρωταρχικών συναισθημάτων με άλλα συναισθήματα, όπως αυτό ακριβώς συμβαίνει με τις εντάσεις των χρωμάτων. Για παράδειγμα, αν συνδυάσουμε το συναίσθημα της χαράς (joy) με εκείνο της εμπιστοσύνης (trust) θα έχουμε το αποτέλεσμα της αγάπης (love), το αντίθετο της οποίας είναι οι τύψεις (remorse). Στη αναπαράσταση του τροχό αυτά που χρήζουν ιδιαίτερης προσοχής είναι η ταξινόμηση των αντιθέτων. Κανείς θα παρατηρήσει πως ο τροχός έχει ταξινομήσει το αντίθετο της αισιοδοξίας (optimism) με εκείνο της αποδοκιμασίας (disapproval), καθώς επίσης το αντίθετο της έκπληξης (surprise) με εκείνο της προσμονής (anticipation).

Μια ακόμη μέθοδος, εκτός από εκείνη του τροχού, είναι η λίστα συναισθημάτων δομημένη με δέντρο Parrott [15], χρησιμοποιεί βασικές κατηγορίες του Plutchik's αλλά τις επεκτείνει διαφορετικά. Χρησιμοποιεί τρία επίπεδα, τα δύο πρώτα από τα όποια εμφανίζονται στον παρακάτω πίνακα.

Primary Emotion	Secondary Emotion
Love	Affection
	Lust/Sexual Desire
	Longing
Joy	Cheerfulness
	Zest
	Contentment
	Pride
	Optimism
	Enthrallment
	Relief
Surprise	Surprise
Anger	Irritability
	Exasperation
	Rage
	Dislike
	Disgust
	Envy
	Torment
Sadness	Suffering
	Depression
	Disappointment
	Shame
	Neglect
	Sympathy
Fear	Horror
	Nervousness

Εικόνα 3.2: Λίστα Συναισθημάτων Δομημένη με Δέντρο Parrott [15]

Μια Τρίτη αναπαράσταση είναι η EARL(Emotion Annotation Representation Language). Αναπτύχθηκε ειδικά για τον σχολιασμό συναισθημάτων από το δίκτυο Αλληλεπίδρασης Ανθρώπου-Μηχανής στο συναίσθημα και ταξινομεί 48 συναισθήματα που φαίνονται στους παρακάτω δυο πίνακες [15].

EARL αναπαράσταση των αρνητικών συναισθημάτων					
Forceful	Anger Annoyance Contempt Disgust Irritation	**Passive**	Boredom Despair Disappointment Hurt Sadness	**Negative Thoughts**	Doubt Envy Frustration Guilt Shame
Not in Control	Anxiety Ambarrassment Fear Helplessness Powerlessness Worry	**Agitation**	Stress Shock Tension		
EARL αναπαράσταση των θετικών συναισθημάτων					
Lively	Amusement Delight Elation Excitement Happiness Joy Pleasure	**Caring**	Affection Empathy Friendliness Love	**Positive Thoughts**	Courage Hope Pride Satisfaction Trust
		Quiest Positive	Calmness Contentment Relaxation Relief Serenity	**Reactive**	Interest Politeness Surprise

Εικόνα 3.3: Αναπαράσταση EARL [15]

Αυτό που πρέπει να λάβουμε υπόψιν είναι πως τα αντίθετα συναισθήματα δεν είναι απαραίτητα τα ίδια με τα αρνητικά συναισθήματα. Ένα απλό παράδειγμα το «δεν είμαι λυπημένος», το αντίθετο της πρότασης δεν σημαίνει πως είναι «είμαι χαρούμενος». Επομένως, η τυπική τεχνική αναστροφής πολικότητας δεν αποτελεί λύση όσον αφορά την ανίχνευση συναισθημάτων. Το ζήτημα δεν έχει εξεταστεί εκτενώς στην βιβλιογραφία για την ανίχνευση συναισθημάτων.

3.2 Σχετικές έρευνες

Η ανάλυση συναισθήματος αποτελεί ένα σημαντικό πεδίο έρευνας καθώς πρόκειται για ένα μεγάλο πρόβλημα κυρίως στις διαδικτυακές κοινότητες και πλατφόρμες κοινωνικής δικτύωσης όπου η προσβλητική γλώσσα κυριαρχεί σε υψηλό βαθμό. Οι περισσότερες έρευνες που υπάρχουν στον τομέα της επεξεργασίας φυσικής γλώσσας αφορούν κυρίως την αγγλική

γλώσσα και αυτό γιατί οι ψηφιακοί αγγλικοί πόροι είναι πολύ περισσότεροι από κάθε άλλη γλώσσα. Η έλλειψη των λεξιλογικών πόρων για την εξόρυξη συναισθήματος δημιουργεί την μεγαλύτερη πρόκληση για τους ερευνητές του πεδίου επεξεργασίας φυσικής γλώσσας.

Κάποιες εργασίες, για να καλύψουν το κενό, δημιούργησαν λεξιλογικούς πόρους εκτός της αγγλικής γλώσσας. Αυτές βασίστηκαν στην μετάφραση λεξικών συναισθημάτων που βασίζεται στην αγγλική γλώσσα και σε χαρτογραφήσεις του συνόλου WordNet. (Jijkoun et al. 2010; Das and Bandyopadhyay 2010; Arora et al. 2012; Perez-Rosas et al. 2012).

Στο Vania et al. (2014), έγινε μια παρόμοια προσέγγιση, ώστε να δημιουργηθεί ένα αρχικό λεξικό της ινδονησιακής γλώσσας, όπου για την επέκτασή του χρησιμοποιήθηκαν διαφορετικοί μέθοδοι, δηλαδή εύρεση λέξεων σε κοινά μοτίβα των τριών γραμμάτων με θετικές/αρνητικές λέξεις σε σώμα.

Ο Perez-Rosas et al. (2012) κατάφερε να γεφυρώσει το γλωσσικό χάσμα μεταξύ της αγγλικής και της ισπανικής γλώσσας χρησιμοποιώντας την πολύγλωσση δομή του WordNet ευθυγραμμισμένη σε επίπεδο αίσθησης δημιουργώντας ένα λεξικό υψηλής ακρίβειας.

Μια ακόμη προσέγγιση είναι εκείνη, με την βοήθεια του αλγορίθμου PageRank, όπου χρησιμοποιήθηκε από τους Jijkoun and Hofmann (2009), για την ανάπτυξη ενός λεξικού της Ολλανδικής γλώσσας με βάση τις σχέσεις των συνόλων WordNet.

Σταδιακά με τον καιρό, έχουν αρχίσει να εμπλουτίζονται και να διερευνώνται, μηχανισμοί και για άλλες γλώσσες, όπως για παράδειγμα τα ισπανικά, τα χίντι και τα γερμανικά. Αντίστοιχα μελέτες που έχουν πραγματοποιηθεί για την επεξεργασία φυσικής γλώσσας και την ανάλυση συναισθήματος σε ελληνικά κείμενα είναι ακόμα πιο περιορισμένες. Ένα παράδειγμα είναι εκείνο του Παλογιαννίδη et al. (2015), που πραγματοποίησε μετάφραση αγγλικών λέξεων από το λεξικο ANEW (Bradley et al. 1999) και έγινε χειροκίνητος σχολιασμός όσο αφορά το σθένος, την διέγερση και την κυριαρχία τους.

Οι Tsakalidis, A., Papadopoulos, S., Voskaki, R., Ioannidou, K., Boididou, C., Cristea, A. I., Liakata, M. & Kompatsiaris, Y. (2018), έχοντας υπόψη τους το λεξικό Τριανταφυλλίδη (1998), μετρόντας 46.747 λήμματα, της νέας ελληνικής γλώσσας, συγκέντρωσαν λέξεις με την πιθανότητα να εμφανίζουν συναισθηματικό φορτίο. Στη συνέχεια χρησιμοποιήσαν βοηθητικά προγράμματα, με στόχο την ανάκτηση λέξεων που χρησιμοποιούνται σε μια ειρωνική πρόταση, υποτιμητική, υβριστική, χλευαστικό ή χυδαίος τόνος, συγκεντρώνοντας 2260 λέξεις.

Καλύπτοντας το χάσμα των λεξιλογικών πόρων, το επόμενο βήμα είναι η αξιοποίηση αυτών των πόρων για την δημιουργία μοντέλων και μηχανισμών όπου ανιχνεύουν το συναίσθημα σε ένα κείμενο.

Η εργασία της αναφοράς [10], αναφέρει μια μέθοδο σχετικά απλή, αφού η αξιολόγηση του συναισθήματος είναι αποτέλεσμα άμεσων υπολογισμών που προέρχονται από τις λέξεις που συντάσσουν ένα tweet. Η μέθοδος στην όποια αναφέρονται, δεν είναι σε θέση να αξιολογήσει σωστά τα tweet που περιέχουν σαρκαστικά σχόλια και ασάφειες. Η μέθοδος που προτείνουν για την αξιολόγηση συναισθήματος ενός tweet, είναι για κάθε λήμμα του λεξικού συναισθήματος το όποιο εντοπίζεται μέσα στο tweet, έχουμε ένα διάνυσμα W όπου περιέχει 6 συστατικά (ένα για κάθε συναίσθημα), έχοντας έτσι ένα διάνυσμα Wi με N στοιχεία

$$\overrightarrow{W_j} = [W_1, W_2, W_3, W_4, W_5, W_6]$$

για j = 1…N, όπου N είναι ο αριθμός των στοιχείων που εντοπίστηκαν μέσα στο tweet. Σχηματίζοντας ένα διάνυσμα T με 6 συστατικών, όπου κάθε συστατικό είναι το αποτέλεσμα του παρακάτω τύπου,

$$\vec{T} = [t_1, t_2, t_3, t_4, t_5, t_6]$$

$$t_i = \sqrt{\frac{\sum_{j=1}^{N} W_i^2}{N}}$$, όπου i = 1,…,6.

Ο παραπάνω τύπος επιστρέφει το τετραγωνικό μέσο όρο των εγγραφών που προσδιορίστηκαν σε κάθε tweet. Έχοντας, στην διάθεσή τους ένα ελληνικό λεξικό συναισθήματος και με την χρήση της παραπάνω μεθόδου, καταφέρνουν να εξάγουν το κυρίαρχο συναίσθημα ενός tweet. polarity

Για παράδειγμα, έστω έχουμε το παρακάτω tweet και το ακόλουθο συναισθηματικό λεξικό όπου έχει τις ακόλουθες στήλες POS1 = ετικέτα του λόγου, SUBJECTIVITY1 = υποκειμενικότητα, POLARITY1 = πόλωση, ANGER1 = θυμός, DISGUST1 = Αηδία, FEAR1 = Φόβος, HAPPINESS1 = Χαρά, SADNESS1 = θλίψη, SURPRISE1 = Έκπληξη.

Tweet: “@USER Είσαι αστέρι βρε Κατερίνα. Και κούκλα εκτός από φοβερή αθλήτρια και φοβερός άνθρωπος. Πάντα και μόνο επιτυχίες εύχομαι.”

Sentiment_lexicon:

TERM	POS1	SUBJECTIVITY 1	POLARITY1	ANGER1	DISGUST1
αστέρι	NOUN	SUBJ+	POS	N/A	N/A
βρε	INTJ	SUBJ+	NEG	4	2
άνθρωπος	NOUN	SUBJ-	POS	1	1

TERM	POS1	FEAR1	HAPPINESS1	SADNESS1	SURPRISE1
αστέρι	NOUN	N/A	4	1	N/A
βρε	INTJ	1	1	1	4
άνθρωπος	NOUN	1	4	1	4

Εικόνα 3.4: Πίνακας Συναισθημάτων

Πριν την χρήση της μεθόδου, γίνεται συντακτική ανάλυση του tweet με στόχο τον εντοπισμό των επιμέρους ετικετών. Για κάθε, λέξη – ετικέτα του λόγου που βρίσκεται μέσα στο λεξικό μου ορίζω το συναίσθημα. Δηλαδή, για την λέξη «αστέρι» θα έχουμε το διάνυσμα, θεωρώντας όπου το N/A = 0

$$W_{αστέρι} = [0, 0, 0, 4, 1, 0]$$

Με τον ίδιο τρόπο, θα έχουμε τα υπόλοιπα διανύσματα,

$$W_{άνρθωπος} = [1, 1, 1, 4, 1, 4]$$

στο συγκεκριμένο παράδειγμα η λέξη «βρε» δεν υπολογίζεται καθώς δεν ταιριάζει η ετικέτα του λόγου με εκείνη του λεξικού.

Στη συνέχεια, θα έχουμε ένα διάνυσμα

$$T = [\sqrt{\frac{1}{2}}, \sqrt{\frac{1}{2}}, \sqrt{\frac{1}{2}}, 4, 1, \sqrt{8} = 2{,}8]$$

Έχοντας το στοιχείο $t_4 = 4$, να είναι κι εκείνο που δηλώνει το βασικό συναίσθημα του tweet, και αυτό δεν είναι άλλο από το συναίσθημα της χαράς.

Άλλες εργασίες, επικεντρώθηκαν στην ανάπτυξη μηχανισμών με χρήση αλγορίθμων ταξινόμησης και μηχανικής μάθησης. Όπως, θα αναφέρουμε σε επόμενη ενότητα, ότι η μηχανική μάθηση, αφορά την μελέτη και την ανάπτυξη αλγορίθμων όπου μαθαίνουν και πραγματοποιούν προβλέψεις πάνω στα δεδομένα που έχουν στην είσοδο τους. Γι' αυτό κυρίως οι εργασίες αυτές, προαπαιτούν την δημιουργία ενός dataset ώστε να δοθεί ως είσοδο στο

μοντέλο προς εκπαίδευση. Το dataset κυρίως είναι δεδομένα που σχολιάζονταν με το χέρι, από σχολιαστές.

Άλλη έρευνα όπου αναγνωρίζουν αν το tweet είναι προσβλητικό ή όχι είναι του Pitenis Z., et al. 2020. Η έρευνα αναφέρει πως το μοντέλο με την καλύτερη απόδοση είναι το LSTM και GRU με σκορ 0.89F1.

Τέλος, άλλες εργασίες παρουσιάζουν το μοντέλο Μηχανών Διανυσμάτων Υποστήριξης (Support Vector Machines SVM) ως μια καλή λύση για τον προσδιορισμό του συναισθήματος μεταξύ των 6 βασικών συναισθημάτων (Tsakalidis, A., et al. 2018).

ΚΕΦΑΛΑΙΟ 4: Συναισθηματική Ανάλυση και Μηχανική Μάθηση

Η μηχανική μάθηση αποτελεί πεδίο της Τεχνικής Νοημοσύνης, όπου ασχολείται με το σχεδιασμό και την ανάπτυξη αλγορίθμων που επιτρέπει στους υπολογιστές (μηχανές) να βελτιώνουν την επίδοση τους με την πάροδο του χρόνου. Επιτρέπει δηλαδή, να μαθαίνουν μέσα από μια επαγωγική διαδικασία που βασίζεται σε δεδομένα που λαμβάνονται από αισθητήρες ή βάσεις δεδομένων.[11] Ένας ακόμα ορισμός που έχει αποδοθεί από το Business Dictionary αναφέρει το εξής, «μηχανική μάθηση είναι η ικανότητα μιας μηχανής να βελτιώνει την επίδοση της μέσα από τη χρήση λογισμικού που εφαρμόζει μεθόδους από το χώρο της τεχνικής νοημοσύνης προκειμένου να μιμηθεί τους τρόπους με τους οποίους φαίνεται πως μαθαίνουν οι άνθρωποι, όπως η επανάληψη και η εμπειρία.»

Στην μηχανική μάθηση, μπορούμε να διακρίνουμε δύο κύριες μεθόδους, την επιβλεπόμενη μάθηση και την μη-επιβλεπόμενη μάθηση. Παρακάτω αναλύονται κάποια από τα μοντέλα, κυρίως της μεθόδου επιβλεπόμενης μάθησης, όπως τα Συνελικτικά Νευρωνικά Δίκτυα Βαθιάς Μάθησης (CNN) και οι Μηχανές Διανυσμάτων Υποστήριξης (SVN). Επιπλέων, θα δούμε τα επαναλαμβανόμενα νευρωνικά δίκτυα, μια τάξη των τεχνικών νευρικών δικτύων.

4.1 Συνελικτικά Νευρωνικά Δίκτυα (Convolution Neural Networks - CNN)

Η εργασία των Hubel και Wiesel (1962, 1977), πάνω στους τοπικά ευαίσθητους και επιλεκτικούς ως προς τον προσανατολισμό νευρώνες του οπτικού φλοιού της γάτας, είναι αυτή που οδήγησε στην ανάπτυξη των CNN. Τα CNN είναι ένα perceptron πολλών επιπέδων, το όποιο έχει σχεδιαστεί έτσι ώστε να αναγνωρίζει σχήματα δισδιάστατα με μεγάλο βαθμό την μη-ευαισθησίας[1], την μετατόπιση, την κλιμάκωση και άλλες μορφές παραμόρφωσης. Τα δίκτυα όμως αυτού του είδους καθίστανται επιρρεπής στην υπέρ- προσαρμογή των δεδομένων. Αυτό οφείλεται στην πλήρη συνδεσιμότητα, που διατρέχει τους νευρώνες μεταξύ των νευρώνων μεταξύ των στρωμάτων. Ένα perceptron πολλών επιπέδων για να αποτρέψει την υπέρ-προσαρμογή, «τιμωρεί» τους παραμέτρους κατά την εκπαίδευση (π.χ. μείωση του βάρους) ή πραγματοποιώντας εγκατάλειψη συνδέσεων. Όμως, τα CNN έχουν υιοθετήσει ένα αρκετά διαφορετικό τρόπο να εκμεταλλεύονται το ιεραρχικό μοτίβο στα δεδομένα και να συναρμολογούν μοτίβα αυξανόμενης πολυπλοκότητας κάνοντας χρήση μικρότερων και απλούστερων μοτίβων ανάγλυφων στα φίλτρα τους. Τα CNN αποδεικνύονται κατάλληλα για την ταξινόμηση προτύπων και είναι ευρέος γνωστά στην ανάλυση οπτικών εικόνων.

Καλό θα ήταν να σημειώσουμε εδώ, πως τα CNN σε σχέση με άλλους αλγορίθμους ταξινόμησης εικόνων, κάνουν χρήση σχετικά λιγότερης προ-επεξεργασίας. Αυτό σημαίνει ότι μέσα από την αυτοματοποιημένη μάθηση, μαθαίνει να βελτιστοποιεί τα φίλτρα (ή τους

πυρήνες), σε αντίθεση με τους παραδοσιακούς αλγορίθμους όπου τα φίλτρα πρέπει να σχεδιαστούν με το χέρι, κερδίζοντας έτσι ένα σημαντικό πλεονέκτημα.

4.2 Επαναλαμβανόμενα νευρωνικά δίκτυα (Recurrent Neural Networks - RNN)

Τα επαναλαμβανόμενα νευρωνικά δίκτυα διαφέρουν από τα παραδοσιακά νευρωνικά, καθώς βασίζονται σε ένα μεγάλο πλεονέκτημα και αυτό είναι ότι επιτρέπουν στις πληροφορίες να διατηρηθούν. Κανείς θα περιέγραφε τα επαναλαμβανόμενα νευρωνικά δίκτυα ως πολλαπλά αντίγραφα του ίδιου δικτύου, όπου το καθένα περνάει ένα μήνυμα στο διάδοχο του [12].

Πιο συγκεκριμένα, έχοντας ένα νευρωνικό δίκτυο A, το όποιο δέχεται μια είσοδο xt και παράγει την έξοδο yt. Το επαναλαμβανόμενο νευρωνικό δίκτυο, μοιάζει με το νευρωνικό δίκτυο A, όμως με μια λεπτομέρεια, διαθέτει επιπλέον έναν βρόγχο, το οποίο επιτρέπει την μετάδοση πληροφορίας από ένα βήμα στο επόμενο.

Τα τελευταία χρόνια, υπήρξε απίστευτη επιτυχία στην εφαρμογή RNN σε μια ποικιλία προβλημάτων όπως, η αναγνώριση ομιλίας, μοντελοποίηση γλώσσας, υπότιτλοι εικόνων κα. Η βασική επιτυχία τον παραπάνω, οφείλετε κυρίως στην χρήστη των LSTM, ενός πολύ ιδιαίτερου είδους επαναλαμβανόμενου νευρωνικού δικτύου που λειτουργεί, για πολλές εργασίες.

4.2.1 Δίκτυα Μακροπρόθεσμη Μνήμη (Long Short-Term Memory - LSTM)

Τα δίκτυα μακροπρόθεσμης βραχυπρόθεσμης μνήμης (LSTM), είναι ένα ειδικό είδος RNN, ικανό να μάθει με μακροπρόθεσμες εξαρτήσεις. Εισήχθησαν από τους Hochreiter & Schmidhuber (1997) και βελτιώθηκαν και διαδόθηκαν από πολλούς ανθρώπους, καθώς επίσης αποδίδουν εξαιρετικά καλά σε μια μεγάλη ποικιλία προβλημάτων και χρησιμοποιούνται πλέον ευρέως.

Τα LSTM έχουν σχεδιαστεί συγκεκριμένα για την αποφυγή του προβλήματος μακροπρόθεσμης εξάρτησης, δηλαδή να μπορούν να θυμούνται πληροφορίες για μεγάλες χρονικές περιόδους.

Τα RNN έχουν την μορφή μιας αλυσίδας, που αποτελείτε από συνδεδεμένες επαναλαμβανόμενες μονάδες νευρωνικού δικτύου. Στα τυπικά RNN, αυτή η επαναλαμβανόμενη μονάδα θα έχει μια απλή μορφή, όπως ένα στρώμα tan. Σε αντίθεση με τα LSTM, όπου η επαναλαμβανόμενη μονάδα έχει διαφορετική δομή. Αντί για ένα ενιαίο επίπεδο νευρωνικού δικτύου, αποτελείται από τέσσερα στρώματα που αλληλοεπιδρούν το ένα με το άλλο.

Το βασικό για τα LSTM είναι η κατάσταση του κελιού, που μοιάζει περισσότερο με μεταφορικό ιμάντα, που διατρέχει όλη την αλυσίδα, με μερικές γραμμικές αλληλεπιδράσεις.

Το LSTM είναι σε θέση να αφαιρεί ή να προσθέτει πληροφορίες στην κατάσταση του κελιού, ρυθμίζοντας με προσοχή δομές που ονομάζονται και πύλες.

Οι πύλες μπορούν περάσουν προαιρετικά κάποιες πληροφορίες. Αποτελούνται από ένα στρώμα σιγμοειδούς νευρικού δικτύου και μια πράξη πολλαπλασιασμού. Το σιγμοειδές στρώμα παράγει τιμές μεταξύ του 0 και 1, δηλώνοντας με αυτό τον τρόπο κατά πόσο ένα στοιχείο θα πρέπει να μην περάσει ή να περάσει αντίστοιχα. Στο LSTM διακρίνουμε 3 από αυτές τις πύλες, για την ασφάλεια και τον έλεγχο της κατάστασης.

Το πρώτο βήμα του LSTM, είναι αυτό το όποιο θα αποφασίσει ποιες πληροφορίες θα αφαιρεθούν από την κατάσταση του κελιού. Για την απόφαση αυτή είναι υπεύθυνο ένα σιγμοειδές στρώμα που ονομάζεται «στρώμα της πύλης ξεχασμού». Το στρώμα ελέγχει τα ht-1 και xt, αποδίδοντας έναν αριθμό μεταξύ 0 και 1 για κάθε αριθμό στην κατάσταση κελιού Ct-1. Με το 0 να δηλώνει την «απαλλαγή του στοιχείου» και 1 να δηλώνει την «διατήρηση του στοιχείου».

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

Στην συνέχεια, το επόμενο βήμα είναι η επιλογή των νέων πληροφοριών που θα διατηρήσουμε στην κατάσταση του κελιού. Ένα σιγμοειδές στρώμα, που ονομάζεται «στρώμα πύλης εισόδου», και είναι υπεύθυνο να αποφασίσει ποιες τιμές θα ενημερωθούν. Ακολουθεί, ένα επίπεδο tanh, για την αναπαραγωγή ενός διανύσματος νέων υποψήφιων τιμών, $\widetilde{C_t}$, που ίσως μπορούσαν να προστεθούν στην κατάσταση. Το επόμενο βήμα είναι ο συνδυασμός αυτών των δύο βημάτων, ώστε να ενημερώσουμε την κατάσταση.

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$$
$$\widetilde{C_t} = \tanh(W_c \cdot [h_{t-1}, x_t] + b_C)$$

Στην συνέχεια, θα πρέπει να ενημερώσουμε την παλιά κατάσταση του κελιού, C_{t-1}, στην νέα κατάσταση κελιού C_t. Αυτό γίνεται πολλαπλασιάζοντας την παλιά κατάσταση με την κατάσταση που δημιούργησε το «στρώμα της πύλης ξεχασμού». Το αποτέλεσμα, την κατάστασης θα προστεθεί με το $i_t * \tilde{C}_t$.

Τέλος, η έξοδος βασίζεται σε μια φλιταρισμένη έκδοση της κατάστασης του κελιού. Αφού πρώτα βάλουμε ένα σιγμοειδές στρώμα να αποφασίσει τα μέρη της κατάστασης του κελιού που πρέπει να εξάγουμε και αποτέλεσμα αυτού του στρώματος το πολλαπλασιάσουμε, με την το αποτέλεσμα της κατάστασης του κελιού, όπου έχει περάσει πρώτα μέσω του tanh. Με αυτό τον τρόπο έχουμε στην έξοδο μας μόνο τα μέρη που αποφασίσαμε.

$$O_t = \sigma\left(W_o\,[h_{t-1}, x_t] + b_o\right)$$
$$h_t = O_t * \tanh(C_t)$$

4.3 Σχετικές έρευνες

Σύμφωνα με την εργασία της αναφορά [16], όπου είχε στόχο την «Κατανόηση των καταστάσεων στον πραγματικό κόσμο πραγματοποιώντας ανάλυση συναισθημάτων στην βάση δεδομένων του Twitter», μας επεξηγούν τον λόγο που προχώρησαν στην χρήση του μοντέλου CNN. Η ιδέα είναι, όπως το μοντέλο έχει την δυνατότητα να εξάγει μια περιοχή χαρακτηριστικών από μια γκάμα πληροφορίων και είναι σε θέση να εξετάσει τη σχέση μεταξύ αυτών των χαρακτηριστικών, έτσι για την ανάλυση συναισθήματος χρειάζεται να εξαχθεί μια πληροφορία από ένα μεγαλύτερο κομμάτι κειμένου. Το CNN δείχνει να αποδίδει στην ταξινόμηση των συναισθημάτων στο Twitter σε σχέση με ορισμένες παραδοσιακές μεθόδους, όπως το SVM και Naïve Bayes.

Στην εργασία της αναφοράς [19], αναφέρεται μια επέκταση του LSTM (το Sentic LSTM) έχοντας στόχο την ενσωμάτωση της ρητής γνώσης με την άρρητη γνώση. Αυτό βοήθησε ένα υβρίδιο του LSTM μετά την επέκταση που πραγματοποιήθηκε στο Sentic LSTM και ένα επαναλαμβανόμενο δίκτυο προσθετικών που προσομοιώνει αισθητικά πρότυπα. Το αποτέλεσμα της εργασίας είναι ότι το προτεινόμενο μοντέλο μπορεί να ενσωματώσει με αποτελεσματικότητα την γνώση της κοινής λογικής στο βαθύ νευρωνικό δίκτυο και να εκπαιδευτεί με τρόπο από άκρο σε άκρο.

Στην συνέχεια ακολουθεί η εργασία της αναφοράς [17] όπου προτείνετε έναν μοντέλο που συνδέει το CNN με το LSTM. Ο στόχος της εν λόγω εργασίας ήταν το προτεινόμενο περιφερειακό CNN να χρησιμοποιείται για τον διαχωρισμό ολοκλήρου του κειμένου της εισόδου σε προτάσεις (περιοχές). Με αυτό τον τρόπο οι χρήσιμες συναισθηματικές πληροφορίες μπορούν να εξαχθούν και να σταθμιστούν ανάλογα με την συμβολή τους. Τα αποτελέσματα της εργασίας φαίνονται στο μοντέλο που υπερτερεί των μεθόδων που βασίζονται σε λεξικά παλινδρόμησης και NN. Επίσης, η εργασία της αναφοράς [20] αναφέρεται με βελτιωμένη ακρίβεια στην ανάλυση συναισθήματος που εντοπίζεται σε αραβικά κείμενα, η χρήση ενός ενοποιημένου μοντέλου CNN – LSTM.

Τέλος, να σημειώσουμε πως οι έρευνες σχετικά με ελληνικά κείμενα είναι πολύ λιγότερες από ότι εκείνες σε άλλε γλώσσες.

ΚΕΦΑΛΑΙΟ 5: Πειραματικό Μέρος

Σ' αυτή την ενότητα θα μιλήσουμε για το πειραματικό μέρος της εργασίας αυτής. Πιο αναλυτικά Θα δούμε τόσο τον εμπλουτισμό του NER πεδίου και την εκπαίδευση του πακέτο spaCy για τα ελληνικά με την χρήση εφαρμογής που αναπτύχθηκε για τον σκοπό αυτό όσο και την κατασκευή μοντέλων μηχανικής μάθησης για την ανίχνευση συναισθήματος. Η εφαρμογή πρόκειται για μια διαδικτυακή εφαρμογή, η όποια αποτελείται από React όσο αφορά την διεπαφή χρήστη και από Python Flask που αποτελεί το back-end μας. Ο λόγος που επιλέξαμε την React, ένα framework της JavaScript, για τον σχεδιασμό της διεπαφής χρήστη, είναι γιατί αποτελεί μια από της πιο δημοφιλής τεχνολογίες σήμερα. Όσο για την επιλογή του Python Flask, είναι γιατί θέλαμε back-end που θα είναι σε θέση να τρέξει python βιβλιοθήκες όπως την spaCy.

Το πειραματικό μέρος, αποτελείται από δύο φάσεις. Η πρώτη φάση αφορά τον εμπλουτισμό του NER πεδίου για τον πακέτο των ελληνικών. Θα μιλήσουμε για το τρόπο που εκπαιδεύεται η spaCy σχετικά με το NER, για τον τρόπο που δημιουργήθηκαν τα δεδομένα μέσω της εφαρμογής, η επανεξέταση των αποτελεσμάτων μετά την εκπαίδευση μέσω της εφαρμογής. Η δεύτερη φάση αφορά την εκπαίδευση μοντέλων μηχανικής μάθησης για τον εντοπισμό του συναισθήματος. Εκεί θα δούμε τον τρόπο με τον όποιο συλλέξαμε δεδομένα από διαδικτυακή πλατφόρμα, οι μηχανισμοί που απαιτούνται για την δημιουργία του dataset, η εκπαίδευση των μοντέλων και τέλος τα αποτελέσματα τους.

Πριν, ξεκινήσουμε με τις φάσεις, θα ρίξουμε για αρχή μια ματιά στην βιβλιοθήκη spaCy, και τα ποιο σημαντικά της αντικείμενα και μεθόδους όπου είναι απαραίτητα για την κατανόηση του πρακτικού μέρους.

5.1 Η βιβλιοθήκη spaCy της Python

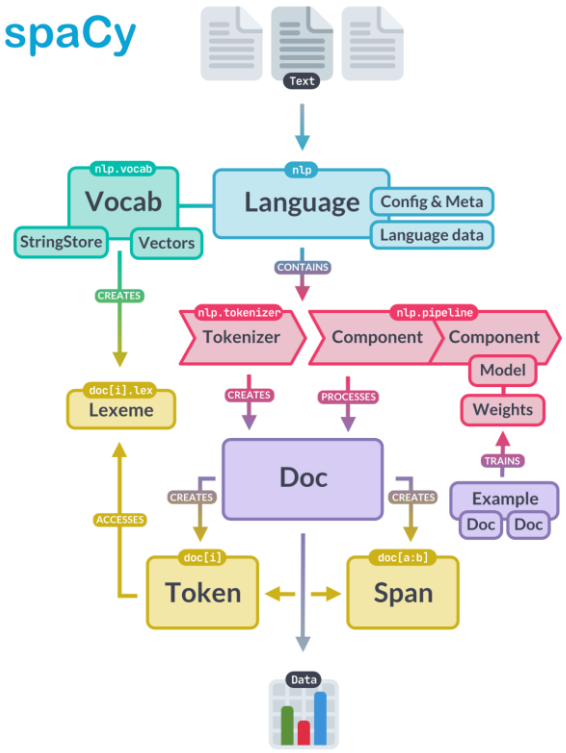
Το spaCy πρόκειται για μια βιβλιοθήκη ανοικτού κώδικα, για την Επεξεργασία Φυσική Γλώσσας (NLP) στην Python. Αρχικά σχεδιάστηκε για παραγωγική χρήση με σκοπό να βοηθήσει στην δημιουργία εφαρμογών που επεξεργάζονται και «κατανοούν» μεγάλους όγκου κειμένου. Επιπλέον, μπορεί να χρησιμοποιηθεί για τη δημιουργία συστημάτων που αφορούν την εξόρυξης πληροφοριών ή την κατανόηση φυσικής γλώσσας. [13]

5.1.1 Αρχιτεκτονική του spaCy

Το spaCy αποτελείται από τρεις βασικές δομές δεδομένων, την κλάση Language, το Vocab και το αντικείμενο Doc. Η κλάση Language είναι υπεύθυνη για την επεξεργασία ενός κειμένου και τη μετατροπή του σε αντικείμενο Doc, όπου συνήθως αποθηκεύεται ως μεταβλητή που ονομάζεται nlp. Το αντικείμενο Doc περιέχει την ακολουθία των tokens και όλους τους σχολιασμούς τους, ενώ στο Vocab συγκεντρώνονται οι συμβολοσειρές, διανύσματα λέξεων και λεξικά χαρακτηριστικά, μην επιτρέποντας την αποθήκευση πολλαπλών

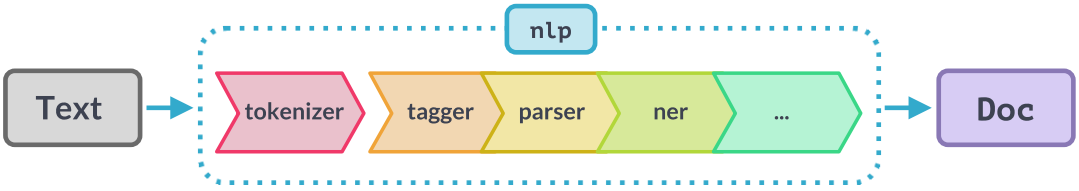
αντιγράφων αυτών των δεδομένων. Με αυτό τον τρόπο διασφαλίζεται η ύπαρξη μιας μοναδικής πηγής αλήθειας και συμβάλει στην εξοικονόμηση μνήμης. [14]

Επίσης, οι σχολιασμοί του κειμένου έχουν σχεδιαστεί για να επιτρέπουν μια ενιαία πηγή αλήθειας. Το Doc κατέχει τα δεδομένα και το Span και το Token είναι προβολές που οδηγούν σε αυτό. Το Doc κατασκευάζεται από το Tokenizer και στη συνέχεια τροποποιείται στη θέση του από τα στοιχεία του pipeline. Ενώ το αντικείμενο Language λαμβάνει ακατέργαστο κείμενο και μέσω ενός pipeline επιστρέφονται οι πληροφορίες του εγγράφου με τα απαραίτητα σχόλια. Επίσης, το αντικείμενο Language οργανώνει την εκπαίδευση και την σειριακή διαδικασία.



Εικόνα 5.1: Αρχιτεκτονικής spaCy (https://spacy.io/api)

Όσο αφορά το pipeline της επεξεργασίας αποτελείται τουλάχιστον από ένα στοιχείο όπου καλούνται στο Doc με διαδοχική σειρά. Το tokenizer είναι το πρώτο που τρέχει πριν από τα στοιχεία του pipeline. Ενώ η βιβλιοθήκη μας επιτρέπει μέσω του Language.add_pipe, να προσθέτουμε στοιχεία στο pipeline. Τέλος, το spaCy μας παρέχει μια σειρά από ενσωματωμένα στοιχεία για διαφορετικές εργασίες επεξεργασίας γλώσσας και επιτρέπει την προσθήκη προσαρμοσμένων στοιχείων.



Εικόνα 5.2: του Processing Pipeline (https://spacy.io/api)

5.1.2 Αντικείμενα και Μέθοδοι

Πριν ξεκινήσουμε, θα θέλαμε να σημειώσουμε εδώ ότι ο στόχος μας είναι ο εμπλουτισμός του NER πεδίου ενός υπάρχοντος μοντέλου και η αποθήκευση του. Στην ουσία δεν θέλουμε να δημιουργήσουμε από το μηδέν ένα νέο μοντέλο, αλλά η ενημέρωση ενός μοντέλου συγκεκριμένα αυτό των ελληνικών. Επομένως, τα παρακάτω αποτελούν κύρια στοιχεία για την εργασία μας.

Μια από σημαντικότερες και από τις βασικές μεθόδους είναι η spacy.load. Η μέθοδος δέχεται σαν είσοδο το path ή το όνομα του πακέτου που θέλουμε να φορτώσουμε στον πρόγραμμα μας το μοντέλο nlp. Για παράδειγμα, `nlp = spacy.load('el_core_news_lg')`. Το nlp που μας επιστρέφετε περιέχει αρκετούς μηχανισμούς και αντικείμενα. Κάποια από τα αντικείμενα αυτά είναι το vocab, αντικείμενο που περιλαμβάνει μηχανισμούς για το λεξικό της εκάστοτε γλώσσας του πακέτου. Επιπλέων, περιέχει όλο το αντικείμενο του pipeline, το αντικείμενο του tokenizer, τα pipe_names (μια λίστα από τα ονόματα των μηχανισμών του pipeline), τα pipe_labels (μια λίστα με ετικέτες για τον αντίστοιχο μηχανισμό του pipeline). Κάποιοι από τους βασικούς μηχανισμούς που διαθέτει είναι το get_pipe και add_pipe.

Το get_pipe, δέχεται σαν είσοδο μια συμβολοσειρά, το όποιο αντιστοιχεί σε κάποιο όνομα ενός από τα pipe names, και επιστρέφει το αντικείμενο του pipeline. Για παράδειγμα, `ner = nlp.get_pipe('ner')`, μου επιστρέφει το NER αντικείμενο του pipeline.

Το add_pipe δέχεται δύο ορίσματα της είσοδο της, ένα όπου αναφέρεται το όνομα που θέλουμε να προσθέσουμε στο pipeline και ένα για να προσδιορίσουμε αν θέλουμε την θέσει του στο pipeline, αν θέλουμε να είναι πρώτο ή τελευταίο. Για παράδειγμα, αν θέλουμε να προσθέσουμε το NER στο τέλος του pipeline, ως τελευταίος μηχανισμός, τότε θα μπορούσαμε να γράψουμε `nlp.add_pipe('ner', last=true)`.

Προηγουμένως, είδαμε την `ner = nlp.get_pipe('ner')` η οποία μου επιστρέφει το αντικείμενο ner, το όποιο με την σειρά του και αυτό διαθέτει τα δικά του αντικείμενα όπως labels (μια λίστα με όλα τα labels που είναι σε θέση να αναγνωρίσει), κ.α..

Η μέθοδος που θα μας απασχολήσει είναι αυτή του ner. add_label, όπου δέχεται μια συμβολοσειρά (string) σαν είσοδο. Με αυτό τον μηχανισμό μπορούμε να εμπλουτίσουμε το αντικείμενο μας NER με labels τα όποια δεν υπάρχουν.

Στη συνέχεια, έχουμε να δούμε το βασικότερο αντικείμενο του πακέτου spaCy και αυτό είναι το αντικείμενο του τύπου Doc. Ένα τέτοιο αντικείμενο δημιουργείται, όταν δώσουμε ως είσοδο του μοντέλου μας nlp ένα κείμενο. Για παράδειγμα, έστω έχουμε το ακόλουθων κείμενο: αν `text = "Που βρίσκεται ο Γιώργος;"`, τότε το `doc = nlp(text)`. Αυτό σημαίνει ότι το κείμενο έχει περάσει από όλες τις λειτουργίες του pipeline και το doc διαθέτει το κείμενο, με τα κατάλληλα σχόλια.

Τέλος, έχουμε τα αντικείμενα του τύπου Example. Είναι πολύ σημαντικός αυτός ο τύπος αντικείμενων, καθώς αποτελεί το βασικό τύπο για την εκπαίδευση του μοντέλου nlp. Η μέθοδος που μας ενδιαφέρει είναι αυτή του Example.from_dict, για να φέρουμε το κείμενο που είναι για εκπαίδευση σε μια μορφή που αντιλαμβάνεται ο μηχανισμός του spaCy. Αν δεν γίνει χρήση του ακόλουθου αντικείμενου, ίσως να μην έχουμε το επιθυμητό αποτέλεσμα που θέλουμε. Η μέθοδος δέχεται δύο ορίσματα σαν είσοδο του, το πρώτο πρέπει να είναι ένα αντικείμενο τύπου Doc και το δεύτερο είναι τα δεδομένα που συνοδεύουν το Doc.

Τώρα, μπορούμε να προχωρήσουμε αναλυτικότερα σχετικά με τον εμπλουτισμό του NER πεδίου για τα ελληνικά που αποτελεί το 1ο μέρος του πειράματος.

5.2 Φάση 1η: Εμπλουτισμός NER του spaCy

Το πρώτο μέρος της εργασίας αυτής, είναι ο εμπλουτισμός του NER πεδίο για την ελληνική γλώσσα του spaCy, καθώς η τελευταία έκδοση του μοντέλου είναι ελλιπής σε σχέση με το μοντέλο άλλης γλώσσας. Για να έχουμε μια εικόνα, αυτή την στιγμή οι οντότητές που αναγνωρίζει το ελληνικό μοντέλο είναι:

EVENT, GPE, LOC, ORG, PERSON, PRODUCT

Ενώ το αγγλικό μοντέλο τα πεδία που αναγνωρίζει είναι:

CARDINAL, DATE, EVENT, FAC, GPE, LANGUAGE, LAW, LOC, MONEY, NORP, OR DINAL, ORG, PERCENT, PERSON, PRODUCT, QUANTITY, TIME, WORK_OF_ART

Αυτό οφείλεται, όπως έχουμε αναφέρει και σε προηγούμενα κεφάλαια, στην έλλειψη των γλωσσικών πόρων της γλώσσας.

Αυτό που προσπαθήσαμε είναι να προσθέσουμε τα ακόλουθα πεδία, DATE, LANGUAGE, LAW, MONEY, PERCENT, QUANTITY, TIME, WORK_OF_ART. Για να γίνει ο εμπλουτισμός αυτός, χρειάστηκε να εκπαιδεύσουμε ένα ελληνικό μοντέλο, έχοντας ως βάση το ήδη υπάρχον, χρησιμοποιώντας ένα ειδικά σχεδιασμένο dataset.

5.2.1 Δημιουργία Δεδομένων Εκπαίδευση NER

Το spaCy είναι σε θέση να δεκτή ένα συγκεκριμένο μοντέλο δεδομένων, για την εκπαίδευση του NER. Ο τρόπος που αντιμετωπίζει το spaCy τα δεδομένα κατά την εκπαίδευση του NER, είναι της μορφής πίνακα τριών τιμών `(StartIndexOfText, EndIndexOfText, Label)`, όπου το Label είναι η ετικέτα της οντότητας (πχ. PERCENT), ενώ το `StartIndexOfText` και το `EndIndexOfText` δηλώνει την αρχή και το τέλος αντίστοιχα, της συμβολοσειράς όπου έχει αναγνωριστεί.

Έστω έχοντας το κείμενο, «Το 30% όσων συμμετείχαν στην έρευνα απάντησε ότι τουλάχιστον άλλο ένα μέλος της οικογενείας του αναζητά εργασία στο εξωτερικό " .», το «30%» αφορά ποσοστό (present), επομένως ο πίνακας θα έχει την παρακάτω μορφή.

```
(3, 6, 'PERCENT')
```

Αλλά ένα κείμενο μπορεί να περιλαμβάνει παραπάνω από ένα NER πεδίο, επομένως το spaCy για ένα κείμενο, περιμένει ως είσοδο το ίδιο το κείμενο και ένα dictionary με κύριο κλειδί το “entities” και value μια λίστα που θα περιέχει πολλούς πίνακες της παραπάνω μορφής (π.χ. `(StartIndexOfText, EndIndexOfText, Label)`). Επομένως, τα δεδομένα εκπαίδευση πρέπει να έχουν την ακόλουθη μορφή,

```
('Text', { 'entities': ["(StartIndexOfText1, EndIndexOfText1,
                         Label1)",
    "(StartIndexOfText2, EndIndexOfText2, Label2)"] })
```

Για παράδειγμα, έστω έχουμε το κείμενο «Αξιοσημείωτο είναι ότι θα είναι ο πρώτος άνδρας μονάρχης της Ολλανδίας μετά από 100 χρόνια», αυτό στο train data θα έχει την παρακάτω μορφή,

```
('Αξιοσημείωτο είναι ότι θα είναι ο πρώτος άνδρας μονάρχης της
            Ολλανδίας μετά από 100 χρόνια .',
 { 'entities': ["(80, 83, 'QUANTITY')", "(61, 70, 'GPE')"] })
```

Τέλος, αφού είδαμε το τρόπο με τον οποίον θα πρέπει να δομηθούν τα δεδομένα για την εκπαίδευση του NER, μπορούμε να προχωρήσουμε στην επόμενη ενότητα, όπου αφορά την εκπαίδευση του NER πεδίου.

5.2.2 Εκπαίδευση του NER

Παρακάτω περιγράφουμε την διαδικασία εκπαίδευση του NER πεδίου, με τον τρόπου που τον εφαρμόσαμε στην εφαρμογή.

Η εκπαίδευση του μοντέλου SpaCy, για την ελληνική γλώσσα, πραγματοποιήθηκε με τέτοιο τρόπο ώστε να γίνεται «επανεκπαίδευση» ενός υπάρχοντος πακέτου, δημιουργώντας ένα νέο πακέτο το όποιο και αποθηκεύουμε. Με λίγα λόγια, σε καμία περίπτωση δεν ενημερώνεται απευθείας το πακέτο, αλλά δημιουργείται ένα νέο πακέτο πανομοιότυπο με το πρωτότυπο, διατηρώντας τα πρωταρχικά στοιχεία του, αλλά έχοντας επιπλέων ενημερωμένα εκείνα τα στοιχεία που επιθυμούμε.

Η διαδικασία της εκπαίδευσης του NER, προαπαιτεί να έχουμε εγκαταστήσει το πακέτο που απευθύνεται στην ελληνική γλώσσα. Για παράδειγμα, εμείς χρησιμοποιήσαμε το “el_core_news_lg”. Με την ολοκλήρωση της εγκατάστασης του πακέτου, θα πρέπει να το φορτώσουμε στο πρόγραμμα μας, ώστε να μπορέσουμε να προχωρήσουμε στις κατάλληλες τροποποιήσεις που απαιτούνται για την εκπαίδευση του NER.

Αυτό που πρώτα καλούμαστε να πραγματοποιήσουμε είναι να εξετάσουμε το processing pipeline που διαθέτει το πακέτο. Θα πρέπει να δούμε αν το processing pipeline διαθέτει το pipe NER. Στην περίπτωση που δεν υπάρχει θα πρέπει να το προσθέσουμε ως τελευταίο στοιχείο του processing pipeline.

Στη συνέχεια, θα πρέπει να ενημερώσουμε το pipe NER με τις ετικέτες που επιθυμούμε να εμπλουτίσουμε το NER πεδίο. Τις ετικέτες μπορεί κανείς να τις δηλώσει ως στατικά δεδομένα ή δυναμικά.

Στην εργασία μας, προτάθηκε η δυναμική δήλωση των ετικετών για να είναι εξαρτώμενα από τα δεδομένα του dataset. Με αλλά λόγια, για κάθε ετικέτα – στοιχείο NER που εντοπίζεται στο dataset και δεν το διαθέτει το pipe NER, το προσθέτουμε.

Το επόμενο βήμα που θα ακολουθήσουμε, αμέσως μετά την διαμόρφωση του processing pipeline και αντίστοιχα του pipe NER, είναι η προ επεξεργασία του dataset. Αυτό που θα χρειαστούμε είναι η μετατροπή κάθε παραδείγματος του dataset μας σε μια δομή που μπορεί να αναγνωρίσει ο μηχανισμός της εκπαίδευσης του spaCy.

Με την βοήθεια συγκεκριμένου αλγορίθμου, που αναπτύχθηκε για τις ανάγκες αυτής της εργασίας, όπου ο αναγνώστης μπορεί να ανατρέξει στο «Παράρτημα Α», καταφέραμε να συγκεντρώσουμε τα δεδομένα του dataset σε μια δομή, διαχείριση από το spaCy.

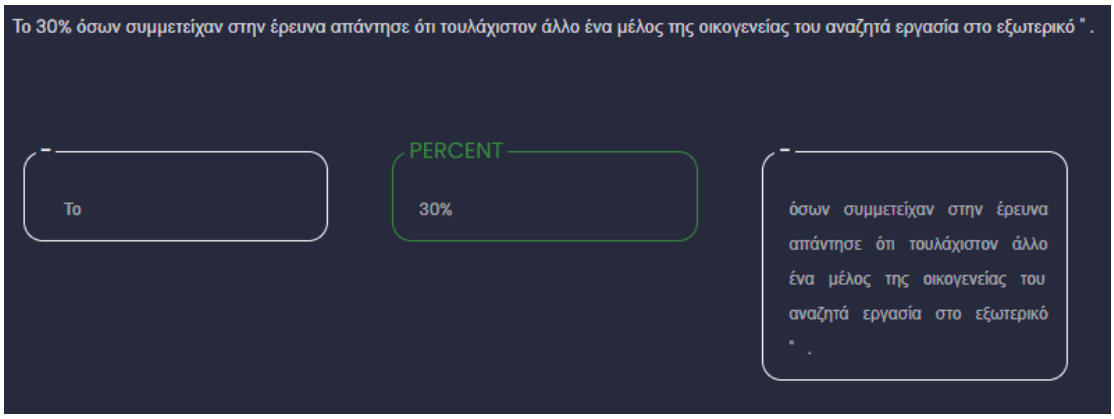
Στο τέλος, δημιουργήθηκε ο αλγόριθμο εκπαίδευσης, όπου ορίστηκε ο αριθμό των εποχών ίσο με 1000 (χίλια), προσθέτοντας την δυνατότητα ώστε σε κάθε εποχή να πραγματοποιείται αναδιάταξη του σχήματος των παραδειγμάτων και λαμβάνονται επιπλέον σε κάθε εποχή την ελάχιστη παρτίδα παραδειγμάτων με ελάχιστο μέγεθος ίσο με 50. Αφού, ολοκληρωθεί η διαδικασία της εκπαίδευσης, θα πρέπει να προχωρήσουμε στην αποθήκευση αυτού του νέου εμπλουτισμένου πακέτου spaCy.

Ο αναγνώστης μπορεί να ανατρέξει στο «Παράρτημα Α», όπου παρουσιάζεται με λεπτομέρεια οι αλγόριθμοι που εφαρμόστηκαν στην εφαρμογής.

5.2.3 Ενδεικτικά Αποτελέσματα

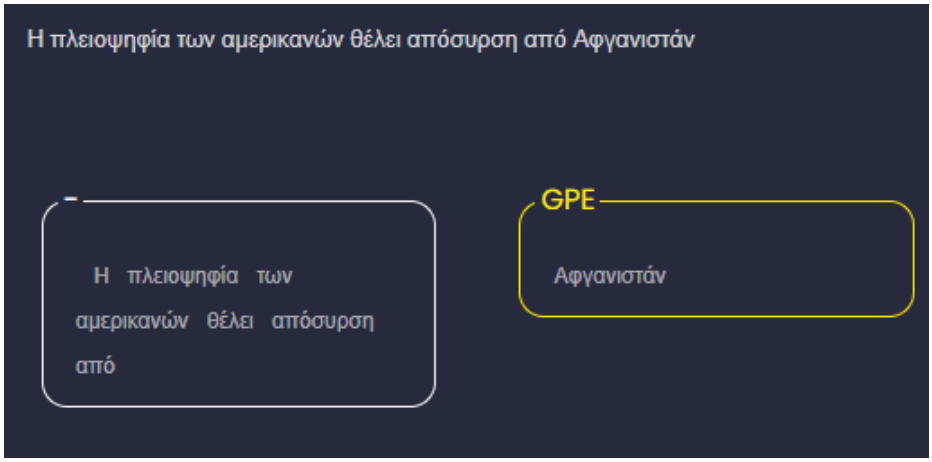
Παρακάτω μπορούμε να δούμε ενδεικτικά κάποια από τα αποτελέσματα, που έχουμε με το πιο ικανοποιητικό μοντέλο.

Παράδειγμα 1: Έχουμε την ακόλουθη πρόταση, «Το **30%** όσων συμμετείχαν στην έρευνα απάντησε ότι τουλάχιστον άλλο ένα μέλος της οικογένειας του αναζητά εργασία στο εξωτερικό», όπου στόχος του νέου πακέτου που εκπαιδεύσαμε είναι να μπορέσει να αναγνωρίσει την ονομαστική οντότητα «**30%**» με το χαρακτηριστικό του «**PERCENT (Ποσοστό)**». Στη παρακάτω εικόνα παρατηρούμε πως η εφαρμογή κατηγοριοποιείσαι σωστά το ποσοστό.



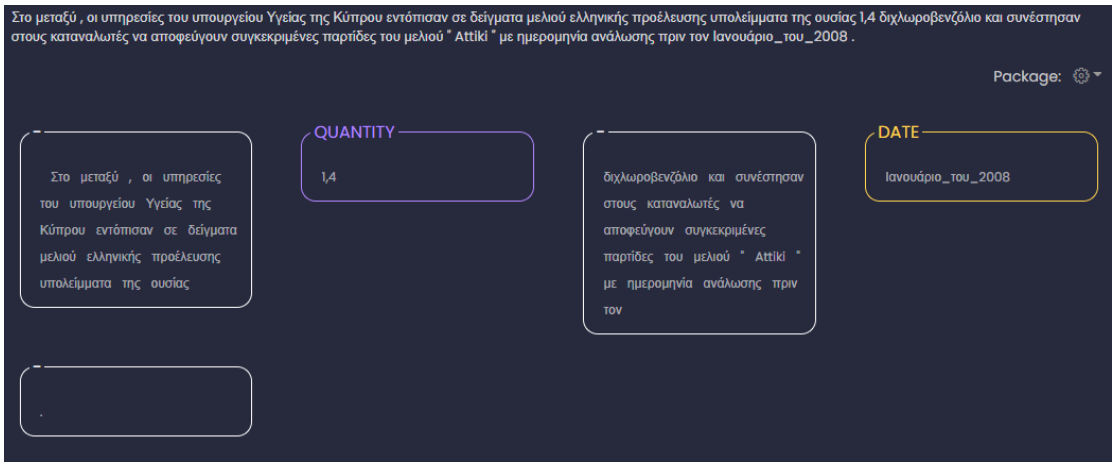
Εικόνα 5.3: Εντοπισμός NER (Παράδειγμα 1)

Παράδειγμα 2: Έχουμε την πρόταση, «Η πλειοψηφία των αμερικανών θέλει απόσυρση από **Αφγανιστάν**». Σε αυτό το παράδειγμα, στόχος του πακέτου είναι να αναγνωρίσει την οντότητα «**Αφγανιστάν**» ως «**GPE (Γεωπολιτική Οντότητα)**», όπως φαίνεται και στην παρακάτω εικόνα.



Εικόνα 5.4: Εντοπισμός NER (Παράδειγμα 2)

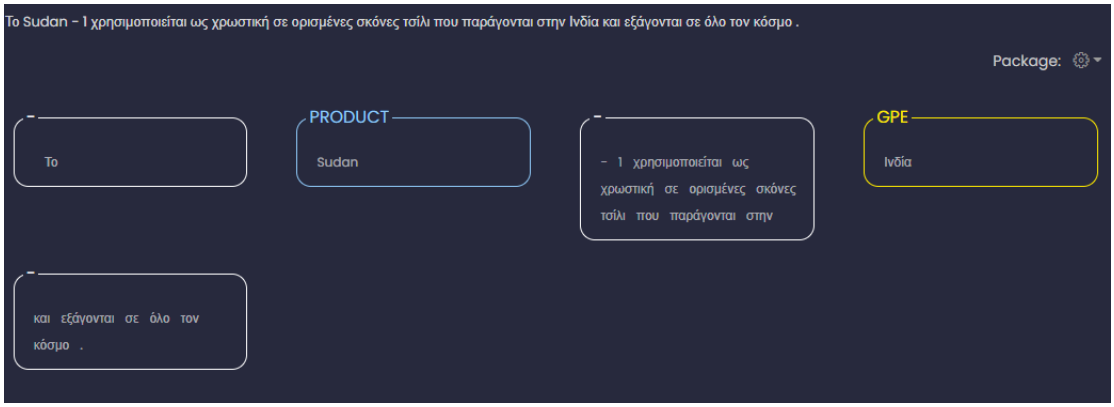
Παράδειγμα 3: Η πρόταση είναι η ακόλουθη «Στο μεταξύ , οι υπηρεσίες του υπουργείου Υγείας της **Κύπρου** εντόπισαν σε δείγματα μελιού ελληνικής προέλευσης υπολείμματα της ουσίας **1,4** διχλωροβενζόλιο και συνέστησαν στους καταναλωτές να αποφεύγουν συγκεκριμένες παρτίδες του μελιού " Attiki " με ημερομηνία ανάλωσης πριν τον **Ιανουάριο_του_2008** .». Σε αυτό το παράδειγμα θα μπορούσε ο αναγνώστης να διακρίνει 3 ονομαστικές οντότητες, την «QUANTITY (ΠΟΣΟΤΗΤΑ)» για τη οντότητα «**1,4**», την «**GPE (Γεωπολιτική Οντότητα)**» για την οντότητα «**Κύπρου**» και την «**DATE (Ημερομηνία)**» για την οντότητα «**Ιανουάριο_του_2008**». Όπως θα παρατηρήσουμε, στην παρακάτω εικόνα, το πακέτο αναγνώρισε πολύ σωστά μόνο τις δύο από τις 3 οντότητες («**QUANTITY**» και «**DATE**») ενώ την «**GPE**» δεν μπόρεσε δύστυχος να την αναγνωρίσει.



Εικόνα 5.5: Εντοπισμός NER (Παράδειγμα 3)

Παράδειγμα 4: Η πρόταση που ακολουθεί είναι «Το **Sudan - 1** χρησιμοποιείται ως χρωστική σε ορισμένες σκόνες τσίλι που παράγονται στην **Ινδία** και εξάγονται σε όλο τον κόσμο.» Στο παράδειγμά αυτό εντοπίζουμε δύο οντότητες το «**Sudan-1**», όπου πρόκειται για μια χημική

ένωση και την οντότητα «**Ινδία**». Το πακέτο μας δεν έχει εκπαιδευτεί σε τέτοιο βαθμό, ώστε να αναγνωρίσει χημικές ενώσεις, αλλά είναι σε θέσει να αναγνωρίσει ότι είναι μια οντότητα που χρησιμοποιείται. Αυτό έχει σαν αποτέλεσμα, που βλέπουμε στην παρακάτω εικόνα, να αναγνωρίσει την οντότητα «**Sudan**» ως «**PRODUCT (Προϊόν)**» και την οντότητα «**Ινδία**» ως «**GPE (Γεωπολιτική Οντότητα)**».



Εικόνα 5.6: Εντοπισμός NER (Παράδειγμα 4)

Τέλος, να σημειώσουμε σε αυτό το σημείο ότι η μη αναγνώριση και η λανθασμένη κατηγοριοποίηση μιας οντότητας οφείλεται, στο σχετικά μικρό σύνολο δεδομένων εκπαίδευσης που χρησιμοποιήθηκε. Παρ' όλα αυτά παρατηρήσαμε μια μεγάλη διαφορά σε σχέση με το πακέτο που προσφέρεται από την spaCy όσο αφορά το NER.

5.3 Φάση 2η: Μηχανική Μάθηση και Εντοπισμός Συναισθήματος

Η δεύτερη φάση του πειραματικού μέρους, αφορά τον σχεδιασμό μοντέλων μηχανικής μάθησης και την εκπαίδευση τους για τον εντοπισμό του συναισθήματος σε ελληνικά κείμενα. Η εκπαίδευση των μοντέλων της μηχανικής μάθησης διαφέρει από εκείνη που είδαμε στην προηγούμενη φάση. Στην πρώτη φάση, χρειαζόμασταν μόνο ένα πακέτο δεδομένων εκπαίδευσης, σε αυτή την φάση θα χρειαστούμε το πακέτο δεδομένων εκπαίδευσης και ένα πακέτο δεδομένων ελέγχου, για την επαλήθευση και την ολοκλήρωση της εκπαίδευσης. Σε αυτή την ενότητα, θα δούμε τον τρόπο συλλογής δεδομένων, την δημιουργία του dataset, τον τρόπο δημιουργίας των πακέτων train και test αντίστοιχα, και τέλος την διαδικασία εκπαίδευση των μοντέλων CNN και LSTM.

5.3.1 Συλλογή Δεδομένων

Για την διαδικασία τόσο της εκπαίδευσης του NER, όσο και την εκπαίδευση τον μοντέλων μηχανικής μάθησης, πραγματοποιήθηκε συλλογή δεδομένων για την δημιουργία των dataset. Η συλλογή των δεδομένων αφορούν tweets, συλλέχθηκαν από την πλατφόρμα του Tweet, με χρήση των εργαλείων που προσφέρει το Developer Platform, τα Tweet API.

Τα κριτήρια, που χρησιμοποιήθηκαν για την ανάκτηση των tweets, διαφέρουν εν μέρη μεταξύ των δύο πειραματικών φάσεων αυτής της εργασίας. Το βασικό κριτήριο για της δύο φάσεις είναι ο προσδιορισμό της γλώσσας που είναι γραμμένο το tweet, εμείς χρειαζόμαστε μόνο τα ελληνικά tweets. Στην πρώτη φάση δεν χρειαζόμαστε κάποια άλλη πληροφορία, σε αντίθεση με την $2^{η}$ φάση , που αφορούσε δεδομένα για τη συναισθηματική ανάλυση.

Στην δεύτερη φάση έπρεπε να προσδιορίσουμε αρκετά το κριτήριο, για την ανάκτηση πληροφορίας καθώς αυτό που θέλαμε να προσδιορίσουμε είναι το συναίσθημα όσο καλύτερα γινόταν. Για τον σκοπό αυτό χρησιμοποιήσαμε το συναισθηματικό λεξικό για ελληνικά. Κάθε λέξει έγινε κριτήριο για τον μηχανισμό ανάκτησης του tweet.

Όλοι η διαδικασία της ανάκτησης, έγινε σε διάρκεια ενός μήνα. Σε αυτό το διάστημα συλλέχθηκαν περίπου 20.000 tweets. Πάνω σε αυτό το σύνολο δεδομένων, πραγματοποιήθηκαν ενέργειες για την διαγραφή τυχών ευαίσθητων πληροφορίων. Αυτό πραγματοποιήθηκε με χρήση ενός απλού κειμενογράφου, και τον σχεδιασμό συγκεκριμένων κανονικών εκφράσεων (Regular expression).

Συγκεκριμένα αυτό που θέλαμε πρώτα, είναι να καθαρίσουμε τα tweets από αναφορές σε ονόματα χρηστών της πλατφόρμας, επομένως δημιουργήσαμε τις παρακάτω κανονικές εκφράσεις:

```
@[A-z]*_?([0-9]*)?[A-z0-9]*:?
```

```
@[a-Z|_|0-9]*:?
```

Στη συνέχεια να σβήσουμε τυχόν URLs που ίσως περιείχαν τα δεδομένα, αυτό το πετύχαμε με τις κανονικές εκφράσεις όπως:

```
https://t.co/[A-z]*([A-z|0-9]*)
```

```
https://t.co…
```

Επιπλέον, ο μηχανισμός για την ανάκτηση του tweet μπορεί να επιστρέφει το ίδιο αποτέλεσμα πάνω από μια φορά. Αυτό είχε ως αποτέλεσμα το σύνολο μας να περιλαμβάνει διπλό – εγγραφές. Όμως, κάτι τέτοιο δεν είναι σωστό και δεν είναι επιθυμητό για την εκπαίδευση ενός συστήματος μηχανικής μάθησης. Για το λόγω αυτό, το σύνολο πέρασε από ένα απλό αλγόριθμο, με σκοπό το καθαρισμό των διπλό-εγγραφών.

Τέλος, το αποτέλεσμα είναι να διαθέτουμε ένα σύνολο δεδομένων μεγέθους περίπου 14.900.

5.3.2 Δημιουργία του Dataset

Για την εκπαίδευση των μοντέλων μηχανικής μάθησης, χρειαζόμαστε ένα πακέτο train που αφορά το training και ένα πακέτο για το test για την επαλήθευση και ολοκλήρωση της εκπαίδευσης. Αλλά πριν την δημιουργία των δυο πακέτων, θα χρειαστεί να κάνουμε μια υπό-εργασία για την δημιουργία του dataset μας. Εμείς, αυτή την στιγμή από που καταφέραμε να κάνουμε είναι να δημιουργήσουμε ένα σύνολο δεδομένων δηλαδή κάποιος κείμενα στα ελληνικά, αυτό που πραγματικά θέλουμε και χρειαζόμαστε είναι να ταξινομηθούν τα κείμενα αυτά προς το συναίσθημα τους. Αυτό μπορεί να γίνει είτε με την χρήση ανθρώπινης παρέμβασης, δηλαδή κάποιος κριτής να σχολιάσει κάθε ένα κείμενο, είτε με αυτοματοποιημένο τρόπο. Στη περίπτωση μας ο σχολιασμό 14.900 αποτελεσμάτων με το χέρι δεν αποτελεί λύση, για αυτό και εμείς ακολουθήσαμε την δεύτερη μέθοδο, δηλαδή στην υλοποίηση και χρήση αλγορίθμου όποιος θα μας βοηθήσει με την ταξινόμηση. Παρακάτω, περιγράφουμε την διαδικασία, επιπλέων παραπέμπουμε στον αναγνώστη να ανατρέξει στα «Παράρτημα Β, Γ», για αναφορές σχετικά με τους αλγορίθμους που χρησιμοποιήθηκαν.

Σε αυτό το σημείο, να υπενθυμίσω ότι ο σκοπός μας είναι να εκπαιδεύσουμε κάποια μοντέλα μηχανικής μάθησης για τον εντοπισμό συναισθήματος σε ελληνικά κείμενα. Το συναίσθημα που θέλουμε να εντοπίσουμε είναι ο θυμός(Anger), η αηδία (Disgust), ο φόβος (fear), η ευτυχία (happiness), η θλίψη (Sadness) και η έκπληξη (Surprise). Επομένως, το σύνολο δεδομένων πρέπει να ταξινομηθεί σε αυτά τα συναισθήματα.

Ο κύριος αλγόριθμος που μας βοήθησε να προχωρήσουμε είναι αυτός που χρησιμοποιήθηκε σύμφωνα με την αναφορά [8]. Ο αλγόριθμος κάνει χρήση ενός λεξικού συναισθήματος και αυτό που απαιτεί είναι το POS tag της λέξης για να προσδιορίσει το συναίσθημα της. Στο τέλος θα προσδιοριστεί το συνολικό συναίσθημα του κειμένου, με βάση του συναισθήματος κάθε λέξης. Το POS tag κάθε λέξεις, εξαχθεί με την χρήση του spaCy του προεπιλεγμένου μοντέλου τον ελληνικών «el_core_news_lg».

Με βάση το κύριο συναίσθημα, προσπαθούμε να δημιουργήσουμε το πακέτο με dataset. Πρώτα έχουμε δημιουργήσει το folder με το όνομα “my_datasets_tweets_sentiment8” και μέσα του δύο υπό-φακέλους, ο πρώτος με το όνομα “train” και ο δεύτερος με το όνομα “test”. Σε κάθε ένα από τους δύο υπό-φακέλους θα πρέπει μέσα σε αυτούς να δούμε υπό-φακέλους ένα για κάθε συναίσθημα.

Για παράδειγμα, αν το συναίσθημα των κειμένων μας, βρίσκεται μέσα στο σύνολο E = {“Anger”, “Happiness”}, τότε και η δομή του dataset μου θα είναι:

- “my_datasets_tweets_sentiment8/train/anger”,
- “my_datasets_tweets_sentiment8/train/happiness”,

- “my_datasets_tweets_sentiment8/test/anger” και
- “my_datasets_tweets_sentiment8/test/happiness”

Αν δεν υπάρχει ο υπό-φάκελος με το όνομα του συναισθήματος, όπου μπορούμε να το ελέγξουμε με την εντολή “os.path.exists” δίνονται σαν είσοδο το path που θέλουμε, τότε δημιουργείται εκείνη την στιγμή με την βοήθεια της εντολής “os.makedirs” δίνοντας στην είσοδο του path που θέλουμε να δημιουργήσουμε.

Έστω έχουμε το παραπάνω παράδειγμα, με το σύνολο συναισθημάτων να E = {“Anger”, “Happiness”}, τότε ο έλεγχος που θα έχουμε είναι

```
    If not
os.path.exists(“my_datasets_tweets_sentiment8/train/anger”) :
    os.makedirs(“my_datasets_tweets_sentiment8/train/anger”)
```

Μετά την δημιουργία του folder με το συναίσθημα, θα πρέπει να αποθηκεύσουμε το κείμενο σε ένα αρχείο. Αυτό το βλέπου στις ακόλουθες τρεις γραμμές:

```
with open('my_datasets_tweets_sentiment8/train/' + sentiment.mainemotion + '/' +
str(countfs) + '.txt', 'a', encoding='UTF-8') as fs:
    fs.write(line)
    fs.close()
```

Όπου το sentiment.mainemotion θα μας καθορίσει το φάκελο με το συναίσθημα του κειμένου και το str(countfs) θα μας δώσει το όνομα του αρχείου όποιος είναι ένας αύξοντα αριθμός.

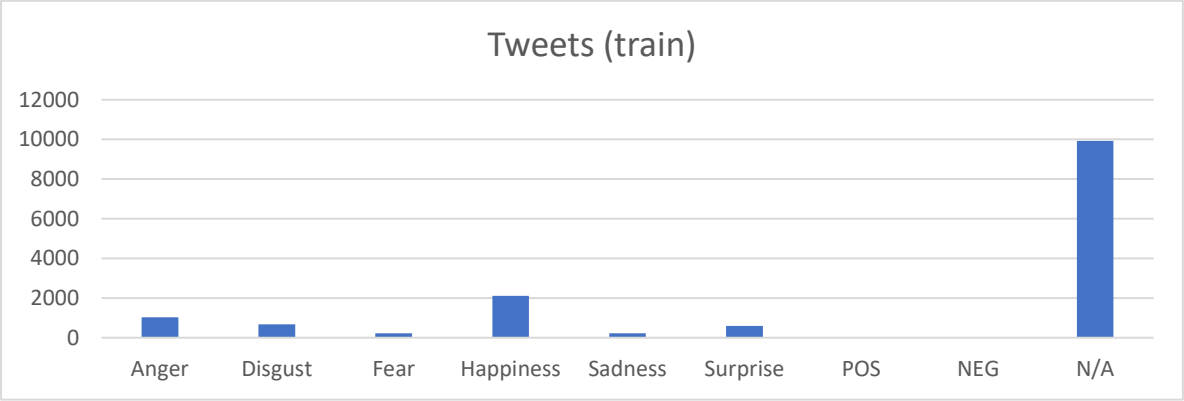
Την ίδια διαδικασία ακολουθούμε και για το folder “test”, με την διαφορά ότι έχουμε προσθέσει τον έλεγχο του counttest αν είναι ίσο με το 10 τότε μπορεί να δημιουργήσει/αποθηκεύσει το κείμενο. Με λίγα λόγια, κάθε 10 κείμενα αποθήκευσε και ένα. Αυτό συμβαίνει γιατί το test δεν θέλαμε να ξεπερνάει σε παραδείγματα αυτά του “train”.

Σε περίπτωση που το κείμενο δεν φέρει κάποιο συναίσθημα και το sentiment.mainemotion είναι ίσο με το “N/A”, είτε λόγω έλλειψης του συναισθηματικού λεξικού είτε για οποιαδήποτε εξαίρεση που δημιουργηθεί κατά την ανάλυση του συναισθήματος, δημιουργούμε το υπό-φάκελο για το train και το test, το “my_datasets_tweets_sentiment8/train/NA” και “my_datasets_tweets_sentiment8/test/NA” αντίστοιχα.

Όλη η παραπάνω διαδικασία μας έδωσε τα ακόλουθα αποτελέσματα:

- Για το train σημειώθηκαν συνολικά 14772 tweets συμπεριλαμβανόμενος και εκείνα τα παραδείγματα όπου δεν ήταν δυνατόν να ανιχνευθεί το συναίσθημα τους και βρίσκονται στην κατηγορία “N/A”.

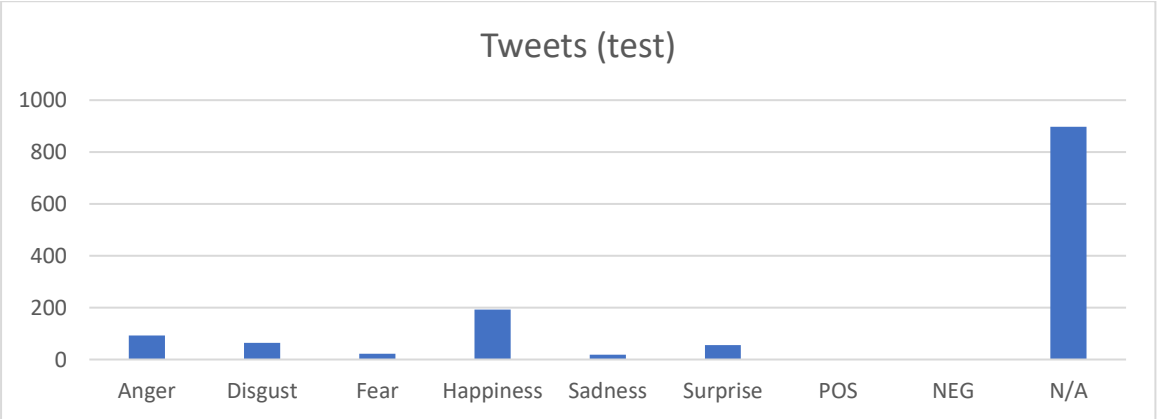
	Anger	Disgust	Fear	Happiness	Sadness	Surprise	POS	NEG	N/A
Tweets (train)	1037	679	218	2105	224	587	0	0	9922



Γράφημα 5.1: Απεικόνιση δεδομένων εκπαίδευσης (train), χωρίς κανονικοποίηση.

- Για το test σημειώθηκαν συνολικά 1342 tweets συμπεριλαμβανόμενος και εκείνα τα παραδείγματα όπου δεν ήταν δυνατόν να ανιχνευθεί το συναίσθημα τους και βρίσκονται στην κατηγορία “N/A”.

	Anger	Disgust	Fear	Happiness	Sadness	Surprise	POS	NEG	N/A
Tweets (test)	93	64	22	193	18	55	0	0	897



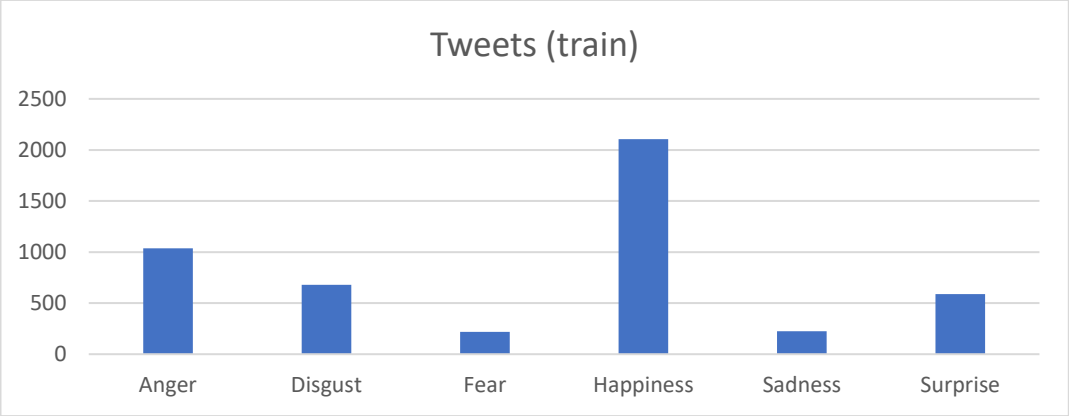
Γράφημα 5.2: Απεικόνιση δεδομένων εκπαίδευσης (test), χωρίς κανονικοποίηση.

Όπως, παρατηρούμε με τα παραπάνω στατιστικά, έχουμε μια αρκετή μεγάλη απόκλιση στην ταξινόμηση των παραδειγμάτων που βρέθηκε συναίσθημα σε σχέση με εκείνων όπου δεν εντοπίστηκε το συναίσθημα τους. Αυτό, θα μας δημιουργήσει αρκετά μεγάλο πρόβλημα στην εκπαίδευση των μοντέλων μηχανικής μάθησης, καθώς θα έχουμε το πρόβλημα της υπερ-εκπαίδευσης. Αυτό σημαίνει ότι τα μοντέλα μας θα ταξινομούν τα κείμενα ως μη αναγνωρίσιμο το συναίσθημα τους ακόμα και αν δεν ισχύει κάτι τέτοιο.

Επομένως, και εμείς αποφασίσαμε να αφαιρέσουμε από το dataset εκείνα που ταξινομήθηκαν ως N/A (not available). Αυτό, έχει σαν αποτέλεσμα την παρακάτω κατανομή.

- Για τα δεδομένα του train σημειώνονται συνολικά 4850 tweets

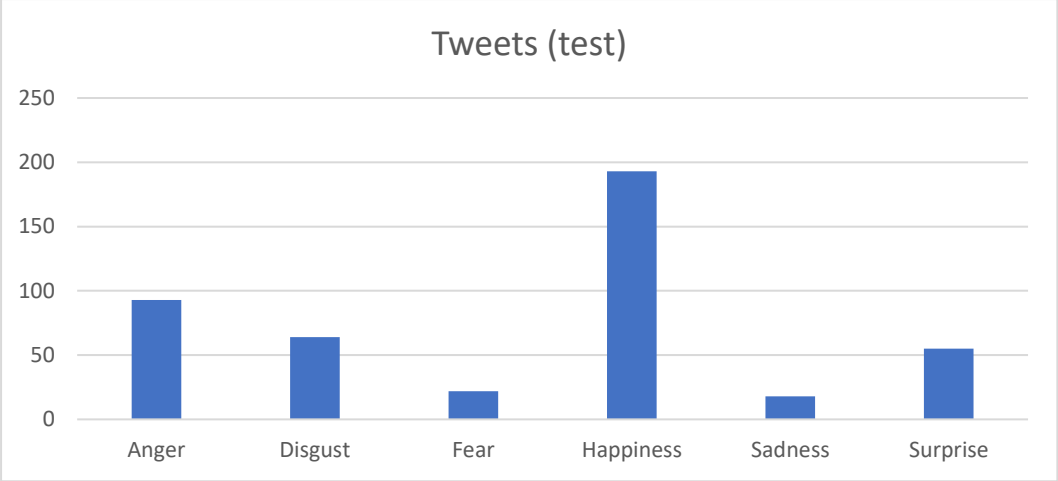
	Anger	Disgust	Fear	Happiness	Sadness	Surprise
Tweets (train)	1037	679	218	2105	224	587



Γράφημα 5.3: Απεικόνιση δεδομένων εκπαίδευσης (train), με κανονικοποίηση

- Για το test σημειώνονται 445 tweets

	Anger	Disgust	Fear	Happiness	Sadness	Surprise
Tweets (test)	93	64	22	193	18	55



Γράφημα 5.4: Απεικόνιση δεδομένων εκπαίδευσης (test), με κανονικοποίηση.

Με την ολοκλήρωση της δημιουργίας του dataset, μπορούμε να προχωρήσουμε στην διαδικασία εκπαίδευση των μοντέλων μηχανικής μάθησης, αφού πρώτα δούμε τον τρόπο με τον όποιο φορτώσαμε τα δεδομένα από το dataset τα χρησιμοποιήθηκαν για την εκπαίδευση.

5.3.3 Φόρτωση Δεδομένων από το Dataset

Σε αυτό το σημείο θα πρέπει να προσδιορίσουμε τις μεταβλητές που απαιτούνται για την διαδικασία της εκπαίδευσης, όπως για παράδειγμα train, validation, test. Οι τρεις αυτές μεταβλητές είναι οι πιο σημαντικές σε κάθε εκπαίδευση μηχανικής μάθησης. Σε κάθε μια από τις παραπάνω μεταβλητές, θα πρέπει να τους δώσουμε κάποια χαρακτηριστικά, όπως για παράδειγμα το batch_size=32, το validation_split=0.1, subset=(training || validation || test) και τέλος το seed=1337.

Στη συνέχεια θα πρέπει να κατασκευάσουμε ένα επίπεδο κανονικοποίησης. Το επίπεδο αυτό έχει στόχο κανονικοποιήση, καταχωρίσει, αντιστοιχήσει τις συμβολοσειρές σε ακέραια δεδομένα. Τα χαρακτηριστικά που δώσαμε στο επίπεδο μας είναι το max_tokens = 3000, output_mode='int' και output_sequence_length=500.

Επιπλέον, θα θέλαμε να παραπέμψουμε στον αναγνώστη να ανατρέξει στο «Παράρτημα Δ», για μια πιο λεπτομερή εικόνα των αλγόριθμων που χρησιμοποιήθηκαν σε αυτή την διαδικασία.

Τέλος, σε αυτό το σημείο ολοκληρώνουμε το κομμάτι που αφορά τα δεδομένα και τους μηχανισμούς που δημιουργήθηκαν, ώστε να είναι σε θέση να τροφοδοτήσουν τα μοντέλα μηχανικής εκπαίδευσης. Παρακάτω, θα δούμε πως δημιουργούμε μοντέλα CNN, LSTM και πως αυτά εκπαιδεύονται.

5.3.4 Εκπαίδευση μοντέλου CNN

Σε αυτό το σημείο θα θέλαμε να επισημάνουμε, πως ο αναγνώστης μπορεί να ανατρέξει στο «Παράρτημα Ε», όπου θα μπορέσει να δει μέρος του κώδικα.

Η βασική βιβλιοθήκη που χρησιμοποιήθηκε είναι αυτή του «tensorflow.keras». Σχετικά τις μεθόδους που θα χρησιμοποιηθούν μπορούμε να σας παραπέμψουμε στο διαδικτυακό εγχειρίδιο που προσφέρεται από Keras.io.

Για αρχή, δημιουργούμε τον τανυστή μας οποίος θα χρησιμοποιηθεί στο μοντέλο μας. Σαν είσοδο δίνουμε το σχήμα και τον τύπο δεδομένων ενώ στη συνέχεια, προσπαθούμε να κτίσουμε τα επίπεδα του μοντέλου μας.

Το πρώτο επίπεδο είναι αυτό της ενσωμάτωσης, όπου μετατρέπει ακέραιους θετικούς αριθμούς σε πυκνά διανύσματα σταθερού μεγέθους. Στη περίπτωση μας, έχουμε δώσει τα πρώτα 2 ορίσματα, όπου το πρώτο δηλώνει το μέγιστο αριθμό λεξιλογίου αυξημένο κατά ένα, και την διάσταση της πυκνής ενσωμάτωσης. Το αμέσως επόμενο, είναι αυτό της εγκατάλειψης, όπου ορίζει τις μονάδες εισόδου στο 0 με συχνότητα ρυθμού σε κάθε βήμα κατά τη διάρκεια του χρόνου εκπαίδευσης, κάτι που βοηθά στην αποφυγή της υπέρ-προσαρμογής.

Στη συνέχεια, προσθέτουμε 2 επίπεδα συνέλιξης 1D. Το καθένα δημιουργεί έναν πυρήνα συνέλιξης που συνελίσσεται με την είσοδο του επιπέδου σε μια ενιαία χωρική (ή χρονική) διάσταση για να παράγει ένα να τανυστή εξόδων. Τα δύο επίπεδα συνέλιξης έχουν την ίδια εικόνα. Έχουμε δηλώσει, ένα φίλτρο (π.χ. 128) πρόκειται για έναν ακέραιο αριθμό φίλτρων εξόδου στη συνέλιξη, ένα ακέραιο ή πλειάδα/λίστα ενός μόνο ακέραιους αριθμού, που καθορίζει το μήκος του παραθύρου συνέλιξης 1D, το padding ως «έγκυρο» (‘Valid’) δηλαδή, χωρίς γέμισμα, ορίσαμε την ενεργοποίηση του «Relu», και τέλος ένα ακέραιο αριθμό ή πλειάδα/λίστα που θα καθορίσει το μήκος του βήματος της συνέλιξης.

Μετά την προσθήκη των δυο παραπάνω επιπέδων, προσθέτου το επίπεδο με την λειτουργία συνολικής μέγιστης συγκέντρωσης για 1D χρονικά δεδομένα.

Ακολουθεί ένα επίπεδο Dense, όπου υλοποιεί τη λειτουργία output = activation(dot(input,kernel)+bias), όπου η ενεργοποίηση είναι η συνάρτηση ενεργοποίησής βάση στοιχείων που μεταβιβάζεται ως όρισμα ενεργοποίησης, ο πίνακας είναι ένας πίνακας βαρών που δημιουργείται από το επίπεδο. Ενώ προσθέτουμε ένα επιπλέον επίπεδο εγκατάλειψης και το τελευταίο επίπεδο που προσθέτουμε είναι ένα ακόμα Dense, έχοντας ορίσει την ως συνάρτηση ενεργοποίησης, μια συνάρτηση σιγμοειδές.

Πριν την εκπαίδευση, χρησιμοποιούμε την μέθοδο Compile,για να ρυθμίσουμε το μοντέλο μας. Στον compile, ορίζουμε την συνάρτηση απώλειας ως «binary_crossentropy», ως

συνάρτηση βελτισποίησης το «adam» και στους μετρητές την συνάρτηση «accuracy». Αμέσως μετά ορίζουμε τον ακέραιο αριθμό που δηλώνουν τις εποχές εκπαίδευσης.

Τέλος, ακολουθεί η εντολή για την εκπαίδευση του μοντέλου, όπου το τροφοδοτούμε με δεδομένα εκπαίδευσης/επικύρωσης, τον αριθμό εποχών και την μεταβλητή επανάκτησης ώστε στο τέλος να μπορεί να αποθηκευτεί.

5.3.5 Εκπαίδευση μοντέλου LSTM

Σ' αυτή την ενότητα θα δούμε την κατασκευή του μοντέλου LSTM, και την εκπαίδευση του. Επιπλέων, θα θέλαμε να επισημάνουμε πως ο αναγνώστης μπορεί να ανατρέξει στο «Παράρτημα ΣΤ», όπου έχουν επισυνάψει μέρος του κώδικα.

Μετά την εισαγωγή των βιβλιοθηκών, θα χρησιμοποιήσουμε την μέθοδο «Sequential», για να δημιουργήσουμε ένα διαδοχικό μοντέλο που είναι κατάλληλο για μια απλή στοίβα στρωμάτων όπου κάθε στρώμα θα έχει ακριβώς έναν τανυστή εισόδου και έναν τανυστή εξόδου.

Το μοντέλο διαθέτη την συνάρτηση «add», η όποια μας δίνει την δυνατότητα προσθήκης επιπέδων. Έτσι το πρώτο επίπεδο που θα τοποθετήσουμε είναι ένα επίπεδο ενσωμάτωσης, όπως είχαμε δώσει και στο παραπάνω μοντέλο.

Αμέσως μετά προσθέσαμε μέσω μια επαναληπτικής μεθόδου από 2 επίπεδα, ένα επίπεδο LSTM και ένα επίπεδο εγκατάλειψης. Το επίπεδο LSTM θα λάβει τους ακόλουθους παραμέτρους, ένα ακέραιο θετικό αριθμό που αντιστοιχούν στους διαστάσεις χώρου εξόδου (π.χ. 128)), το σχήμα εισόδου (π.χ. (None, None, 128)), η δήλωση λειτουργίας ενεργοποίησης ως υπερβολική εφαπτομένη ('tanh'), η δήλωση λειτουργίας ενεργοποίησης για το επαναλαμβανόμενο βήμα, ως σιγμοειδής, και τέλος η δήλωση για το αν θα επιστρέψουμε την τελευταία έξοδο στην ακολουθία εξόδου. Μετά από το LSTM ακολουθεί το επίπεδο εγκατάλειψης με ρυθμό 0.5.

Τα τελευταία επίδενα που προσθέτουμε είναι το Dense με συνάρτηση ενεργοποίησης την «Relu» και ένα επίπεδο εγκατάλειψης με ρυθμό 0.5. Όπως στο μοντέλο CNN έτσι και στο μοντέλο LSTM., εκτελούμε το compile και το fit για την ρύθμιση του μοντέλου και την εκπαίδευση του αντίστοιχα.

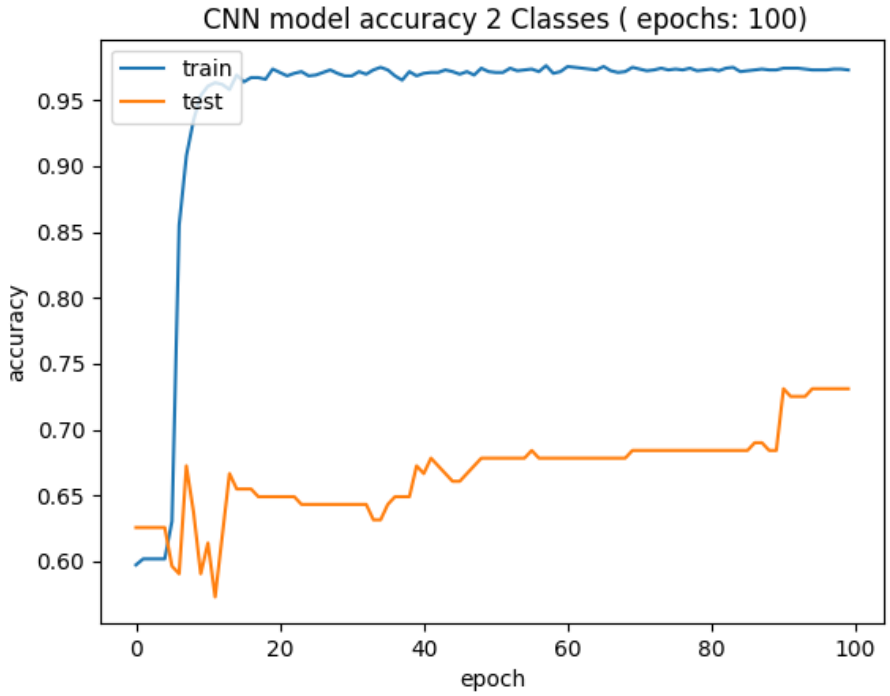
5.3.6 Ανάλυση Αποτελεσμάτων

Παρακάτω παραθέτουμε τα αποτελέσματα των δύο παραπάνω μοντέλων. Η πρώτη σειρά δοκιμών πραγματοποιήθηκαν με τις ακόλουθες συνθήκες,

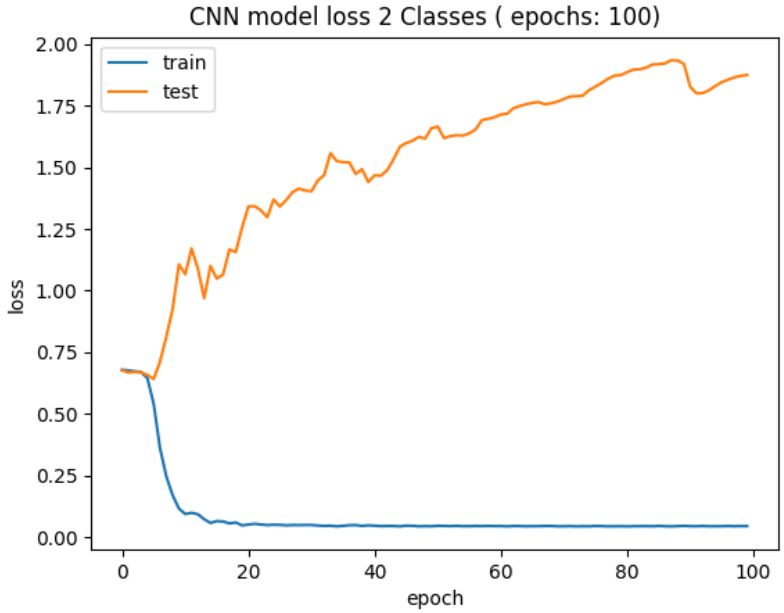
- Πακέτο εκπαίδευσης με 2 κύρια συναισθήματα: Anger (θυμό), Disgust (Αηδία)

- Αριθμός εποχών: 100

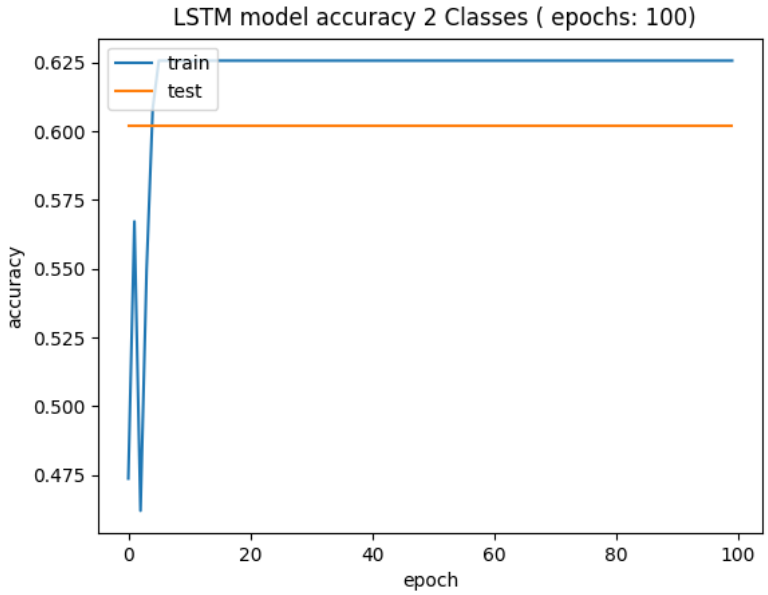
Έχοντας τις παραπάνω συνθήκες παρατηρούμε το CNN, προσεγγίζει πολύ καλύτερα τις δυο κλάσεις σε σχέση εκείνου του μοντέλου LSTM. Το CNN αποδίδει accuracy 0.7310 και loss 1.8743 ενώ το LSTM δίνει accuracy 0.6019.



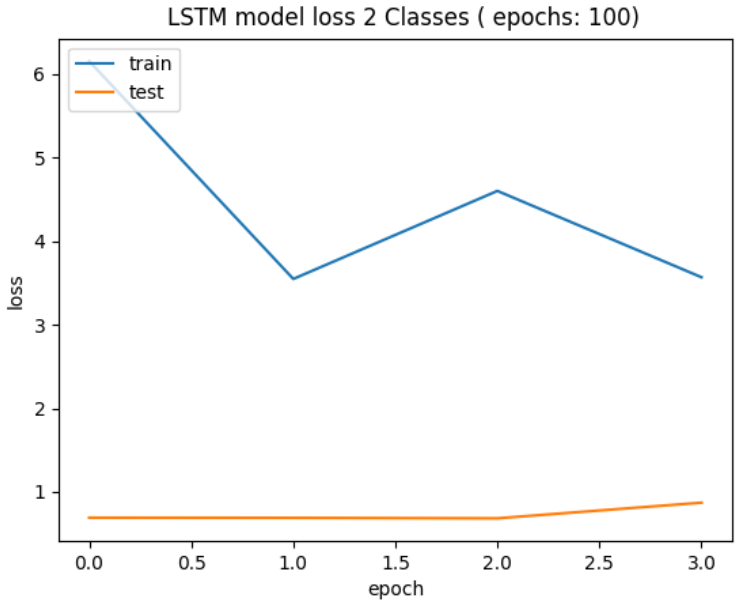
Γράφημα 5.5: Απόδοση CNN 2 Κλάσεων Συν. (100 Εποχές)



Γράφημα 5.6: Απώλεια CNN 2 Κλάσεων Συν. (100 Εποχών)



Γράφημα 5.7: Απόδοση LSTM 2 κλάσεων Συν. (100 Εποχών)



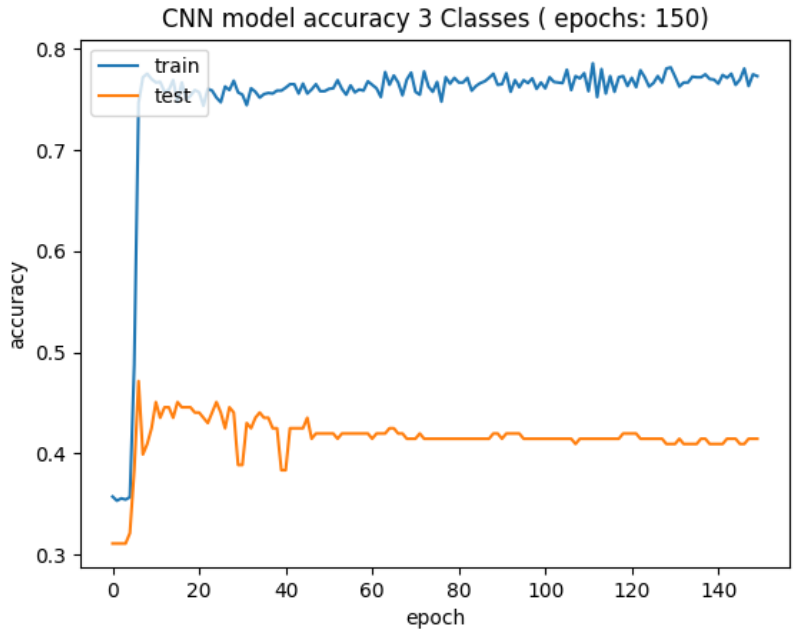
Γράφημα 5.8: Απώλεια LSTM2 κλάσεων (100 εποχών)

Στη συνέχεια, προχωρήσαμε σε αναπροσαρμογή των αρχικών συνθηκών με τις ακόλουθες,

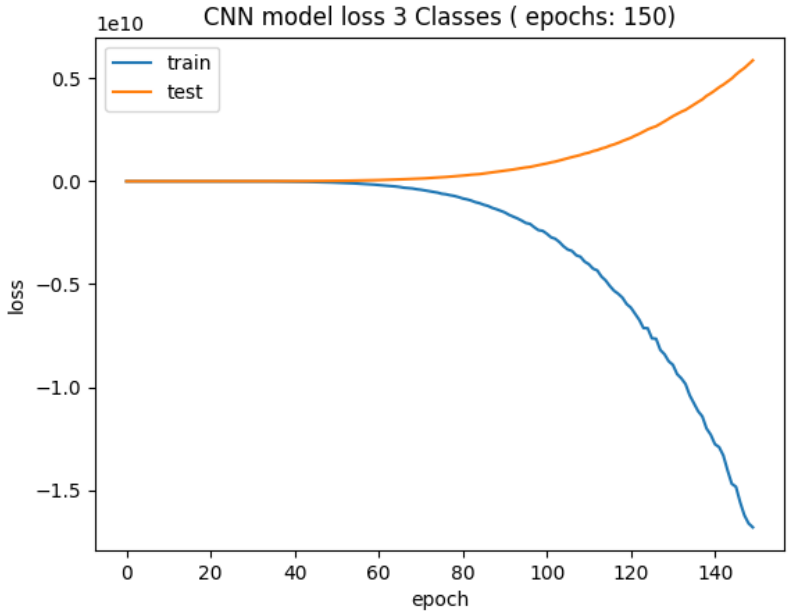
- Πακέτο εκπαίδευσης με 3 κύρια συναισθήματα: Anger (θυμό), Disgust (Αηδία), Fear (Φόβος)
- Αριθμός εποχών: 150

Με βάση των παραπάνω συνθηκών, παρατηρείται (όπως φαίνονται και στα παρακάτω διαγράμματα) μια αρκετά μεγάλη μείωση του accuracy (0.4145) στο μοντέλο CNN σε σχέση

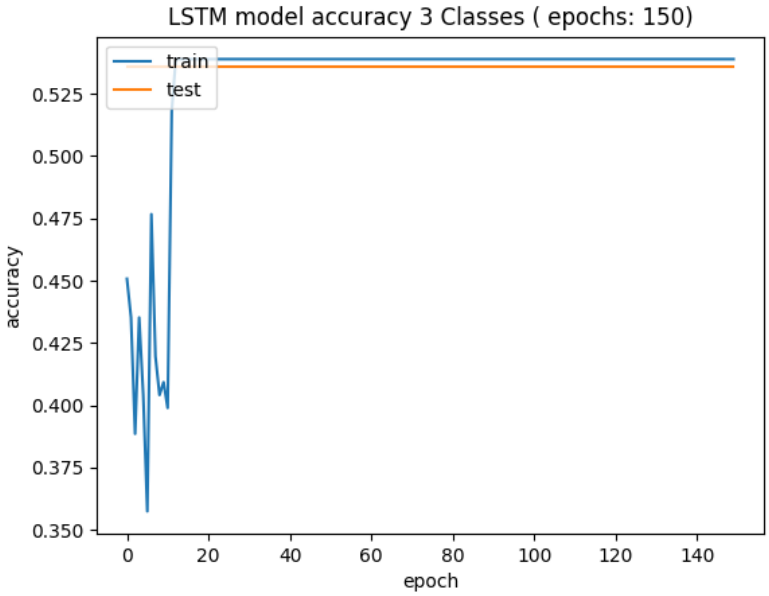
με τα προηγούμενα αποτελέσματα. Η μείωση παρατηρείται και στο LSTM μοντέλο αλλά σε μικρότερη κλίμακα, σημειώνοντας accuracy 0.5359 κάνοντας το LSTM να αποδίδει καλύτερα απ' ότι το CNN.



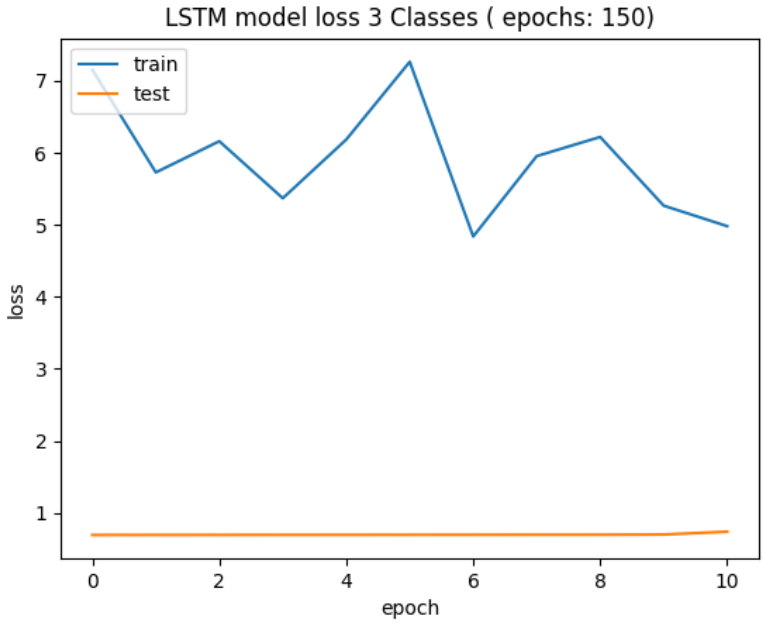
Γράφημα 5.9: Απόδοση CNN 3 κλάσεων Συν. (150 εποχές)



Γράφημα 5.10: Απώλεια CNN3 κλάσεων Συν. (150 εποχών)



Γράφημα 5.11: Απόδοση LSTM 3 κλάσεων Συν. (150 εποχών)

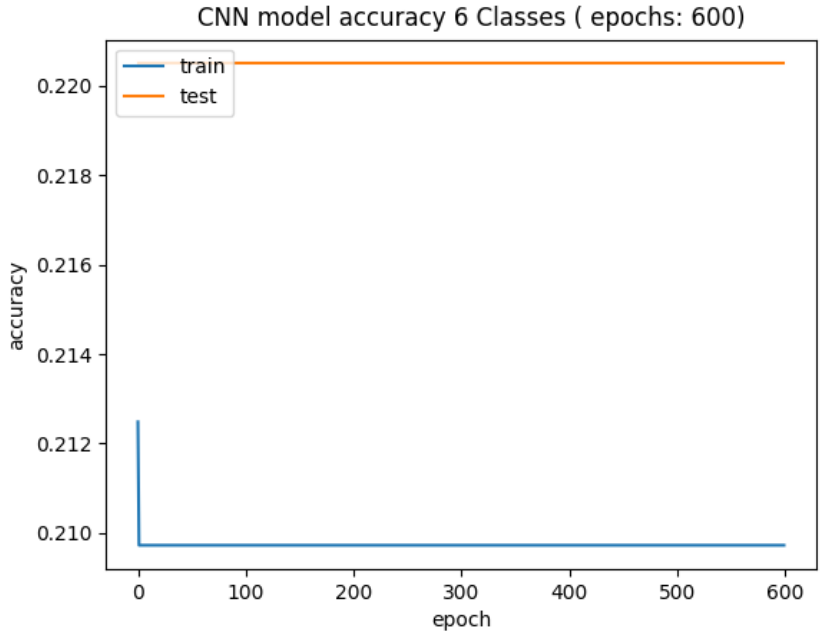


Γράφημα 5.12: Απώλεια LSTM 3 κλάσεων Συν. (150 εποχών)

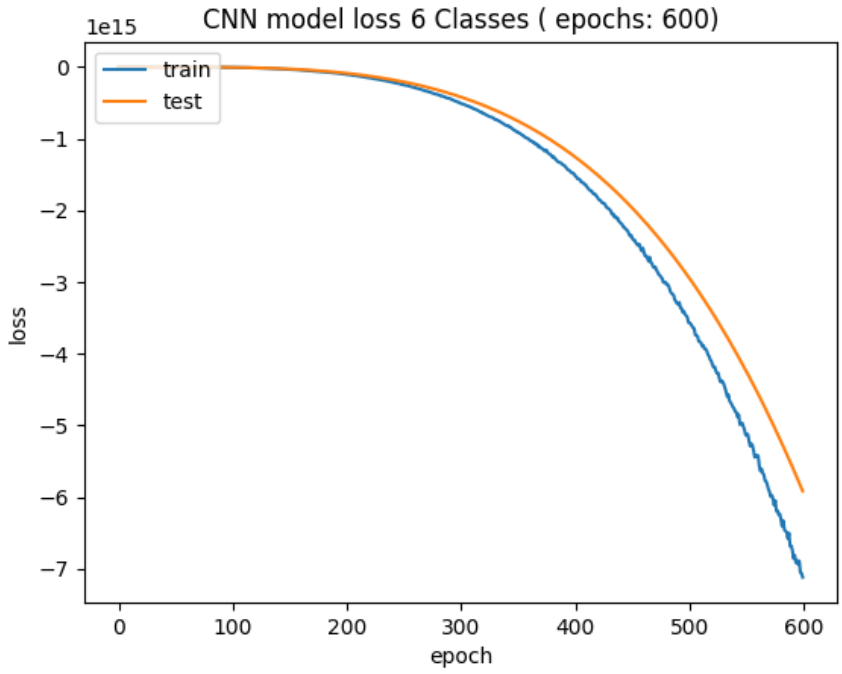
Αμέσως μεταβάλλαμε τις συνθήκες με τα ακόλουθα δεδομένα,

- Πακέτο εκπαίδευσης με 6 κύρια συναισθήματα: Anger (θυμό), Disgust (Αηδία), Fear (Φόβος), Happiness (Ευτυχία), Sadness (Θλίψη), Surprise (Έκπληξη)
- Αριθμός εποχών: 300

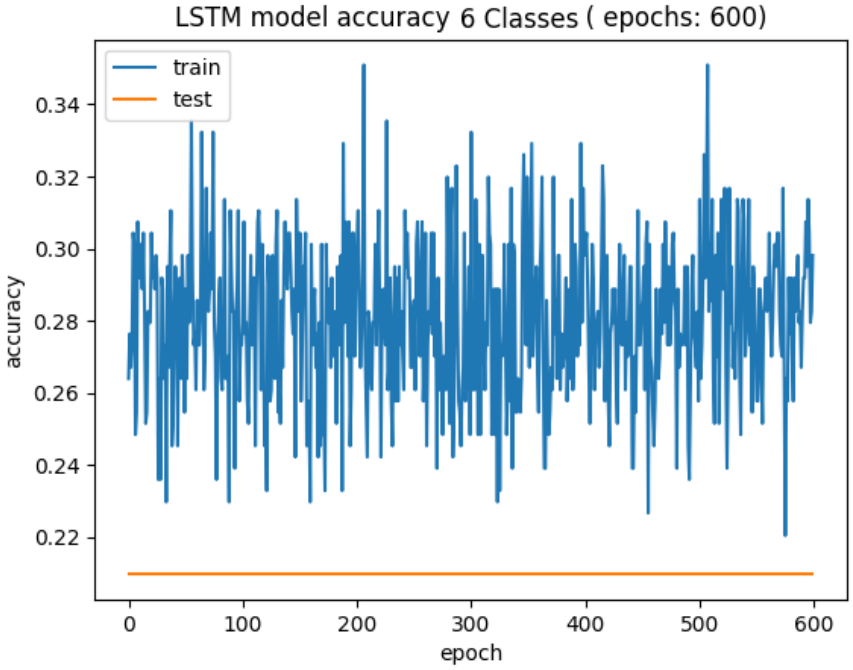
Τα αποτελέσματα που φαίνονται στα παρακάτω διαγράμματα, δείχνει το μοντέλο CNN να σημειώνεται accuracy ίσο με εκείνο του LSTM στα 0.220.



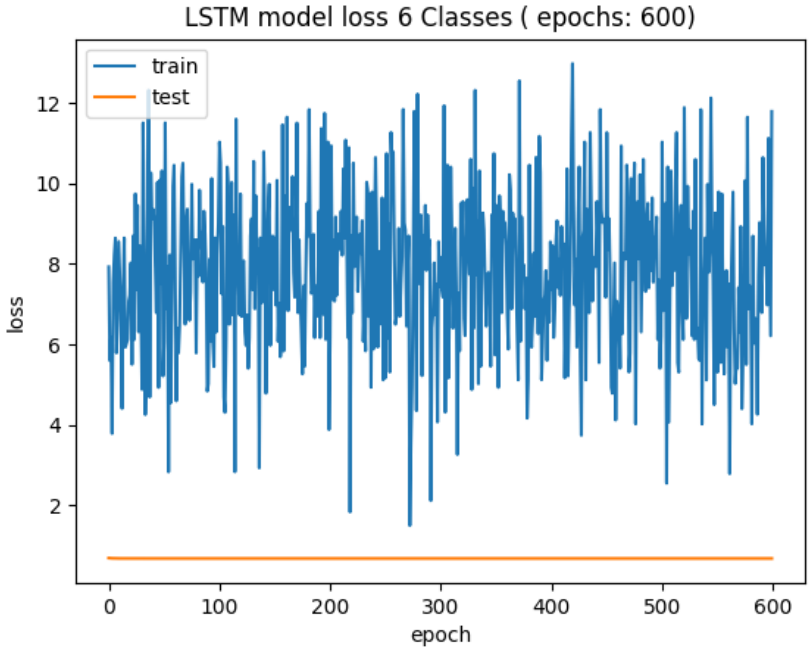
Γράφημα 5.13: Απόδοση CNN 6 κλάσεων Συν. (600 εποχές)



Γράφημα 5.14: Απώλεια CNN 6 κλάσεων Συν. (600 εποχές)



Γράφημα 5.15: Απόδοση LSTM 6 κλάσεων Συν. (600 εποχές)



Γράφημα 5.16: Απώλεια LSTM 6 κλάσεων Συν. (600 εποχές)

Τέλος, σύμφωνα με τα ακόλουθα αποτελέσματα θα παρατηρούσε κανείς πως τα μοντέλα ΄έχουν θετική απόδοση στην αναγνώριση 2 κλάσεων συναισθημάτων, ενώ και τα δύο μοντέλα η απόδοση τους μειώθηκε σημαντικά με την αύξηση των συναισθηματικών κλάσεων.

Σχετικά με την μειώσει της απόδοσης των μοντέλων, αυτό μπορεί να οφείλεται στα dataset που χρησιμοποιήθηκαν για την εκπαίδευση τους.

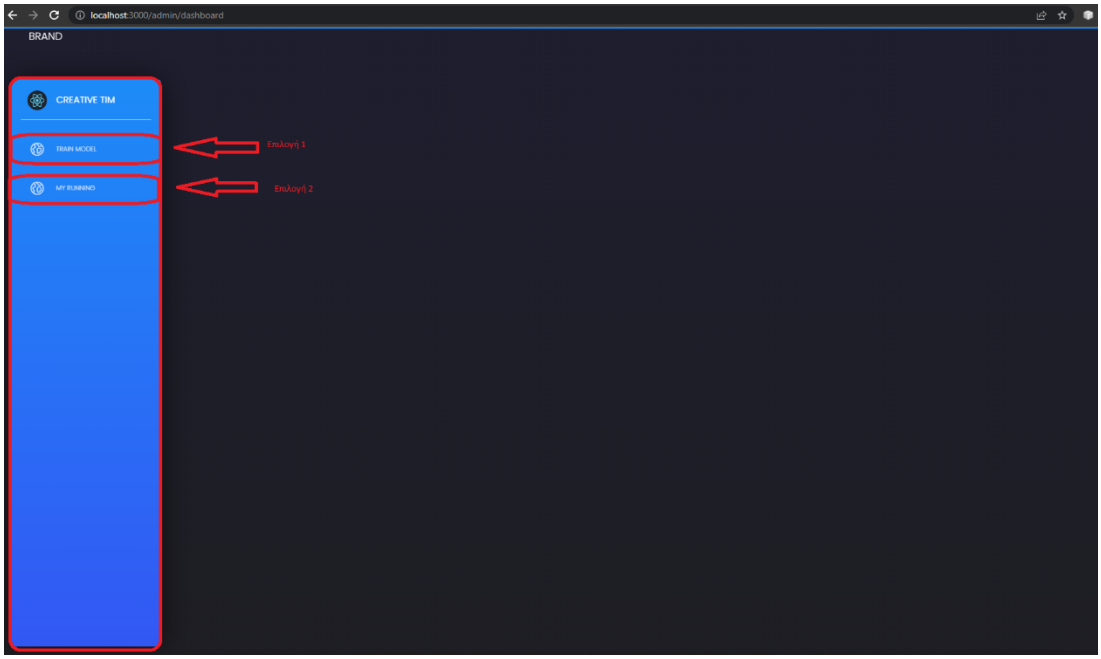
5.4 Ανάπτυξη Εφαρμογής

Για τις ανάγκες της εργασίας αυτής, σχεδιάσαμε μια εφαρμογή με βασικές λειτουργίες με σκοπό να οπτικοποιήσουμε την διαδικασία εκπαίδευσης τόσο του NER πεδίου όσο και του εμπλουτισμού ενός συναισθηματικού λεξικού. Εκτός από τις ενέργειες σχετικά με τις εκπαιδεύσεις, προσθέσαμε και μια γενική λειτουργία με σκοπό το επανέλεγχο για την σωστή λειτουργία των αποτελεσμάτων. Σε αυτή την ενότητα, αναφέρουμε τα εργαλεία και τις τεχνολογίες με τις οποίες ασχοληθήκαμε, αλλά επίσης παρουσιάζουμε ένα demo της εφαρμογής. Επιπλέον, αν ο αναγνώστης αν θέλει να εμβαθύνει σε ποιο τεχνικά κομμάτια, μπορεί να ανατρέξει στα «Παραρτήματα Z, H»

Για την ανάπτυξη της εφαρμογής, η όποια πρόκειται για μια διαδικτυακή εφαρμογή, χρησιμοποιήθηκαν δύο βασικές γλώσσες προγραμματισμού JavaScript και Python. Πιο συγκεκριμένα η εφαρμογή μας αποτελείται από το Front-End μέρος και το Back-End. Το Front-End πρόκειται για την διεπαφή χρήστη, ενώ το Back-End είναι το σύστημα που πραγματοποιεί τους υπολογισμούς. Για την επικοινωνία μεταξύ του Front-End και Back-End, θα χρησιμοποιήσουμε τον τύπο JSON, δηλαδή το request/response θα είναι σε μορφή JSON. Τα εργαλεία που χρησιμοποιήθηκαν, πρόκειται για το Visual Studio Code της Microsoft και το PyCharm της JetBrains.

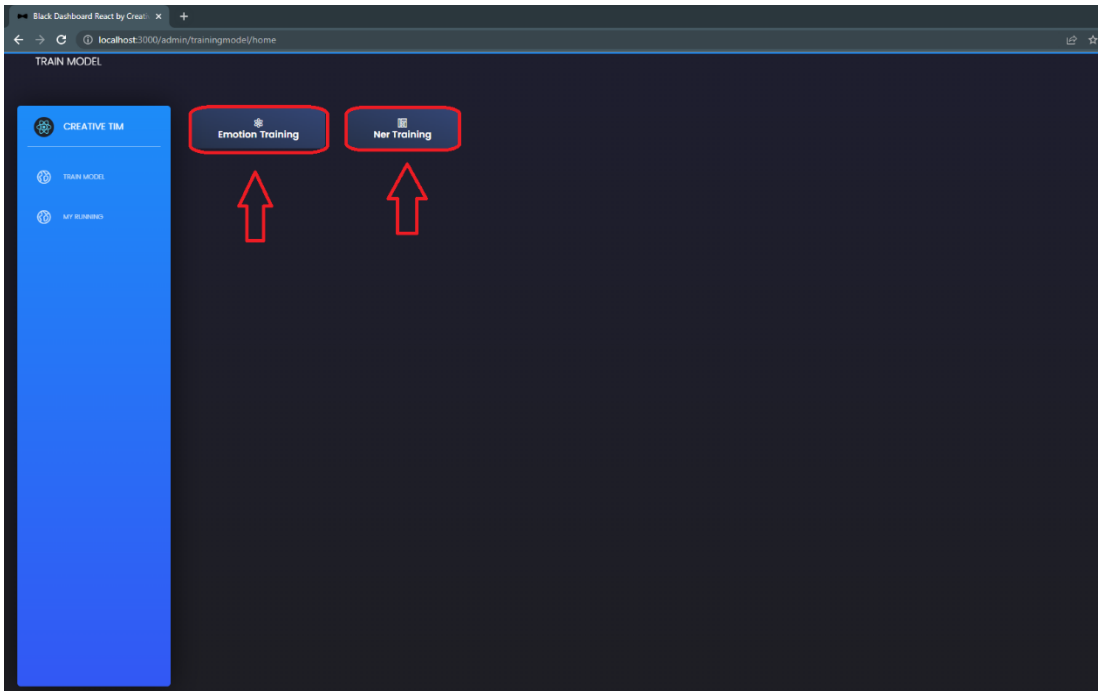
5.4.1 Εκτέλεση Εφαρμογής

Στην είσοδο της εφαρμογή ο χρήστης θα διακρίνει στην οθόνη του, στο αριστερό μέρος το κύριο μενού με τις δυνατές ενέργειες που διαθέτη αυτή η στιγμή η έκδοση. Η εφαρμογή διαθέτει δύο επιλογές την «TRAIN MODEL» και την «MY RUNNING», όπως φαίνεται στην παρακάτω εικόνα:



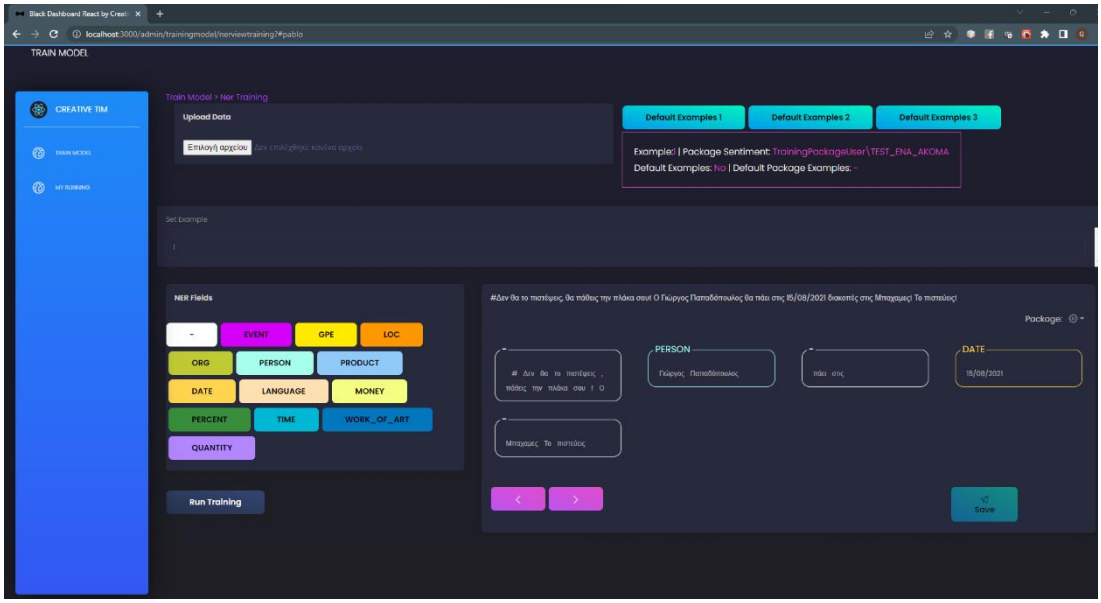
Εικόνα 5.7: Κύριο Μενού της Εφαρμογής

Στη πρώτη επιλογή «TRAIN MODEL», δύναται η δυνατότητα της εκπαίδευσης του NER πεδίου, αλλά και η δυνατότητα εμπλουτισμού ή ακόμα και δημιουργίας ενός νέου συναισθηματικού λεξικού. Επιλέγοντας, το «TRAIN MODEL» ο χρήστης διακρίνει δύο υπολειτουργίες, «Emotion Training» και «NER Training», όπως εμφανίζεται στην παρακάτω εικόνα:



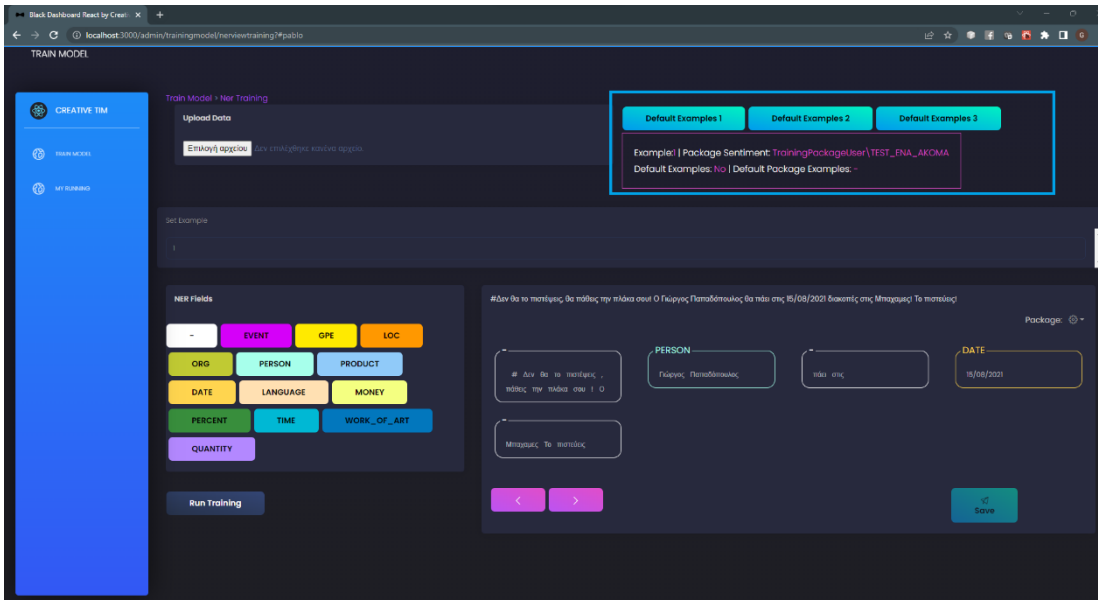
Εικόνα 5.8: Βασικές Λειτουργίες Εκπαίδευσης (NER Training & Emotion Training)

Η επιλογή «NER Training», θα μας οδηγήσει στην οθόνη, όπου δίνεται οι λειτουργίες σχετικά με την εκπαίδευση του NER πεδίου, όπως φαίνεται παρακάτω:



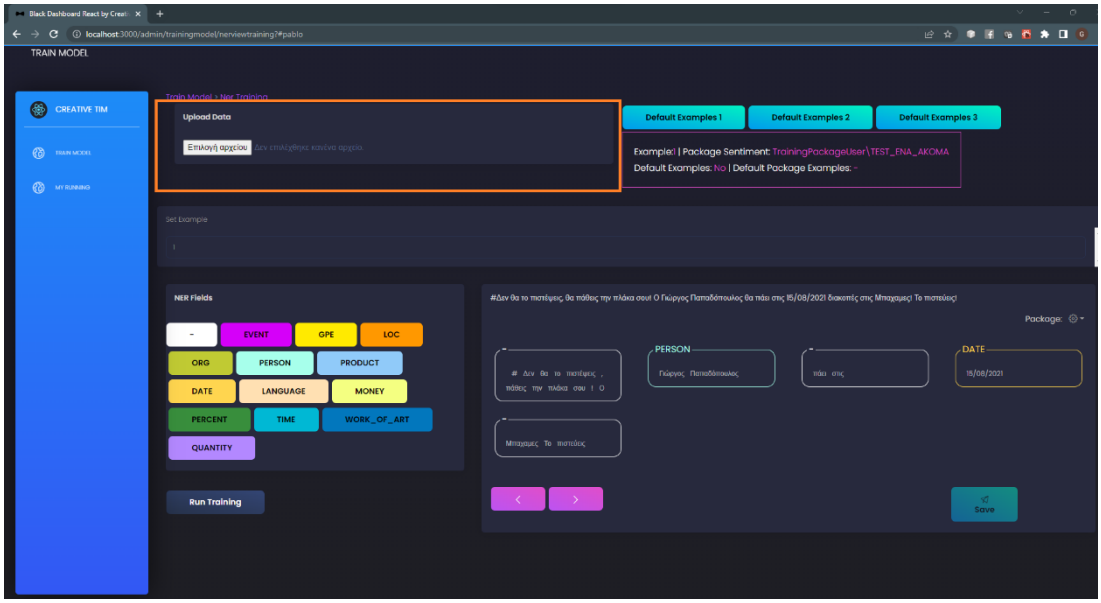
Εικόνα 5.9.1: Οθόνη εκπαίδευσης NER

Ας κάνουμε μια περιήγηση στην παραπάνω οθόνη, στο πάνω δεξιά μέρος της οθόνης (στο μπλε πλαίσιο, όπως φαίνεται στην εικόνα), εμφανίζονται βοηθητικές ενέργειες και πληροφορίες για την οθόνη. Τα 3 buttons, αφορούν τα by Default παραδείγματα που είναι διαθέσιμα προς τον χρήστη, ενώ αμέσως από κάτω από αυτά εμφανίζονται οι πληροφορίες, όπως τον αριθμό του παραδείγματος που εξετάζει, το πακέτο NER που χρησιμοποιείται, αν έχει ενεργοποιηθεί κάποιο default πακέτο με παραδείγματα και αν είναι ενεργό ποιο πακέτο είναι αυτό.



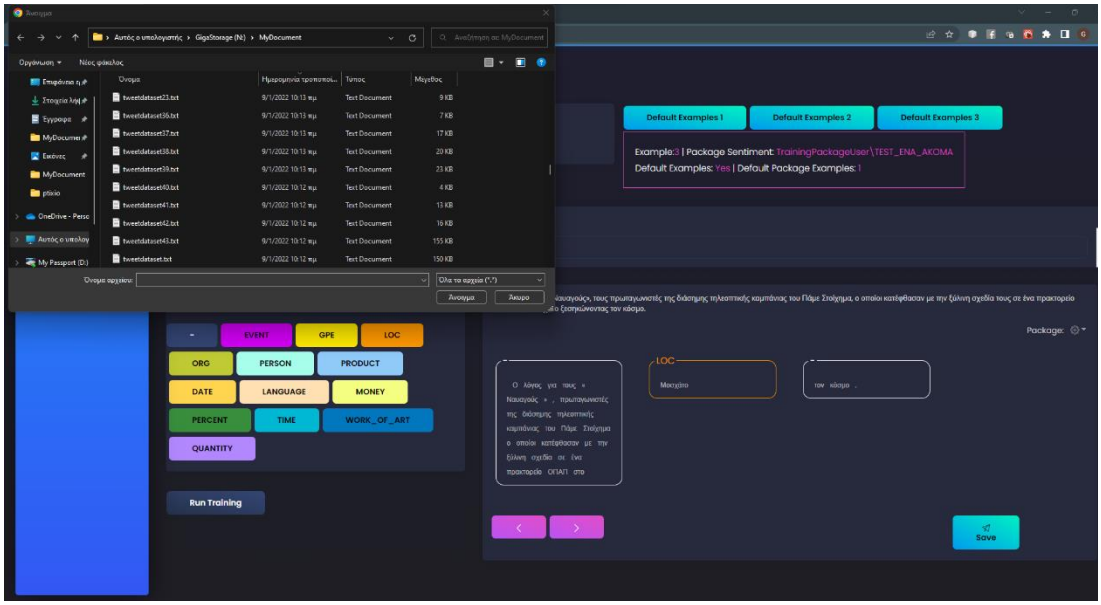
Εικόνα 5.9.2: Οθόνη εκπαίδευσης NER

Ενώ, πάνω αριστερό μέρος της οθόνης (στο πορτοκαλή πλαίσιο, όπως φαίνεται στην εικόνα), δύναται η δυνατότητα στον χρήστη να ανεβάσει το δικό του αρχείο, με τα δικά του παραδείγματα.



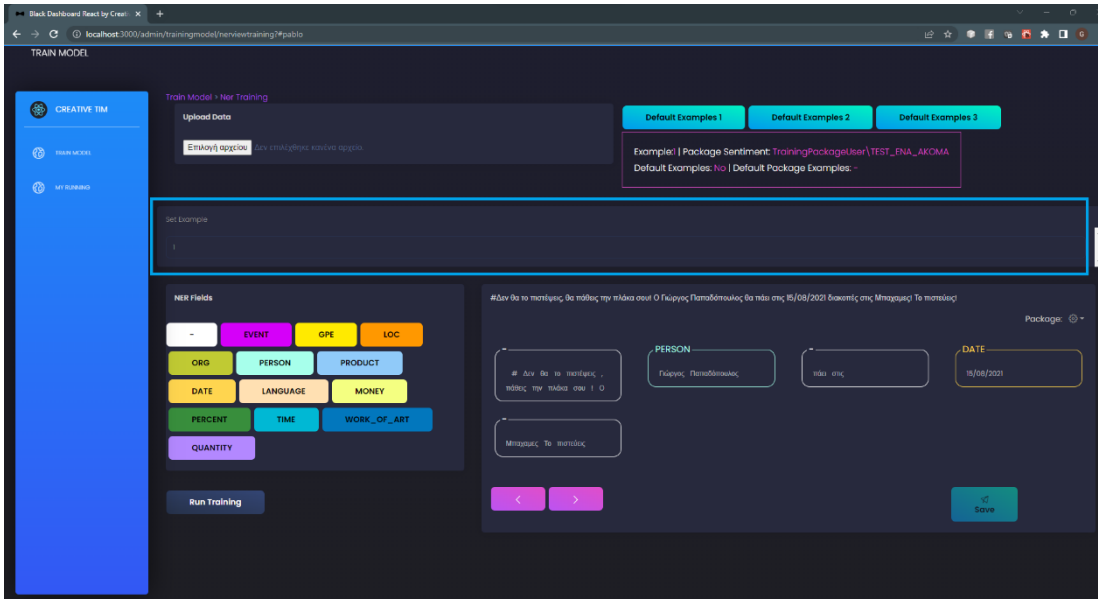
Εικόνα 5.9.3: Οθόνη εκπαίδευσης NER

Με την επιλογή του button «Επιλογή Αρχείου», εμφανίζεται το παράθυρο διαλόγου για την επιλογή του αρχείου.



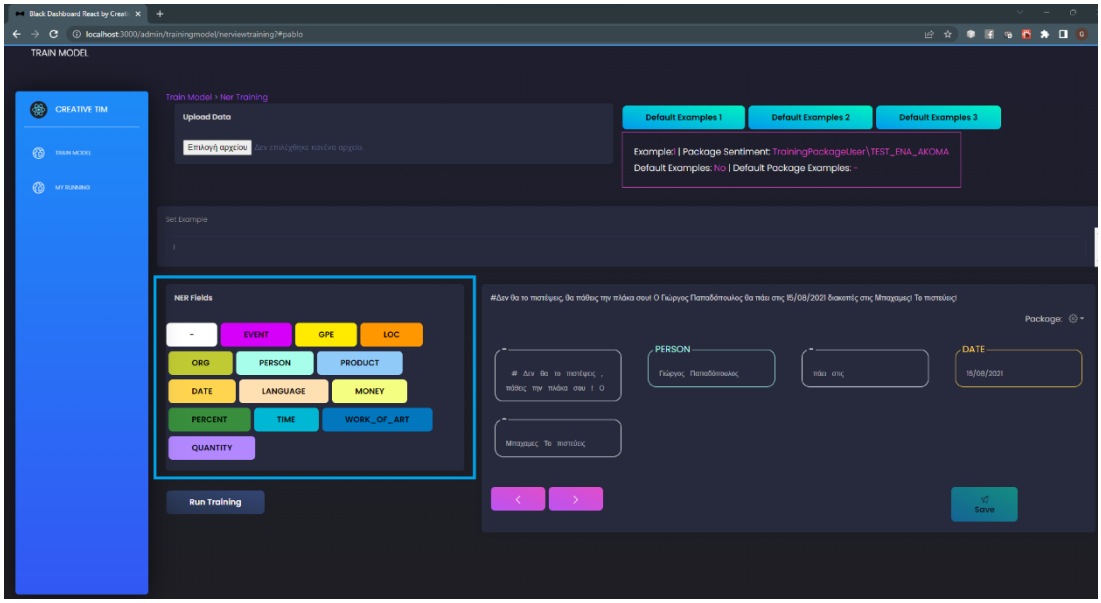
Εικόνα 5.9.4: Οθόνη εκπαίδευσης NER

Αμέσως, μετά από τα παραπάνω ακολουθεί ένα πεδίο ελεύθερου πεδίου, όπου ο χρήστη μπορεί να επιλέξει απευθείας τον αριθμό του παραδείγματος, που θέλει να εξετάσει.



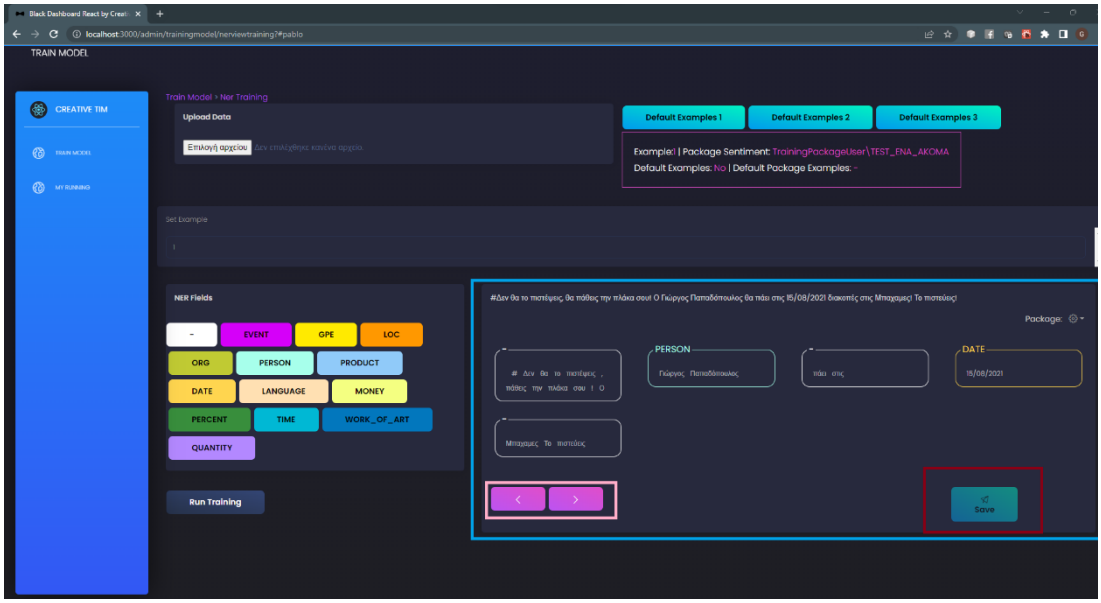
Εικόνα 5.9.5: Οθόνη εκπαίδευσης NER

Στη συνέχεια όπως φαίνεται στην παρακάτω εικόνα, έχουμε μια καρτέλα με όλες τις δυνατές κατηγορίες NER που θέλουμε να εκπαιδεύσουμε το πακέτο.



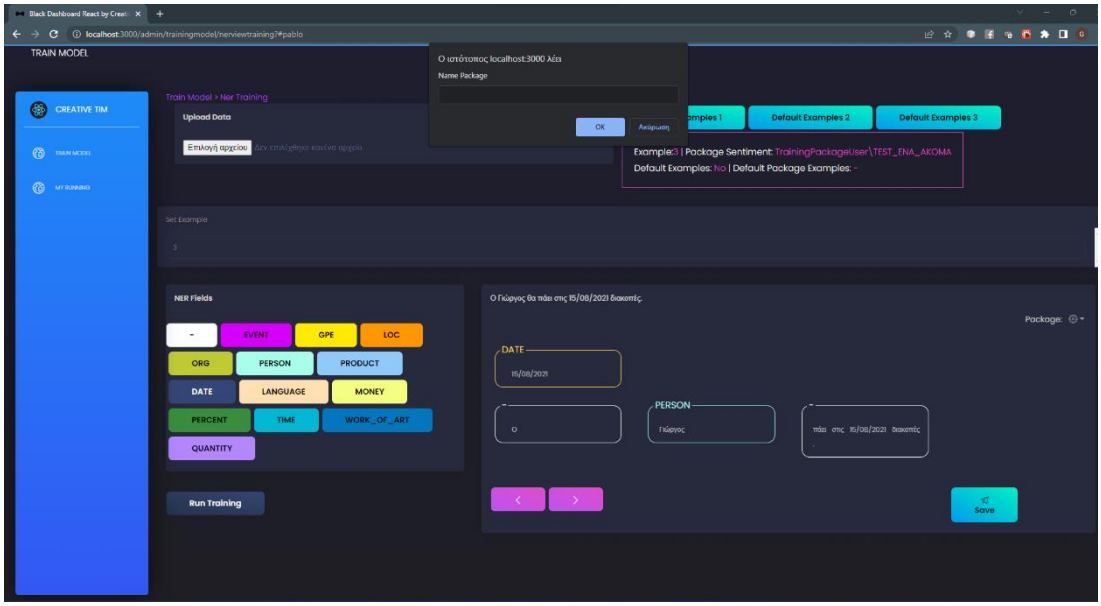
Εικόνα 5.9.6: Οθόνη εκπαίδευσης NER

Ενώ, δίπλα από το παραπάνω πλαίσιο, έχουμε την καρτέλα με τα αποτελέσματα, του παραδείγματος που εκτελείται εκείνη την στιγμή, καθώς επιπλέον δίνονται η ενέργειες μέσω των buttons «<, >» να προχωρήσει στο επόμενο ή το προηγούμενο παράδειγμα. Ένα ακόμα button που διαθέτει η καρτέλα είναι αυτό του «Save», όπου αφορά την αποθήκευση του παραδείγματος μετά από διορθώσεις που πραγματοποιήθηκαν.



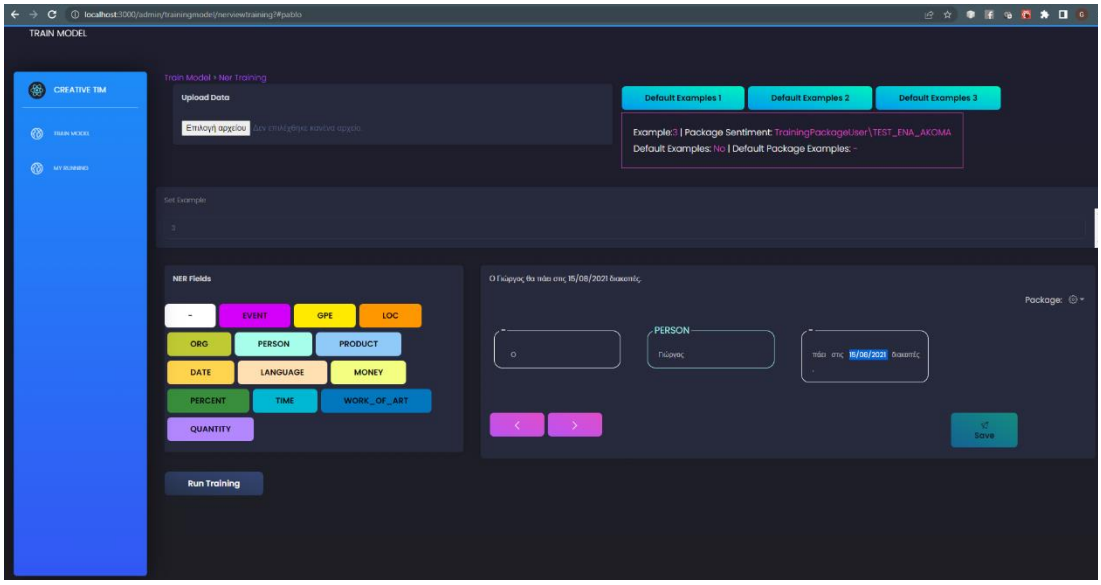
Εικόνα 5.9.7: Οθόνη εκπαίδευσης NER

Τέλος, κάτω αριστερά θα διακρίνει ο χρήστη το button «Run Training», όπου χρήστης μπορεί να εκτελέσει την έναρξη της εκπαιδεύσεις του πακέτου. Με την επιλογή, «Run Training», ανοίγει το παράθυρο διαλόγου, για να πληκτρολογήσει ο χρήστης το όνομα του πακέτου με όποιο θέλει να αποθηκεύσει, όπως φαίνεται στην παρακάτω εικόνα.



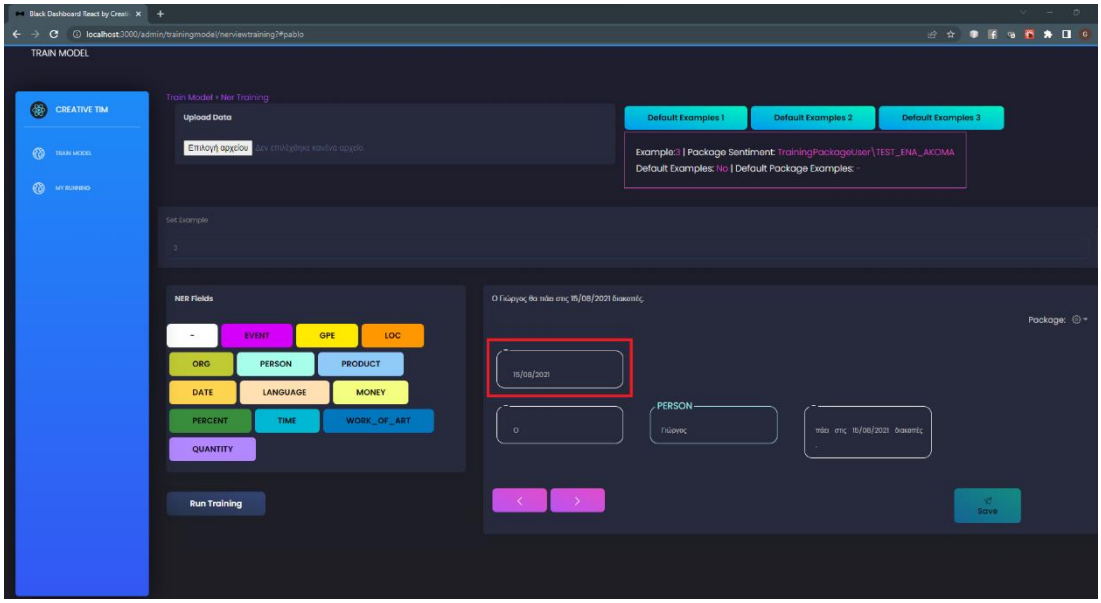
Εικόνα 5.9.8: Οθόνη εκπαίδευσης NER

Αφού είδαμε τις βασικές λειτουργίες της επιλογής «NER Training», παρακάτω θα δούμε πως ο χρήστη μπορεί να προχωρήσει σε διόρθωση του παραδείγματος. Έστω η πρόταση «Ο Γιώργος θα πάει στις 15/08/2021 διακοπές», όπως θα δούμε στην παρακάτω οθόνη το πακέτο δεν μπόρεσε να κατηγοριοποιήσει το «15/08/2021» ως “DATE”.



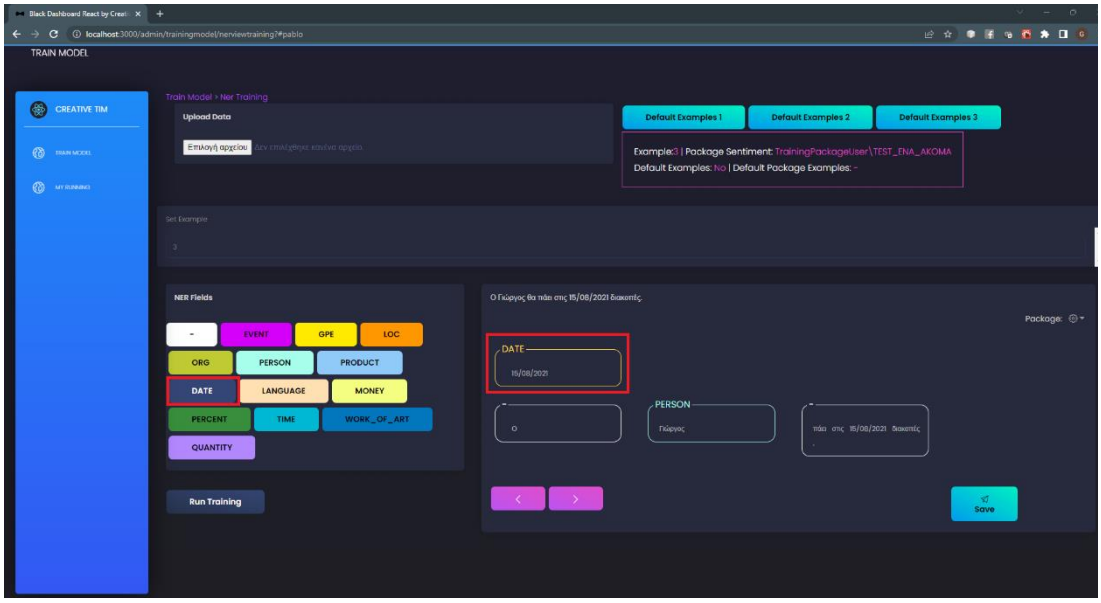
Εικόνα 5.9.9: Οθόνη εκπαίδευσης NER

Αυτό που έχει να κάνει ο χρήστης για να διορθώσει το σφάλμα, είναι να επιλέξει το λεκτικό για να δημιουργηθεί η νέα φούσκα.



Εικόνα 5.9.10: Οθόνη εκπαίδευσης NER

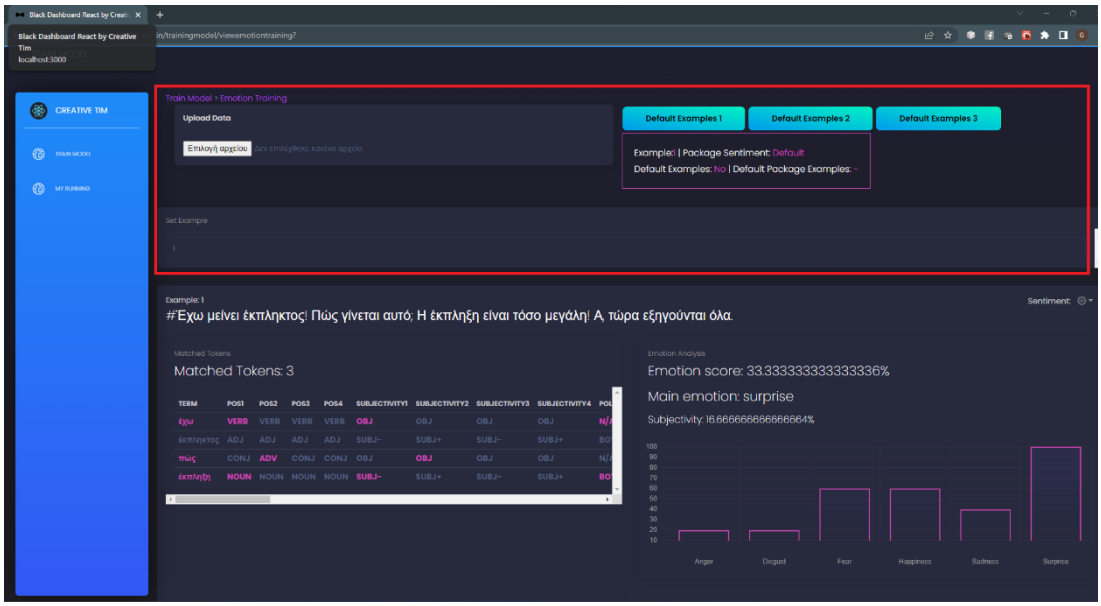
Στη συνέχεια, θα μπορεί να επιλέξει από τις κατηγορίες NER το «DATE» και την φούσκα στην όποια θέλει να ορίσει την ετικέτα, όπως φαίνεται στην παρακάτω εικόνα:



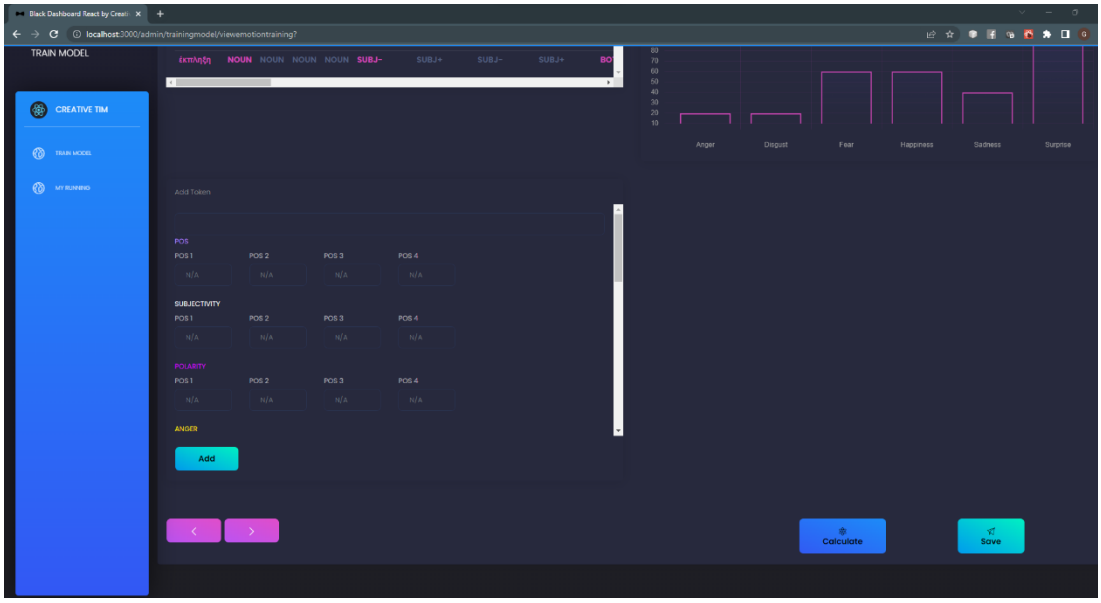
Εικόνα 5.9.11: Οθόνη εκπαίδευσης NER

Στο τέλος, θα πρέπει να επιλέξει το button «Save», για να αποθηκευτή στο αρχείο που προορίζεται για την εκπαίδευση του πακέτου spaCy.

Όσο αφορά, την λειτουργία για τον εμπλουτισμό/δημιουργία του συναισθηματικού λεξικού, ο χρήστης θα διακρίνει την παρακάτω κεντρική οθόνη.

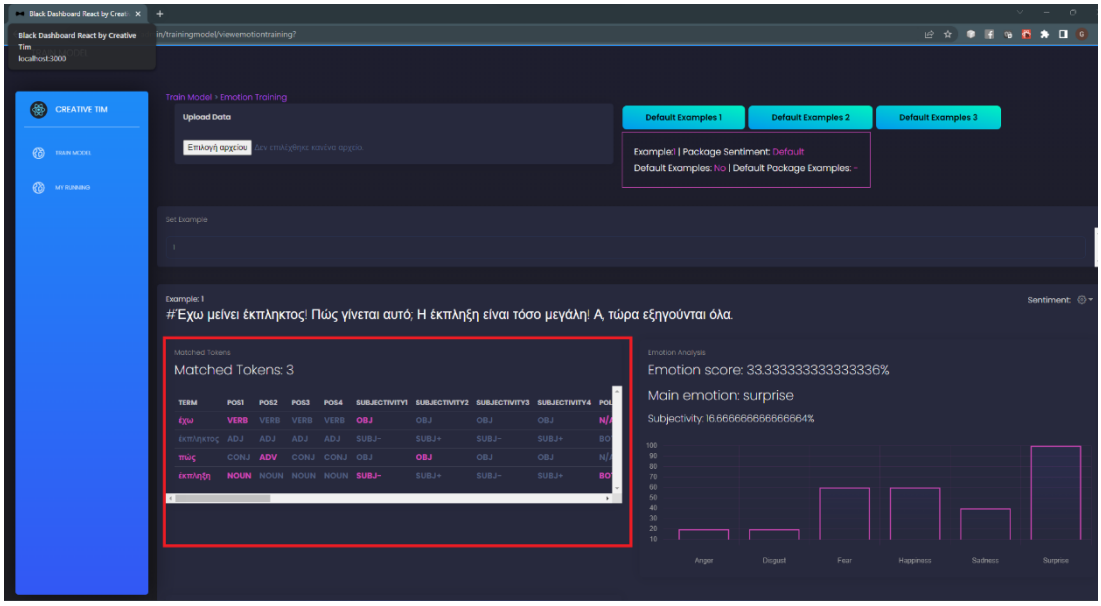


Εικόνα 5.10.1: Οθόνη εκπαίδευσης Emotions



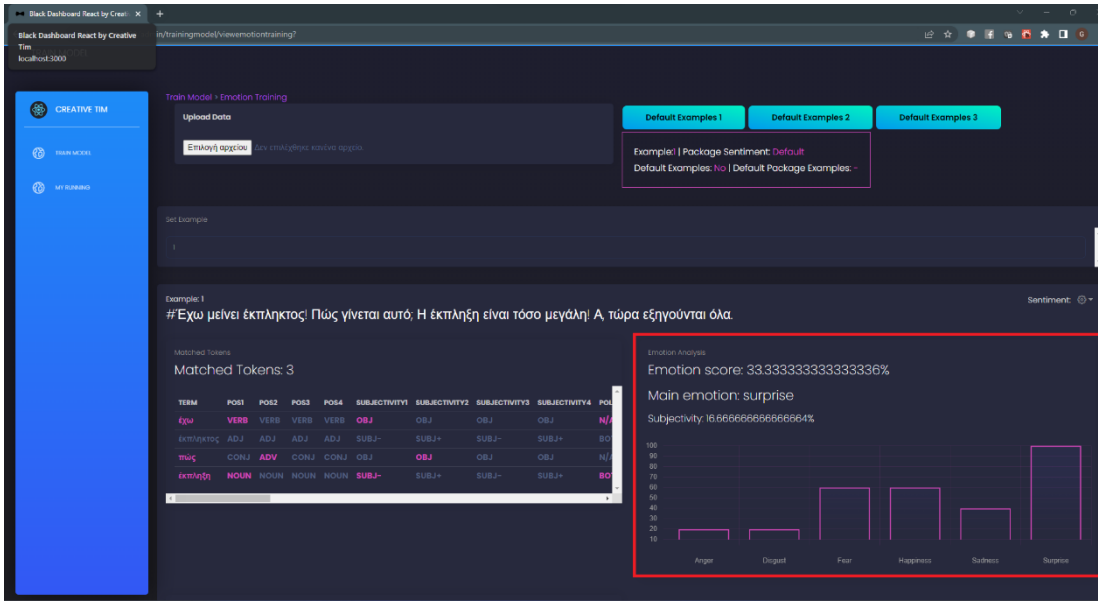
Εικόνα 5.10.2: Οθόνη εκπαίδευσης Emotions

Σε αυτή την οθόνη, έχουν αποδοθεί οι βασικές λειτουργίες (αυτές που είναι μέσα στο κόκκινο πλαίσιο) που περιγράψαμε προηγούμενος, ενώ κάτω από αυτές διαθέτουμε δυο καρτέλες. Η πρώτη καρτέλα περιέχει μια λίστα με όλα τα tokens που εντοπίστηκαν στο παράδειγμα, ενώ ως μωβ χαρακτηρίστηκαν τα tokens που συμμετέχουν στο συναίσθημα της πρότασης.



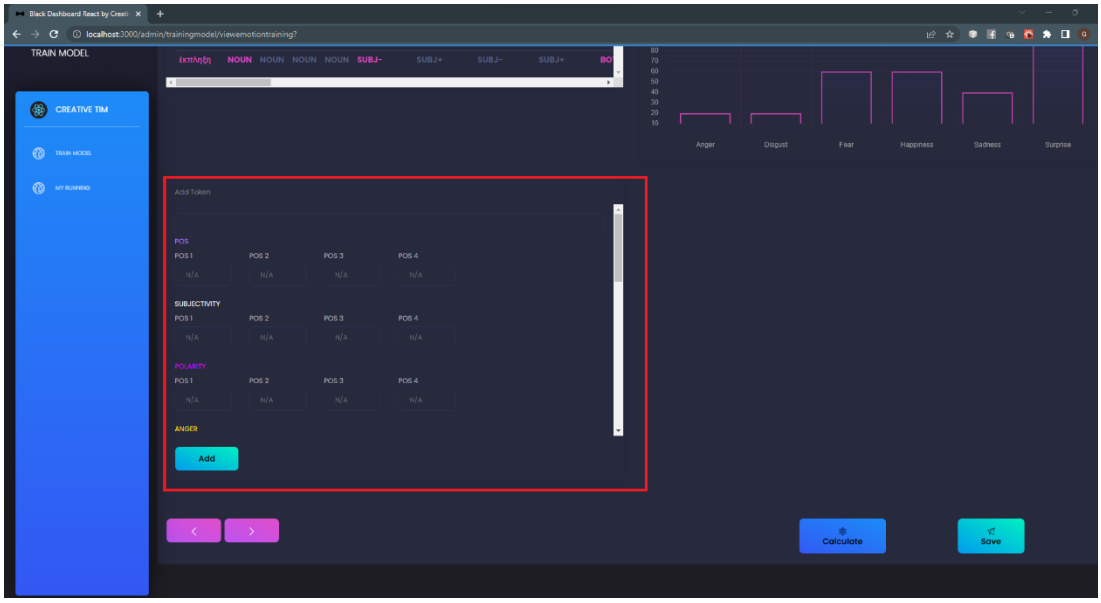
Εικόνα 5.10.3: Οθόνη εκπαίδευσης Emotions

Στην δεύτερη, καρτέλα παρατηρούμε ένα γράφημα, με τις τιμές που κυμαίνονται αν συναίσθημα, καθώς επίσης αναγράφονται πληροφορίες, όπως το κύριο συναίσθημα που εντοπίστηκε, το συνολικό ποσοστό του συναισθήματος και το ποσοστό υποκειμενικότητας.



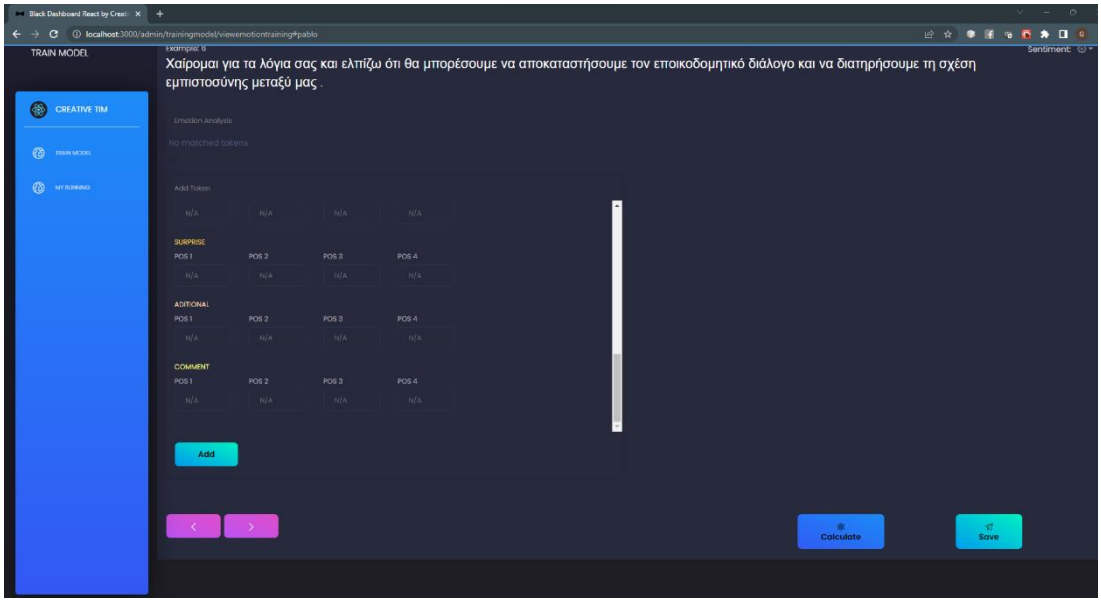
Εικόνα 5.10.4: Οθόνη εκπαίδευσης Emotions

Στην συνέχεια, ακολουθεί ένα πίνακας όπου δίνεται στον χρήστη η δυνατότητα, να προσθέσει λέξεις, όπου θεωρεί πως κανονικά θα έπρεπε να συμμετέχουν στο συναίσθημα, αλλά δεν εντοπίστηκαν είτε γιατί το συναισθηματικό λεξικό είναι έλλειπες, είτε γιατί δεν περιέχει σωστές πληροφορίες.



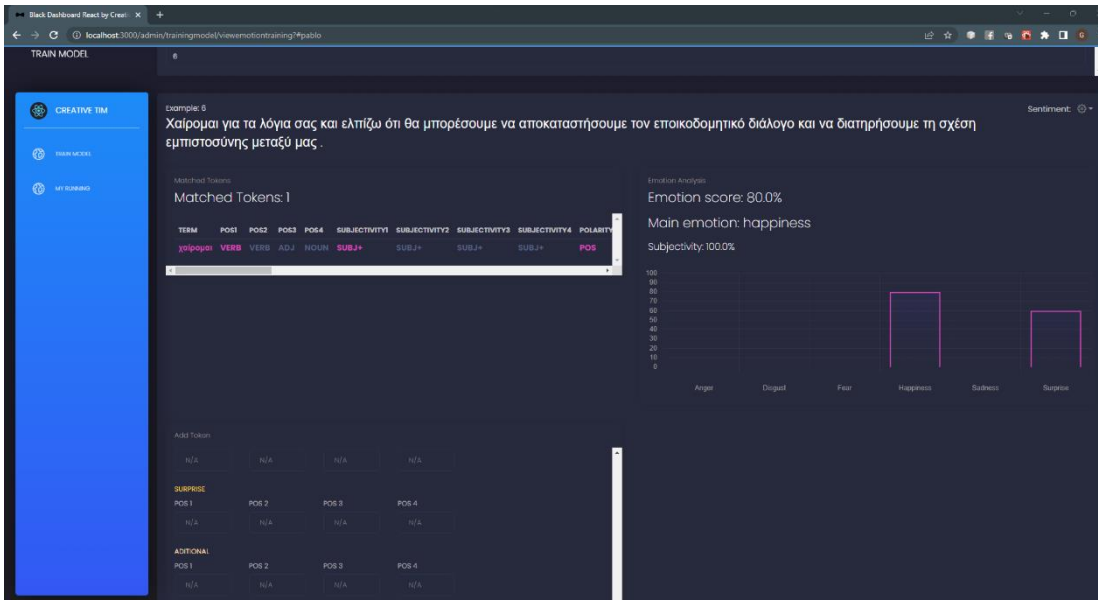
Εικόνα 5.10.5: Οθόνη εκπαίδευσης Emotions

Ας δούμε ένα παράδειγμα, όπου το συναισθηματικό λεξικό είναι ελλιπές και δεν φέρνει σωστά αποτελέσματα, όπως φαίνεται στην παρακάτω εικόνα:



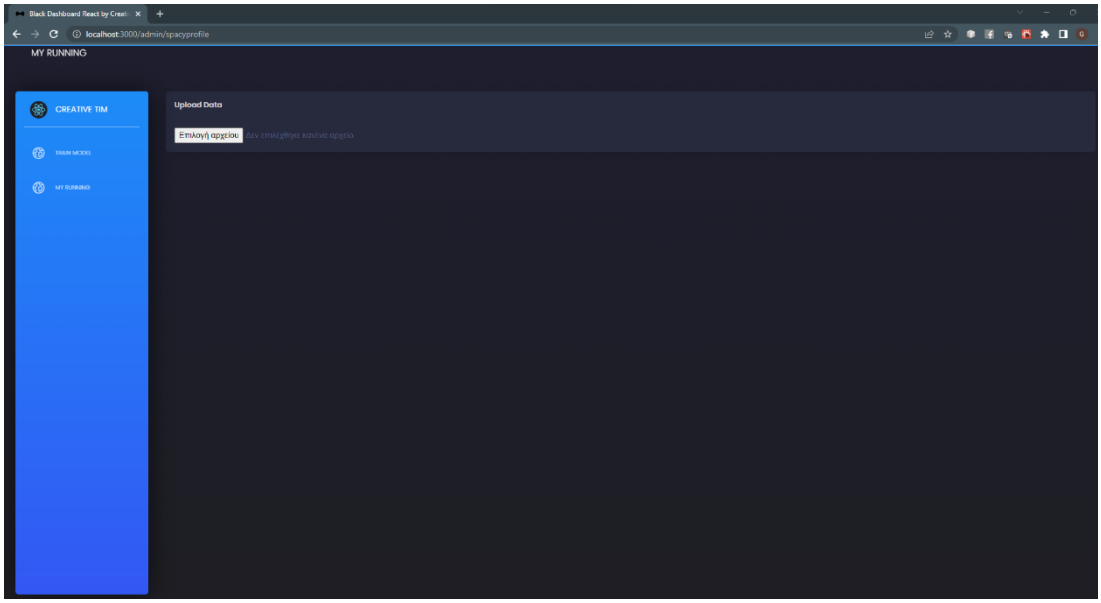
Εικόνα 5.10.6: Οθόνη εκπαίδευσης Emotions

Ο χρήστη μπορεί να προσθέσει την λέξη «Χαίρομαι», με τα στοιχεία που θεωρεί ότι πρέπει να οριστεί, με την επιλογή «Add» προστίθεται στο πίνακα με τα «Matched Tokens» και επιλέγοντας το button «Calculate» πραγματοποιείται εκ νέου υπολογισμός του συναισθήματος, όπως φαίνεται στην παρακάτω εικόνα.



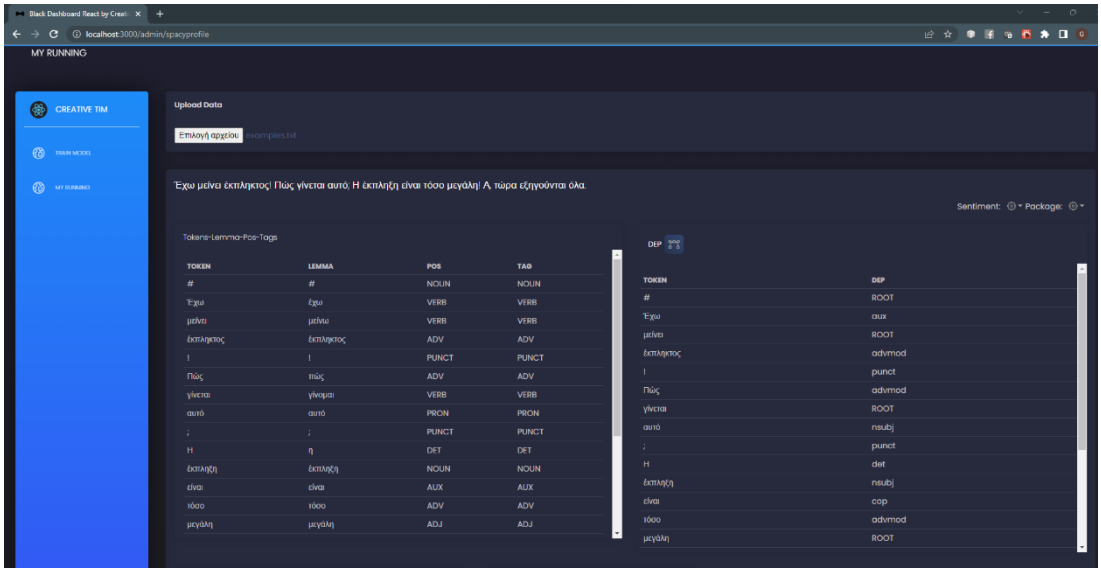
Εικόνα 5.10.7: Οθόνη εκπαίδευσης Emotions

Τέλος, έχουμε και την επιλογή «My Running», όπου ο χρήστης μπορεί να φορτώσει το αρχείο προς εξέταση και να εμφανίσει τα σχετικά αποτελέσματα.

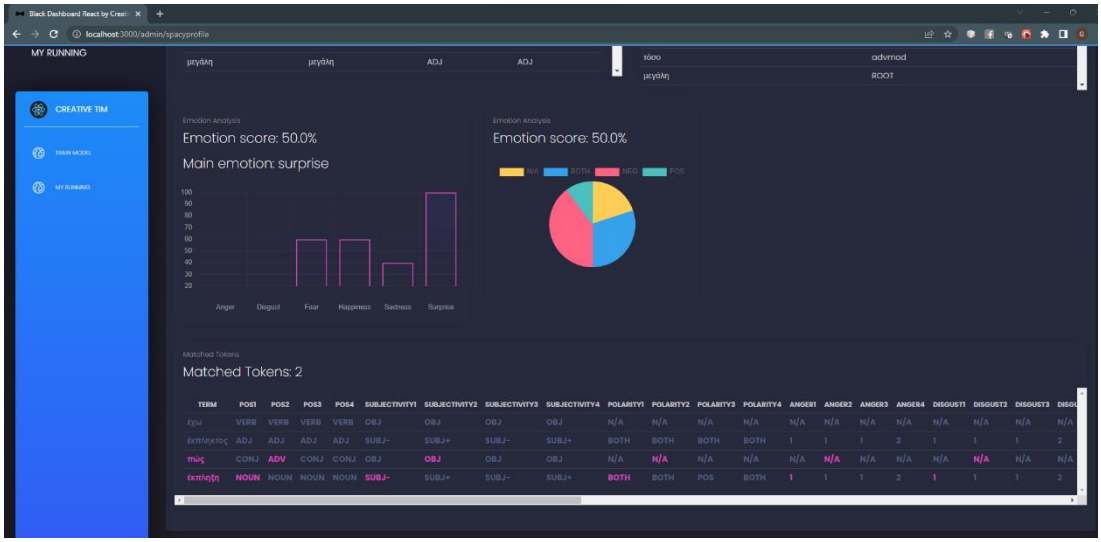


Εικόνα 5.11.1: Οθόνη «My Running»

Τα αποτελέσματα έρχονται στην ακόλουθη μορφή,

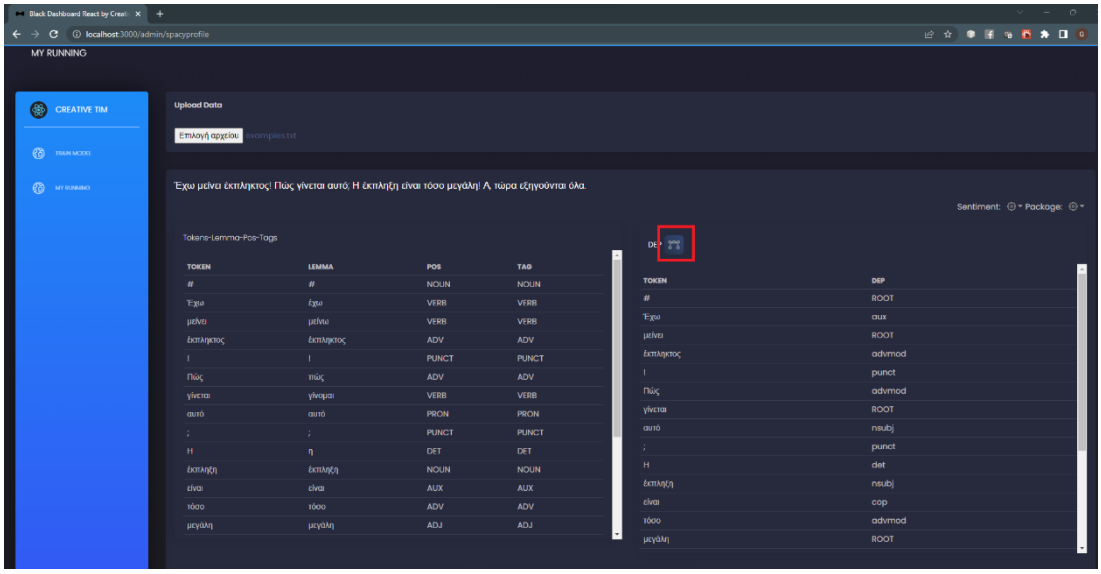


Εικόνα 5.11.2: Οθόνη «My Running»

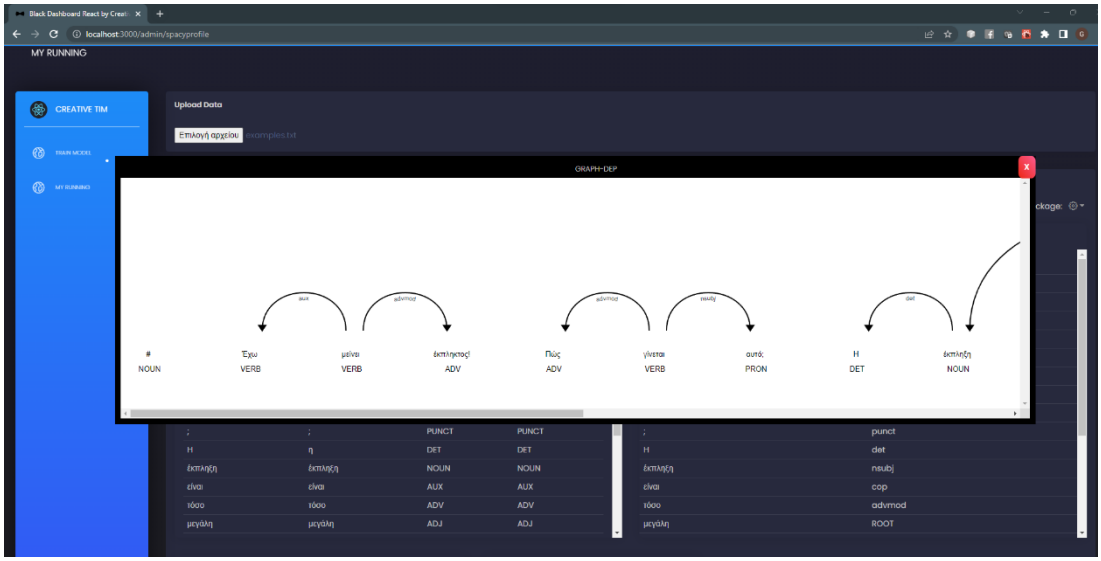


Εικόνα 5.11.3: Οθόνη «My Running»

Επιπλέων, δύναται η δυνατότητα εμφάνισης γραφήματος με τις συντακτικές εξαρτήσεις του κάθε κειμένου που εξετάζεται.

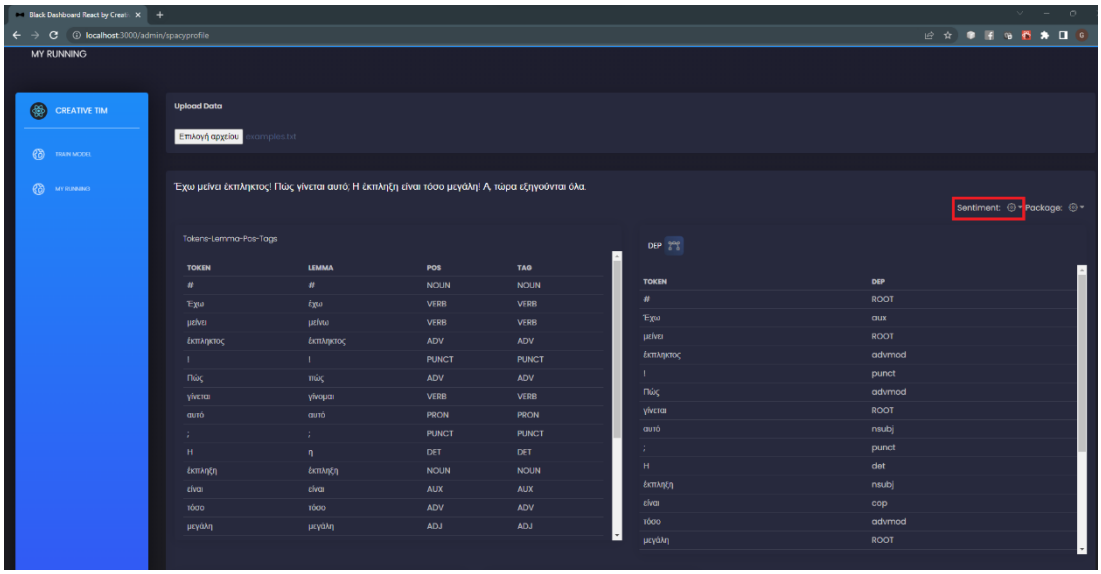


Εικόνα 5.11.4: Οθόνη «My Running»

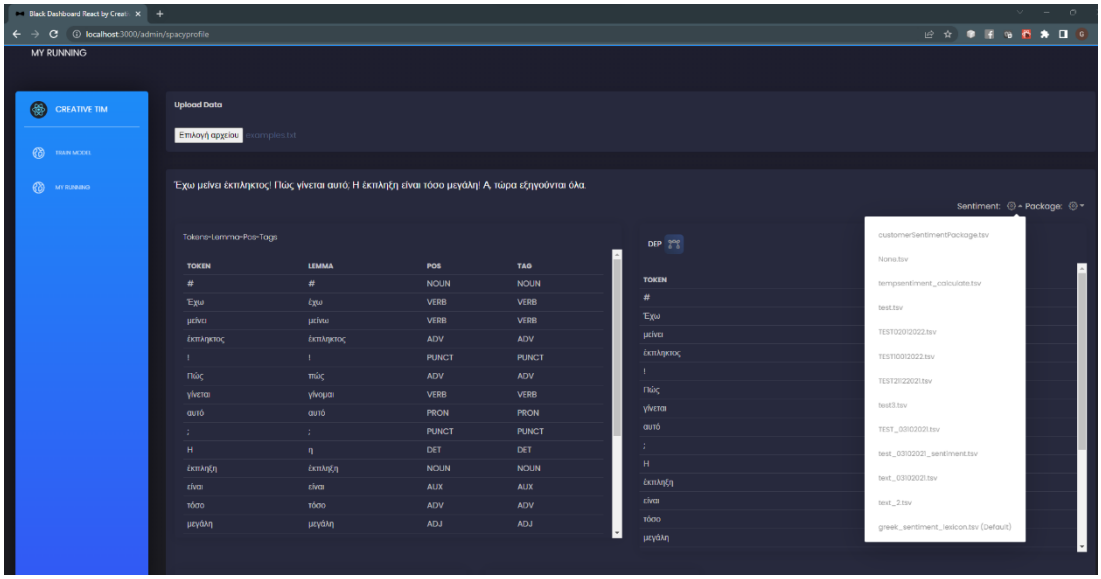


Εικόνα 5.11.5: Οθόνη «My Running»

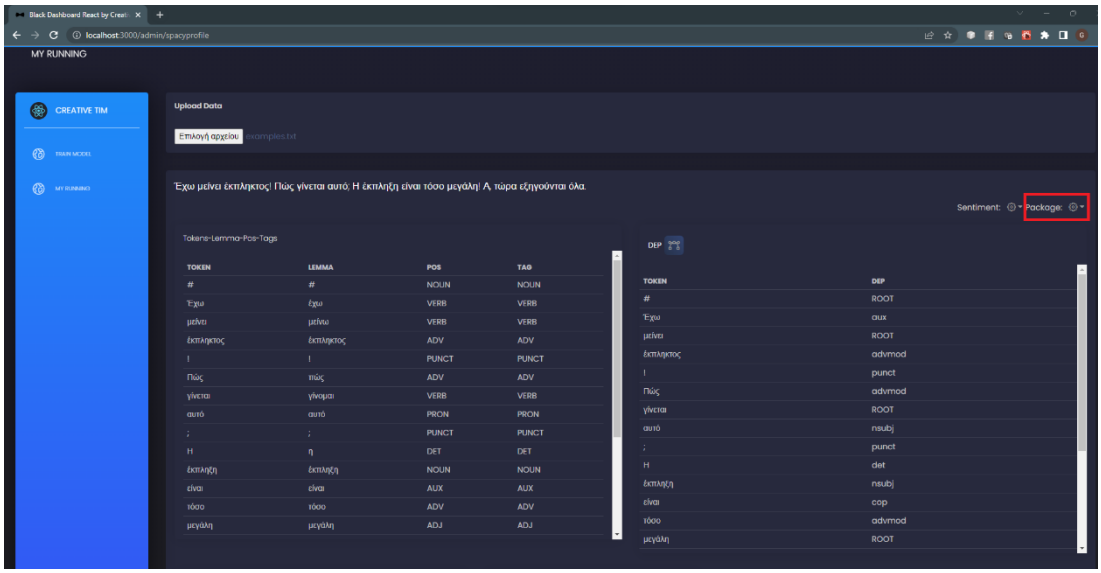
Τέλος, σε κάθε αποτέλεσμα δύναται η δυνατότητα να επιλέξει ο χρήστης τόσο το συναισθηματικό λεξικό όσο και το πακέτο spaCy που θα πραγματοποιήσουν την ανάλυση.



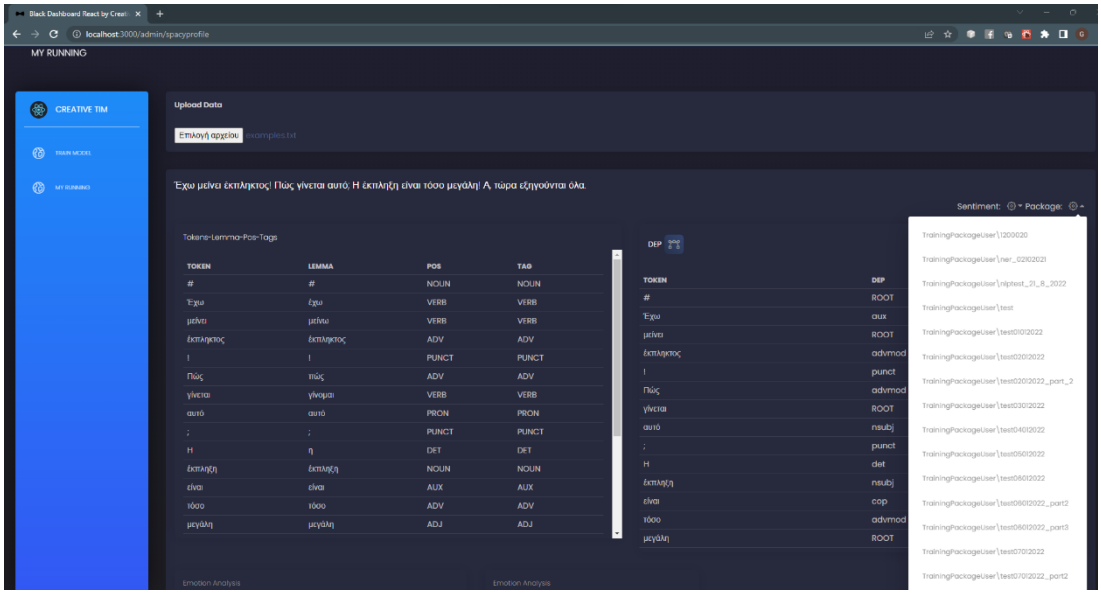
Εικόνα 5.11.6: Οθόνη «My Running»



Εικόνα 5.11.7: Οθόνη «My Running»



Εικόνα 5.11.8: Οθόνη «My Running»



Εικόνα 5.11.9: Οθόνη «My Running»

ΚΕΦΑΛΑΙΟ 6: Συμπεράσματα

Η σημερινή ψηφιακή πραγματικότητα όντας πολιορκημένη από την ταχύτατη ανάπτυξη της τεχνολογίας, και την παράλληλη έξαρση των κοινωνικών φαινομένων, καθιστά αναγκαία τον έλεγχο και ενδεχομένως την ασφάλεια αυτής. Ακριβώς γι' αυτό τον λόγο, η επεξεργασία φυσικής γλώσσας θα μπορούσε να αποτελέσει μια από τις βασικότερες οδούς για την διασφάλιση των παραπάνω.

Μέσα από την έρευνά μας εξάγουμε δύο συμπεράσματα: πρώτον, η βιβλιοθήκη spaCy είναι μια αρκετά υποσχόμενη εργαλειοθήκη για την επεξεργασία φυσικής γλώσσας κυρίως για τα ελληνικά κείμενα. Οι μηχανισμοί που προσφέρει η βιβλιοθήκη spaCy είναι αρκετά κατανοητοί και επαναχρησιμοποιήσιμοι. Αυτό αποτελεί μεγάλο πλεονέκτημα αφού διευκολύνει την διαδικασία του εμπλουτισμού NER, επιτρέποντας έτσι τα αποτελέσματα να είναι πολύ ενθαρρυντικά. Δεύτερον, η προσομοίωση των μοντέλων μηχανικής μάθησης (CNN - LSTM) για την ανίχνευση του συναισθήματος σε ελληνικά κείμενα δεν εκπληρώνει απόλυτα τις προσδοκίες μας. Πιο συγκεκριμένα, τα μοντέλα αποδίδουν θετικά σε μικρής κλίμακας συναισθημάτων, ενώ η ανταπόκριση αυτών σε μεγαλύτερης κλίμακας συναισθημάτων είναι χαμηλότερη.

Συνεπώς, τα συμπεράσματα της μελέτης αυτής αποτελούν τον οδηγό για την διερεύνηση και αξιολόγηση μελλοντικών μοντέλων μηχανικής μάθησης. Παραδείγματα αυτών των μοντέλων θα μπορούσαν να είναι το BERT (Bidirectional Encoder Representations from Transformers), το Bi-LSTM (Bidirectional LSTM), το ULMFiT (Universal Language Model Fine-tuning), το GPT-2 (Generative Pre-trained Transformer 2) και το XGBoost (eXtreme Gradient Boosting).

Επίσης, μια μελλοντική επέκταση αυτής της μελέτης θα μπορούσε να είναι ο συνδυασμός των δύο πειραματικών φάσεων. Πιο συγκεκριμένα, ο εντοπισμός του συναισθήματος για κάθε NER ενός κειμένου. Ο σκοπός αυτού είναι ο εντοπισμός του συναισθήματος (Θυμός, Χαρά, Λύπη κ.α.) για ετικέτες NER όπως PERSON (ΑΤΟΜΟ) ή και PRODUCT (ΠΡΟΙΟΝ) μέσα σε ένα κείμενο.

Εν κατακλείδι, στην εξόρυξη συναισθημάτων ενός κειμένου θα πρέπει να λάβουμε άμεσα υπόψιν μας το κριτήριο της ειρωνείας. Η ειρωνεία μπορεί να επηρεάσει άμεσα το αποτέλεσμα του συναισθήματος, καθώς αν το συναίσθημα χαρακτηρίζεται με θετικό πρόσημο μπορεί να έχει εννοιολογικά αρνητικό αποτέλεσμα. Πρόκειται για ένα πολύπλοκο ζήτημα που έχει προβληματίσει την επιστημονική κοινότητα, αφού δεν έχει μελετηθεί τόσο σε μεγάλο βαθμό, ότι αφορά τα ελληνικά κείμενα, και η δύναμη του όποιου συνεχίζει να υπερτερεί στον ψηφιακό κόσμο.

Αναφορές

[1] R. Grishman and B. Sundheim. Message understanding conference-6: A brief history. In Proc. of COLING. Association for Computational Linguistics, 1995. DOI: 10.3115/992628.992709. 25, 26, 38

[2] Erik F. Tjong, Kim Sang, and Fien De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In Walter Daelemans and Miles Osborne, Eds., Proc. of NAACL-HLT, pages 142–147, 2003. 27, 32

[3] Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. ACE 2005 multilingual training corpus. Linguistic Data Consortium, Philadelphia, 2006. 26, 27

[4] Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. Ontonotes: The 90% solution. In Proc. of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers, pages 57–60, New York City, 2006. Association for Computational Linguistics. 27

[5] V. Crescenzi, G. Mecca, P. Merialdo, et al. Roadrunner: Towards automatic data extraction from large web sites. In Proc. of the International Conference on Very Large Data Bases, pages 109–118. Citeseer, 2001. 30

[6] D. E. Appelt. The common pattern specification language. Technical report, SRI International, Artificial Intelligence Center, 1996. DOI: 10.3115/1119089.1119095. 31

[7] Maynard, D., Bontcheva, K. & Augenstein, I., 2017. Natural Language Processing for the Semantic Web. A Publication in the Morgan & Claypool Publishers series: SYNTHESIS LECTURES ON THE SEMANTIC WEB: THEORY AND TECHNOLOGY: Morgan & Claypool Publishers.

[8]https://colab.research.google.com/drive/14-xuOilquuYYNGQxtwaZnEKThyjqKqzc?usp=sharing#scrollTo=CCPpzat-d8Px

[9] https://medium.com/mysuperai/what-is-named-entity-recognition-ner-and-howcan-i-use-it-2b68cf6f545d

[10] Mallis, D., Kalamatianos, G., Nikolaras, D. & Symeonidis, S., 2015. Sentiment Analysis of Greek Tweets and Hashtags using Sentiment Lexicon.

[11]https://el.wikipedia.org/wiki/%CE%9C%CE%B7%CF%87%CE%B1%CE%BD%CE%B9%CE%BA%CE%AE_%CE%BC%CE%AC%CE%B8%CE%B7%CF%83%CE%B7

[12] https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[13] https://spacy.io/usage/facts-figures

[14] https://spacy.io/api

[15] W. Gerrod Parrott. Emotions in Social Psychology: Essential Readings. Psychology Press, 2001. 79

[16] Liao, S. et al., 2017. CNN for situations understanding based on sentiment analysis of twitter data.

[17] Wang, J., Yu, L.-C., Lai, K. R. & Zhang, Z., 2016. Dimensional Sentiment Analysis Using a Regional CNN-LSTM Mode.

[18] Rhanoui, M., Mikram, M., Yousfi, S. & Barzali, S., 2019. A CNN-BiLSTM Model for Document-Level.

[19] Ma, Y., Peng, H., Khan, T. et al. Sentic LSTM: a Hybrid Network for Targeted Aspect-Based Sentiment Analysis.

[20] Alayba, A.M., Palade, V., England, M., Iqbal, R. (2018). A Combined CNN and LSTM Model for Arabic Sentiment Analysis.

[21] Pitenis, Z., Zampieri, M. & Ranasinghe, T., 2020. Offensive Language Identification in Greek. 18 Mar.

[22] https://sites.google.com/site/offensevalsharedtask/results-and-paper-submission

[23] Zheng Lin, C., Ptaszynski, M. & Masui, F., 2019. Exploring Machine Learning Techniques for Irony Detection.

[24] https://towardsdatascience.com/text-classification-using-k-nearest-neighbors-46fa8a77acc5

[25] https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html

[26] http://nlp.cs.aueb.gr/software.html

[27] https://www.npmjs.com/package/react-switch

[28] https://keras.io/examples/nlp/text_classification_from_scratch/

[29] https://reactjs.org/

[30] https://flask.palletsprojects.com/en/2.2.x/

[31] Haykin, S., 2009, Neural Networks and Learning Machines, Third Edition

Παράρτημα Α

Η εκπαίδευση του μοντέλου spaCy για την ελληνική γλώσσα, πραγματοποιήθηκε με τρόπο ώστε να γίνεται «επανεκπαίδευση» του υπάρχοντος και η αποθήκευση του νέου πακέτου. Για να γίνει αυτό, έχουμε πρώτα κατεβάσει και φορτώσει το ήδη υπάρχον πακέτο με τις παρακάτω εντολές:

```
spacy.cli.download("el_core_news_lg")
nlp = spacy.load('el_core_news_lg')
```

όπου το nlp διαθέτει την πληροφορία του υπάρχοντος μοντέλου.

Στη συνέχεια, θα πρέπει να προσθέσουμε στο τέλος του processing pipeline, το στοιχείο του NER. Αυτό μπορεί να γίνει με τον μηχανισμό που μας προσφέρεται από την βιβλιοθήκη τον Language.add_pipe, όπου στο παράδειγμα μας έχουμε το εξής:

```
self.nlp.add_pipe('ner', last=True)
```

Αφού, έχουμε προσθέσει το στοιχείο στο pipeline, θα πρέπει να προσθέσουμε τις ετικέτες του για το NER πεδίο. Σε μια μεταβλητή κρατάμε το στοιχείο NER του pipeline, στο όποιο θα προσθέσουμε τις ετικέτες. Παρακάτω θα δούμε την διαδικασία αυτή:

```
ner = self.nlp.get_pipe('ner')
```

```
for ele in self.TRAIN_DATA_JSON:
    o, s = ast.literal_eval(ele)
    for ent in s.get('entities'):
        ner.add_label(ent[2])
```

Αυτό που γίνεται , είναι ότι για κάθε ένα στοιχείο του dataset μου, προσπαθώ να εντοπίσω όλες τις οντότητες που μπορεί να διαθέτη, δηλαδή όλες τα array (start, end , Label), ώστε να θα πάρουμε το όνομα του label και να το προσθέσω στο ner.

Στην συνέχεια, θα χρειαστούμε μια λίστα και ένα dictionary, όπου στην λίστα μας θέλουμε να περάσουμε τα παραδείγματα που διαθέτη το dataset μας, ενώ το dictionary πρόκειται να δεχθεί πληροφορίες που είναι απαραίτητες για τα παραδείγματα που θα προστεθούν στην λίστα.

Για κάθε μια γραμμή που διαθέτη το dataset μας, θα πρέπει να έχουμε το κείμενο και τις οντότητες του. Το κείμενο θα είναι το input για της nlp, ώστε να πάρουμε το αντικείμενο Doc και αυτό αποθηκεύεται σε μια μεταβλητή docel, όπως δημιουργείτε από το υφιστάμενο μοντέλο, το όποιο έχει πληροφορίες που θα θέλαμε να διατηρήσουμε και να μεταφέρουμε στον νέο μοντέλο. Οι πληροφορίες αυτές μπορεί να είναι το μορφολογικό μέρος(morphologizer), το

λήμμα(lemma), λέξεις (words- tokens) κ.α. και αυτές τις πληροφορίες είναι που θα τις προσθέσουμε στο dictionary μας.

Ενώ έχουμε φτιάξει ένα νέο αντικείμενο Doc, δίνοντας στην μεταβλητή το όνομα doctemp, όπου σαν input έχουμε περάσει το vocab του nlp μας και τα words, όπου πρόκειται για μια λίστα με τα tokens που διαθέτη το docel.

Επομένως, αυτό που έχει μείνει είναι να φορτώσουμε τη λίστα μας με τα παραδείγματα. Το spaCy για να μπορέσει να προχώρηση στην εκπαίδευση, θα πρέπει τα παραδείγματα να είναι συγκεκριμένου τύπου αντικειμένου αυτός είναι το spacy.training.example. Εμείς, χρησιμοποιήσαμε το μηχανισμό Example.from_dict και σαν input την πρώτη φορά περνάμε το doctemp και τις οντότητες από το dataset., ενώ την δεύτερη φορά περνάμε το doctemp και το dictionary. Παρακάτω ακολουθεί ο αντίστοιχος κώδικας:

```
self.examples = []
self.datadoc_ = {}
```

```
for ele in self.TRAIN_DATA_JSON:
    text, ent = ast.literal_eval(ele)
    docel = self.nlp(str(text))
    tokens_ = []
    lemma_ = []
    morph_ = []
    doctemp = Doc(self.nlp.vocab, words=[token.text for token in docel])
    for token in docel:
        try:
            tokens_.append(str(token))
            lemma_.append(token.lemma_)
            morph_.append(str(token.morph))
        except KeyError:
            continue
    self.examples.append(Example.from_dict(doctemp, ent))
    try:
        self.datadoc_ = {'WORDS': tokens_,
                         'LEMMA': lemma_,
                         'MORPH': morph_
                        }
        self.examples.append(Example.from_dict(doctemp, self.datadoc_))
    except:
        continue
```

Τέλος, αφού έχουμε κάνει την προεργασία μας που χρειάζεται, έχουμε το κομμάτι της εκπαίδευση. Για την εκπαίδευση έχουμε ορίσει έναν αριθμό εποχών ίσο με 1000, σε κάθε επανάληψη ανακατεύουμε τα παραδείγματα μέσα στην λίστα των examples, και πραγματοποιούμε το update πάνω στο nlp μας. Αφού, ολοκληρωθεί η διαδικασία αυτό που μένει είναι να αποθηκεύσουμε το νέο μοντέλο μας. Η ακόλουθη διαδικασία φαίνεται στον παρακάτω κώδικα.

```
print('start_training...')
# optimizer = nlp.begin_training()
optimizer = self.nlp.create_optimizer()
for itn in range(n_iter):
    random.shuffle(self.examples)
    losses = {}
    # losses = morphologizer.update(examples, sgd=optimizer)
    for batch in minibatch(self.examples, size=50):
        try:
            self.nlp.update(
                batch,
                drop=0.5,
                sgd=optimizer,
                losses=losses)
        except:
            continue
    print(losses)
self.nlp.to_disk("TrainingPackageUser/"+self.package_name)
```

Παράρτημα Β

Για τον σκοπό αυτό σχεδιάσαμε τρεις βασικές κλάσεις, την CreateData για την δημιουργία dataset, την EmotionData για το φόρτωμα των δεδομένων από το συναισθηματικό λεξικό, και τέλος την CalculateSentiment για τον υπολογισμό του συναισθήματος μιας πρότασης.

EmotionData

Η κλάση αυτή είναι υπεύθυνη να φορτώσει τα δεδομένα από το συναισθηματικό λεξικό να τα διατηρήσει σε μεταβλητή, όπου να είναι διαθέσιμη όπου χρειαστεί. Για την κλάση αυτή απαιτείται η βιβλιοθήκη pandas της Python.

Άρα, πρώτα κάνουμε ***import pandas as pd***, για να εισάγουμε την βιβλιοθήκη pandas στο πρόγραμμα μας και προσδιορίζοντας την ονοματικής συντόμευση ***pd***. Να σημειωθεί ότι πολλές φορές γίνεται χρήση του ‘as’, καθώς κάποιες από της βιβλιοθήκες έχουν μεγάλα ονόματα και μας είναι δύσχρηστα στην εφαρμογή τους σε όλο τον πηγαίο κώδικα μας, για να προσδιορίσουμε και συντομογραφία της βιβλιοθήκης που θέλουμε να χρησιμοποιήσουμε. Επιπλέων, έχουμε προσθέσει δύο μεθόδους την GetIndexes και την GetDF. Η πρώτη μέθοδος, θα μας επιστρέψει ένα σχήμα με τις θέσεις των λέξεων που βρίσκονται στο συναισθηματικό λεξικό, όπως για παράδειγμα η πρώτη λέξη είναι «αβάφτιστος» στην θέση μηδέν. Ενώ η δεύτερη, είναι αυτή που μου επιστρέφει όλο το σχήμα που έχει το συναισθηματικό λεξικό, μαζί με τα headers.

Παρακάτω, παραθέτουμε τον κώδικα:

```
class EmotionData:
    indexes = {}
    df = None
    file_name = None

    def __init__(self, file_name=None):
        self.indexes = {}
        self.df = None
        self.file_name = None
        self.file_name = file_name
        if file_name is None or str(file_name).__contains__('(Default)'):
            self.df = pd.read_csv('greek_sentiment_lexicon.tsv', sep='\t')
            self.df = self.df.fillna('N/A')
        else:
            self.df = pd.read_csv(self.file_name, sep='\t')
            self.df = self.df.fillna('N/A')
        for index, row in self.df.iterrows():
            self.df.at[index, "Term"] = row["Term"].split(' ')[0]
            self.indexes[self.df.at[index, "Term"]] = index

    def GetIndexes(self):
        return self.indexes

    def GetDF(self):
        return self.df
```

CalculateSentiment

Η κλάση είναι υπεύθυνη για τον εντοπισμό του συναισθήματος μια πρότασης. Αυτή η δομή περιλαμβάνει έναν αρχικοποιητή («__init__») και μια μέθοδο για τον υπολογισμό του συναισθήματος την Analysis. Στον αρχικοποιητή, έχουμε δηλώσει τέσσερα στοιχεία στην είσοδο του, ένα είναι το κείμενο (text) που θα αναλύσουμε, το επόμενο είναι ο nlp ο μηχανισμός που θα χρησιμοποιηθεί για την ανάλυση του κειμένου (π.χ. POS Tags), το τρίτο όρισμα αφορά τα δεδομένα του λεξικού με τα συναισθήματα και τέλος το μια λίστα από με την θέση που βρίσκονται οι λέξεις στο πίνακα του λεξικού. Ο αρχικοποιήτης διαθέτει επιπλέον στοιχεία που δεν λαμβάνονται από την είσοδο αλλά είναι αρχικός ορισμένα, αφού δεν πρόκειται αλλάξουν κατά την εκτέλεση του προγράμματος. Τα στοιχεία ειναι αυτά που περιλαμβάνονται και στην αναφορά [8], όμως λίγο τροποποιημένα. Παρακάτω, παραθέτουμε τον κωδικά του αρχικοποιητή.

```
def __init__(self, text, nlp, df, indexes):
    self.resp_data = {}
    self.text = text
    self.nlp = nlp
    self.df = df
    self.indexes = indexes
    self.mainemotion = ""
    self.mypolarity = ""

    self.subj_scores = {
        'N/A': 0,
        'OBJ': 0,
        'SUBJ-': 0.5,
        'SUBJ+': 1
    }
```

```
    self.emotion_scores = {
        'N/A': 0,
        '0.0': 0,
        '1.0': 0.2,
        '2.0': 0.4,
        '3.0': 0.6,
        '4.0': 0.8,
        '5.0': 1,
        '0': 0,
        '1': 0.2,
        '2': 0.4,
        '3': 0.6,
        '4': 0.8,
        '5': 1,
    }

    self.polarity_scores = {
        'N/A': 0,
        'BOTH': 0,
        'NEG': -1,
        'POS': 1
    }

    self.polaritymyscore = {
        0: 'N/A',
        -1: 'NEG',
        1: 'POS'
    }
```

Επεξήγηση:

- ***self.resp_data = {}*** : Το σχήμα δεδομένων που θέλουμε να επιστρέψουμε.
- ***self.text*** : Το κείμενο που θα αναλύσουμε (δίνετε στην είσοδος)
- ***self.nlp*** : Το πακέτο nlp που θα χρεισιμοποιησουμε (δίνετε στην είσοδο)
- ***self.df*** : Τα δεδομένα από το συναισθηματικό λεξικό (δίνετε στην είσοδο)
- ***self.indexes*** : Το σχήμα με τις θέσεις των λέξεων (δίνετε στην είσοδο)
- ***self.mainemotion :*** Στοιχείο που διατειρήτε το κυρίως συναίσθημα
- ***self.mypolarity*** : Στοιχείο που θα αποθηκεύει το γενικό συναίσθημα του κειμένου (Θετικό/αρνητικό/ουδέτερο)

Ακολουθούν τα σχήματα που αντιστοιχούν τα λεκτικά που περίεχει το συναισθηματικό λεξικό με τιμές.

- ***self.subj_scores*** : βαθμολογία υποκειμενικότητας
- ***self.polarity_scores*** : βαθμολογίες πολικότητας
- ***self.polaritymyscores*** : βαθμολογίες πολικότητας, δεν περιλαμβάνει το BOTH.

- ***self.emotion_scores*** : βαθμολογίες συναισθημάτων

Η μέθοδος Analysis, δεν διαθέτει κάποιο στοιχείο ως είσοδο καθώς δεν θεωρήσαμε ότι χρειάζεται καθώς όλα τα απαραίτητα δεδομένα τα έχουμε από τον αρχικοποιητή μας. Αλλά και εδώ πραγματοποιήθηκαν κάποιες τροποιήσεις σχετικά με τον αλγόριθμό της εργασίας [8]. Παραθέτουμε τον κώδικα σταδιακά, με τις επεξηγήσεις τους.

```
def Analysis(self):
    doc = self.nlp(self.text)
    subjectivity_score = 0
    anger_score = 0
    disgust_score = 0
    fear_score = 0
    happiness_score = 0
    sadness_score = 0
    surprise_score = 0
    mypolarity_scores = 0
```

Επεξήγηση: Για αρχή θα δούμε τις τοπικές μεταβλητές.

Το doc είναι ένα αντικείμενο τύπου Doc όπου περιέχει στοιχεία του αναλυμένου κειμένου, θα το χρειαστούμε για να πάρουμε τα Tokens και τα POS tags του.

Έχουμε μεταβλητές για κάθε συναίσθημα όπου θα διατηρήσουμε το άθροισμα των συναισθημάτων από κάθε λέξη του κειμένου.

Στη επόμενη εικόνα βλέπουμε την βασική ρουτίνα, η όποια είναι αυτή που θα μαζέψει την βαθμολογία των συναισθημάτων. Αυτό που κάνει είναι, για κάθε token που διαθέτει το κείμενο μου, εξετάζω το λήμμα του για το αν βρίσκεται στο σχήμα με τις θέσεις που έχουμε από την είσοδο του αρχικοποιητή. Αν το βρούμε στο σχήμα με τα indexes τότε παίρνουμε την θέση που βρίσκεται (π.χ. στην 15η). Στην συνέχεια, θα πρέπει να δούμε αν το POS tag του token αντιστοιχεί σε ένα από τα 4 POS που διαθέτει το συναισθηματικό λεξικό. Αφού εντοπίσουμε το POS tag και έχουμε δηλαδή το pos_flag = true, τότε εντοπίζουμε για ποια κολώνα μιλάμε. Για παράδειγμα, έχουμε εντόπιση το POS tag αντιστοιχεί στο POS3, άρα και εμείς πρέπει να πάρουμε την βαθμολογία που αντιστοιχεί στο συναίσθημα-3 (π.χ. Anger3, Fear3, Disgust3, Happiness3 κτλ.). Με αυτό τον τρόπο, υπολογίζουμε αθροιστικά τα συναισθήματα σε όλο το κείμενο.

```
for token in doc:
    lemmatized_token = token.lemma_
    if lemmatized_token in self.indexes:
        indx = self.indexes[lemmatized_token]
        pos_flag = False
        for col in ["POS1", "POS2", "POS3", "POS4"]:
            if token.pos_ == self.df.at[indx, col]:
                match_col_index = [int(s) for s in col if s.isdigit()][0]
                pos_flag = True
                break
        if pos_flag == True:
            match_col_index = [int(s) for s in col if s.isdigit()][0]  # exei parei to POS1, psaxnei na vrei
            # to 1 (2, 3, 4)
            subjectivity_score += self.subj_scores[self.df.at[indx, 'Subjectivity' + str(match_col_index)]]
            anger_score += self.emotion_scores[str(self.df.at[indx, 'Anger' + str(match_col_index)])]
            disgust_score += self.emotion_scores[str(self.df.at[indx, 'Disgust' + str(match_col_index)])]
            fear_score += self.emotion_scores[str(self.df.at[indx, 'Fear' + str(match_col_index)])]
            happiness_score += self.emotion_scores[
                str(self.df.at[indx, 'Happiness' + str(match_col_index)])]
            sadness_score += self.emotion_scores[str(self.df.at[indx, 'Sadness' + str(match_col_index)])]
            surprise_score += self.emotion_scores[str(self.df.at[indx, 'Surprise' + str(match_col_index)])]
            mypolarity_scores += self.polarity_scores[str(self.df.at[indx, 'Polarity' + str(match_col_index)])]
```

Τέλος, σε ένα σχήμα, το self.emotions, αντιστοιχούμε το συναίσθημα με το αντίστοιχη συνολική βαθμολογία. Το ***self.mypolarity*** θα έχουμε μια από τις τιμές «POS, NEG, N/A», και σε μια τοπική μεταβλητή περνάμε το κύριο συναίσθημα, κάνοντας χρήση της συνάρτησης max της python, όπου μας επιστρέφει την μέγιστη βαθμολογία από του σχήμα self.emotions. Επειδή, πολλές φορές ένα κείμενο μπορεί κανένα από τα token του να μην αντιστοιχεί στο συναισθηματικό λεξικό, είτε επειδή δεν έχει συμπεριληφθεί είτε αυτό είναι σωστό, μπορεί το κύριο συναίσθημα να είναι μηδέν. Σε αυτή την περίπτωση, θα θέλαμε να ορίζουμε το κύριο συναίσθημα ως “N/A”.

```
try:
    self.emotions = {'anger': anger_score, 'disgust': disgust_score, 'fear': fear_score,
                     'happiness': happiness_score,
                     'sadness': sadness_score, 'surprise': surprise_score}
    self.mainemotion = ""
    self.mypolarity = self.polaritymyscore.get(mypolarity_scores)
    emotion = max(self.emotions.items(), key=operator.itemgetter(1))[0]
    if max(self.emotions.items(), key=operator.itemgetter(1))[1] == 0:
        self.mainemotion = "N/A"
    else:
        self.mainemotion = emotion
except:
    print('No matched tokens')
```

CreateData

Η κλάση είναι κορμός για την δημιουργία του dataset μας, διαβάζει το σύνολο δεδομένων και με την βοήθεια των προηγούμενων κλάσεων θα πραγματοποιήσει την ταξινόμηση που επιθυμούμε. Η Κλάση αυτή χρησιμοποιεί τρεις βασικές βιβλιοθήκες, την "os" η οποία μας δίνει την δυνατότητα που αφορά το λειτουργικό σύστημα (π.χ. εντοπισμός ένα file path, δημιουργία ενός folder κτλ), την "spacy" για την ανάλυση του κειμένου και την "pandas" για την φόρτωση δεδομένων από σύνολο. Αυτή η κλάση διαθέτει έναν αρχικοποιητή και μια βοηθητική μέθοδο. Η βοηθητική μέθοδος πρόκειται για τον καθαρισμό των δεδομένων από διπλό-εγγραφές και την χρησιμοποιούμε σε περίπτωση το σύνολο μας περιέχει τέτοιες εγγραφές.

```
def fixtheData(self):
    self.listlines = []
    for line in self.df['text'].values:
        if line not in self.listlines:
            self.listlines.append(line)
    print(str(len(self.listlines)))
```

Αυτή είναι η βοηθητική μέθοδος, για κάθε κείμενο που περιέχεται στο σύνολο μας, το προσθέτουμε σε μια τοπική λίστα "listlines" με την προϋπόθεση ότι δεν υπάρχει ήδη μέσα στην λίστα. Στο τέλος εμφανίζεται το πλήθος των δεδομένων.

Παρακάτω βλέπουμε το πρώτο μέρος του κώδικα του αρχικοποιητή, οποίος διαβάζει το σύνολο δεδομένο μας που είναι αποθηκευμένο στο αρχείο «CLEARDataSetsTweetISEtiment2.csv» με την βοήθεια της βιβλιοθήκης «pandas» και του μηχανισμού "read_csv", προσδιορίζονται το delimiter και την κωδικοποίηση που αρχείου. Αφού έχουμε φορτώσει όλα τα δεδομένα στην μεταβλητή "self.df", προσορμίζουμε τα ονόματα των επικεφαλίδων κάθε στήλης. Εμείς διαθέτουμε δύο στήλες, η πρώτη αφορά το κείμενο «text» και η δεύτερη την κατηγοριοποίηση του κειμένου. Η δεύτερη στήλη, δεν μας απασχολεί και δεν θα ασχοληθούμε καθόλου.

Στη συνέχεια, χρησιμοποιούμε την βοηθητική μέθοδο, ώστε να πάρουμε τα κείμενα που θέλουμε. Ακολουθούν οι εντολές "emotion = EmotionData()", όπου φορτώνουμε τα δεδομένα από το συναισθηματικό λεξικό και την "nlp = sp.load('el_core_news_lg')", όπου φορτώνουμε το πακέτο για την φυσική επεξεργασία ελληνικών κειμένων. Τέλος, ορίζουμε δύο μετρητές, τον countfs για το όνομα με το όποιο θα αποθηκεύσουμε το κείμενο, και τον counttest για την "τυχαία" κατανομή τον δεδομένων του train και του test.

```
def __init__(self):
    self.df = pd.read_csv('CLEARDataSetsTweetISEtiment2.csv', delimiter=';', encoding='UTF-8')
    col = ['text', 'category']
    self.df = self.df[col]
    self.df = self.df[pd.notnull(self.df['category'])]
    self.df.columns = ['text', 'category']
    self.df['category_id'] = self.df['text'].factorize()[0]

    self.fixtheData()

    emotion = EmotionData()
    nlp = sp.load("el_core_news_lg")
    countfs=0
    counttest = 0
```

Στη συνέχεια, αυτό που πραγματοποιούμε είναι ο εντοπισμός του συναισθήματος του κάθε κείμενου και την ταξινόμηση του. Η ταξινόμηση γίνεται βάση του συναισθήματος, και μπορεί να πραγματοποιηθεί με τον παρακάτω κώδικα. Όπου για κάθε ένα κείμενο, που έχουμε στο “self.listlines”, φτιάχνουμε ένα αντικείμενο του τύπου “CalculateSentiment” δίνοντας στην είσοδο του, το κείμενο, το μοντέλο nlp, τα δεδομένα του συναισθηματικού λεξικού και το σχήμα με τις θέσεις των λέξεων μέσα στο λεξικό. Το αποτέλεσμα αυτού του αντικείμενου αποθηκεύεται στην μεταβλητή «sentiment». Το επόμενο βήμα είναι η χρήση της μεθόδου “CalculateSentiment.Analysis”, για να υπολογίσουμε το συναίσθημα του κειμένου. Αυτό πραγματοποιείται με την εντολή “sentiment.Analysis()”, όπου θα έχει ως αποτέλεσμα το στοιχείο mainemotion να έχει το κύριο συναίσθημα όλου του κειμένου.

Παράρτημα Γ

Ο παρακάτω ο αλγόριθμος πραγματοποιεί την δημιουργία των φακέλων και αρχείο σε μια ιεραρχική δομή, για την δημιουργία του dataset μας.

```
for line in self.listlines:
    print('''==> '''+line+'''\n''')
    sentiment = CalculateSentiment(text=line, nlp=nlp, df=_emotion.GetDF(), indexes=emotion.GetIndexes())
    sentiment.Analysis()
    print(sentiment.mainemotion)
    print(sentiment.mypolarity)
    countfs = countfs + 1
    if sentiment.mainemotion == "N/A":
        sentiment.mainemotion = "NA"
    if not os.path.exists('my_datasets_tweets_sentiment8/train/' + sentiment.mainemotion):
        os.makedirs('my_datasets_tweets_sentiment8/train/' + sentiment.mainemotion)
    with open('my_datasets_tweets_sentiment8/train/' + sentiment.mainemotion + '/' + str(countfs) + '.txt', 'a',_encoding='UTF-8') as fs:
        fs.write(line)
        fs.close()
    if counttest == 10:
        counttest = 0
        if not os.path.exists('my_datasets_tweets_sentiment8/test/' + sentiment.mainemotion):
            os.makedirs('my_datasets_tweets_sentiment8/test/' + sentiment.mainemotion)
        with open('my_datasets_tweets_sentiment8/test/' + sentiment.mainemotion + '/' + str(countfs) + '.txt',
                  'a', encoding='UTF-8') as fs:
            fs.write(line)
            fs.close()
    else:
        counttest += 1
```

Παράρτημα Δ

Παρακάτω παρουσιάζουμε αναλυτικά την διαδικασία, με την όποιο μπορέσαμε να φορτώσουμε το dataset για να μπορέσουμε να εκπαιδεύσουμε τα μοντέλα μηχανικής μάθησης.

Στη παρακάτω διαδικασία, η βασική ρουτίνα που χρησιμοποιήθηκε είναι η "Kerras.preprocessing.text_dataset_from_directory", όπου σαν είσοδο έχουμε περάσει τα ακόλουθα ορίσματα, το path του dataset, ένα batch_size, το validation_split, ένα όρισμα που δηλώνει το είδος των δεδομένων (π.χ., "training", "validation" κτλ.), τέλος το πεδίο του seed.

Παρακάτω παραθέτουμε και τον κώδικα, όπου σε τρεις μεταβλητές θα πάρουμε θα τα δεδομένα από το dataset. Μια μεταβλητή για το training, μια για το validation και μια μεταβλητή για το test.

```
path = "my_datasets_tweets_sentiment8"


    raw_train_ds = tf.keras.preprocessing.text_dataset_from_directory(
        path+"/train",
        batch_size=batch_size,
        validation_split=0.1,
        subset="training",
        seed=1337,
    )
    raw_val_ds = tf.keras.preprocessing.text_dataset_from_directory(
        path+"/train",
        batch_size=batch_size,
        validation_split=0.1,
        subset="validation",
        seed=1337,
    )
    raw_test_ds = tf.keras.preprocessing.text_dataset_from_directory(
        path+"/test", batch_size=batch_size
    )
```

Στην συνέχεια, δημιουργούμε μια μέθοδο με σκοπό την διανυσματοποίηση των δεδομένων. Φτιάχνουμε, το επίπεδο της διανυσματοποίησης, όπου θα είναι σε θέση να κανονικοποίησει, διαχωρίσει, αντιστοιχήσει τις συμβολοσείρες ακαίραια δεδομένα. Όπως βλέπουμε στον

παρακάτω απόσπασμα κώδικα, αυτό θα γίνει με χρήση της μεθόδου, ‘tensorflow.Keras.layers.TextVectorization. Επομένως, έχουμε δημιουργήσει το επίπεδο κανονικοιήσης, με την βοήθεια της παραπάνω μεθόδου, όπου σαν είσοδο, έχουμε δώσει το max_tokens =3000, output_mode=“int”, output_sequence_length= ‘500’. Το επίπεδο αυτό καλείται από την συνάρτηση, vectorise_text((text, label)). Η συνάρτηση στην είσοδο της θα πρέπει να δέχεται το κείμενο και την ετικέτα όπου βρίσκεται το κείμενο, έπειτα προσθέτει στο text που δόθηκε στην είσοδο, ενώ στο τέλος μα επιστρέφει το σχήμα με το text (μετά την κανονικοιήση του) και με την ετικέτα της κατηγορίας που ανοίκει το κείμενο.

```
vectorize_layer = tf.keras.layers.TextVectorization(
        max_tokens=max_features,
        output_mode="int",
        output_sequence_length=sequence_length
    )

    def vectorize_text(text, label):
        text = tf.expand_dims(text, -1)
        return vectorize_layer(text), label
```

Τέλος, αυτό που θέλουμε είναι να διαμορφωθούν τα δεδομένα σύμφωνα με τη παραπάνω συνάρτηση κανονικοποίησης, και αυτό πετυχαίνεται με το παρακάτω κώδικα.

```
train_ds = raw_train_ds.map(vectorize_text)
val_ds = raw_val_ds.map(vectorize_text)
test_ds = raw_test_ds.map(vectorize_text)
```

Παράρτημα Ε

```
inputs = tf.keras.Input(shape=(None,), dtype="int64")
max_features = 30000
embedding_dim = 128

x = layers.Embedding(max_features,embedding_dim)(inputs)
x = layers.Dropout(0.5)(x)
x = layers.Conv1D(128, 7, padding="valid", activation="relu", strides=3)(x)
x = layers.Conv1D(128, 7, padding="valid", activation="relu", strides=3)(x)
x = layers.GlobalMaxPooling1D()(x)
x = layers.Dense(128, activation="relu")(x)
x = layers.Dropout(0.5)(x
predictions = layers.Dense(1, activation="sigmoid", name="predictions")(x)
model = tf.keras.Model(inputs, predictions)
model.compile(loss="binary_crossentropy", optimizer="adam", metrics=["accuracy"])
epochs = 5

checkpoint_path = "training_model1/cp.ckpt"
checkpoint_dir = os.path.dirname(checkpoint_path)

cp_callback = tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint_path,
                                                 save_weights_only=True,verbose=1)
model.fit(train_ds, validation_data=val_ds, epochs=epochs, callbacks=[cp_callback])
```

Παράρτημα ΣΤ

```
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Dense, SpatialDropout1D, Embedding, Dropout
from keras.layers import LSTM
from keras.layers import TimeDistributed
from keras.layers import RepeatVector
self.model = Sequential()
self.model.add(Embedding(30000, 128))
for i in range(6):
    self.model.add(LSTM(128, input_shape=(None, None, 128), activation='tanh', recurrent_activation='sigmoid', return_sequences=True))

    self.model.add(Dropout(0.5))
self.model.add(Dense(12, activation='relu'))
self.model.add(Dropout(0.5))
self.model.compile(loss='sparse_categorical_crossentropy',
                   optimizer='adam', metrics=['accuracy'])

print(self.model.summary())

checkpoint_path = "training_model_lstm/cp.ckpt"
checkpoint_dir = os.path.dirname(checkpoint_path)

cp_callback = tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint_path,
                                                 save_weights_only=True,
                                                 verbose=1)
self.model.fit(Data, validation_data=val_dat, epochs=3)
```

Παράρτημα Ζ

Το παράρτημα Γ, αναφέρουμε την τεχνολογίες, τα εργαλεία και τις τεχνικές που συνέβαλαν στην δημιουργία του back-end web application.

Back-End

Στην ενότητα αυτή θα δούμε το back-end μέρος της εφαρμογής όπου χρησιμοποιήθηκε το Flask, το οποίο πρόκειται για ένα web framework γραμμένο σε Python. Θα δούμε βήμα βήμα την δημιουργία ενός Project Flask με την χρήση του IDE PyCharm.

Με την δημιουργία του project, μπορεί κάποιος να παρατηρήσει ότι φτιάχνονται αυτόματα κάποια αρχεία που είναι απαραίτητα για την εκκίνηση του Server. Ένα από τα βασικά αρχεία που θα μας απασχολήσει είναι το «app.py». Το αρχείο είναι αυτό που διαθέτει όλα τα api routes, δηλαδή τις διαδρομές όπου μπορεί το front-end να επικοινωνήσει με το back-end.

Ενώ, έχουμε δημιουργήσει και προσθέσει φακέλους και αρχεία βοηθητικά, για την σωστή οργάνωση και εικόνα του Server. Δημιουργήσαμε τους ακόλουθους φακέλους «NLPSpacyModel» και «NLPSpacyCommone». Στο πρώτο φάκελο δημιουργήσαμε τα αρχεία που περιέχουν μηχανισμούς για ότι σχετίζεται με το υπολογιστικό μέρος της εφαρμογής, όπως ο εντοπισμός του NER, η ανάλυση συναισθήματος, η εκπαίδευση του NER και άλλες βοηθητικές μεθόδους. Σε ότι αφορά το φάκελο «NLPSpacyCommone», φτιάξαμε αρχεία που σχετίζονται μετά τα Request και Response από/προς στο Front-End.

Τέλος, έχουμε δημιουργήσει ένα ακόμα αρχείο στο project το «settings.ini», πρόκειται για ένα αρχείο παραμετροποιήσης (configuration), το όποιο περιέχει πληροφορίες όπως για παράδειγμα διαδρομές αρχείο κτλ.

NLPSpacyModel

NLPSpacyHelper.py

Πρόκειται για το αρχείο, όπου περιέχει βοηθητικές κλάσεις που μπορεί να χρησιμοποιούνται καθόλη την εκτέλεση της διαδικτυακής εφαρμογής. Το αρχείο περιλαμβάνει τρεις κλάσεις την «Spacy», την «JSONSerialize», την «ReadConfigSettings». Η βιβλιοθήκες που έχουμε εισάγει είναι ακόλουθες.

```
import json


import spacy
from configparser import ConfigParser
from json_serializer.Serializer import Serializer
```

NLPSpacyHelper.Spacy

Η κλάση είναι υπεύθυνη για την φόρτωση του πακέτου Spacy που έχουμε εκπαίδευση ή το προεπιλεγμένο, δηλαδή αυτό που υπάρχει σήμερα. Η κλάση διαθέτη ένα χαρακτηριστικό, και τρεις μεθόδους. Παρακάτω, παραθέτουμε τον κώδικα:

class Spacy:
  nlp = None
 	
  def __init__(self, nlp=None):
  if nlp is None or nlp == '(Default)':
  conf = ConfigParser()
  configFilePath = r'settings.ini'
  conf.read(configFilePath)
  self.nlp = spacy.load(str(conf.get("nlp_pgk", 'path')))
  else:
  self.nlp = spacy.load(str(nlp))
 	
  def Get(self):
  return self.nlp
 	
  def GetDefault(self):
  conf = ConfigParser()
  configFilePath = r'settings.ini'
  conf.read(configFilePath)
  self.nlp = spacy.load('el_core_news_lg')
  return self.nlp
	Nlp: είναι το χαρακτηριστικό όπου περιλαμβάνει το πακέτο nlp που του έχουμε φορτώσει.
	__init__(self, nlp=None): πρόκειται για τον αρχικοποιητή οποίος έχει ένα όρισμα στην είσοδο του και αυτό είναι το όνομα του αρχείου που θέλουμε να φορτώσουμε. Στην περίπτωση που η είσοδος είναι 'None' ή '(Default)' τότε το πακέτο που θα φορτώσουμε είναι αυτό που έχει οριστεί ως default από το αρχείο «settings.ini». Σε κάθε άλλη περίπτωση, αυτό που θα φορτωθεί στην είναι αυτό που όριζεται στην είσοδο.
	
	Get(self): μας επιστρέφει το φορτωμένο πακέτο.
	
	GetDefault(self): μας επιστρέφει το πακέτο που είναι διαθέσιμο αυτή στιγμή σήμερα.

NLPSpacyHelper.JSONSerialize

Η κλάση αυτή είναι σε θέση να μετατρέπει αντικείμενα σε JSON συμβολοσειρές και αντίστροφο αντίστοιχα. Η μετατροπή αυτή μας βοηθάει στην άμεση επεξεργασία των δεδομένων ως αντικείμενα. Η βιβλιοθήκη που μας βοήθησε είναι «json_serializer.Serializer».

Η κλάση διαθέτη δύο μεθόδους, την μετατροπή του αντικειμένου σε συμβολοσειρά JSON και την αντίστροφη. Παρακάτω παραθέτουμε τον κώδικα:

class JSONSerialize:
  def GetJson(self, obj):
  serializer = Serializer([
  obj
  ])
  string = serializer.serialize(obj)
  return string
 	
  def GetObject(self, strjson):
  object_data = json.loads(strjson)
  return object_data
	GetJSON(self, obj): μετατροπή του αντικειμένου σε συμβολοσειρά JSON. Δέχεται σαν είσοδο ένα αντικείμενο. Επιστρέφει την συμβολοσειρά JSON.
	
	GetObject(self, strjson): μετατροπή της συμβολοσειράς JSON σε αντικείμενο. Δέχεται στην είσοδο μια συμβολοσειρά JSON. Επιστρέφει το αντικείμενο.

NLPSpacyHelper.ReadConfigSettings

Η κλάση αυτή πρόκειται για έναν βοηθητικό μηχανισμό ώστε να επιστρέφει άμεσα τα δεδομένα από το παραμετρικό αρχείο «settings.ini». Διαθέτη μια μόνο μέθοδο, όπου έχει ως είσοδο την περιοχή (session), το κλειδί (key) και το όνομα του αρχείου με προεπιλεγμένη τιμή αυτή του «settings.ini».

Παραθέτουμε τον κώδικα:

```
class ReadConfigSettings:
  def GetData(self, session, key, file=r'settings.ini'):
    conf = ConfigParser()
    conf.read(file)
        return str(conf.get(session, key))
```

NERModel.py

Το αρχείο περιέχει τις κλάσεις και μεθόδους σχετικά με το μοντέλο του NER. Πρόκειται μια από τις βασικές κλάσεις καθώς, διαχειρίζεται τις απαιτήσεις του NER τόσο για το σχήμα που χρειάζεται για την προβολή του, όσο και για την εκπαίδευση του. Το αρχείο διαθέτει τρεις κλάσεις, την NerToken, NerModel, NerTrainingModel, NerSaveDataModel. Οι βιβλιοθήκες που έχουμε προσθέσει είναι η ακόλουθες.

```
from NLPSpacyModel.NLPSpacyHelper import Spacy as sp   #βλ. Προηγούμενη ενότητα
from NLPSpacyModel.NLPSpacyHelper import JSONSerialize #βλ. Προηγούμενη ενότητα
import re
```

NERModel.NerTokens

Η δομή διαθέτει χαρακτηριστικά που διαθέτη ένα token, για την αναγνώριση του token. Αυτό που χρειαζόμαστε για την εφαρμογή ότι αφορά τον εντοπισμό του NER είναι το κείμενο του token και η ετικέτα όπου έχει κατηγοριοποιηθεί.

Παραθέτουμε τον κώδικα:

class NerTokens:
text = ""
NER = ""
 	
def __init__(self, text=None, ner=None):
self.text = text
self.ner = ner
	text: το κείμενο, όπου περιέχει το token.
	ner: η ετικέτα που έχει κατηγοριοποιηθεί το κείμενο
	
	__init__(self, text=None, ner=None): πρόκειται για τον αρχικοποιητή, όπου στην είσοδο του δέχεται το κείμενο και την ετικέτα του token. Για τα ορίσματα στην είσοδο είναι αρχικοποιημένα με «None»

NERModel.NerModel

Αφορά την βασική δομή για την προβολή τον tokens, που διαθέτη το κείμενο, με την ετικέτα NER στην όποια έχει κατηγοριοποιηθεί. Επιπλέων, διαθέτη μηχανισμούς που είναι απαραίτητοι για την εκπαίδευση του NER. Πρόκειται για μια δομή αρκετούς μηχανισμούς, θα την δούμε σιγά σιγά.

Για αρχή έχουμε τα χαρακτηριστικά και τον αρχικοποιητή που διαθέτη:

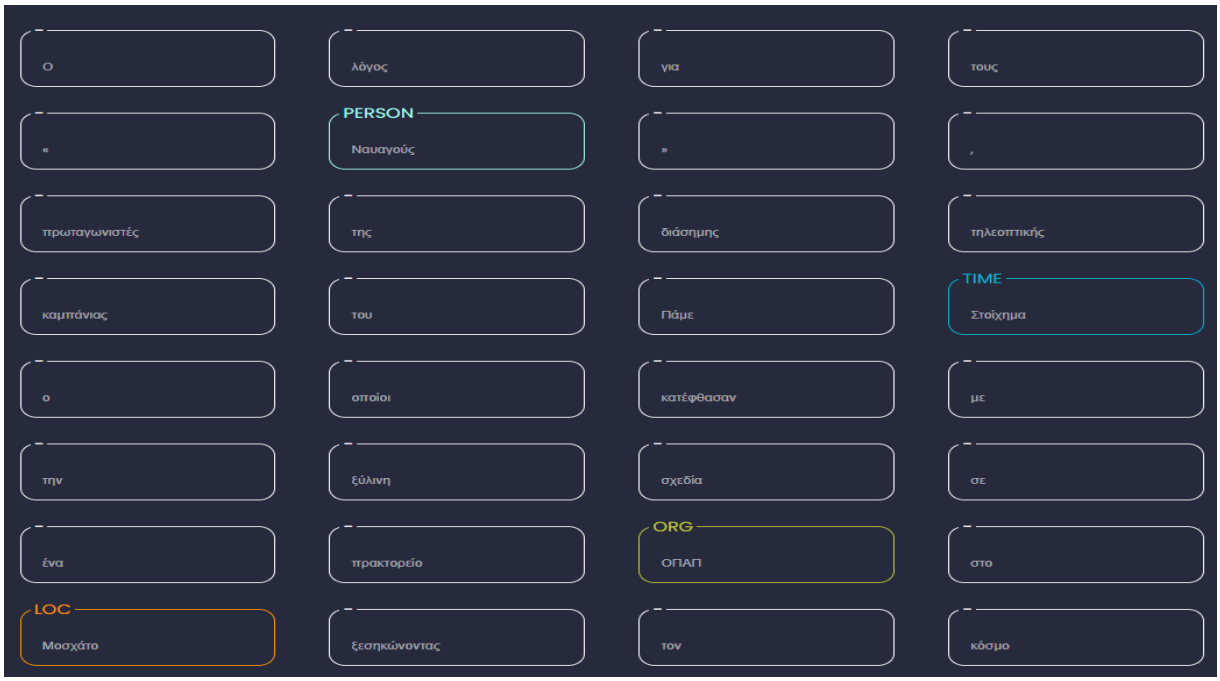
class NerModel:
id = ""
paragraph = ""
tokens = []
 	
def __init__(self, id=None, paragraph=None, tokens=[], package=None):
self.nlp = sp(nlp=package).Get() #φορτώνουμε το πακέτο, με το όποιο θα γίνει ανάλυση του κειμένου.
self.id = id
self.paragraph = paragraph
self.tokens = tokens
	Χαρακτηριστικά
	id: ένα μοναδικό κωδικό.
	paragraph: το κείμενο προς ανάλυση και εντοπισμό των NERs.
	tokens: πίνακας, όπου διαθέτει αντικείμενα NerToken (βλ. δομή NERModel.NerToken).
	
	Αρχικοποιητής
	__init(self, id=None, paragraph=None, tokens=[], package=None): ο αρχικοποιητής διαθέτη τέσσερα ορίσματα στην είσοδο του, με το καθένα να έχει προεπιλεγμένη τιμή το None ή άδειο κενό. Στην είσοδο μπορεί να λάβει το id, το paragraph, το tokens και το

self.GetTokensNer() #κλίση μηχανισμού, όπου θα πραγματοποιήσει τον εντοπισμό του NER για κάθε token	package. Τα πρώτα τρία είναι αυτά που διαθέτει στα χαρακτηριστικά της δομής του. Το package πρόκειται για ένα όρισμα, όπου μπορούμε να δηλώσουμε με ποιο πακέτο θέλουμε να γίνει ανάλυση του κειμένου.

```
def GetTokensNer(self):
    self.tokens = []
    doc = self.nlp(str(self.paragraph))
    has_ent = []
    isAppended = []
    for ent in doc.ents:
        has_ent.append(str(ent.text))
    print(has_ent)
    for token in doc:
        for one_ent in has_ent:
            if str(token) in one_ent and (str(token) not in isAppended and (one_ent) not in isAppended):
                for ent in doc.ents:
                    if str(token) in str(ent.text) and (str(token) not in isAppended):
                        enttxtsplit = ent.text.split()
                        if len(enttxtsplit) > 1 and len(str(token)) != len(str(ent.text)) \
                                or (len(enttxtsplit) == 1 and len(str(token)) == len(str(ent.text))):
                            isAppended.append(str(token))
                            isAppended.append(str(ent.text))
                            self.tokens.append(NerTokens(text=ent.text, ner=ent.label_))
        if str(token) not in isAppended:
            isAppended.append(str(token))
            self.tokens.append(NerTokens(text=str(token), ner="-"))
    print(isAppended)
    self.FixeDataNerToken()
    for token in self.tokens:
        print(token.text + " | " + token.ner + "\n")
```

GetTokensNer(self): Εντοπισμός των ners σε ένα κείμενο. Με βάση το πακέτο που έχουμε επιλέξει από την αρχή, θα πραγματοποιηθεί η του κειμένου (paragraph), και όπου μας επιστρέφει μια δομή doc. Με την σειρά της η δομή αυτή διαθέτη το χαρακτηριστικό ents, όπου περιέχει τα NER που έχουν εντοπιστεί.
Τα κείμενα από τα ents τα διατηρούμε στην has_ent. Ώστε να πραγματοποιήσουμε αμέσως μετά τον έλεγχο για κάθε token του κειμένου αν βρίσκεται στο has_ent.

Η συνάρτηση «GetTokensNer», αφού εντοπίσει τα NERs για κάθε token του κειμένου, καλεί με την σειρά της, την βοηθητική συνάρτηση «FixeDataNerToken» με σκοπό να δημιουργήσει μια ομοιόμορφη κατανομή της εικόνας του κειμένου προς τα NERs. Παρακάτω παραθέτουμε ένα παράδειγμα χρήσης και μη της συνάρτησης «FixeDataNerToken», για τον εντοπισμό των NER στο κείμενο : «Ο λόγος για τους «Ναυαγούς» , πρωταγωνιστές της διάσημης τηλεοπτικής καμπάνιας του Πάμε Στοίχημα ο οποίοι κατέφθασαν με την ξύλινη σχεδία σε ένα πρακτορείο ΟΠΑΠ στο Μοσχάτο ξεσηκώνοντας τον κόσμο».



Μη χρήση FixeDataNerToken

Χρήση FixeDataNerToken

def FixeDataNerToken(self):
m_tokens = self.tokens.copy()
self.tokens.clear()
text = ""
prevner = ""
nernow = "-"
for token in m_tokens:
if prevner != "" and prevner != nernow:
if text != "" and text != " ":
self.tokens.append(NerTokens(text=text, ner=nernow))
text = str(token.text)
elif str(token.ner) == nernow:
prevner = nernow
text = text + " " + str(token.text)
else:
self.tokens.append(NerTokens(text=text, ner=nernow))
text = str(token.text)
nernow = str(token.ner)
self.tokens.append(NerTokens(text=text, ner=nernow))
	FixeDataNerToken(self):
	Στόχος του μηχανισμού είναι να μπορεί να εντοπίζει αν υπάρχουν γειτονικά tokens με την ίδια ετικέτα ner, ώστε να τα τοποθετηθούν μαζί.

def FixWordSpaces(self, word):
rword = ""
words = word.split(" ")
for w in words:
if w == "" or w == " ":
continue
rword += w + " "
return rword
	FixWordSpaces(self, word):
	Στόχος είναι να διορθώσουμε το κείμενο όπου περιέχει πάνω από ένα κενό χαρακτήρα
	Είσοδος: μια λέξη/κείμενο
	Έξοδος: την νέα συμβολοσείρα χωρίς περιττά κενά.

def FindPositionInParagraph(self, paragraph, word):
word = self.FixWordSpaces(word).strip()
for match in re.finditer(word, paragraph):
start = match.start()
end = match.end()
return str("""" + str(start) + "", "" + str(end) + """")
	FindPositionInParagraph(self, paragraph, word):
	Η συνάρτηση έχει στόχο να εντοπίσει το συμβολοσείρες μέσα στο κείμενο.
	Είσοδος: το κείμενο, την συμβολοσείρα (που θέλουμε να εντοπίσουμε)
	Έξοδος: επιστρέφει τη θέση του πρώτου/τελευταίου χαρακτήρα της συμβολοσειράς μέσα στο κείμενο.

Οι επόμενοι μηχανισμοί που περιλαμβάνονται στη NerModel, χρησιμοποιούνται κυρίως στο κομμάτι της διόρθωσης/μετατροπής του αποτελέσματος. Μέχρι τώρα, βλέπουμε το κείμενο να αναλύεται και να απεικονίζει τα NER που έχουν εντοπιστεί. Για να μπορέσουμε να επαν-εκπαιδεύσουμε το πακέτο με διορθώσεις που απαιτούνται δημιουργήθηκαν οι παρακάτω μέθοδοι, όπου με βάσει τις διορθώσει που έστειλε το front-end, δημιουργεί την αντιστοιχεί γραμμογράφηση που απαιτείται για την εκπαίδευση του πακέτου.

def ClearDuplicateEntities(self, listentities): return list(dict.fromkeys(listentities))	ClearDuplicateEntities(self, listentities): Στόχος του μηχανισμού είναι ο καθαρισμός των NER που αναφέρονται στο ίδιο token. Είσοδος: μια λίστα με τα entities. Έξοδος: την λίστα με τα μοναδικά entities.

def GetEntities(self):
restr = []
for token in self.tokens:
if token is None:
continue
if token['ner'] == '-':
continue
restr.append('("' + self.FindPositionInParagraph(self.paragraph, word=str(token['text'])) +
'", \"" + token['ner'] + "\')"')
return self.ClearDuplicateEntities(restr)
	GetEntities(self):
	Η συνάρτηση μας επιτρέπει για κάθε token να βρούμε την θέση τους και την ετικέτα του NER που έχει εντοπιστεί.
	Έξοδος: επιστρέφει την λίστα με τα entities.

def ClearData(self): self.paragraph = self.paragraph.replace("'", "'\\\'")	ClearData(self): Σκοπός του μηχανισμού είναι να αντικαταστήσει τα «'» με την γραμμογράφηση «\\\'», και αυτό γιατί το κείμενο θα πρέπει να περιλαμβάνει μόνα εισαγωγικά στην αρχή και στο τέλος του κειμένου. Οπουδήποτε αλλού μέσα στο κείμενο θα πρέπει να έχει διαφορετική γραμμογράφηση.

def is_Corected(self, result):
if result is [] or len(result) == 0 or result is None:
return False
else:
return True
	is_Corected(self, result):
	Προτού προχωρήσουμε στην ενημέρωση του αρχείου με νέα εγγραφή, καλό θα ήταν να ελέγξουμε ότι έχουμε δεδομένα για να αποθηκεύσουμε, δηλαδή αν το Entities περιέχει δεδομένα.
	Έξοδος: επιστρέφει False αν δεν υπάρχουν δεδομένα, σε κάθε άλλη περίπτωση True.

def WriteInFile(self):
resultentities = self.GetEntities()
self.ClearData()
if self.is_Corected(result=resultentities):
	WriteInFile(self):
	Πρόκειται για τον μηχανισμό, οποίος γράφει την νέα εγγραφή στο

resultline = '"(\"" + self.paragraph + "\', { \'entities\': '" + str(resultentities) + '" })'" with open("trainingnerfile.txt", "a", encoding='UTF-8') as f: f.write(resultline + "\n") f.close()	αρχείο που προορίζεται για την εκπαίδευση.

NERModel. NerTrainingModel

Η κλάση «πατέρας», είναι η δομή που περιέχει τα χαρακτηριστικά όπως ένα id και ένα αντικείμενο την δομής NerModel.

class NerTrainingModel:
id = 0
nermodel = NerModel()
 	
def __init__(self, id, text, package=None):
self.id = id
self.text = text
self.nermodel = NerModel(id, paragraph=text, package=package)
	NerTrainingModel:
	Χαρακτηριστικά: ένα id, και το μοντέλο NerModel.
	Κατασκευαστής:
	Είσοδος: ένα id, το κείμενο και το πακέτο που θα χρησιμοποιηθεί για την ανάλυση του κειμένου.

NERModel. NerSaveDataModel

Αυτή η δομή πρόκειται για μια βοηθητική κλάση, με σκοπό τη μετατροπή του json σχήματος σε ένα obj. Αυτό μας βοήθησε στην διαδικασία της ενημέρωσης του αρχείου που είναι προς εκπαίδευση. Στόχος μας ήταν η διεπαφή να δημιουργήσει ένα αντίστοιχο σχήμα όπως εκείνο του response που λαμβάνει, σχετικά με τα tokens. Όπως, είδαμε πιο πάνω δημιουργούμε ένα μοντέλο του τύπου “text: «κείμενο», ner: «Ner_Type»”, θα πρέπει και το request που αφορά την μεταβολή του αρχείου να έχει την αντίστοιχή εικόνα, ώστε να μην χρειάζεται να κάνουμε διαδικασίες που ίσως επηρέαζαν τα δεδομένα.

class NerSaveDataModel:
obj = None
 	
def __init__(self, data=None):
self.obj = JSONSerialize().GetObject(strjson=data)
	NerSaveDataModel:
	Χαρακτηριστικά: ένα αντικείμενο
	Κατασκευαστής:
	Είσοδος: τα δεδομένα
	Μετατρέπει τα δεδομένα από Json string σε αντικείμενο.

def Get(self): return self.obj	

NLPSpacyGeneralModel.py

Το επόμενο αρχείο περιέχει δομές και μηχανισμού πιο γενικούς και σχετίζονται με κάποιες από τις λειτουργίες που διαθέτει επιπλέον η εφαρμογή. Η εφαρμογή εκτός από την εκπαίδευση του NER πεδίου και την συναισθηματική ανάλυση, διαθέτη μια ακόμα λειτουργία πιο γενική, όπου χρήστης μπορεί να δώσει κείμενα γραμμένα στα ελληνικά και να πραγματοποιηθεί συντακτική/συναισθηματική ανάλυση ακόμα των συντακτικών εξαρτήσεων.

Για αρχή έχουμε προσθέσει τις παρακάτω βιβλιοθήκες, όπως θα δείτε πολλές από αυτές είναι από το NLPSpacyModel που αναφέραμε προηγούμενος. Εκτός από αυτά τα δικά μας μοντέλα έχουμε 3 επιπλέων βιβλιοθήκες το displacy από το spacy, το glob και το json.

Το αρχείο περιέχει οκτώ δομές, τις NLPSpacyDoc, NLPSpacyGetToken, NLPSpacyAllDoc, NLPSpacyTokenDep, NLPSpacyAllDep, NerTraining, EmotionAnalysisTraining, GetAllFilesSentiment, GetAllSpacyPackage, όπου θα τις αναλύσουμε παρακάτω.

```
import glob
from spacy import displacy
from NLPSpacyModel.NLPSpacyHelper import Spacy as sp
from NLPSpacyModel.NLPSpacyHelper import JSONSerialize
from NLPSpacyModel.NLPSpacyHelper import ReadConfigSettings as RCS
from NLPSpacyModel.NERModel import NerTrainingModel
import json
```

NLPSpacyGeneralModel.NLPSpacyDoc

Η κλάση αυτή πρόκειται για μια γενικευμένη δομή, όπου περιέχει τα πιο βασικά στοιχεία ενός κειμένου μετά την ανάλυση του.

class NLPSpacyDoc:
  paragraph = None
  tokens = []
  lemma = []
  pos = []
  tags = []
	NLPSpacyDoc:
	Η δομή με τα αποτελέσματα των βασικών στοιχείων του κειμένου μετά από την ανάλυση του.
	Χαρακτηριστικά: το κείμενο (paragraph), μια λίστα με τα tokens,

def __init__(self, paragraph=None, tokens=[], lemma=[], pos=[], tags=[]): self.paragraph = paragraph self.tokens = tokens self.lemma = lemma self.pos = pos self.tags = tags	μια λίστα με τα λήμματα, μια λίστα με τα pos και μια λίστα με τα tags. Κατασκευαστής: στόχος να μεταφέρει την πληροφορία από την είσοδο του στα χαρακτηριστικά του.

NLPSpacyGeneralModel.NLPSpacyGetToken

class NLPSpacyGetToken:
nlplistdoc = []
alltext = None
 	
def __init__(self, texts=None, package=None):
self.nlplistdoc = []
self.alltext = None
if texts is None:
with open("runuploadfile.txt", "r", encoding='UTF-8') as f:
texts = f.readlines()
f.close()
self.text = texts
self.nlp = sp(nlp=package).Get()
for txt in self.text:
doc = self.nlp(txt)
if txt[0] == '#':
txt = txt.replace('#', '')
bsplit = True
else:
bsplit = False
if len(list(doc.sents)) > 2 and bsplit is not True:
self.alltext = str(doc.text)
for sent in list(doc.sents):
self.tokens = [str(do) for do in sent]
self.lemma = [str(lm.lemma_) for lm in [token for token in sent]]
	NLPSpacyGetToken:
	Έχει σκοπό την ανάλυση του κειμένου και να συλλέξει όλα τα απαραίτητα χαρακτηριστικά.
	Χαρακτηριστικά: μια λίστα από αντικείμενα του τύπου NLPSpacyDoc, όλο το κείμενο που έχει δοθεί.
	Κατασκευαστής: στην είσοδο του μπορεί να λάβει το κείμενο και το όνομα του πακέτου που θα χρησιμοποιηθεί στην ανάλυση.
	Αν το κείμενο δεν δοθεί τότε θα προσπαθήσει να διαβάσει από τα δεδομένα που έχουν ανέβει μέσω της εφαρμογής.
	Επίσης, κατασκευαστής διαθέτει δύο ροές, μπορεί να αναλύσει όλο το κείμενο ή μπορεί να αναλύσει κάθε πρόταση του κειμένου αν αυτό διαθέτει πάνω από μια.

self.pos = [str(lm.pos_) for lm in [token for token in sent]]
self.tag = [str(lm.tag_) for lm in [token for token in sent]]
self.nlplistdoc.append(NLPSpacyDoc(paragraph=str(sent), tokens=self.tokens,
lemma=self.lemma, pos=self.pos, tags=self.tag))
else:
self.tokens = [str(do) for do in doc]
self.lemma = [str(lm.lemma_) for lm in [token for token in doc]]
self.pos = [str(lm.pos_) for lm in [token for token in doc]]
self.tag = [str(lm.tag_) for lm in [token for token in doc]]
self.nlplistdoc.append(NLPSpacyDoc(paragraph=txt, tokens=self.tokens, lemma=self.lemma, pos=self.pos,
tags=self.tag))
 	
def GetReturnNLPList(self):
return self.nlplistdoc
 	
def GetReturnAllText(self):
return self.alltext
	Μέθοδοι:
	GetReturnNLPList(self):
	Επιστρέφει την λίστα που περιέχει τα αντικείμενα του τύπου NLPSpacyDoc.
	
	GetReturnAllText(self):
	Επιστρέφει όλο το κείμενο αν αυτό διαθέτει πάνω από μια πρόταση.

Η δομή πραγματοποιεί την ανάλυση του κειμένου, κάνοντας χρήση το πακέτο που επιθυμεί ο χρήστης. Να θυμίσουμε εδώ ότι το προεπιλεγμένο πακέτο είναι αυτό που εκπαιδεύσαμε στην παρούσα εργασία.

NLPSpacyGeneralModel. NLPSpacyAllDoc

Πρόκειται για μια δομή, όπου διαθέτει ένα χαρακτηριστικό αυτό του τύπου «NLPSpacyGetToken» και έναν μηχανισμό.

class NLPSpacyAllDoc:
nlpspacygettoken = None
 	
def __init__(self, texts=None, package=None):
self.nlpspacygettoken = NLPSpacyGetToken(texts=texts, package=package)
 	
def Get(self):
return JSONSerialize().GetJson(obj=self.nlpspacygettoken)
	NLPSpacyAllDoc:
	Σκοπός του είναι να δημιουργήσει ένα αντικείμενο τύπου NLPSpacyGetToken και με την βοήθεια της μεθόδου Get αυτό να είναι αμέσα διαθέσιμο στην μορφή ενός json σχήματος.

	Χαρακτηριστικά: ένα αντικείμενο τύπου NLPSpacyGetToken Μέθοδοι: Get(self): Επιστρέφει το json σχήμα του χαρακτηριστικού του.

NLPSpacyGeneralModel. NLPSpacyTokenDep

Η δομή διαθέτει μια γενική εικόνα σχετικά με την συντακτική εξάρτηση που υπάρχει μεταξύ των tokens ενός κειμένου.

class NLPSpacyTokenDep:
  tokens = []
  dep = []
  graphs = None
 	
  def __init__(self, tokens=[], dep=[], graphs=None):
  self.tokens = tokens
  self.dep = dep
  self.graphs = graphs
	NLPSpacyTokenDep:
	Η γενική εικόνα των συντακτικών εξαρτήσεων.
	Χαρακτηριστικά: μια λίστα tokens, μια λίστα με τις συντακτικές εξαρτήσεις και ένα αντικείμενο που θα αποθηκεύσουμε το γράφημα.
	Κατασκευαστής: θα πρέπει να μεταφέρει την πληροφορία από την είσοδο του, στα χαρακτηριστικά του.

NLPSpacyGeneralModel. NLPSpacyAllDep

Η δομή πραγματοποιεί ελέγχους για την συντακτική εξάρτηση κάθε κειμένου που έχει δοθεί ως είσοδο.

class NLPSpacyAllDep:
  tokens_dep = []
 	
  def __init__(self, texts=None, package=None):
  if texts is None:
  with open("runuploadfile.txt", "r", encoding='UTF-8') as f:
  texts = f.readlines()
  f.close()
  self.text = texts
  self.nlp = sp(nlp=package).Get()
	NLPSpacyAllDep:
	Υπολογίζει τις συντακτικές εξαρτήσεις όλων των κειμένων.
	Κατασκευαστής: στην είσοδο μπορεί να λάβει το κείμενο και το πακέτο για την ανάλυση του. Ο κατασκευαστής μοιάζει με εκείνον του «NLPSpacyGetToken», σχετικά με το αν έχει δοθεί ή όχι κείμενο αλλά και την δυνατότητα να αναλύσει όλο το

self.tokens_dep = []
for txt in self.text:
doc = self.nlp(txt)
if txt[0] == '#':
txt = txt.replace('#', '')
bsplit = True
else:
bsplit = False
if len(list(doc.sents)) > 2 and bsplit is not True:
for sent in list(doc.sents):
self.token = [str(do) for do in sent]
self.dep = [str(lm.dep_) for lm in [token for token in sent]]
self.svg = displacy.render(sent, style="dep", jupyter=False)
 	
self.tokens_dep.append(NLPSpacyTokenDep(tokens=self.token, dep=self.dep, graphs=self.svg))
else:
self.token = [str(do) for do in doc]
self.dep = [str(lm.dep_) for lm in [token for token in doc]]
self.svg = displacy.render(doc, style="dep", jupyter=False)
 	
self.tokens_dep.append(NLPSpacyTokenDep(tokens=self.token, dep=self.dep, graphs=self.svg))
 	
def Get(self):
return JSONSerialize().GetJson(self.tokens_dep)
	κείμενο ή κάθε πρόταση του κειμένου. Η ανάλυση αφορά μόνο τις συντακτικές εξαρτήσεις. Στο τέλος, θα έχουμε μια λίστα του τύπου «NLPSpacyTokenDep».
	Μέθοδοι:
	Get(self):
	Επιστρέφει το json σχήμα του χαρακτηριστικού του.

NLPSpacyGeneralModel.NerTraining

Η δομή αφορά το μέρος της εκπαίδευσης του NER πεδίου. Με βάση τα δεδομένα που έχει λάβει επιστρέφει την εικόνα που έχει σχηματίσει το «NerTrainingModel». Τα δεδομένα, μπορεί να είναι είτε από αρχείο που έχει γίνει uploaded από τον χρήστη είτε από αρχεία (που χρησιμοποιήθηκαν στο gsoc2018) που είναι διαθέσιμα στην εφαρμογή.

class NerTraining: nertraining = [] count = 0	NerTraining: Σκοπός είναι να επιστρέφει το NerTrainingModel με

default_examples = False
package_default_example = '1'
 	
def __init__(self, texts=None, count=0, package=None, default_examples=False, package_default_example='1'):
self.count = count
self.nertraining = []
self.package_default_example = package_default_example
if texts is None:
if default_examples:
ner_datasets_json = RCS().GetData('training_ner_data', 'path_ner' + package_default_example +'_parts')
with open(ner_datasets_json, 'r') as File:
datasets_ner_file = File.readlines()
TRAIN_DATA_NER = [json.loads(st) for st in datasets_ner_file]
for position, jsonobj in enumerate(TRAIN_DATA_NER):
if position == self.count:
self.text = str(jsonobj.get('text'))
break
else:
with open("runuploadfile.txt", "r", encoding='UTF-8') as f:
for position, line in enumerate(f):
if position == self.count:
self.text = str(line).replace('\n', '')
break
f.close()
self.nertraining.append(NerTrainingModel(id=self.count, text=str(self.text), package=package))
 	
def Get(self):
return JSONSerialize().GetJson(obj=self.nertraining)
	βάσει τα δεδομένα που έχει δώσει ο χρήστης.
	Χαρακτηριστικά: μια λίστα που περιέχει αντικείμενα του τύπου «NerTrainingModel», ένας μετρητής, μια μεταβλητή για το αν γίνεται χρήση τον παραδειγμάτων που διαθέτει η εφαρμογή και τέλος το προεπιλεγμένο αρχείο παραδειγμάτων που είναι ορισμένο με το πρώτο.
	Κατασκευαστής: στην είσοδο του μπορεί να λάβει όσο και τα χαρακτηριστικά του, αλλά και το όνομα του πακέτου που θα κάνει τον εντοπισμό του NER.
	Πραγματοποιεί δύο ενέργειες ανάλογα αν θα διαβάσει από το ανεβασμένο αρχείο ή από ήδη υπάρχοντα αρχεία. Ανεξάρτητα από που θα διαβάσει, αυτό που μας ενδιαφέρει είναι το κείμενο και το πακέτο, όπου θα δοθούν ως είσοδο στο «NerTrainingModel»
	
	Μέθοδοι:
	Get(self): επιστρέφει το json σχήμα της λίστας με τα αντικείμενα «NerTrainingModel»

NLPSpacyGeneralModel. EmotionAnalysisTraining

Η δομή είναι αντίστοιχη με αυτή που είδαμε προηγουμένως («NerTraining»), με την διαφορά ότι δεν μας ενδιαφέρει η λίστα με τα «NerTrainingModel». Αυτό που μας ενδιαφέρει είναι ο μηχανισμό του κατασκευαστή του «NerTraining», για την επιλογή του κειμένου είτε από αρχείο του χρήστη είτε από τα ήδη αρχεία που είναι αποθηκευμένα. Η δομή χρησιμοποιείται για την λειτουργία της εκπαίδευσης του συναισθήματος, για αυτό το λόγο θα θέλαμε να το διαχωρίσουμε από την προηγούμενη δομή.

```
class EmotionAnalysisTraining:
    text = None
    default_examples = False
    package_default_example = '1'

    def __init__(self, count=0, default_examples=False, package_default_example='1'):
        self.count = int(count) - 1
        if default_examples:
            ner_datasets_json = RCS().GetData('training_ner_data', 'path_ner' + package_default_example + '_parts')
            with open(ner_datasets_json, 'r') as File:
                datasets_ner_file = File.readlines()
            TRAIN_DATA_NER = [json.loads(st) for st in datasets_ner_file]
            for position, jsonobj in enumerate(TRAIN_DATA_NER):
                if position == self.count:
                    self.text = str(jsonobj.get('text'))
                    break
        else:
            with open("runuploadfile.txt", "r", encoding='UTF-8') as f:
                for position, line in enumerate(f):
                    if position == self.count:
                        self.text = line
                        break
                f.close()
        print(self.text)

    def GetText(self):
        return self.text
```

EmotionAnalysisTraining:

Σκοπός είναι ο εντοπισμός του κειμένου με βάση τον μετρητή που έδωσε ο χρήστης.

Χαρακτηριστικά: το κείμενο, μια μεταβλητή για τον αν θα γίνει χρήση των παραδειγμάτων που διαθέτει η εφαρμογή.

Κατασκευστής: στην είσοδο του δέχεται την πληροφορία για το αν θα διαβάσουμε από τα υπάρχων αρχεία, με από ποιο αρχείο που είναι διαθέσιμα και το μετρητή για την επιλογή του παραδείγματος.

Μέθοδοι:

GetText(self): επιστρέφει το κείμενο που έχει βρεθεί.

NLPSpacyGeneralModel. GetAllFilesSentiment και NLPSpacyGeneralModel. GetAllSpacyPackage

Τέλος, ακολουθούν δυο επιπλέων δομές η «GetAllFilesSentiment» και η «GetAllSpacyPackage», όπου είναι καθαρά βοηθητικές και ο σκοπό έχουν να εντοπίσουν τα πακέτα που είναι διαθέσιμα για τον χρήστη, όπως τα συναισθηματικά και τα πακέτα ανάλυσης (spacy) αντίστοιχα.

class GetAllFilesSentiment:
file_list = []
 	
def __init__(self):
self.file_list = [str(f) for f in glob.glob("*.tsv")]
self.file_list.append('greek_sentiment_lexicon.tsv (Default)')
 	
def GetList(self):
return self.file_list
	GetAllFilesSentiment:
	Σκόπος της δομής είναι ο εντοπισμός των συναισθηματικών λεξικών.
	
	Χαρακτηριστικά: μια λίστα με τα ονόματα των λεξικών
	
	Κατασκευαστής: με την βοήθεια της βιβλιοθήκης «glob» αναζητάει σε όλο το project τα αρχεία που έχουν κατάληξη «.tsv»
	
	Μέθοδοι:
	GetList(self): επιστρέφει την λίστα με τα ονόματα.

class GetAllSpacyPackage:
dir_list = []
 	
def __init__(self):
self.dir_list = [str(f) for f in glob.glob("TrainingPackageUser/*")]
self.dir_list.append('(Default)')
 	
def GetList(self):
return self.dir_list
	GetAllSpacyPackage:
	Σκόπος της δομής είναι ο εντοπισμός των πακέτων spacy που έχουν δημιουργηθεί μετά την εκπαίδευση τους.
	
	Χαρακτηριστικά: μια λίστα με τα ονόματα των πακέτων
	
	Κατασκευαστής: με την βοήθεια της βιβλιοθήκης «glob» αναζητάει σε συγκεκριμένο φάκελο του project τα αρχεία.

	Μέθοδοι: GetList(self): επιστρέφει την λίστα με τα ονόματα των πακέτων.

TrainingNerField.py

Στη συνέχεια έχουμε το αρχείο όπου αποτελεί κορμό για τον μηχανισμό εκπαίδευσης του NER πεδίου. Το αρχείο περιέχει δομές και μηχανισμού, ώστε να εντοπίσουν τα νέα παραδείγματα και να πραγματοποιηθεί η εκ νέου η εκπαίδευση του. Το αρχείο περιέχει δύο δομές, την «ReadTrainingData» και την «TrainingNer» ενώ παρακάτω παραθέτουμε τις βιβλιοθήκες που χρειάστηκαν να εισάγουμε.

```
import ast

from numpy import random
from spacy.tokens import Doc, DocBin
from spacy.training import Example
from spacy.util import minibatch

from NLPSpacyModel.NLPSpacyHelper import Spacy as sp
from NLPSpacyModel.NLPSpacyHelper import ReadConfigSettings as rcs
```

TrainingNerField.ReadTrainingData

Η δομή έχει στόχο να διαβάσει τα παραδείγματα που έχουν δημιουργηθεί μέσω της εφαρμογής, ώστε να τροφοδότηση το σύστημα για την εκπαίδευση του NER.

class ReadTrainingData:
  TRAIN_DATA = None
 	
  def __init__(self):
  try:
  with open("trainingnerfile.txt", "r", encoding='UTF-8') as f:
  self.TRAIN_DATA = f.readlines()
  f.close()
  except:
  self.TRAIN_DATA = None
	ReadTrainingData:
	Σκοπό της δομής να διαβάσει τα δεδομένα που έχουν αποθηκευτή μέσω της εφαρμογής.
	Χαρακτηριστικά: μια λίστα με όλες τις εγγραφές του αρχείου “trainingnerfile”.
	
	Κατασκευαστής: διαβάζει από αρχείο «trainingnerfile» τις εγγραφές, όπου αποθηκεύονται στο χαρακτηριστικό της δομής.
	
	Μέθοδοι:

def GetData(self):
  if self.TRAIN_DATA is None:
  return None
  else:
  return str(self.TRAIN_DATA).replace('\'', '')
	GetData(self): μας επιστρέφει την λίστα ως μια συμβολοσείρα ενώ έχουμε «καθαρίσει» τα δεδομένα από δίπλα εισαγωγικά (‘‘).

TrainingNerField.TrainingNer

Η ακόλουθη δομή αποτελεί το βασικό μοντέλο, που πραγματοποιεί την εκπαίδευση του NER πεδίου και την δημιουργία ενός νέου πακέτου.

class TrainingNer:
  nlp = None
  Error = None
  Message = None
  examples = []
  datadoc_ = {}
 	
  def __init__(self, package_name=None):
  self.package_name = package_name
  self.TRAIN_DATA = ReadTrainingData().GetData()
  self.nlp = sp().GetDefault()
  # self.nlp = sp("TrainingPackageUser/test09012022_part2").Get()
  self.TRAIN_DATA_JSON = ast.literal_eval(str(self.TRAIN_DATA))
  try:
  self.start_Training()
  except KeyError:
  self.Error = True
  self.Message = KeyError
 	
  def start_Training(self):
  n_iter = 1000
  # n_iter = 500
  if 'ner' not in self.nlp.pipe_names:
  self.nlp.add_pipe('ner', last=True)
  NER = self.nlp.get_pipe('ner')
	TrainingNer:
	Βασική δομή, για την εκπαίδευση του NER πεδίου.
	
	Χαρακτηριστικά: ένα αντικείμενο nlp, όπου θα περιέχει το πακέτου που θα χρησιμοποιηθεί για την ανάλυση του κειμένου, μια μεταβλητή Boolean για τον υπάρχει κάποιο σφάλμα στην διαδικασία, το μήνυμα που θα επιστρέψουμε, μια λίστα με τα παραδείγματα που θα χρησιμοποιηθούν για την εκπαίδευση, τέλος μια μεταβλητή με την δομή «{'WORDS': tokens_, 'LEMMA':lemma_, 'MORPH': morph_}».
	
	Κατασκευαστής: δέχεται στην είσοδο το όνομα του πακέτου όπου θα χρησιμοποιήσουμε για την ανάλυση του κειμένου. Επιπλέον, χρησιμοποιεί την «ReadTrainingData» για να λάβει

```
self.examples = []
self.datadoc_ = {}
if self.TRAIN_DATA_JSON is not None:
    for ele in self.TRAIN_DATA_JSON:
        o, s = ast.literal_eval(ele)
        for ent in s.get('entities'):
            ner.add_label(ent[2])
    for ele in self.TRAIN_DATA_JSON:
        text, ent = ast.literal_eval(ele)
        docel = self.nlp(str(text))
        tokens_ = []
        lemma_ = []
        morph_ = []
        doctemp = Doc(self.nlp.vocab, words=[token.text for token in docel])
        for token in docel:
            try:
                tokens_.append(str(token))
                lemma_.append(token.lemma_)
                morph_.append(str(token.morph))
            except KeyError:
                continue
        self.examples.append(Example.from_dict(doctemp, ent))
        try:
            self.datadoc_ = {'WORDS': tokens_,
                             'LEMMA': lemma_,
                             'MORPH': morph_
                            }
            self.examples.append(Example.from_dict(doctemp, self.datadoc_))
        except:
            continue
print('start_training...')
# optimizer = nlp.begin_training()
optimizer = self.nlp.create_optimizer()
for itn in range(n_iter):
    random.shuffle(self.examples)
    losses = {}
```

τα δεδομένα ενώ στο τέλος καλεί την μέθοδο «start_Training», όπου με την σειρά ξεκινάει την διαδικασία εκπαίδευσης.

Μέθοδοι:

start_Training(self): εκκίνηση διαδικασίας και αποθήκευσης νέου πακέτου. Η διαδικασία έχει αναλυθεί στη ενότητα «Εκπαίδευση του NER».

Get(self): επιστρέφει πως υπάρχει error και το μήνυμα του σφάλματος αν υπάρχει, σε κάθε άλλη περίπτωση επιστρέφει το «OK»

```
        # losses = morphologizer.update(examples, sgd=optimizer)
        for batch in minibatch(self.examples, size=50):
            try:
                self.nlp.update(
                    batch,
                    drop=0.5,
                    sgd=optimizer,
                    losses=losses)
            except:
                continue
        print(losses)
    self.nlp.to_disk("TrainingPackageUser/"+self.package_name)

def Get(self):
    if self.Error:
        return {'Error': self.Error, 'Message': self.Message}
    return "OK"
```

EmotionModel.py

Τέλος, θα δούμε το αρχείο που διαθέτει τις δομές για το ότι έχει να κάνει με την συναισθηματική ανάλυση που πραγματοποιεί η εφαρμογή. Το αρχείο, διαθέτει τέσσερεις δομές, το «EmotionData», το «EmotionAnalysis», το «TempSaveEmotionsAndCalculate» και τέλος την «ReqTextDataModel». Όσο αφορά τις δύο πρώτες δομές από «EmotionData», το «EmotionAnalysis», είναι ίδιες που είδαμε και αναλύσαμε σε προηγούμενη ενότητα «Φάση 2η: Μηχανική Μάθηση και Εντοπισμός Συναισθήματος», επομένως συνιστούμε τον αναγνώστη να ανατρέξετε σε εκείνη την ενότητα για να θυμηθεί τα μοντέλα.

EmotionModel.TempSaveEmotionsAndCalculate και EmotionModel.ReqTextDataModel

Ας δούμε, όμως τις άλλες δύο πιο απλές δομές την «TempSaveEmotionsAndCalculate», όπου αφορά την προσωρινή αποθήκευση εγγραφής ενός νέου token που μπορεί να συμμετέχει στην συναισθηματική ανάλυση και την «ReqTextDataModel», όπου αποσκοπεί στην επιστροφή του κειμένου.

class TempSaveEmotionsAndCalculate:
matched_token = None
text = None
package = None
savetempfile = 'tempsentiment_calculate.tsv'
 	
def __init__(self, matched_tokens=None, text=None, package=None):
self.matched_token = matched_tokens
self.text = text
self.package = package
 	
def saveTempMatchedTokens(self):
try:
with open(self.savetempfile, 'w+', encoding='UTF-8') as f:
 	
f.write('"Term\tPOS1\tPOS2\tPOS3\tPOS4\tSubjectivity1\tSubjectivity2\tSubjectivity3\tSubjectivity4
 	
\tPolarity1\tPolarity2\tPolarity3\tPolarity4\tAnger1\tAnger2\tAnger3\tAnger4
 	
\tDisgust1\tDisgust2\tDisgust3\tDisgust4\tFear1\tFear2\tFear3\tFear4
 	
\tHappiness1\tHappiness2\tHappiness3\tHappiness4\tSadness1\tSadness2\tSadness3\tSadness4
 	
\tSurprise1\tSurprise2\tSurprise3\tSurprise4\tAditional1\tAditional2\tAditional3\tAditional4
 	
\tComments1\tComments2\tComments3\tComments4\t"')
f.write('\n')
for lines in self.matched_token:
f.write(str(lines) + '\t\n')
	TempSaveEmotionsAndCalculate:
	Σκοπός η προσωρινή αποθήκευση εγγραφής ενός νέου token που συμμετέχει στην συναισθηματική ανάλυση ενός κειμένου.
	
	Χαρακτηριστικά: μια λίστα με τα tokens που συμμετέχουν στην συναισθηματική ανάλυση, το κείμενο που εξετάζεται, το όνομα του συναισθηματικού λεξικού και το όνομα του αρχείου που αποθηκεύονται προσωρινά.
	
	Κατασκευαστής: στην είσοδο του λαμβάνει την λίστα με τα tokens, το κείμενο και το όνομα του συναισθηματικού λεξικού.
	Με τα δεδομένα από δημιουργεί το προσωρινό αρχείο.
	
	Μέθοδοι:
	Get(self): επιστρέφει την νέα συναισθηματική ανάλυση που πραγματοποιήθηκε κάνοντας χρήση το προσωρινό αρχείο.

f.close()
return True
except:
return False
 	
def Get(self):
if not self.saveTempMatchedTokens():
return "Error"
emotioanalysis = EmotionAnalysis(text=self.text, file_name=self.savetempfile)
emotioanalysis.Analysis()
resp = emotioanalysis.Get()
return resp	

class ReqTextDataModel:
text = None
 	
def __init__(self, text=None):
self.text = text
 	
def Get(self):
return self.text
	ReqTextDataModel:
	Σκοπό η δημιουργία μιας δομής με ένα χαρακτηριστικό που είναι το κείμενο.
	
	Χαρακτηριστικά: μια μεταβλητή για το κείμενο.
	
	Κατασκευαστής: δέχεται στην είσοδο του ένα κείμενο.
	
	Μέθοδοι:
	Get(self): επιστρέφει το κείμενο.

Αφού, ολοκληρώσαμε την μορφή του φακέλου “NLPSpacyModel” και είδαμε όλα τα αρχεία και τις δομές που χρησιμοποιεί η εφαρμογή του back-end. Στην συνέχεια, θα δούμε το φάκελο «NLPSpacyCommone» όπου περιέχει βασικές δομές που γίνονται χρήση από το back-end κατά την επικοινωνία με το front-end.

NLPSpacyCommone

Ο συγκεκριμένος φάκελος, δημιουργήθηκε με σκοπό να περιέχει αρχεία που να είναι κοινά σε άλλα επίπεδα του project. Πιο συγκεκριμένα, δημιουργήσαμε δύο αρχεία το «NLPSpacyCommoneAbsReqResp» όπου περιέχει γενικευμένες δομές για την διαχείριση των αιτημάτων/απαντήσεων (request/response) και το «NLPSpacyReqResp» όπου περιέχει μαζεμένα όλα τα αιτήματα/απατήσεις που διαχειρίζεται το back-end.

NLPSpacyCommoneAbsReqResp.py

Το αρχείο περιέχει γενικευμένες δομές, όπου αποσκοπούν στα μοντέλα για την επικοινωνία αιτήματος-απάντησης. Το αρχείο διαθέτει τρεις δομές, την «NLPSpacyCommoneAbs», την «NLPSpacyReq» και την «NLPSpacyResp». Επιπλέον, γίνεται χρήση μόνο μίας βιβλιοθήκης «from flask import jsonify», με σκοπό να μετατρέπει τα δεδομένα σε σειριακά της μορφής json και τα να «τυλίγει» σε ένα τύπο flask.Response ορίζοντας το mimetype σε application/json. Παρακάτω θα αναλύσουμε όλες τις δομές και μηχανισμούς που διαθέτουν.

NLPSpacyCommoneAbsReqResp.NLPSpacyCommoneAbs

Η δομή, αποτελεί ένα γενικό μοντέλο, όπου κληρονομείτε από μεταγενέστερες δομές και είναι αυτό που διευκολύνει στην εικόνα ενός ερωτήματος/απάντησης.

class NLPSpacyCommoneAbs:
response = None
obj = None
 	
def __init__(self, obj):
self.obj = obj
try:
self.response = jsonify(obj)
self.response.headers.add("Access-Control-Allow-Origin", "*")
except:
self.response = obj
 	
def GetSimpleObj(self):
return self.obj
	NLPSpacyCommoneAbs:
	Γενική δομή για την εικόνα στην επικοινωνία ενός ερωτήματος/απάντησης.
	
	Χαρακτηριστικά: μια μεταβλητή για την απάντηση και μια μεταβλητή που περιέχει το αντικείμενο, όπου θα μετατραπεί σε σειριακά δεδομένα της μορφής json.
	
	Κατασκευαστής: λαμβάνει στην είσοδο του ένα αντικείμενο και το μετατρέπει σε σειριακά δεδομένα όπου αποθηκεύονται στην μεταβλητή “response”.
	
	Μέθοδοι:
	GetSimpleObj(self): επιστρέφει το αντικείμενο που έλαβε.

Το παραπάνω μοντέλο κληρονομείτε στις επόμενες δύο δομές. Τα επόμενα δύο μοντέλα αναλαμβάνου κάθε άκρο ενός κανάλι επικοινωνίας, το ένα άκρο είναι το «ερώτημα» και το άλλο πρόκειται για την «απάντηση».

NLPSpacyCommoneAbsReqResp.NLPSpacyReq **και** NLPSpacyCommoneAbsReqResp. NLPSpacyResp

class NLPSpacyReq(NLPSpacyCommoneAbs):
def __init__(self, idName="Req", dataReq=None):
NLPSpacyCommoneAbs.__init__(self, {'idName': idName})
try:
if dataReq is not None:
self.dataReq = dataReq.decode('UTF-8')
except:
self.dataReq = dataReq
 	
def GetObject(self):
return jsonify(self.dataReq)
	NLPSpacyReq(NLPSpacyCommoneAbs):
	Η δομή έχει κληρονομηθεί με την σειρά της σε κάποια μοντέλα που έχουν άμεση σχέση με το «ερώτημα» που λαμβάνει το back-end.
	
	Χαρακτηριστικά: αυτά που έχει κληρονομήσει από την δομή «NLPSpacyCommoneAbs»
	
	Κατασκευαστής: λαμβάνει στην είσοδο του ένα κωδικό ονόματος (idName), συνήθως το όνομα της δομής που το έχει κληρονομήσει, και τα δεδομένα του «ερωτήματος». Αν έχει λάβει κάποια δεδομένα τότε τα αποκωδικοποιεί σε “UTF-8”.
	
	Μέθοδοι: αυτά που κληρονόμησε από τον πατέρα.
	GetObject(self): επιστρέφει το «ερώτημα» σε δομή Json.

class NLPSpacyResp(NLPSpacyCommoneAbs):
def __init__(self, idName="Resp", Resp=None, Error=False, Message=None):
self.Error = Error
self.Message = Message
self.Resp = Resp
	NLPSpacyResp(NLPSpacyCommoneAbs):
	Η δομή έχει κληρονομηθεί με την σειρά της σε κάποια μοντέλα που έχουν άμεση σχέση με την «απάντηση» που πρέπει να δώσει το back-end.

NLPSpacyCommoneAbs.__init__(self, {'idName': idName, 'Resp': self.Resp, 'Error': self.Error, 'Message': self.Message})	Χαρακτηριστικά: αυτά που έχει κληρονομήσει από την δομή «NLPSpacyCommoneAbs», μια μεταβλητή που δηλώνει αν υπάρχει σφάλμα, μια μεταβλητή για το μήνυμα του σφάλματος και την απάντηση που θα στείλουμε. Κατασκευαστής: λαμβάνει στην είσοδο του ένα κωδικό ονόματος (idName), συνήθως το όνομα της δομής που το έχει κληρονομήσει, τα δεδομένα της «απάντησης», το σφάλμα, και το μήνυμα του σφάλματος. Τα δεδομένα αυτά τοποθετούνται σε μια δομή, όπου δίνετε ως είσοδο στο μοντέλο πατέρα. Μέθοδοι: αυτά που κληρονόμησε από τον πατέρα.

Το επόμενο αρχείο διαθέτει τις βασικές δομές που χρησιμοποιούνται σε κάθε ερώτημα που δέχεται το back-end. Διαθέτη με λίγα λόγια για κάθε services ένα μοντέλο «ερωτήματος» όπου περιμένει και τη αντίστοιχη δομή της «απάντησης» που πρέπει να στείλει ο μηχανισμός.

NLPSpacyReqResp.py

Για κάθε services έχουν δημιουργηθεί αντίστοιχα μοντέλα για την διαχείριση του «ερώτημας» και της «απάντησης». Στο αρχείο αυτό θα δούμε και όλες τις δομές που αναλύσαμε στην αρχή, και που χρησιμοποιούνται. Πρώτα, παραθέτουμε όλες τις βιβλιοθήκες που πρέπει να εισάγουμε.

```
import urllib  #ανέβασμα αρχείων
from configparser import ConfigParser
from NLPSpacyCommone.NLPSpacyCommoneAbsReqResp import NLPSpacyReq, NLPSpacyResp
from NLPSpacyModel.NERModel import NerSaveDataModel, NerModel
from NLPSpacyModel.EmotionModel import ReqTextDataModel, EmotionAnalysis, TempSaveEmotionsAndCalculate
from NLPSpacyModel.NLPSpacyHelper import JSONSerialize
from NLPSpacyModel.NLPSpacyGeneralModel import NLPSpacyAllDoc, NLPSpacyAllDep, NerTraining, \
```

```
EmotionAnalysisTraining, GetAllFilesSentiment, GetAllSpacyPackage
import zipfile  #βιβλιοθήκη διαχείρησης συμπιεσμένων αρχείων
from NLPSpacyModel import TrainingNerField as tnf
```

NLPSpacyReqResp.NLPNerTrainingReq και NLPSpacyReqResp.NLPNerTrainingResp

Οι δομές αφορούν το api service "/nertraining", και έχει σχέση με την διαδικασία της εκπαίδευσης του NER πεδίου, όπου επιστρέφει τα NER που έχουν εντοπιστεί σε ένα κείμενο.

class NLPNerTrainingReq(NLPSpacyReq):
def __init__(self, datareq=None, package=None, default_example=False, package_default_example='1'):
self.package = package
self.default_example = default_example
self.package_default_examples = package_default_example
NLPSpacyReq.__init__(self, idName="NLPNerTrainingReq", dataReq=datareq)
 	
def Get(self):
resp = NLPNerTrainingResp(count=self.dataReq, package=self.package, default_example=self.default_example,
 	
package_default_example=self.package_default_examples)
return resp.Get()
	NLPNerTrainingReq(NLPSpacyReq):
	Η δομή διαχειρίζεται το «ερώτημα» που έλαβε από το api «/nertraining».
	
	Κατασκευαστής: δέχεται τα δεδομένα που έστειλε το front-end.
	
	Μέθοδοι:
	Get(self): επιστρέφει την απάντηση της δομής «NLPNerTrainingResp(NLPSpacyResp)»

class NLPNerTrainingResp(NLPSpacyResp):
 	
def __init__(self, count=None, package=None, default_example=False, package_default_example='1'):
self.count = count
self.package = package
self.default_example = default_example
self.package_default_example = package_default_example
# NLPSpacyResp.__init__(self, idName="NLPNerTrainingResp") 	
# respdata = NTTD.NerTrainingTest() 	
respdata = NerTraining(count=int(self.count), package=self.package, default_examples=self.default_example,
	NLPNerTrainingResp(NLPSpacyResp):
	Η δομή διαχειρίζεται την «απάντηση» που πρέπει να στείλει μέσου του api «/nertraining».
	
	Κατασκευαστής: δέχεται τα δεδομένα από το ερώτημα, πραγματοποιεί τους υπολογισμούς.
	
	Μέθοδοι:
	Get(self): επιστρέφει την απάντηση της.

package_default_example=self.package_default_example).Get() NLPSpacyResp.__init__(self, idName="NLPNerTrainingResp", Resp=respdata) def Get(self): return self.response	

NLPSpacyReqResp.NLPNerTrainingSaveDataReq και NLPSpacyReqResp.NLPNerTrainingSaveResp

Οι δομές γίνονται χρήση για το api " /nertraining/savenerdata", όταν θέλουμε να διορθώσουμε την εικόνα του παραδείγματος είτε γιατί βρέθηκε λάθος NER είτε γιατί θέλουμε να το εμπλουτίσουμε με νέο πεδίο.

class NLPNerTrainingSaveDataReq(NLPSpacyReq):
def __init__(self, datareq=None):
NLPSpacyReq.__init__(self, idName="NLPNerTrainingSaveDataReq", dataReq=datareq)
obj = NerSaveDataModel(data=datareq).Get()
reqdata = NerModel().update(dict=obj['data'])
reqdata.WriteInFile()
 	
def Get(self):
resp = NLPNerTrainingSaveResp()
return resp.Get()
	NLPNerTrainingSaveDataReq(NLPSpacyReq):
	Η δομή διαχειρίζεται το «ερώτημα» που έλαβε από το api «/nertraining/savenerdata».
	
	Κατασκευαστής: δέχεται τα δεδομένα που έστειλε το front-end.
	
	Μέθοδοι:
	Get(self): επιστρέφει την απάντηση της δομής «NLPNerTrainingSaveResp(NLPSpacyResp)»

class NLPNerTrainingSaveResp(NLPSpacyResp):
 	
def __init__(self):
NLPSpacyResp.__init__(self, idName="NLPNerTrainingResp")
 	
def Get(self):
return self.response
	NLPNerTrainingSaveResp(NLPSpacyResp):
	Η δομή διαχειρίζεται την «απάντηση» που πρέπει να στείλει μέσου του api «/nertraining/savenerdata».
	
	Κατασκευαστής: δέχεται τα δεδομένα από το ερώτημα, πραγματοποιεί τους υπολογισμούς.
	
	Μέθοδοι:
	Get(self): επιστρέφει την απάντηση της.

NLPSpacyReqResp.NLPUploadFileReq και NLPSpacyReqResp.NLPUploadFileResp

Οι δομές γίνονται χρήση από το api "/exespacy/uploadfile", και πρόκειται για το service, όπου «ανεβάζει» ο χρήστης το αρχείο με τα παραδείγματα του και αποθηκεύεται προσωρινά στο server.

```
class NLPUploadFileReq(NLPSpacyReq):
    resp = None

    def __init__(self, dataReq=None):
        NLPSpacyReq.__init__(self,
idName="NLPUploadFileReq")
        response = urllib.request.urlopen(dataReq)
        try:
            with open('runuploadfile.txt', 'wb') as f:
                f.write(response.file.read())
                f.close()
            resp = "OK"
        except:
            resp = "Error"

    def Get(self):
        return self.resp
```

```
class NLPUploadFileResp(NLPSpacyResp):
    def __init__(self):
        NLPSpacyResp.__init__(self,
idName="NLPUploadFileResp", Resp="OK")

    def Get(self):
        return "OK"
```

NLPSpacyReqResp.NLPGetTokensReq και NLPSpacyReqResp.NLPGetTokensResp

Οι δομές χρησιμοποιούνται για το api "/exespacy/gettokens ", ένα services υπεύθυνο για την γενική εικόνα της ανάλυσης ενός κειμένου.

```
class NLPGetTokensReq(NLPSpacyReq):
    def __init__(self, package=None):
        self.package = package
        NLPSpacyReq.__init__(self,
idName="NLPGetTokensReq")

    def Get(self):
        resp =
NLPGetTokensResp(package=self.package).Get()
```

```
class NLPGetTokensResp(NLPSpacyResp):
    def __init__(self, texts=None, package=None):
        nlptoken = NLPSpacyAllDoc(texts=texts,
package=package)
        resp = nlptoken.Get()
        NLPSpacyResp.__init__(self,
idName="NLPGetTokensReq", Resp=resp)

    def Get(self):
```

return resp	return self.response

NLPSpacyReqResp.NLPSpacyGetDepReq και NLPSpacyReqResp.NLPSpacyGetDepResp

Τα μοντέλα αυτά χρησιμοποιούνται από το api " /exespacy/getdep", όπου δίνει στο front-end την εικόνα των συντακτικών εξαρτήσεων ενός κειμένου.

```
class NLPSpacyGetDepReq(NLPSpacyReq):
    def __init__(self, package=None):
        self.package = package
        NLPSpacyReq.__init__(self, idName="NLPSpacyGetDepReq")

    def Get(self):
        resp = NLPSpacyGetDepResp(package=self.package).Get()
        return resp
```

```
class NLPSpacyGetDepResp(NLPSpacyResp):
    def __init__(self, texts=None, package=None):
        nlpdep = NLPSpacyAllDep(texts=texts, package=package)
        resp = nlpdep.Get()
        NLPSpacyResp.__init__(self, idName="NLPGetTokensReq", Resp=resp)

    def Get(self):
        return self.response
```

NLPSpacyReqResp.NLPSpacyGetEmotionAnalysisReq και NLPSpacyReqResp. NLPSpacyGetEmotionAnalysisResp

Χρήση τον ακόλουθων μοντέλων γίνονται από το api " /exespacy/emotionalysis", όπου μπορεί και πραγματοποιεί την συναισθηματική ανάλυση. Το συγκεκριμένο api γίνεται χρήση μόνο από την επιλογή που έχει να κάνει την εκτέλεση/γενική εικόνα στην ανάλυση των κειμένων. Ενώ παρακάτω θα δούμε τις δομές που έχουν να κάνουμε με την λειτουργία της εκπαίδευσης του συναισθήματος (ο εμπλουτισμός του λεξικού συναισθήματος).

```
class NLPSpacyGetEmotionAnalysisReq(NLPSpacyReq):
    def __init__(self, datareq=None, sentiment_file=None):
        self.sentiment_file = sentiment_file
        NLPSpacyReq.__init__(self, idName="NLPSpacyGetEmotionAnalysisReq", dataReq=datareq)
        self.text = ReqTextDataModel(text=datareq).Get()
        print(self.text)

    def Get(self):
```

```
class NLPSpacyGetEmotionAnalysisResp(NLPSpacyResp):
    def __init__(self, text=None, sentiment_file=None):
        emotioanalysis = EmotionAnalysis(text=text, file_name=sentiment_file)
        emotioanalysis.Analysis()
        resp = emotioanalysis.Get()
        NLPSpacyResp.__init__(self, idName="NLPSpacyGetEmotionAnalysisResp", Resp=resp)

    def Get(self):
```

```
        resp = NLPSpacyGetEmotionAnalysisResp(text=self.text, sentiment_file=self.sentiment_file).Get()
        return resp
```

```
        return self.GetSimpleObj()
```

NLPSpacyReqResp.NLPSpacyGetEmotionAnalysisTrainingReq και NLPSpacyReqResp. NLPSpacyGetEmotionAnalysisTrainingResp

Οι δομές χρησιμοποιούνται για το api "/nertraining/emotionalysistraining", όπου αφορά την συναισθηματική ανάλυση του κειμένου στην λειτουργία για τον εμπλουτισμό του λεξικού.

```
class NLPSpacyGetEmotionAnalysisTrainingReq(NLPSpacyReq):
    def __init__(self, datareq=None, file_name=None, default_example=False, package_default_example='1'):
        NLPSpacyReq.__init__(self, idName="NLPSpacyGetEmotionAnalysisTrainingReq", dataReq=datareq)
        self.count = ReqTextDataModel(text=datareq).Get()
        self.file_name = file_name
        self.default_example = default_example
        self.package_default_example = package_default_example

    def Get(self):
        resp = NLPSpacyGetEmotionAnalysisTrainingResp(count=self.count, file_name=self.file_name,
                                                      default_example=self.default_example,
                                                      package_default_example=self.package_default_example).Get()
        return resp
```

```
class NLPSpacyGetEmotionAnalysisTrainingResp(NLPSpacyResp):
    def __init__(self, count=1, file_name=None, default_example=False, package_default_example='1'):
        text = EmotionAnalysisTraining(count=count, default_examples=default_example,
                                       package_default_example=package_default_example).GetText()
        emotioanalysis = EmotionAnalysis(text=text, file_name=file_name)
        emotioanalysis.Analysis()
        resp = emotioanalysis.Get()
        NLPSpacyResp.__init__(self, idName="NLPSpacyGetEmotionAnalysisTrainingResp", Resp=resp)

    def Get(self):
        return self.GetSimpleObj()
```

NLPSpacyReqResp.NLPSpacyGetEmotionAnalysisTrainingUploadFileReq και NLPSpacyReqResp.NLPSpacyGetEmotionAnalysisTrainingUploadFileResp

Στη συνέχεια έχουμε τις δομές, όπου γίνονται χρήση στο api "/nertraining/emotionalysistrainingsavedata", για την αποθήκευση ενός νέου token που συμμετέχει στην συναισθηματική ανάλυση.

```
class NLPSpacyGetEmotionAnalysisTrainingUploadFileReq(NLPSpacyReq):
    resp = None

    def __init__(self, dataReq=None):
        NLPSpacyReq.__init__(self, idName="NLPSpacyGetEmotionAnalysisTrainingUploadFileReq")
        ob = JSONSerialize().GetObject(strjson=dataReq)
        existedemontiondata = []
        try:
            conf = ConfigParser()
            with open(str(conf.get("emotion_data", 'path')), 'r', encoding='UTF-8') as f:
                existedemontiondata = f.readlines()[1:]
                f.close()
        except:
            resp = "Error"
        try:
            with open(str(ob['filename']) + '.tsv', 'w+', encoding='UTF-8') as f:
                f.write('"Term\tPOS1\tPOS2\tPOS3\tPOS4
\tSubjectivity1\tSubjectivity2\tSubjectivity3\tSubjectivity4
\tPolarity1\tPolarity2\tPolarity3\tPolarity4
\tAnger1\tAnger2\tAnger3\tAnger4
\tDisgust1\tDisgust2\tDisgust3\tDisgust4
\tFear1\tFear2\tFear3\tFear4
\tHappiness1\tHappiness2\tHappiness3\tHappiness4
\tSadness1\tSadness2\tSadness3\tSadness4
\tSurprise1\tSurprise2\tSurprise3\tSurprise4
\tAditional1\tAditional2\tAditional3\tAditional4
\tComments1\tComments2\tComments3\tComments4\t"')
                f.write('\n')
                for lines in ob['data']:
                    f.write(str(lines) + '\t\n')
                for lines in existedemontiondata:
```

```
                f.write(str(lines) + '\t\n')
            f.close()
        resp = "OK"
    except:
        resp = "Error"

def Get(self):
    return self.resp
```

```
class NLPSpacyGetEmotionAnalysisTrainingUploadFileResp(NLPSpacyResp):
    def __init__(self):
        NLPSpacyResp.__init__(self, idName="NLPSpacyGetEmotionAnalysisTrainingploadFileResp", Resp="OK")

    def Get(self):
        return "OK"
```

NLPSpacyReqResp.NLPGetAllSentimentLexiconReq/Resp και NLPSpacyReqResp.NLPGetAllSpacyPackageReq/Resp

Οι ακόλουθες δομές αφορούν λειτουργίες της εφαρμογής που συμμετέχουν σε παραμετροποιήσεις. Πιο συγκεκριμένα, τα μοντέλα έχουν σκοπό να επιστρέψουν στο front-end τα συναισθηματικά λεξικά που είναι διαθέσιμα και τις διάφορες εκδόσεις πακέτων spacy που διαθέτει η εφαρμογή. Τα αντίστοιχα api services είναι «/nertraining/getsentimentlexicon» και «/nertraining/getspacypackage». Παραθέτουμε παρακάτω τον κώδικα.

```
class NLPGetAllSentimentLexiconReq(NLPSpacyReq):
    def __init__(self):
        NLPSpacyReq.__init__(self, idName="NLPGetAllSentimentLexiconReq")

    def Get(self):
        resp = NLPGetAllSentimentLexiconResp().Get()
        return resp
```

```
class NLPGetAllSpacyPackageReq(NLPSpacyReq):
    def __init__(self):
        NLPSpacyReq.__init__(self, idName="NLPGetAllSpacyPackageReq")

    def Get(self):
        resp = NLPGetAllSpacyPackageResp().GetSimpleObj()
        return resp
```

```
class NLPGetAllSentimentLexiconResp(NLPSpacyResp):
    def __init__(self):
        file_list = GetAllFilesSentiment().GetList()
```

```
class NLPGetAllSpacyPackageResp(NLPSpacyResp):
    def __init__(self):
        dir_list = GetAllSpacyPackage().GetList()
```

```
        NLPSpacyResp.__init__(self,
idName="NLPGetAllSentimentLexiconResp",
Resp=file_list)

    def Get(self):
        return self.response
```

```
        NLPSpacyResp.__init__(self,
idName="NLPGetAllSpacyPackageResp", Resp=dir_list)

    def Get(self):
        return self.response
```

NLPSpacyReqResp.NLPSpacyTrainingReq και NLPSpacyReqResp.NLPSpacyTrainingResp

Οι δομές γίνονται χρήση από το api «/nertraining/trainingPackage», όπου εκτελείται η εκπαίδευση του NER. Σαν είσοδο δέχεται το όνομα του αρχείου που θα έχει το πακέτο. Παραθέτουμε τον αντίστοιχο κώδικα.

```
class NLPSpacyTrainingReq(NLPSpacyReq):
    def __init__(self, package_name=None):
        self.package_name = package_name
        NLPSpacyReq.__init__(self,
idName="NLPSpacyTrainingReq")

    def Get(self):
        resp =
NLPSpacyTrainingResp(package_name=self.package_na
me).GetSimpleObj()
        return resp
```

```
class NLPSpacyTrainingResp(NLPSpacyResp):
    def __init__(self, package_name=None):
        self.resp =
tnf.TrainingNer(package_name=package_name).Get()
        if self.resp != 'OK':
            NLPSpacyResp.__init__(self,
idName="NLPSpacyTrainingResp",
Error=self.resp['Error'], Resp=self.resp,
                                Message=self.resp['Message'])
        else:
            NLPSpacyResp.__init__(self,
idName="NLPSpacyTrainingResp", Resp=self.resp)

    def Get(self):
        return self.response
```

NLPSpacyReqResp.NLPCalculateEmotionReq και NLPSpacyReqResp.NLPCalculateEmotionReq

Οι δομές γίνονται χρήση από το api «/nertraining/emotionalysiscalculated», όπου μπορεί να γίνει εκ νέου υπολογισμός του συναισθήματος σε ένα κείμενο.

```
class NLPCalculateEmotionReq(NLPSpacyReq):
    def __init__(self, dataReq=None):
        ob = JSONSerialize().GetObject(strjson=dataReq)
```

```
        self.package_name = ob['package']
        self.text = ob['text']
        self.matched_token = ob['matched_token']
        NLPSpacyReq.__init__(self, idName="NLPCalculateEmotionReq")

    def Get(self):
        resp = NLPCalculateEmotionResp(package_name=self.package_name, text=self.text,
                                       matched_token=self.matched_token).GetSimpleObj()
        return resp
```

```
class NLPCalculateEmotionResp(NLPSpacyResp):
    def __init__(self, package_name=None, text=None, matched_token=None):
        emotioncalculate = TempSaveEmotionsAndCalculate(matched_tokens=matched_token, text=text,
package=package_name)
        resp = emotioncalculate.Get()
        NLPSpacyResp.__init__(self, idName="NLPCalculateEmotionResp", Resp=resp)

    def Get(self):
        return self.response
```

Ολοκληρώνοντας με όλες τις δομές και τους μηχανισμούς που δημιουργήθηκαν στο back-end για τις ανάγκες της εφαρμογής, ερχόμαστε να στήσουμε τα api services, όπου θα δέχεται τις κλήσεις από το front-end. Όλα τα api routes βρίσκονται στο αρχείο που δημιουργήθηκε μαζί με το project και αυτό είναι το «app.py».

App.py

Το αρχείο διαθέτει όλα τα api services, όπου είναι διαθέσιμα από το front-end. Η βιβλιοθήκες που χρησιμοποιήθηκαν είναι οι εξής:

```
from flask import Flask, jsonify, request
from NLPSpacyCommone import NLPSpacyReqResp
from flask_cors import CORS, cross_origin
import os
```

Για να μπορέσει να δεχτεί κλήσεις το back-end από το front-end χρειάστηκε να προσθέσουμε το ακόλουθο κώδικα.

```
app = Flask(__name__)
app.config['SECRET_KEY'] = 'the quick brown fox jumps over the lazy   dog'
app.config['CORS_HEADERS'] = 'Content-Type'
# CORS(app)
# cors = CORS(app, resources={r"/*": {"origins": "*"}})
cors = CORS(app, resources={r"/*": {"origins": "http://localhost:port"}})
```

Ενώ παρακάτω παραθέτουμε όλα τα apis, με σχόλια για το τι είναι το καθένα.

```
@app.route('/', methods=['GET'])
def hello_world():
    response = jsonify(message="Simple server is running")
    response.data = ["error"]
    response.headers.add("Access-Control-Allow-Origin", "*")
    return response


@app.route('/nertraining', methods=['Get']) #επιστρέφει μια εικόνα με τα ner που έχει εντοπίσει στο κείμενο
def Ner_Training():
    req = NLPSpacyReqResp.NLPNerTrainingReq(datareq=request.values.get('count_text'),
                                            package=request.values.get('package'),
                                            default_example=request.values.get('default_examples') == 'true',
                                            package_default_example=request.values.get('package_example'))
    resp = req.Get()
    return resp


@app.route('/exespacy/uploadfile', methods=['POST']) #ο χρήστη μπορεί να ανεβάσει τα δικά του test δεδομένων
# @cross_origin()
@cross_origin(origin='localhost', headers=['Content- Type', 'Authorization'])
def Exe_Spacy():
    req = NLPSpacyReqResp.NLPUploadFileReq(dataReq=request.form['files'])
    return "OK"


@app.route('/exespacy/gettokens', methods=['Get']) #επιστρέφει μια γενική εικόνα του κειμένου που έχει αναλυθεί
def Get_Tokens():
    req = NLPSpacyReqResp.NLPGetTokensReq(package=request.values.get('package'))
    resp = req.Get()
    return resp


@app.route('/exespacy/getdep', methods=['Get'])  #επιστρέφει τις συντακτικές εξαρτήσεις του κειμένου
def Get_Dep():
    req = NLPSpacyReqResp.NLPSpacyGetDepReq(package=request.values.get('package'))
    resp = req.Get()
    return resp
```

```python
@app.route('/exespacy/emotionalysis', methods=['Get']) #επιστρέφει την εικόνα της συναισθηματικής ανάλυσης του κειμένου
# @cross_origin()
@cross_origin(origin='localhost', headers=['Content- Type', 'Authorization'])
def Get_Emotion_Analysis():
    req = NLPSpacyReqResp.NLPSpacyGetEmotionAnalysisReq(datareq=request.values.get('text'),
                                                        sentiment_file=request.values.get('sentiment_file'))
    resp = req.Get()
    print(resp)
    return resp


@app.route('/nertraining/savenerdata', methods=['POST']) #δυνατότητα να αποθηκεύσει την διορθωμένη εικόνα
                                                         #των εντοπισμένων ner
# @cross_origin()
@cross_origin(origin='localhost', headers=['Content- Type', 'Authorization'])
def Ner_Training_Save_Data():
    req = NLPSpacyReqResp.NLPNerTrainingSaveDataReq(datareq=request.data)
    # req = NLPSpacyReqResp.NLPUploadFileReq(dataReq=request.form['files'])
    return "OK"


@app.route('/nertraining/emotionalysistraining', methods=['Get']) #επιστρέφει την εικόνα της συναισθηματικής
                                                                  # ανάλυσης του κειμένου, αφορά την ενέργεια που εχεί να
                                                                  # κάνει με τον εμπλουτισμό του συναισθηματικού λεξικού
# @cross_origin()
@cross_origin(origin='localhost', headers=['Content- Type', 'Authorization'])
def Get_Emotion_Analysis_Training():
    req = NLPSpacyReqResp.NLPSpacyGetEmotionAnalysisTrainingReq(datareq=request.values.get('count_text'),
                                                                file_name=request.values.get('file_name'),
                                                                default_example=request.values.get('default_example') == 'true'
                                                                package_default_example=request.values.get('package_example'))
    resp = req.Get()
    return resp
```

```python
@app.route('/nertraining/emotionalysistrainingsavedata', methods=['POST']) #αποθήκευση ενός νέου token που συμμετέχει
                                                                           # στην συναισθηματική ανάλυση
# @cross_origin()
@cross_origin(origin='localhost', headers=['Content- Type', 'Authorization'])
def Emotion_Analysis_Training_Save_Data():
    req = NLPSpacyReqResp.NLPSpacyGetEmotionAnalysisTrainingUploadFileReq(dataReq=request.data)
    return "OK"


@app.route('/nertraining/getsentimentlexicon', methods=['Get']) #επιστρέφει λίστα με τα συναισθηματικά
                                                                # λεξικά που εχει η εφαρμογή
def Get_All_File_Sentiment_Lexicon():
    req = NLPSpacyReqResp.NLPGetAllSentimentLexiconReq()
    resp = req.Get()
    return resp


@app.route('/nertraining/getspacypackage', methods=['GET'])
# @cross_origin()
@cross_origin(origin='localhost', headers=['Content- Type', 'Authorization']) #επιστρέφει μια λίστα με τα πακέτα spacy
def Get_ALL_Packagies_Spacy():
    req = NLPSpacyReqResp.NLPGetAllSpacyPackageReq()
    resp = req.Get()
    # req = NLPSpacyReqResp.NLPUploadFileReq(dataReq=request.form['files'])
    return resp
```

```python
@app.route('/nertraining/trainingPackage', methods=['GET']) #η ενέργεια όπου ξεκινάει η διαδικασία
                                                            # την εκπαίδευσης του ner
# @cross_origin()
@cross_origin(origin='localhost', headers=['Content- Type', 'Authorization'])
def Training_Package_Spacy():
    req = NLPSpacyReqResp.NLPSpacyTrainingReq(package_name=str(request.values.get('package_name')))
    resp = req.Get()
    return "OK"


@app.route('/nertraining/emotionalysiscalculated', methods=['POST']) #δυνατότητα προσωρινής αποθήκευσης ενός νέου
                                                                      # token που συμμετέχει
                                                                      # στην συναισθηματική ανάλυση του κειμένου
                                                                      # και εκ νέου υπολογισμός του συναισθήματος.
# @cross_origin()
@cross_origin(origin='localhost', headers=['Content- Type', 'Authorization'])
def Get_Emotion_Analysis_Calculated():
    req = NLPSpacyReqResp.NLPCalculateEmotionReq(dataReq=request.data)
    resp = req.Get()
    return resp
```

Τέλος, στο αρχείο κλείνει με τον ακόλουθο κώδικα, όπου ορίζουμε την χρήση secret_key, την ip και την port στην οποία θα τρέχει.

```python
if __name__ == '__main__':
    # app.run()
    app.secret_key = os.urandom(64)
    app.run(host="0.0.0.0", port="5000", debug=True)
```

Τέλος, το τελευταίο αρχείο που έμεινε είναι αυτό του «settings.ini», το οποίο πρόκειται για ένα αρχείο διαμορφωτή.

Settings.ini

Το αρχείο περιέχει στατικά δεδομένα, όπως διαδρομές αρχείων, φακέλων, ονόματα πακέτων κτλ. Παρακάτω, δείχνουμε την εικόνα με τα περιεχόμενα του αρχείου αυτού.

```
[nlp_pgk]
path = TrainingPackage/my_el_continue_model_pgk2
default = "el_core_news_lg"

[emotion_data]
path = TrainingPackage/greek_sentiment_lexicon.tsv

[test_data]
path = "NLPSpacyTestData/"
test_data_dev = "el_gdt-ud-dev.spacy"
test_data_train = "el_gdt-ud-train.spacy"

[training_ner_data]
path_train_spacy = ./TrainingDataSet/output-train.spacy/el_gdt-ud-train.spacy
path_dev_spacy = ./TrainingDataSet/output-train.spacy/el_gdt-ud-dev.spacy
path_ner1_parts = ./TrainingDataSet/ner_parts/ner1.jsonl
path_ner2_parts = ./TrainingDataSet/ner_parts/ner2.jsonl
path_ner3_parts = ./TrainingDataSet/ner_parts/ner3.jsonl
```

Παράρτημα Η

Το παράρτημα Γ, αναφέρουμε την τεχνολογίες, τα εργαλεία και τις τεχνικές που συνέβαλαν στην δημιουργία του front-end web application.

Front-End

Σχετικά με το front-end, όπου θα μιλήσουμε για την γραφική διεπαφή του χρήστη, πρόκειται για μια υλοποίηση που πραγματοποιήθηκε με την χρήση της React. Για διευκόλυνση μας, προτιμήσαμε να χρησιμοποιήσουμε κάποιο υπάρχων project που βρίσκεται διαθέσιμο στο διαδίκτυο, και να το προσαρμόσουμε στα δικά μας δεδομένα και για το σκοπό μας. Το project που χρησιμοποιήσαμε το βρήκαμε στην διεύθυνση https://www.creative-tim.com/product/black-dashboard-react. Επιπλέον, το εργαλείο που χρησιμοποιήσαμε για τον εμπλουτισμό του project, με νέα components και ότι άλλο απαιτείται, είναι το Visual Studio Code. Όσο αφορά την επικοινωνία με το back-end, έγινε χρήση της βιβλιοθήκης “axios”, καθώς την βρήκαμε ποιο ευέλικτή από κάποια άλλη.

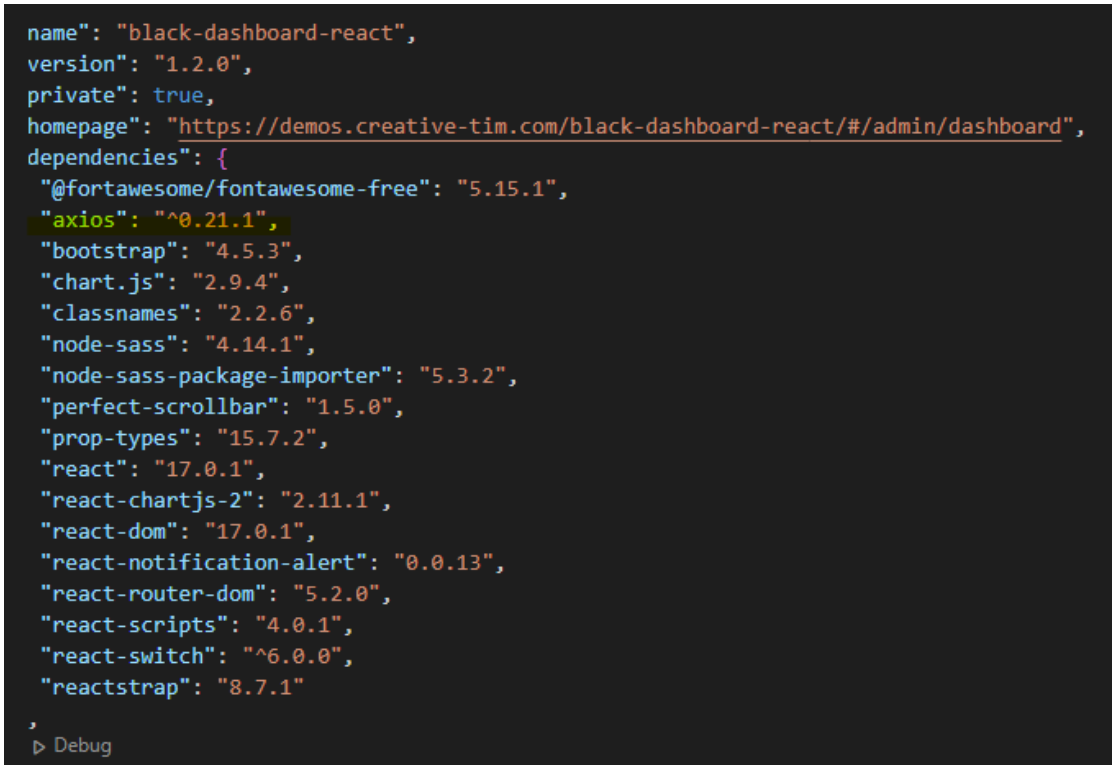
Μέσω της εφαρμογής δώσαμε τρεις βασικές λειτουργίες, όπου μπορεί ο χρήστης να εργαστεί. Μια από τις λειτουργίες και πιο βασική είναι αυτή της δυνατότης εκπαίδευσης του NER πεδίου. Η επόμενη λειτουργία δίνει στον χρήστη δυνατότητα να εμπλουτίσει το συναισθηματικό λεξικό. Τέλος, δίνουμε μια ποιο γενική λειτουργία, όπου μπορεί να μελετήσει ο χρήστη τα πόσο λειτουργικά είναι τα πακέτα που δημιουργήθηκαν κατά την εκπαίδευση τους, δηλαδή δίνεται στον χρήστη να φορτώσει παραδείγματα, και να δει βασικές πληροφορίες της ανάλυσης συντακτικής, γραμματικής, συναισθηματικής των κειμένων αυτών.

Παρακάτω θα δούμε σταδιακά τα components και τα αρχεία που διηγήθηκα ή μεταβλήθηκαν ώστε να φτάσουμε στην τελική εικόνα της εφαρμογής. Η διαδικασία που θα ακολουθήσουμε είναι από έξω προς τα μέσα. Δηλαδή, θα ξεκινήσουμε με τα αρχεία από την αρχή του project και θα προχωράμε σταδιακά στους υπό-φακέλους του.

Για αρχή θα πρέπει αν εγκαταστήσουμε το axios, αυτό μπορεί να γίνει τρέχοντας την εντολή:

```
npm install axios
```

Αμέσως μετά την ολοκλήρωση της εγκατάστασης, θα πρέπει να προσθέσουμε αν δεν υπάρχει στο αρχείο package.json, όπως φαίνεται στην παρακάτω εικόνα:



Το axios θα μας βοηθήσει να πραγματοποιήσουμε κλήσεις προς το back-end. Αφού ολοκληρώσαμε την παραπάνω διαδικασία, στην συνέχεια θα ασχοληθούμε με ότι βρίσκεται στο φάκελο "src". Στο φάκελο αυτό συνήθως τοποθετούνται κυρίως μέρη του πηγαίου κώδικα. Στο συγκεκριμένο φάκελο θα διακρίνει κανείς τους υπό-φακέλους που διαθέτει, όπως για παράδειγμα το «assets», «components», «views», «layouts» κτλ. Εμείς, όμως θα ασχοληθούμε με αυτά.

Επίσης, στο φάκελο «src», θα διακρίνουμε και άλλα αρχεία όπως το «index.js», «routes» κτλ. Εμείς θα ξεκινήσουμε με το index.js, αφού είναι το πρώτο αρχείο που τρέχει και μπορούμε αν θέλουμε θα δηλώσουμε το επίπεδο του route που θα γίνεται από εκεί και μετά render. Με λίγα λόγια, μπορούμε να ορίσουμε άμα θέλουμε κάποιες εφαρμογές να βρίσκονται κάτω από ένα κύριο path και να διαφέρουν από ένα άλλο πλήθος εφαρμογών. Για παράδειγμα, αν θέλαμε να διαχωρίσουμε την εκπαίδευση του NER πεδίου, από επιλογή εμπλουτισμού συναισθηματικού λεξικού, ορίζοντας στο καθένα διαφορετικό βασικό route, έστω η πρώτη επιλογή «http://localhost:3000/nertraining/home» ενώ για την δεύτερη «http://localhost:3000/emotiontraining/home».

Στην περίπτωση μας όμως, όλα τα τοποθετήσαμε κάτω από το route «http://localhost:3000/admin». Παρακάτω παραθέτουμε την εικόνα που έχει το index.js, όπου κυρίως αφαιρέσαμε άλλα βασικά routes.

```
import React from "react";
import ReactDOM from "react-dom";
import { BrowserRouter, Route, Switch, Redirect } from "react-router-dom";

import AdminLayout from "layouts/Admin/Admin.js";

import "assets/scss/black-dashboard-react.scss";
import "assets/demo/demo.css";
import "assets/css/nucleo-icons.css";
import "@fortawesome/fontawesome-free/css/all.min.css";

import ThemeContextWrapper from "./components/ThemeWrapper/ThemeWrapper";
import BackgroundColorWrapper from "./components/BackgroundColorWrapper/BackgroundColorWrapper";

ReactDOM.render(
  <ThemeContextWrapper>
    <BackgroundColorWrapper>
      <BrowserRouter>
        <Switch>
          <Route path="/admin" render={(props) => <AdminLayout {...props} />} />
          {/* <Route path="/rtl" render={(props) => <RTLLayout {...props} />} /> */}
          <Redirect from="/" to="/admin/dashboard" />
        </Switch>
      </BrowserRouter>
    </BackgroundColorWrapper>
  </ThemeContextWrapper>,
  document.getElementById("root")
);
```

Το επόμενο βήμα είναι να αλλάξουμε το αρχείο «routes.js», όπου εκεί εμπεριέχονται όλα τα route που «κρέμονται» κάτω από κάποιο βασικό. Εμείς, κόψαμε της τρεις λειτουργίες της εφαρμογές σε δύο βασικές. Σε εκείνες που αφορούν την εκπαίδευση είτε του NER πεδίου είτε συναισθηματική και εκείνη της γενικής λειτουργίας. Το αρχείο μετά από αλλαγές πρέπει να έχει την ακόλουθη εικόνα:

```
import TrainModel from "views/TrainModel";
import SpacyProfile from "views/SpacyProfile";

var routes = [
  {
    path: "/trainingmodel",
    name: "Train Model",
    icon: "tim-icons icon-world",
    component: TrainModel,
    layout: "/admin",
  },
  {
    path: "/spacyprofile",
    name: "My Running",
    icon: "tim-icons icon-world",
    component: SpacyProfile,
    layout: "/admin",
  },
];
export default routes;
```

Επιπλέον, στο «src» δημιουργήσαμε κάποια επιπλέον αρχεία το «dataner.js», το «sentimentDataFields.js» και το «viewstrainingmode.js». Τα δύο πρώτα αρχεία, πρόκειται να περιέχουν δεδομένα τα οποία είναι στατικά και δεν μεταβάλλονται κατά την εκτέλεση της εφαρμογής. Η πληροφορία που έχουν τα δύο αυτά αρχεία είναι μια λίστα από δομές που ως

χαρακτηριστικά έχουν ένα id, ένα όνομα ως ετικέτα και το χρώμα. Ενώ το «viewstrainingmode.js» έχει παρόμοια εικόνα με εκείνη του «routes.js» περιέχει routes που χρησιμοποιούνται από το «TrainModel». Τα αρχεία εχουν την ακόλουθη εικόνα:

Εικόνα dataner.js

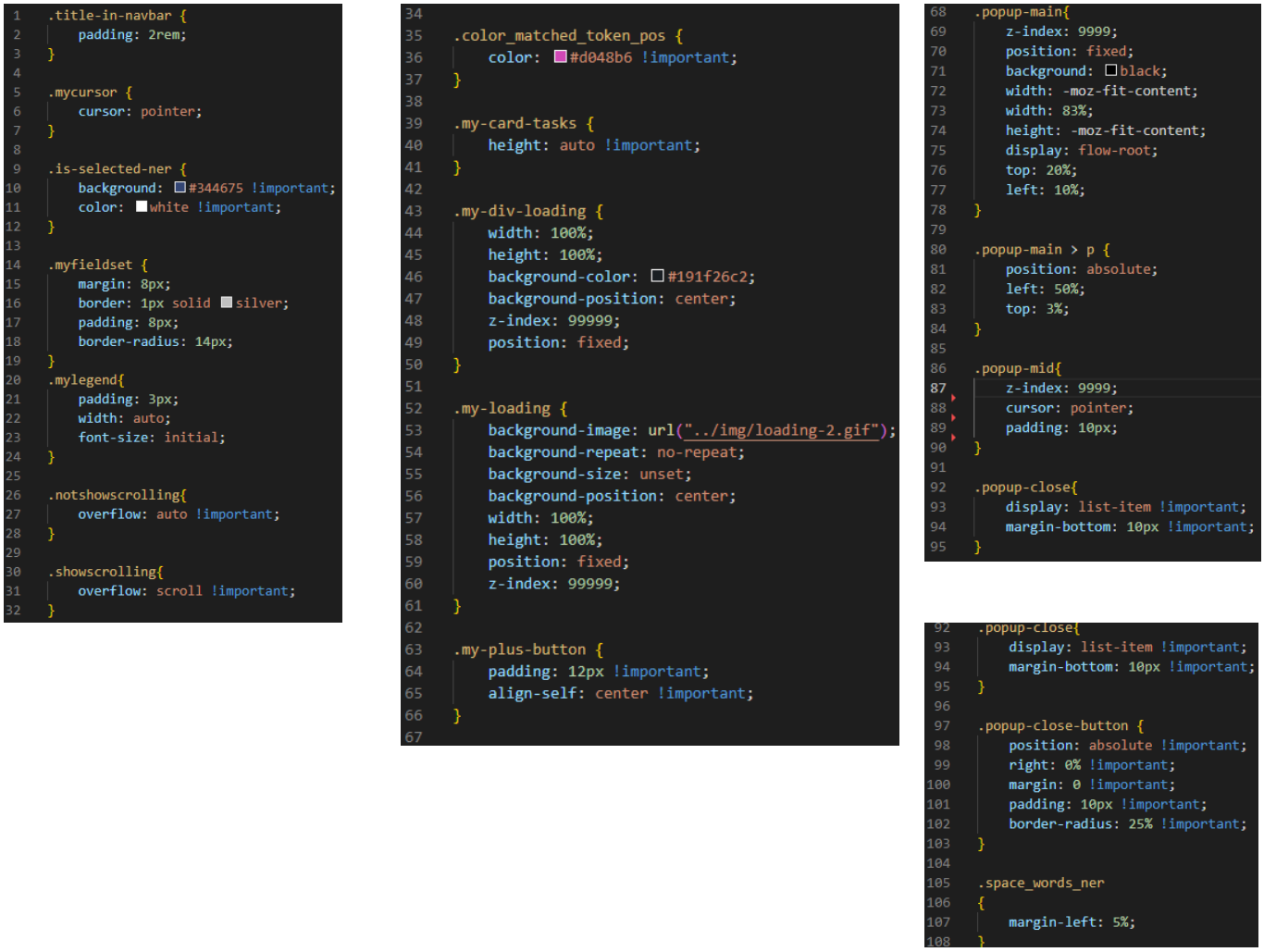
```
var SentimentDataFields = [
    {
        id: 98,
        name: "POS",
        color: "#B388FF"
    },
    {
      id: 99,
      name: "SUBJECTIVITY",
      color: "white"
    },
    {
      id:1,
      name: "POLARITY",
      color: "#D500F9",
    },
    {
      id:2,
      name: "ANGER",
      color: "#FFEA00",
    },
    {
      id:3,
      name: "DISGUST",
      color: "#FF9800",
    },
    {
      id:4,
      name: "FEAR",
      color: "#C0CA33",
    },
    {
      id:5,
      name: "HAPPINESS",
      color: "#A7FFEB",
    },
    {
      id:6,
      name: "SADNESS",
      color: "#90CAF9",
    },
    {
      id:7,
      name: "SURPRISE",
      color: "#FFD54F",
    },
    {
      id:8,
      name: "ADITIONAL",
      color: "#FFE0B2",
    },
    {
      id:9,
      name: "COMMENT",
      color: "#F4FF81",
    },
];

export default SentimentDataFields;
```

Εικόνα sentimentDataFields.js

```
import NerViewTraining from 'views/NerViewTraining';
import EmotiontrainingView from 'views/EmotionTrainingView';
import TrainingViewHome from 'views/TrainingViewHome'

var viewstrainingmode = [
    {
      path: "/home",
      name: "Training Model Home",
      icon: "tim-icons icon-chart-pie-36",
      component: TrainingViewHome,
      layout: "/admin/trainingmodel",
    },
    {
      path: "/viewemotiontraining",
      name: "Emotion Training",
      icon: "tim-icons icon-atom",
      component: EmotiontrainingView,
      layout: "/admin/trainingmodel",
    },
    {
      path: "/nerviewtraining",
      name: "Ner Training",
      icon: "tim-icons icon-components",
      component: NerViewTraining,
      layout: "/admin/trainingmodel",
    }
]

export default viewstrainingmode;
```

Εικόνα viewstrainingmode.js

Στην συνέχεια, έχουμε το φάκελο assets, όπου εκεί τοποθετούνται ότι έχει να κάνει με το στιλ, εικόνες γραμματοσειρές κτλ. Εμείς προσθέσαμε το δικό μας css αρχείο που περιέχει τα δικά μας styles για της ανάγκες της εφαρμογής. Ακολουθεί εικόνα του αρχείου:



```css
.title-in-navbar {
    padding: 2rem;
}

.mycursor {
    cursor: pointer;
}

.is-selected-ner {
    background: #344675 !important;
    color: white !important;
}

.myfieldset {
    margin: 8px;
    border: 1px solid silver;
    padding: 8px;
    border-radius: 14px;
}
.mylegend{
    padding: 3px;
    width: auto;
    font-size: initial;
}

.notshowscrolling{
    overflow: auto !important;
}

.showscrolling{
    overflow: scroll !important;
}

.color_matched_token_pos {
    color: #d048b6 !important;
}

.my-card-tasks {
    height: auto !important;
}

.my-div-loading {
    width: 100%;
    height: 100%;
    background-color: #191f26c2;
    background-position: center;
    z-index: 99999;
    position: fixed;
}

.my-loading {
    background-image: url("../img/loading-2.gif");
    background-repeat: no-repeat;
    background-size: unset;
    background-position: center;
    width: 100%;
    height: 100%;
    position: fixed;
    z-index: 99999;
}

.my-plus-button {
    padding: 12px !important;
    align-self: center !important;
}

.popup-main{
    z-index: 9999;
    position: fixed;
    background: black;
    width: -moz-fit-content;
    width: 83%;
    height: -moz-fit-content;
    display: flow-root;
    top: 20%;
    left: 10%;
}

.popup-main > p {
    position: absolute;
    left: 50%;
    top: 3%;
}

.popup-mid{
    z-index: 9999;
    cursor: pointer;
    padding: 10px;
}

.popup-close{
    display: list-item !important;
    margin-bottom: 10px !important;
}

.popup-close-button {
    position: absolute !important;
    right: 0% !important;
    margin: 0 !important;
    padding: 10px !important;
    border-radius: 25% !important;
}

.space_words_ner
{
    margin-left: 5%;
}
```

Εικόνα mystyle.css

Το επόμενο που θα δούμε είναι το components. Στα components, τοποθετούμε αρχεία που ίσως είναι επαναχρησιμοποιήσιμα και έχουν συγκεκριμένες λειτουργίες. Η ιδέα είναι ότι έχω ένα view αλλά τα ένα components μπορεί να χρησιμοποιείται από πολλά views. Ένα component μπορεί να είναι ένα γράφημα, ένα button, μια ομάδα από πίνακες κτλ. Εμείς δημιουργήσαμε νέα components ειδικά σχεδιασμένα τις ανάγκες μας και τα οποία χρησιμοποιήθηκαν στην εφαρμογή μας.

Αυτό που θα ξεκινήσουμε είναι με το «/src/components/NERField». Το component NERField αποτελείται από δύο αρχεία, το «ButtonNerField.js» και το «NERField.js». Αυτό χρειαζόμασταν είναι να έχουμε σε ένα συγκεκριμένο σημείο όλες τις ετικέτες NER που σκοπεύουμε να αναγνωρίζει η εφαρμογή μας. Το «ButtonNerField» έχει σχεδιαστή έχει ώστε να δέχεται μια δομή του τύπου, όπως φαίνεται στην εικ. “Dataner.js”. Παρακάτω παραθέτουμε τους κώδικες.

```jsx
import React, { Component } from 'react';
import { Button } from 'reactstrap';

import "assets/css/mystyle.css";

class ButtonNerField extends Component{
    constructor(props){
        super(props);
        this.state = {
            style: { background: this.props.nerobject.color, color: "Black" }  //by default style
        }
    }

    handleClick = () => { //function για την εκτέλεση του onClick του component όπου περνάει ένα index
        this.props.onClick(this.props.index)
    }

    render(){
        let ner = this.props.nerobject; //λαμβάνει ένα nerobject
        return(
            <Button className = { this.props.isActive ? 'is-selected-ner' : '' } //αν έχει επιλεχθεί τότε
                                                                                  //προσθέτουμε το ονόμα της
                                                                                  //κλάσης "is-selected-ner"
                    style={ this.state.style } //το style που έχει δοθεί στον κατασκευαστή
                    onClick={ this.handleClick } //η συνάρτηση που θα εκτελεστεί με το πάτημα
                    value = { ner.name }  //η τιμή που θα πάρει το button είναι αυτή το nerobject
                    >
                {ner.name}
            </Button>
        )
    }
}

export default ButtonNerField;
```

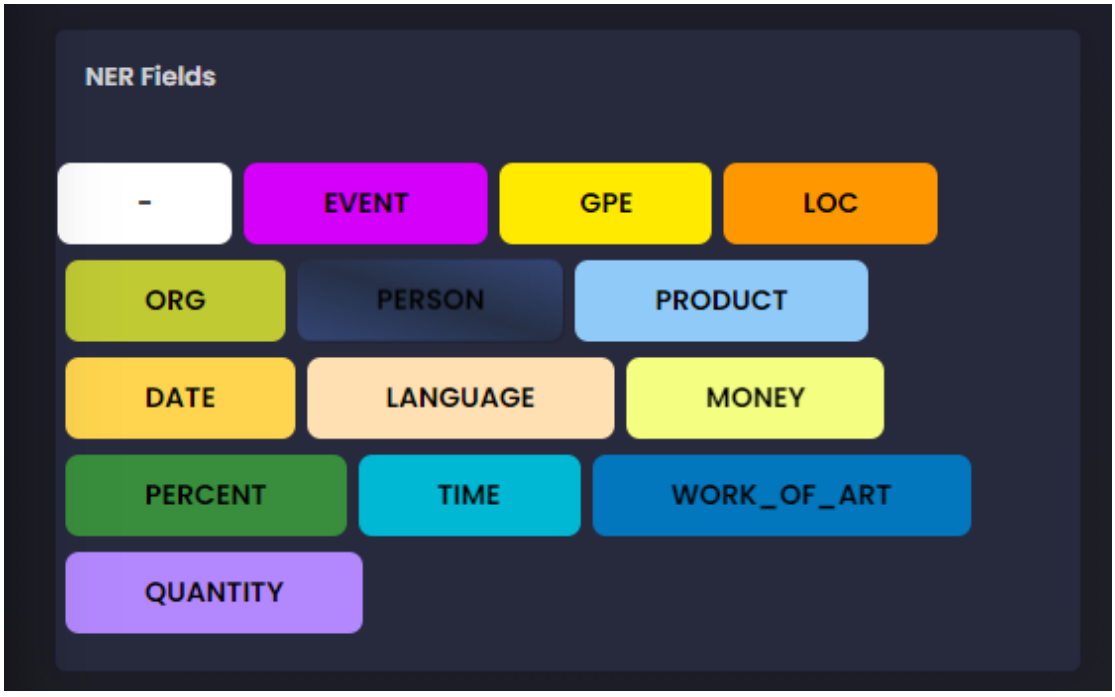
Εικόνα ButtonNerField.js

```jsx
import React, { Component } from 'react';
import { Row, Card, CardBody, CardHeader, Form } from 'reactstrap';
import ButtonNerField from './ButtonNerField';

class NERField extends Component {
    constructor(props){
        super(props);
        this.state = {
            data: props.nerfields, //ολη την λίστα με τα διαθέσημα ner.
            activeIndex: null //δήλωση του ενεργού button, στον κατασκευαστή το έχουμε ως null
        }
    }

    handleClick = (index) => { //συνάρτηση όπου αλλάζει το state δηλώνοντας το ενεργό button.
        this.setState({ activeIndex: index })
    }

    getDataNERFields = (dataner) => { //συνάρτηση όπου επιστρέφει μια λίστα από ButtonNerField όσα έχει το datane
        return dataner.map((ner, key) => {
            return(
                <>
                <ButtonNerField nerobject = { ner }
                                key={ key }
                                index={ ner.id }
                                isActive={ this.state.activeIndex === ner.id }
                                onClick={ this.handleClick } />
                </>
            )
        })
    }
```

```
31      render() {
32          return( //επιστρέφει μια καρτέλα με όλα τα button
33              <Card>
34                  <CardHeader>
35                  <h5 className="title">NER Fields</h5> {/**τίτλος */}
36                  </CardHeader>
37                  <CardBody>
38                      <Form>
39                          <Row>
40                              { this.getDataNERFields(this.state.data) } {/**χρήση της συναρτησης */
41                          </Row>
42                      </Form>
43                  </CardBody>
44              </Card>
45          );
46      }
47  }
48
49  export default NERField;
```

Εικόνα NERField.js

Το αποτέλεσμα όλου του component, είναι η παρακάτω εικόνα:



Το επόμενο components είναι ένα από τα πιο βασικά. Αυτό είναι το «/src/components/FieldToken», όπου είναι υπεύθυνο για την απεικόνιση του token με την ετικέτα NER όπως αυτή του έχει αποδοθεί. Παρακάτω, παραθέτουμε τον κώδικα:

```
import React, { Component } from 'react';
import {
    Col,
} from "reactstrap";
import "assets/css/mystyle.css";

import dataner from "dataner.js";
import NotificationAlert from "react-notification-alert";

class FieldToken extends Component {
    constructor(props){
        super(props);
        this.state = {
            text: this.props.text, //το κείμενο
            ner: this.props.token.ner.toUpperCase(), //η ετικέτα ner που ανήκει
            hasError: false, //αν υπάρχει κάποιο σφάλμα
            selectedToken: document.onmouseup = (e) => { //συνάρτηση όπου επιστρέφει τις επιλεγμένες λέξεις
                try
                {
                    let idelement = window.getSelection().focusNode.childNodes[0].id;
                    if(idelement.includes("not-selected"))
                    {
                        return;
                    }
                }
                catch(error)
                {
                }
                var selection = window.getSelection();
                selection.setBaseAndExtent(window.getSelection().anchorNode, window.getSelection().anchorOffset.value,
                window.getSelection().focusNode, window.getSelection().focusOffset.value);
                var text = selection.toString();
                if(text === null || text === "" || text === " ")
                {
                    return;
                }
```

```
                    text = text.replace(/\s+/g, " ").replace("-", " ");
                    text = this.clearText(text);
                    this.props.wordsClick(text);
                    document.getSelection().empty();
                }
            }
            this.notificationhandler = this.notificationhandler.bind(this);
        }

        static rmsg(){ //στατική συνάρτηση για την επιστροφή του μηνύματος
            return "Select first ner field!"
        }
        setStyle(nername){ //συνάρτηση για την αλλαγή του style, ανάλογα της ετικέτας ner
            let colorner = dataner.find(da => da.name.toLowerCase() === nername.toLowerCase())
            this.setState({
                stylefieldset: '1px solid ' + colorner.color,
                stylelegend: colorner.color
            })
        }

        componentDidCatch(error, info){
            console.error(error);
            console.info(info);
        }

        componentDidMount(){
            this.setStyle(this.state.ner);
        }
```

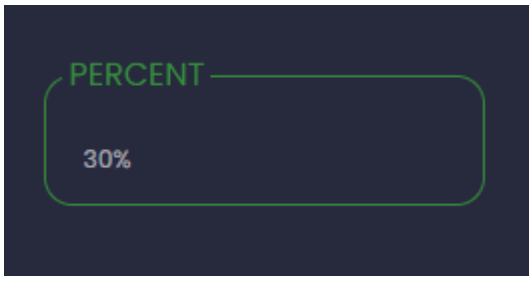
```
    notificationhandler = () => { //συνάρτηση για την εμφάνισει μηνυμάτων.
        const options = {
            place: "tr",
            message: (
                <div style={{ 'userSelect': 'none' }}>
                <div style={{ 'userSelect': 'none' }}>
                    Selecte first <b style={{ 'userSelect': 'none' }}>NER field</b>!
                </div>
                </div>
            ),
            type: "warning",
            icon: "tim-icons icon-bell-55",
            autoDismiss: 3,
            }
            this.refs.notify.notificationAlert(options);
    }
    butonClick = (e) => { //συνάρτηση όπως αλλάζει το ner πεδίο με αυτό που εχει επιλεγχθεί από την λίστα
        var elemner = document.documentElement.getElementsByClassName("is-selected-ner")[0];

        try{
            this.setState({
                ner: elemner.getAttribute("value"),
                hasError: false
            })
            this.setStyle(elemner.getAttribute("value"));
            this.props.onClick(false);
        }catch(error)
        {
            this.setState({
                hasError: true
            })
            this.notificationhandler()
        }
    }
```

```
    clearText = (text) => { //συνάρτηση που καθαρίζει το επιλεγμένο κείμενο
        var totaltext = "";
        var splitext = text.split(" ");
        for(var i = 0; i < splitext.length; i++)
            if(!this.isNerField(splitext[i]))
                totaltext = totaltext + " " + splitext[i];
        return totaltext;
    }
    isNerField = (text) => { //συναρτηση για τον έλεγχος για τον έχει επιλεγχθεί και λεκτικό εκτος του κειμενου.
        if(text === "" || text === null || text === undefined) return false;
        let data = dataner.find(da => da.name.toLowerCase() === text.toLowerCase());
        if(data !== undefined)
            return true;
        else
            return false;
    }
```

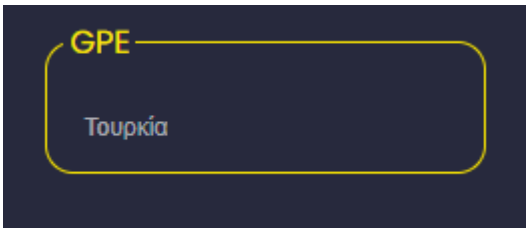
```
117     render() {
118         return(
119             <>
120             <Col className = 'col-3' key={this.props.token.id} >
121                 <fieldset id={ this.props.id } className = 'myfieldset' style = {{ border: this.state.stylefieldset }}
122                         onClick={(e) => this.butonClick(e)}
123                         >
124                     <legend className = 'mylegend'
125                             style = {{ color: this.state.stylelegend, 'userSelect': 'none' }}>
126                             { this.state.ner }
127                     </legend>
128                     {this.props.token.text.split(" ").map((tok) => {
129                         return(
130                             <>
131                             <label id={ this.props.id + "#" + tok } style={{'cursor': 'text'}}
132                                 className="space_words_ner">
133                                     { tok }
134                             </label>
135                             <label><br></br></label>
136                             </>
137                         );
138                     })
139                     }
140                 </fieldset>
141                 <NotificationAlert ref="notify"/>
142             </Col>
143             </>
144         );
145     }
146 }
147
148 export default FieldToken;
```

Εικόνα FieldToken.js

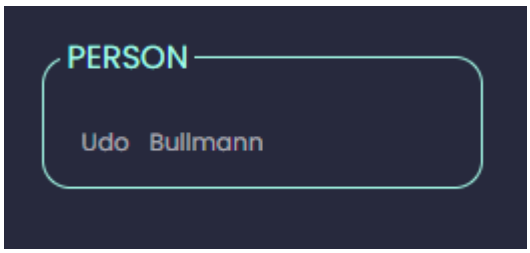
Το αποτέλεσμα του παραπάνω κώδικά, φαίνεται στις ακόλουθες εικόνες:



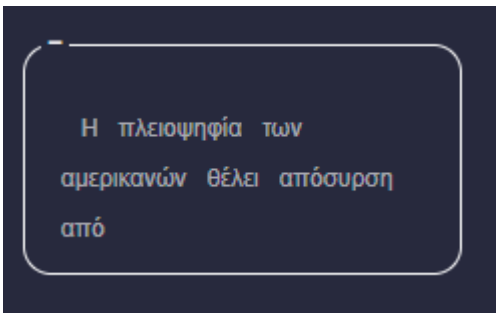
Α.



Β.



Γ.



Δ.

Ε.

Θ.

Στη συνέχεια, έχουμε το components «/src/components/GeneralOptions», σε αυτό υπάρχουν κυρίως γενικευμένα components. Εδώ θα δούμε τα components για ένα popup παράθυρο, το component για το ανέβασμα ενός αρχείου κτλ.

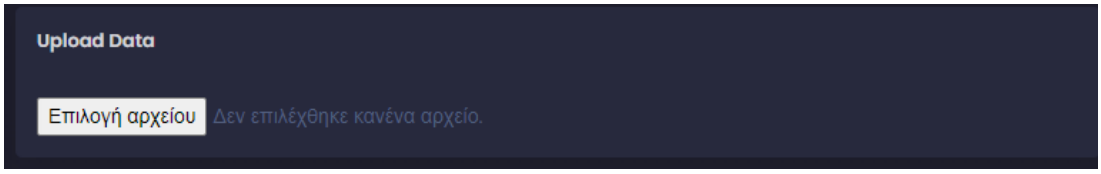
Με αυτό που θα ξεκινήσουμε είναι το με το components που δίνει τι δυνατότητα στον χρήστη να ανεβάσει το αρχείο του «/src/components/GeneralOptions/SelectedFiles».

```
import React, { Component } from 'react';
import { Card, CardHeader, CardBody, Form, Row, Col, Input } from 'reactstrap';

import "assets/css/mystyle.css";

class SelectedFile extends Component{
    constructor(props){
        super(props);
        this.state = {
            title: null //μια ετικέτα της καρτέλας που αφορού το uploader
        }
    }

    componentDidMount = () => {
        this.setState({
            title: this.props.title
        })
    }

    handleClickFilePackage = (e) =>{  //ο handler με την επιλογή του αρχείου, ο οποιος καλεί αυτό που εχει δοθε
                                      //απο τον πατέρα
        this.props.onChange(e);
    }
```

```
24
25      render(){
26          return(
27              <Col>
28                  <Card>
29                      <CardHeader>
30                          <h5 className="title">{this.state.title}</h5>
31                      </CardHeader>
32                      <CardBody>
33                          <Form>
34                          <Row>
35                              <Col>
36                              <Input type="file" onChange={(e) => this.handleClickFilePackage(e)}/
37                              </Col>
38                          </Row>
39                          </Form>
40                      </CardBody>
41                  </Card>
42              </Col>
43          )
44      }
45  }
46
47  export default SelectedFile;
```

Εικόνα SelectedFiles.js

Το αποτέλεσμα του παραπάνω κώδικα είναι:



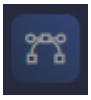
Εικόνα SelectedFiles.js(2)

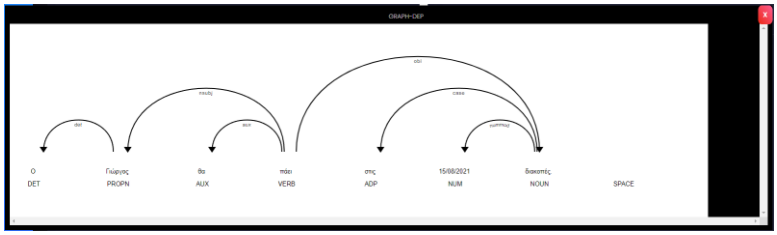
Έπειτα θα προχωρήσουμε στο component που σχετίζεται με το άνοιγμα ενός popup παραθύρου «/src/components/GeneralOptions/PopupWindow.js».

```jsx
import React, { Component } from 'react';
import {
    Button
  } from "reactstrap";
import "assets/css/mystyle.css";

class PopupWindow extends Component{
    constructor(props){
        super(props);
        this.state = { indexKey: this.props.indexKey}; //ορίζουμε στο state ένα indexkey, όπου θεωρητικά ένα κλεδ
    }

    handleClick = (event) =>{
        this.props.onClick(this.state.indexKey); //ο listener για το click, το άνοιγμα του popup
    }

    handleClose = (event) =>{ //ο listener για το κλείσιμο του popup
        this.props.handleClose(event);
    }

    render(){
        return(
            <>
            {
            this.props.openPopup ?
            <div className="popup-main">
                <p>GRAPH-DEP</p>
                <div className="popup-close">
                    <Button color="danger" className="popup-close-button" onClick={this.handleClose}>X</Button>
                </div>
                <div className="popup-mid">
                    { this.props.children }
                </div>
            </div>
            :
                <Button style={{padding: '1%'}} onClick={this.handleClick}>
                <i className="tim-icons icon-vector"/>
                </Button>
            }
            </>
        )
    }
}export default PopupWindow;
```

Εικόνα PopupWindow.js

Το αποτέλεσμα του παραπάνω κώδικα είναι το παρακάτω ένα button με icon:





Εικόνα PopupWindow.js(2)

Το επόμενο component έχει να κάνει με την εμφάνιση των προεπιλεγμένων πακέτων με παραδείγματα και διάφορων άλλο πληροφορίων (π.χ. ποιο πακέτο χρησιμοποιείται, τον αριθμό του παραδείγματος κτλ). Παρακάτω ακολουθεί ο αντίστοιχος κώδικας.

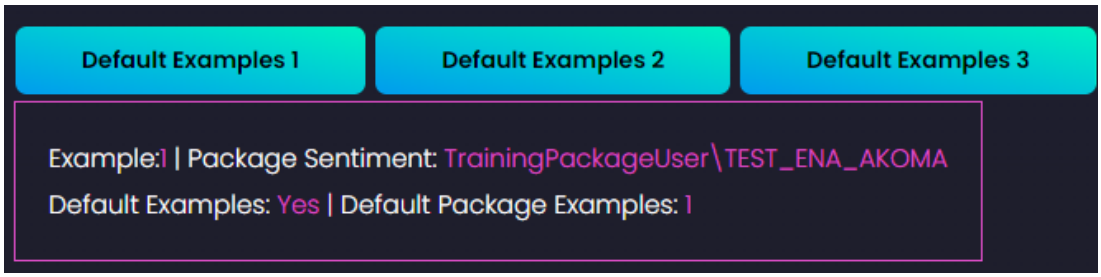
```jsx
import React, { Component } from 'react';
import { Row, Button } from 'reactstrap';
import "assets/css/mystyle.css";

class ButtonExamplesAndInf extends Component{
    constructor(props){
        super(props);
        this.state = {
            title: props.title,
            index_package: ["1", "2", "3"] //index για τα πακέτα, με τα υπάρχων παραδείγματα
        }
    }

    componentDidMount() { //ενημερωση του state με τις αλλαγές
        this.setState({
            title: this.props.title,  //τίτλος
            text_num: this.props.text_num, //ο αριθμός του παραδείγματος
            file_name_package: this.props.file_name_package, //το όνομα του πακέτου spacy που χρησιμοποιείτα
            packagedefaultexample: this.props.packagedefaultexample, //αν το παράδειγμα ειναι από τα default
            loaddefaultexamples: this.props.loaddefaultexamples //ποιο index απο τα default πακέτα
        })
    }


    LoadDefaultExamples = (index) =>{  //listener για την φόρτωση ενός από τα default πακέτα
        this.props.onClick(index);
    }
```

```jsx
    render(){
        return(
            <>
                <Row>
                    <Button color="success" style={{ color:"black" }}
                        onClick= { (e) => this.LoadDefaultExamples(this.state.index_package[0]) }
                    >
                        Default Examples 1 {/**1ο Default πακέτο */}
                    </Button>
                    <Button color="success" style={{ color:"black" }}
                        onClick= { (e) => this.LoadDefaultExamples(this.state.index_package[1]) }
                    >
                        Default Examples 2 {/**2ο Default πακέτο */}
                    </Button>
                    <Button color="success" style={{ color:"black" }}
                        onClick= { (e) => this.LoadDefaultExamples(this.state.index_package[2]) }
                    >
                        Default Examples 3 {/**3ο Default πακέτο */}
                    </Button>
                </Row>
                <Row>
```

```
                    <blockquote>
                        <p className="blockquote blockquote-primary">
                        <h9 style={{ color:"white" }}>Example:</h9>
                        { this.props.text_num }
                        <h9 style={{ color:"white" }}> | Package Sentiment:
                        </h9> { this.props.file_name_package == null ?
                        'Default'
                        :  this.props.file_name_package }
                        <br />
                        <h9 style={{ color:"white" }}>Default Examples:
                        </h9> {  this.props.loaddefaultexamples ? 'Yes' : 'No' }
                        <h9 style={{ color:"white" }}> | Default Package Examples:
                        </h9> {  this.props.packagedefaultexample == null ?
                        '-' :  this.props.packagedefaultexample }<br/>
                        </p>
                    </blockquote>
                </Row>
            </>
        )
    }
}

export default ButtonExamplesAndInf;
```

Εικόνα ButtonExamplesAndInf.js

Το αποτέλεσμα του παραπάνω component είναι το ακόλουθο:



Εικόνα ButtonExamplesAndInf.js(2)

Τέλος, στο «/src/components/GeneralOptions» θα βρούμε ένα πιο γενικό component που περιέχει κάποια από τα παραπάνω με επιπλέων ένα πεδίο, όπου μπορεί ο χρήστης να δώσει απευθείας το αριθμό του παραδείγματος που θέλει να εξετάσει (π.χ. το 21ο παράδειγμα). Το όνομα του component είναι το «GeneralSettings» και ο κώδικας είναι ο παρακάτω:

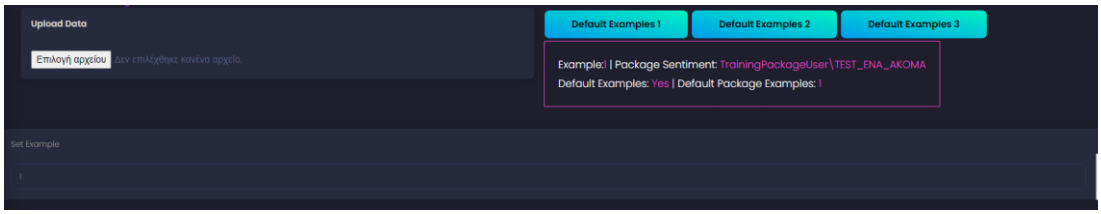
```
import React, { Component } from 'react';
import { Col, Row, Card, CardBody, CardHeader, Input } from 'reactstrap';
import SelectedFile from './SelectedFiles';
import ButtonExamplesAndInf from './ButtonExamplesAndInfo';
import "assets/css/mystyle.css";

class GeneralSettings extends Component{
    constructor(props){
        super(props);
        this.state = {
            title: null
        }
        this.handleFileSelect = this.handleFileSelect.bind(this);
        this.LoadDefaultExamples = this.LoadDefaultExamples.bind(this);
        this.changeCounter = this.changeCounter.bind(this);
    }

    handleFileSelect = event => {
        this.props.handleFileSelect(event);
    }

    LoadDefaultExamples = event => {
        this.props.LoadDefaultExamples(event);
    }

    changeCounter = event => {
        this.props.changeCounter(event);
    }
```

```
    render(){
        return(
            <Row>
                <Col md="6">
                    <SelectedFile title='Upload Data' onChange={(e) => this.handleFileSelect(e)}/>
                </Col>
                <Row>
                    <Col>
                        <ButtonExamplesAndInf onClick={ (e) => this.LoadDefaultExamples(e) }
                            text_num={ this.props.text_num }
                            file_name_package={ this.props.package_name }
                            loaddefaultexamples={ this.props.loaddefaultexamples }
                            packagedefaultexample={ this.props.packagedefaultexample }
                            />
                    </Col>
                </Row>

                <Card>
                    <CardHeader>
                        <h5 className="card-category">Set Example</h5>
                    </CardHeader>
                    <CardBody key="table_for_add_word_sentiment"
                                style={{ 'maxHeight': "400px", 'overflowY': 'scroll' }}>
                        <Row>
                        <Col>
                            <Input id="change-counter-user-not-selected" type="text"
                                placeholder={this.props.text_num} onBlur={(e) => this.changeCounter(e) }
                            </Input>
                        </Col>
```

```
59                                  </Row>
60                                </CardBody>
61                              </Card>
62                            </Row>
63                        )
64                    }
65                }
66
67                export default GeneralSettings;
```

Εικόνα GeneralSettings.js

Το αποτέλεσμα του παραπάνω component είναι το ακόλουθο:



Εικόνα GeneralSettings.js(2)

Το επόμενο component έχει να κάνει με τα δεδομένα της συναισθηματικής ανάλυσης. Χρησιμοποιείται στην δεύτερη κεντρική επιλογή της εφαρμογής και έχει να κάνει μόνο με την παρουσίαση των αποτελεσμάτων συναισθηματικής ανάλυσης ενός κειμένου. Το όνομα του component είναι «EmotionsChart» και βρίσκεται στα «/src/components/Emotions». Παρακάτω θα δείτε τον κώδικα και τα αποτελέσματα του:

```
import React, { Component } from 'react';
import {
    Col,
    Card,
    CardHeader,
    CardTitle,
    CardBody,
    Table
} from "reactstrap";
import "assets/css/mystyle.css";

import
{
  Bar,
  Pie
} from "react-chartjs-2";

import axios from "axios";


```

```
class EmotionsChart extends Component {
    constructor(props){
        super(props);
        this.state = {
            text: props.text, //το κείμενο που αναλύθηκε
            emotion_score: null, //το συναισθηματικό σκόρ
            result: null, //το αποτέλεσμα
            emotions: null, //οι βαθμολογίες ανά συναίσθημα
            list_matched_token: null, //η λιστα με τα tokens που υπάρχουν στο κείμενο και στο συναισθηματικό λεξικό
            matched_token: null, //αριθμός των tokens που συμετέχουν στο τελικό συναισθήμα
            matched_tokens_pos: null, //τα tokens που συμμετέχουν στο τελικό συναιθσήμα
            matched_tokens_pos_col: null, //οι τιμές του κάθε token που συμμετέχει στο τελικό συναίσθημα
            main_emotion: null, //το κύριο συναίσθημα
        }

        this.chartExample3 = { //το style των δεδομένων που θα εμφανίζονται στο πρώτο γραάφημα
            data: (canvas) => {
              let ctx = canvas.getContext("2d");

              let gradientStroke = ctx.createLinearGradient(0, 230, 0, 60);

              gradientStroke.addColorStop(1, "rgba(72,72,176,0.1)");
              gradientStroke.addColorStop(0.4, "rgba(72,72,176,0.0)");
              gradientStroke.addColorStop(0, "rgba(119,52,169,0)"); //purple colors
```

```
46
47              return {
48                labels: ["Anger", "Disgust", "Fear", "Happiness", "Sadness", "Surprise"],
49                datasets: [
50                  {
51                    label: "Emotion",
52                    fill: true,
53                    backgroundColor: gradientStroke,
54                    hoverBackgroundColor: gradientStroke,
55                    borderColor: "#d048b6",
56                    borderWidth: 2,
57                    borderDash: [],
58                    borderDashOffset: 0.0,
59                    data: this.state.emotions,
60                  },
61                ],
62              };
63            },
64            options: {
65              maintainAspectRatio: false,
66              legend: {
67                display: false,
68              },
69              tooltips: {
70                backgroundColor: "#f5f5f5",
71                titleFontColor: "#333",
72                bodyFontColor: "#666",
73                bodySpacing: 4,
74                xPadding: 12,
75                mode: "nearest",
```

```
      intersect: 0,
      position: "nearest",
    },
    responsive: true,
    scales: {
      yAxes: [
        {
          gridLines: {
            drawBorder: false,
            color: "rgba(225,78,202,0.1)",
            zeroLineColor: "transparent",
          },
          ticks: {
            suggestedMin: 60,
            suggestedMax: 100,
            padding: 10,
            fontColor: "#9e9e9e",
          },
        },
      ],
      xAxes: [
        {
          gridLines: {
            drawBorder: false,
            color: "rgba(225,78,202,0.1)",
            zeroLineColor: "transparent",
          },
          ticks: {
            padding: 20,
            fontColor: "#9e9e9e",
          },
        },
      ],
    },
  },
};
```

```
113            this.chartExample4 = { //το αντίστοιχο style που 2ου γραφήματος
114              data: (canvas) => {
115                let ctx = canvas.getContext("2d");
116
117                let gradientStroke = ctx.createLinearGradient(0, 230, 0, 60);
118
119                gradientStroke.addColorStop(1, "rgba(72,72,176,0.1)");
120                gradientStroke.addColorStop(0.4, "rgba(72,72,176,0.0)");
121
122                return {
123                  labels: ["N/A", "BOTH", "NEG", "POS"],
124                  datasets: [
125                    {
126                      label: "Emotion",
127                      fill: true,
128                      hoverBackgroundColor: gradientStroke,
129                      backgroundColor: [
130                        'rgba(255, 206, 86, 1)',
131                        'rgba(54, 162, 235, 1)',
132                        'rgba(255, 99, 132, 1)',
133                        'rgba(75, 192, 192, 1)',
134                      ],
135                      borderColor: [
136                        'rgba(255, 206, 86, 1)',
137                        'rgba(54, 162, 235, 1)',
138                        'rgba(255, 99, 132, 1)',
139                        'rgba(75, 192, 192, 1)',
140                      ],
141                      borderWidth: 2,
142                      data: [20,30,40,10],
143                    },
144                  ],
145                };
146              }
147            };
148        }
```

```
getGetEmotionAnalysis = () => { //μετά την αρχικοποιήση του state, πραγματοποιείται κλήση στο back-end
                                //ώστε να πάρουμε τα νέα δεδομένα.
  this.setState({
        emotion_score: null,
        result: null,
        emotions: null,
        matched_token: null,
        list_matched_token: null,
        matched_tokens_pos: null,
        main_emotion: null
    }, () => {
            let formdata = //η δομή με τα δεδομένα που στέλνουμε
            {
              text: this.state.text, package:
              this.props.package_name,
              sentiment_file: this.props.sentiment_file
            }
            axios.get("http://127.0.0.1:5000/exespacy/emotionalysis", {params: formdata},
                {
                    headers: {
                        'Content-Type': 'application/json',
                        "Access-Control-Allow-Origin": "*"
                    }
                }
                ).then(response => { //όταν παρουμε απάντηση δημιουργούμε το νέο state
```

```
                    console.log(response.data)
                    this.setState({
                        emotion_score: response.data.Resp.emotion_score,
                        result: response.data.Resp.result,
                        emotions: this.getEmotions(response.data.Resp.emotions),
                        matched_token: response.data.Resp.matched_tokens_num,
                        list_matched_token: response.data.Resp.matched_tokens_list,
                        matched_tokens_pos: response.data.Resp.matched_tokens_pos,
                        matched_tokens_pos_col: response.data.Resp.matched_tokens_pos_col,
                        main_emotion: response.data.Resp.main_score
                    })
                })
                .catch(error => { //αν υπάρχει σφάλμα
                    console.error(error)
                });
            })
  }
```

```
getEmotions = (emotions) => { //μεττρέπουμε τους βαθμούς ανα συναίσθημα ώστε να είνα %.
    return emotions != null ? [
        emotions['anger'] * 100,
        emotions['disgust'] * 100,
        emotions['fear'] * 100,
        emotions['happiness'] * 100,
        emotions['sadness'] * 100,
        emotions['surprise'] * 100,
    ]
    : []
}
findPosMatched = (cell, key, count_pos_matched) => { //συνάρτηση για τον εντοπίσμο τον βαθμών
                                                     //συναισθήματος ανά λέξεων σύμφωνα με τα δεδομένα
  const findIndexElement = (element) => element === cell[0];
  const pos = this.state.matched_tokens_pos_col[this.state.matched_tokens_pos.findIndex(findIndexElement)]
  if( key === 0 || key === Number(pos) || key === count_pos_matched + 5)
  {
      return true;
  }
  return false;
}
getMatchedList = (list) => { //αχεδιάζει τα δεδομένα ενός πίνακα
```

```
getMatchedList = (list) => { //σχεδιάζει τα δεδομένα ενός πίνακα
  var count_pos_matched = 0;
  return list.map((ele, key) => (
      this.inMatchedListPos(ele[0]) ?
      <tr key= {ele[0] + key} id={count_pos_matched = key}>
          { ele.map((cell, keycell) => (
              this.findPosMatched(ele, keycell, count_pos_matched) ?
                  <th className="color_matched_token_pos" key={ele[0]+key+keycell}
                  id={count_pos_matched = keycell-1}>
                      { cell }
                  </th>
              :
              <th key={ele[0]+key+keycell} >
                  { cell }
              </th>
          ))
          }
      </tr>
      : <tr key= {key} /*contentEditable={ 'true' }*/>
          { ele.map((cell, keycell) => (
              <th key={ele[0]+key+keycell}>{cell}</th>
          ))}
      </tr>
  ));
}
inMatchedListPos = (cell) => { //αν το λεκτικό ανήκει στα ματσαρισμένα tokens
    return this.state.matched_tokens_pos.some(x => x === cell);
}
```

```
componentDidMount = () =>{
    this.setState({
        emotion_score: null,
        result: null,
        emotions: null,
        matched_token: null,
        list_matched_token: null,
        matched_tokens_pos: null,
        main_emotion: null
    })
    this.getGetEmotionAnalysis();
}
```

```
render() {
    if(this.state.result != null)
    {
        return(
            <Col lg="4">
                <Card className="card-chart">
                    <CardHeader>
                    <h5 className="card-category">Emotion Analysis</h5>
                    </CardHeader>
                    <CardBody>
                        { this.state.result }
                    </CardBody>
                </Card>
            </Col>
        );
    }
```

```
270         else if( this.state.emotion_score != null)
271         {
272             return (
273                 <>
274                 <Col md="4">
275                     <Card className="card-chart">
276                         <CardHeader>
277                         <h5 className="card-category">Emotion Analysis</h5>
278                         <CardTitle tag="h3">
279                             { this.state.emotion_score }
280                         </CardTitle>
281                         <CardTitle tag="h3">
282                             { this.state.main_emotion }
283                         </CardTitle>
284                         </CardHeader>
285                         <CardBody>
286                         <div className="chart-area">
287                             <Bar
288                             data={this.chartExample3.data}
289                             options={ this.chartExample3.options}
290                             />
291                         </div>
292                         </CardBody>
293                     </Card>
294                 </Col>
295                 <Col md="3">
296                   <Card className="card-chart">
297                         <CardHeader>
298                         <h5 className="card-category">Emotion Analysis</h5>
299                         <CardTitle tag="h3">
300                             { this.state.emotion_score }
301                         </CardTitle>
302                         </CardHeader>
303                         <CardBody>
304                         <div className="chart-area">
305                             <Pie data={this.chartExample4.data}/>
306                         </div>
307                         </CardBody>
308                     </Card>
309                   </Col>
310                 <Col md="12">
311                     <Card>
312                       <CardHeader>
313                         <h5 className="card-category">Matched Tokens</h5>
314                         <CardTitle tag="h3">
315                           { this.state.matched_token }
316                         </CardTitle>
317                       </CardHeader>
318                       <CardBody style={{ 'maxHeight': "500px", 'overflowY': 'scroll' }}>
319                         <Table className="tablesorter">
320                         <thead className="text-primary">
321                           <tr>
322                             <th className="text-center">Term</th>
323                             <th className="text-center">POS1</th>
324                             <th className="text-center">POS2</th>
325                             <th className="text-center">POS3</th>
```

```
                                    <th className="text-center">POS4</th>
                                    <th className="text-center">Subjectivity1</th>
                                    <th className="text-center">Subjectivity2</th>
                                    <th className="text-center">Subjectivity3</th>
                                    <th className="text-center">Subjectivity4</th>
                                    <th className="text-center">Polarity1</th>
                                    <th className="text-center">Polarity2</th>
                                    <th className="text-center">Polarity3</th>
                                    <th className="text-center">Polarity4</th>
                                    <th className="text-center">Anger1</th>
                                    <th className="text-center">Anger2</th>
                                    <th className="text-center">Anger3</th>
                                    <th className="text-center">Anger4</th>
                                    <th className="text-center">Disgust1</th>
                                    <th className="text-center">Disgust2</th>
                                    <th className="text-center">Disgust3</th>
                                    <th className="text-center">Disgust4</th>
                                    <th className="text-center">Fear1</th>
                                    <th className="text-center">Fear2</th>
                                    <th className="text-center">Fear3</th>
                                    <th className="text-center">Fear4</th>
                                    <th className="text-center">Happiness1</th>
                                    <th className="text-center">Happiness2</th>
                                    <th className="text-center">Happiness3</th>
                                    <th className="text-center">Happiness4</th>
                                    <th className="text-center">Sadness1</th>
                                    <th className="text-center">Sadness2</th>
                                    <th className="text-center">Sadness3</th>
                                    <th className="text-center">Sadness4</th>
```

```
                                    <th className="text-center">Surprise1</th>
                                    <th className="text-center">Surprise2</th>
                                    <th className="text-center">Surprise3</th>
                                    <th className="text-center">Surprise4</th>
                                    <th className="text-center">Aditional1</th>
                                    <th className="text-center">Aditional2</th>
                                    <th className="text-center">Aditional3</th>
                                    <th className="text-center">Aditional4</th>
                                    <th className="text-center">Comments1</th>
                                    <th className="text-center">Comments2</th>
                                    <th className="text-center">Comments3</th>
                                    <th className="text-center">Comments4</th>
                                </tr>
                            </thead>
                            <tbody>
                                { this.getMatchedList(this.state.list_matched_token) }
                            </tbody>
                            </Table>
                        </CardBody>
                    </Card>
                </Col>
            </>
        );
    }
    else
    {
        return null
    }
```

```
383        }
384    }
385
386    export default EmotionsChart;
```

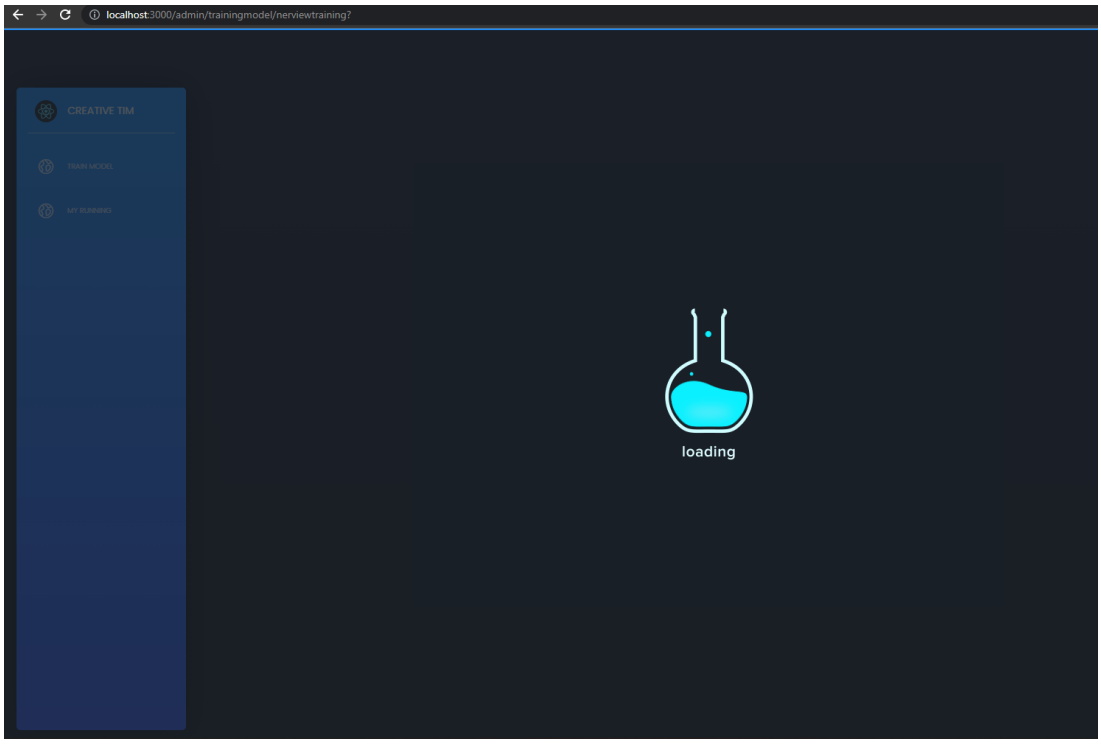
Εικόνα EmotionsChart.js

Εικόνα EmotionsChart.js(2)

Επιπλέον, ως component δημιουργήσαμε ένα παράθυρο φόρτωσης, θα θέλαμε όταν τρέχει το back-end την εκπαίδευση του συστήματος να μην εκτελείται κάτι άλλο παράλληλα. Αυτή την δουλειά την κάνει το «LoadingProgress», παρακάτω παρουσιάζουμε των κώδικα και τα αποτελέσματα:

```
1
2   import React, { Component } from "react";
3
4   import "assets/css/mystyle.css";
5
6   class LoadingProgress extends Component {
7       render(){
8           return (
9               <>
10              <div className="my-div-loading loading">
11                  <div className="my-loading"/>
12              </div>
13              </>
14          );
15      }
16  }
17
18  export default LoadingProgress;
19
```

A.



B.

Εικόνα LoadingProgress.js

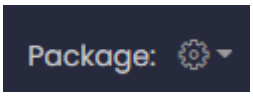
Τέλος, δημιουργήσαμε δυο ακόμα components, με σκοπό να μπορεί ο χρήστης να επιλέξει το πακέτο ή το συναισθηματικό λεξικό με το οποίο θέλει να εκτελέσει τις ενέργειες

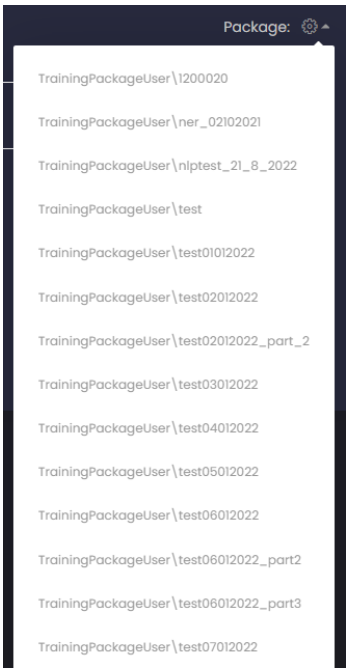
του. Με αυτό τρόπο μπορέσαμε να συγκρίνουμε τα πακέτα μεταξύ τους και κατά πόσο η εκπαίδευση του είναι προοδευτική ή όχι. Τα components είναι τα «SettingSpacyPackage» και «SettingSentimentLexicons» για την επιλογή του spacy ή του συναισθηματικού λεξικού αντίστοιχα, ενώ κώδικα και το αποτελέσματα αυτό εμφανίζονται παρακάτω.

```
1
2   import React, { Component } from "react";
3
4   import {
5     UncontrolledDropdown,
6     DropdownToggle,
7     DropdownMenu,
8     DropdownItem
9   } from "reactstrap";
10
11  import axios from "axios";
12  import "assets/css/mystyle.css";
13
14  class SettingSpacyPackage extends Component {
15      constructor(props){
16          super(props);
17          this.state = {
18              packagies: null, //η λίστα με τα διαθέσημα πακέτα του back-end
19              package_name: null //το όνομα του πακέτου
20          }
21      }
22      componentDidMount = () =>{
23          this.getAllSentimentLexicons(); //κλήση συνάρτησης για να μου επιστρέχει τα πακέτα.
24      }
25      getAllSentimentLexicons = () => { //συνάρτηση που καλεί το back-end και αυτό μου
26                                        //επιστρέφει τα διαθέσιμα πακέτα
27          axios.get("http://127.0.0.1:5000/nertraining/getspacypackage").then(response => {
28              this.setState({
29                  packagies: response.data.Resp
30              });
31          })
```

```
32          .catch(error => {
33              console.error(error)
34          });
35      }
36      setPackagiesSpacy = (list) => { //για κάθε πακέτο σχεδιάζεται ως ενα dropdownitem
37          return list.map((file_name, key) => {
38              return (
39                  <DropdownItem key = { key }
40                      href="#pablo"
41                      onClick={(e) => this.buttonchangepackage(e)}
42                      >
43                      { file_name }
44                  </DropdownItem>
45              )
46          })
47      }
48      buttonchangepackage = Event => { //ο listener μολις επιλέξουμε κάποιο πακέτο
49          this.props.onClick(Event.target.innerText)
50      }
```

```
51    render(){
52      return (
53          <>
54          <UncontrolledDropdown>
55          Package:
56              <DropdownToggle
57                  caret
58                  className="btn-icon"
59                  color="link"
60                  data-toggle="dropdown"
61                  type="button"
62              >
63                  <i className="tim-icons icon-settings-gear-63" />
64              </DropdownToggle>
65              <DropdownMenu aria-labelledby="dropdownMenuLink" right>
66                  {
67                      this.state.packagies != null ? this.setPackagiesSpacy(this.state.packagies
68                      : <DropdownItem
69                              href="#pablo"
70                              >
71                      </DropdownItem>
72                  }
73              </DropdownMenu>
74          </UncontrolledDropdown>
75          </>
76      );
77    }
78  }
79
80  export default SettingSpacyPackage;
```



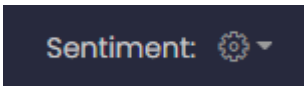


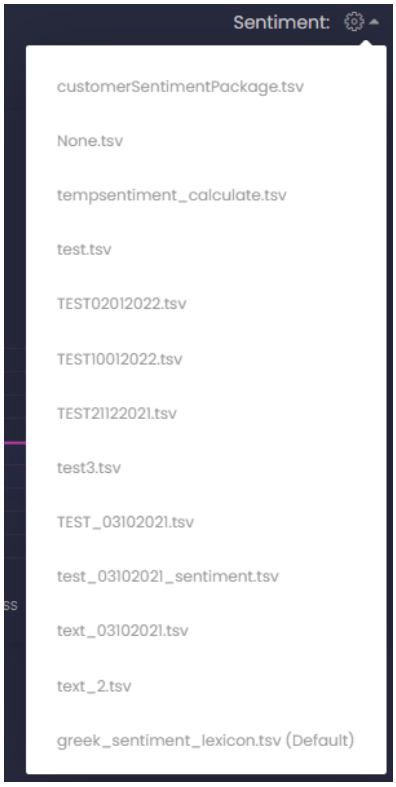
Εικόνα SettingSpacyPackage.js

Ο κώδικα δεν διαφέρει από το παραπάνω, εκτός από μέρος που καλείται το back-end.

```jsx
import React, { Component } from "react";

// reactstrap components
import {
  UncontrolledDropdown,
  DropdownToggle,
  DropdownMenu,
  DropdownItem
} from "reactstrap";

import axios from "axios";
import "assets/css/mystyle.css";

class SettingSentimentLexicons extends Component {
    constructor(props){
        super(props);
        this.state = {
            sentiment_lexicons: null,
            file_name_sentiment: null
        }
    }
    componentDidMount = () =>{
        this.getAllSentimentLexicons();
    }
    getAllSentimentLexicons = () => {
        axios.get("http://127.0.0.1:5000/nertraining/getsentimentlexicon").then(response => {
            //console.log(response)
            this.setState({
                sentiment_lexicons: response.data.Resp
            });
        })
            .catch(error => {
                console.error(error)
            });
    }
    setSentimentLexicons = (sentiment_list) => {
        return sentiment_list.map((file_name, key) => {
            return (
                <DropdownItem key = { key }
                    href="#pablo"
                    onClick={(e) => this.buttonchangefileSentiment(e)}
                    >
                    { file_name }
                </DropdownItem>
            )
        })
    }
    buttonchangefileSentiment = Event => {
        this.props.onClick(Event.target.innerText)
    }
```

```
render(){
  return (
      <>
      <UncontrolledDropdown>
          Sentiment:
          <DropdownToggle
              caret
              className="btn-icon"
              color="link"
              data-toggle="dropdown"
              type="button"
          >
              <i className="tim-icons icon-settings-gear-63" />
          </DropdownToggle>
          <DropdownMenu aria-labelledby="dropdownMenuLink" right>
              {
                  this.state.sentiment_lexicons != null ? this.setSentimentLexicons(this.state.sentiment_lexicons
                  : <DropdownItem
                          href="#pablo"
                          >
                  </DropdownItem>
              }
          </DropdownMenu>
      </UncontrolledDropdown>
      </>
  );
 }
}

export default SettingSentimentLexicons;
```





Εικόνα SettingSentimentLexicons.js

Σε αυτό το σημείο, έχουμε ολοκληρώσει με τα components, στην συνέχεια θα παρουσιάσουμε όλα τα views που σχεδιάστηκαν για την εφαρμογή. Όλα τα παρακάτω που θα αναφέρουμε έχουν σχέση με τις κύριες οθόνες της εφαρμογής και βρίσκονται στο «/src/views».

Για αρχή θα δούμε το αρχείο «TrainingModel.js», όπου αποτελεί την κύριο layout με τα routes μας, αυτά του αρχείου «viewstrainingmode.js» που είδαμε στην αρχή αυτού του

κεφαλαίου. Είναι ο βασικό οδηγός για το path «/admin/trainingmodel» και κάθε πρόσθετο του. Η χρήση του γίνεται από το αρχείο «routes.js», και αποτελεί ένα από τις βασικές επιλογές του κύριου μενού. Παρακάτω παραθέτουμε τον κώδικα:

```
import React from "react";


import viewmodes from 'viewstrainingmode.js';
import {
  Route,
  Switch,
  Redirect,
} from "react-router-dom";


function TrainModel(props) {
  const getRoutes = (routes) => { //συνάρτηση οπου επιστρέφει τα routes παιδία του "viewmodes
    return routes.map((prop, key) => {
      if (prop.layout === "/admin/trainingmodel") {
        return (
          <Route
            path={prop.layout + prop.path} //το path που κτίζεται
            component={prop.component} //το component που θα χρησιμοποιηθεί
            key={key} //ενα κλειδί
          />
        );
      } else {
        return (
          null
        );
      }
    });
  };
```

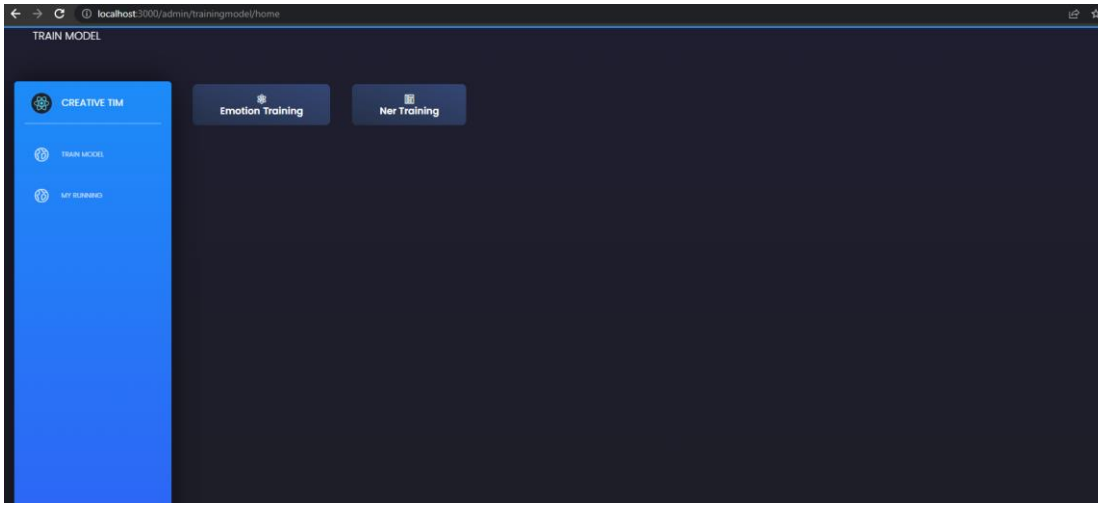
```
  return (
    <>
      <div className="content">
        <Switch>
          {getRoutes(viewmodes)} {/**κλήση συνάρτησης */}
          <Redirect from="*" to="/admin/trainingmodel/home" /> {/**ανακατεύθυνση*/
        </Switch>
      </div>
    </>
  );
}

export default TrainModel;

```

Εικόνα TrainingModel.js

Όπως μπορούμε να διακρίνουμε παραπάνω, το «TrainingModel.js» φορτώνει τα παιδία routes αλλά κάνει αμέσως ανακατεύθυνση (Redirect) στο «/admin/trainingmodel/home». Το path κάνει χρήση του επόμενο αρχείο «TrainingViewHome.js», όπου περιέχει τις δύο υπό-επιλογές στο χρήστη, αυτή της εκπαίδευσης του NER πεδίου και αυτή του εμπλουτισμού του συναισθηματικού λεξικού. Ο κώδικας αναφαίρεται παρακάτω, όπως και αποτελέσματα:

```jsx
import React from "react";

import {
  Button,
  Form,
  Row,
  Col
} from "reactstrap";

import viewmodes from 'viewstrainingmode.js';


function TrainingViewHome(props) {

  const showMode = (viewmodes) => { //συνάρτηση όπου για κάθε αντικείμενο του viewmodes
                                    //δημουργούμαι μια φόρμα με ένα button.
    console.log(this);
    return viewmodes.map((vm, key) => {
      return (
        vm.path !== "/home" ?
        <Form action={vm.layout + vm.path} key={ key }>
          <Col>
            <Button type="submit">
              <i className={ vm.icon }></i> <br/>
              { vm.name }
            </Button>
          </Col>
        </Form>
        : null
      );
    })
  }
```

```jsx
  return (
    <>
      <Row>
        { showMode(viewmodes) } {/**Κλήση συνάρτησης */}
      </Row>
    </>
  );
}

export default TrainingViewHome;
```



Εικόνα TrainingViewHome.js

Το επόμενο αρχείο έχει να κάνει με την οθόνη της εκπαίδευσης του NER πεδίου. Είναι αυτό από το χρησιμοποιείται στο path «/admin/trainingmodel/nerviewtraining». Το αρχείο είναι το «NerViewTraining.js» και παρακάτω παραθέτουμε τον κώδικα:

```
1
2   import React from "react";
3   import ReactDOM from 'react-dom';
4   import {
5     Button,
6     Card,
7     CardHeader,
8     CardBody,
9     CardFooter,
10    FormGroup,
11    Form,
12    Row,
13    Col,
14  } from "reactstrap";
15
16  import dataner from "dataner.js";
17  import FieldToken from "components/FieldToken/FieldToken";
18  import NERField from "components/NERField/NERField";
19  import {
20    Link,
21  } from "react-router-dom";
22
23  import axios from "axios";
24  import SettingSpacyPackage from "components/SettingsPackageSentiment/SettingSpacyPackage"
25  import LoadingProgress from "components/ScreenSettings/LoadingProgress";
26  import GeneralSettings from "components/GeneralOptions/GeneralSettings";
```

```
27
28  function NerViewTraining(props) {
29
30    //Τα states της οθόνης
31    const [loaddefaultexamples, setDefaultExamples] = React.useState(false); //αν τα δεδομένα ειναι απο τα default.
32    const [packagedefaultexample, setPackageDefaultExample] = React.useState(null);  //το επιλεγμένο πακέτο παραδειγμάτω
33    const [count, setCount] = React.useState(1); //ένα counter για το παράδειγμα που εξετάζουμε
34    const [newtoken, setNewToken] = React.useState(null); //η δημιουργία ενός νέου χαρακτηρισμού token
35    const [dataresp, setDataResp] = React.useState(null); //τα αποτελέσματα του response από το back-end
36    const [enableSaveButton, setEnableSaveButton] = React.useState(true); //δυνατότητα ενεργού button για την αποθήκευση
37    const [package_name, setPackageName] = React.useState(null); //το όνομα του πακέτου spacy που χρησιμοποιείται
38
```

```
39
40  const saveButton = Event => { //η συνάρτηση για την απθήκευση τον δεδομένων.
41    var elem = document.getElementsByClassName("myfieldset") //εντοπίζουμε τα όλα fields
42    var para = document.getElementById("paragraph").innerText; //το κείμενο του παραδείγματος
43    console.log(elem);
44    console.log(para);
45
46    let listob = []
47    for(let i = 0; i < elem.length; i++) //για κάθε πεδίο που έχουμε, είτε διαθέτει ner είτε όχι
48    {
49      let ner = elem[i].getElementsByClassName("mylegend")[0].innerText; //η ετικέτα του ner ανα πεδίο
50      if(ner === '-') continue; //κάνουμε προσπερνάμε(skip) ότι δεν έχει ner ετικέτα
51      let val = "" //ορίζουμε το val με το κενό
52      for(let j=0; j < elem[i].getElementsByTagName("label").length; j++) //για κάθε token θα προστεθού στο va
53      {
54        if(j ===  elem[i].getElementsByTagName("label").length-1)
55          val += elem[i].getElementsByTagName("label")[j].innerText.replace('\n', '');
56        else
57          val += elem[i].getElementsByTagName("label")[j].innerText.replace('\n', ' ');
58      } //στο τέλος έχουμε όλο το περιχόμενο του πεδίου
59      listob[i] = {ner: ner, text: val} //μια δομή του ner-text προστίθεται σε μια λίστα
60    }
61    let saveData = {paragraph: para, tokens: listob} //το παράδειγμα και η λιστα με τις δομές ner-text,
62                                                     //τοποθετούνται σε μια νέα δομή paragraph-tokens
63    let dataform = {data: saveData} //η παραπανω δομή θα μπεί σε νέα δομή όπου έχει ένα property το data
64    console.log(dataform)
65    //αποστολη των δεδομένων στο back-end
66    axios.post("http://127.0.0.1:5000/nertraining/savenerdata", dataform).then(response => {
67      // let resp = JSON.parse(response.data.Resp);
68      console.log(response);
69      alert("The change succces saved...");
70    })
71    .catch(error => {
72      console.error(error)
73    });
74  }
```

```
75
76  const buttonchangefilepackage = input_file => { //συνάρτηση επιλογής το πακέτου spacy
77    console.log(input_file)
78    setPackageName(input_file); //ενημέρωση του package_name με το επιλεγμάνο
79    setNewToken(null); //καθαρισμός δεδομένων
80    setDataResp(null); //καθαρισμός δεδομένων
81    getData(); //φόρτωσει εκ νέου δεδομένα
82  }
83
84  const nextButton = () => { //συνάρτηση για την επιλογή του επόμενου παραδείγματος
85      setCount(count+1); //αυξάνουμε κατά ένα το counter
86      setNewToken(null);
87      setDataResp(null);
88      getData();
89  }
90
91  const backButton = () => {//συνάρτηση για την επιλογή του προηγούμενου παραδείγματος
92    if(count === 0) return //αν είμαστε στο 1ο παράδειγμα δεν μπορούμε να πάμε πίσω
93      setCount(count-1) //αφαιρούμε κατά ένα το counter
94      setNewToken(null);
95      setDataResp(null);
96      getData();
97  }
98
99  const changeCounter = (e) =>{ //συνάρτηση που ο χρήστη επιλέγει το παράδειγμα,
100                                //ορίζοντας το counter
101   var elemval = parseInt(document.getElementById("change-counter-user-not-selected").value)
102   console.log(elemval);
103   setCount(elemval);
104   setNewToken(null);
105   setDataResp(null);
106   getData();
107 }
```

```
const changeNerField = (hasChanges) => { //συνάρτηση όπου ελέγχει αν έχουν γίνει αλλαγές,
                                         //ώστε να ενεργοποιηθεί το button save
  setEnableSaveButton(hasChanges); //ενεργοποιήσει του button save
}

const wordClick = (child) => { //συνάρτηση για την δημιουργία νέου token,
                               //από τον χρηστη για τον προσδιορισμού ner
    setNewToken(null);
    let splittoken = { ner: '-', text: child}; //δομή ner-text, όπου ner το κενο ('-')
                                               //και text ότι ειναι επιλεγμένο από τον χρήστη
    let allnewtoken = newtoken; //τα νεα token
    if(allnewtoken === null) allnewtoken = []; //αν δεν υπάρχουν τότε περνάμε μια κενή λίστα
    allnewtoken.push(splittoken); //προσθέτουμε την νεα δομή
    setNewToken(allnewtoken); //ορίζουμε το νέο token
}

const getDataTestToken = (tokens) => { //συνάρτηση όπου για κάθε token επιστρέφει μια λίστα από
                                       //component FieldToken, ενό έχουμε περάσει στο
                                       //onClick -> changeNerField
                                       //και στο wordsClick -> wordClick
  return tokens.map((token, key) => {
    return (
      <FieldToken onClick={ changeNerField } key={key} id={key} token = { token } wordsClick={ (child) => wordClick(child) }/
    )
  })
}
```

```
const handleFileSelect = (e) => { //συνάρτηση για το ανέβασμα του αρχείου από τον χρήστη
  console.log(e.target.files);
  let files = e.target.files;
  let reader = new FileReader();
  reader.readAsDataURL(files[0]);
  reader.onload = (e) => {
    console.log(e.target.result);
    let formData = new FormData();
    formData.append("files", e.target.result);
    axios.post("http://127.0.0.1:5000/exespacy/uploadfile", formData).then(response => { //μετά την απάντησ
      console.log(response)
      setDefaultExamples(false);
      setCount(1); //ορισμός του παραδείγματος από το 1
      setNewToken(null);
      setDataResp(null);
      getData();
    })
    .catch(error => {
      console.error(error)
    });
  }
}
```

```
const getData = () => { //συνάρτηση για το υπολογισμό/εντοπισμό των ner fields ενός κειμένου
  if (dataresp === null)
  {
    //η δομή με τα δεδομένα που στέλνουμε στο back-end
    let formdata = {count_text: count, package: package_name, default_examples: loaddefaultexamples, package_example: packagedefaultexampl
    axios.get("http://127.0.0.1:5000/nertraining", { params: formdata}, {
      headers: {
        'Content-Type': 'application/json',
        "Access-Control-Allow-Origin": "*"
    }
    }).then(response => { //οταν παρουμε απάντηση
      let resp = JSON.parse(response.data.Resp); //μετατρέπου σε μορφή json
      console.log(resp);
      setDataResp(resp); //στο state περνάμε την απάντηση
    })
    .catch(error => {
      console.error(error)
    });
  }
}
```

```
const RunTrainingButton = () => { //συνάρτηση για την εκτέλεση της εκπαίδευσης
  // let copyparent = document.getElementsByClassName('wrapper')[0];
  //render το LoadingProgresss
  ReactDOM.render(
    <LoadingProgress/>,
    document.getElementsByClassName('main-panel')[0]
  );
  let name_package = prompt('Name Package'); //ερωτηση στον χρήστη ώστε να ορίσει το ονομα που επιθυμεί
  console.log(name_package);
  let dataform = {package_name: name_package} //δεδεομέν που θα στείλουμε
  axios.get("http://127.0.0.1:5000/nertraining/trainingPackage", {params: dataform}).then(response => {
      alert("The change succces saved...");
      window.location.reload(false);
    })
    .catch(error => {
    console.error(error);
    window.location.reload(false);
  });
}
```

```
const getDataTest = (datatest, count) => { //συνάρτηση για την εμφάνηση τον αποτελεσμάτων
  return datatest.map((prop, key) => {
    /** η δομή που του datatest είναι η παρακάτω,
     * αλλά εμείς θέλουμε το nermodel
     * {
        "id": 2,
        "nermodel": {
            "id": 2,
            "paragraph": "#Είσαι πολύ **** τελικά! Αλλά τι περιμένεις από έναν **** του χωριού!",
            "tokens": [
                {
                    "ner": "-",
                    "text": " # Είσαι πολύ **** τελικά ! Αλλά τι περιμένεις από έναν **** του χωριού"
                }
            ]
        }
    } */
```

```
    prop = prop.nermodel //πέρνουμε το nermodel
      return (
        prop.id === count ? //θα πρέπει το id να είναι ίδιο του count
        <>
        <Card className="card-tasks my-card-tasks" key = {key}>
          <CardHeader style={{'userSelect': 'none'}}>
            <h5 id="paragraph" className="title">{ prop.paragraph }</h5> {/*το κείμενο της παραγράφου*/}
            <SettingSpacyPackage onClick={ buttonchangefilepackage }/> {/**όλα τα διαθέσημα spacy πακέτα */}
          </CardHeader>
          <CardBody>
            <Form>
              <Row>
                <Col>
                  <FormGroup>
                    <Row id="row-new-token"> {/** προσθήκη του νέου token */}
                      { newtoken !== null ? getDataTestToken(newtoken) : null}
                    </Row>
                    <Row id="row-tokens-fields">
                        { getDataTestToken(prop.tokens) } {/**εμφάνισει όλων των ner-tokens */}
                    </Row>
                  </FormGroup>
                </Col>
              </Row>
            </Form>
          </CardBody>
```

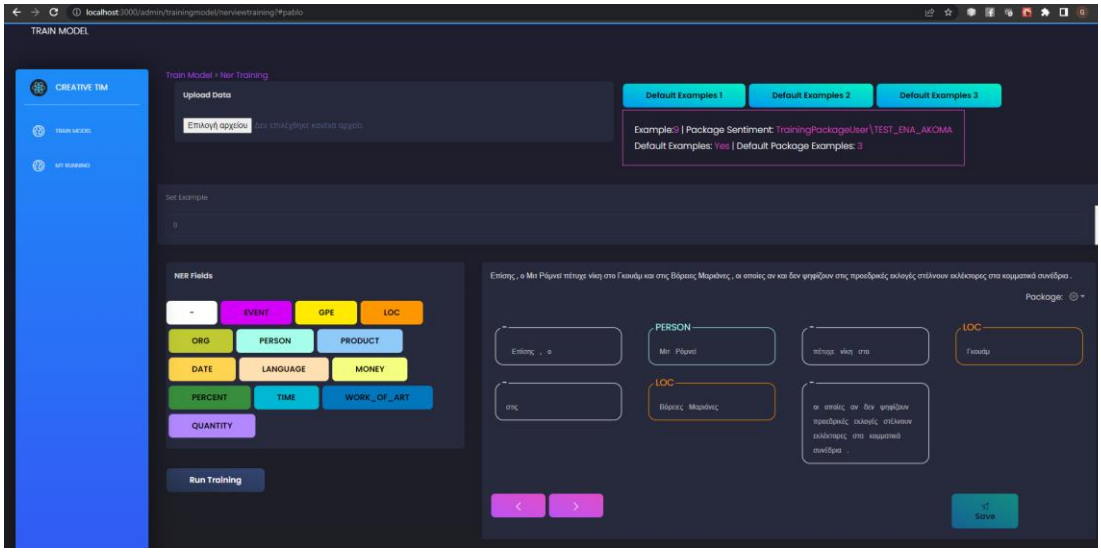
```
                <CardFooter>
                  <Row>
                    <Col>
                      <Button className="btn-fill" color="primary" type="submit"
                              onClick={ backButton }>
                                <i className="tim-icons icon-minimal-left"></i>
                      </Button>
                      <Button className="btn-fill" color="primary" type="submit"
                              onClick={ nextButton }>
                                <i className="tim-icons icon-minimal-right" />
                      </Button>
                    </Col>
                    <Col md="3">
                      <Button disabled={ enableSaveButton } className="btn-fill" color="success" style={{ color:"black" }} type="submit
                              onClick={ (e) => saveButton(e) }>
                                <i className="tim-icons icon-send"/> <br/>
                                Save
                      </Button>
                    </Col>
                  </Row>
                </CardFooter>
              </Card>
              </> : null
            )
          });
        };
```

```
        const LoadDefaultExamples = (value) => { //συνάρτηση επιλογής από τα διαθέσημα default πακέτα παραδειγματω
          setPackageDefaultExample(value);
          setDefaultExamples(true);
          setCount(1);
          setNewToken(null);
          setDataResp(null);
          getData();
        }

```

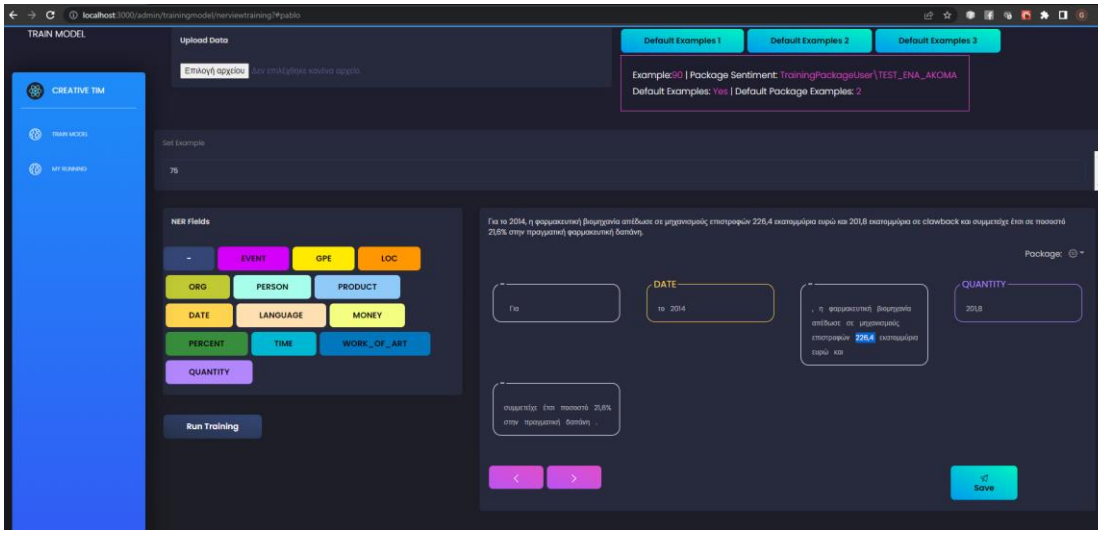
```
        return (
          <>
            <Row>
              <Col>
                <Link to="/admin/trainingmodel"> Train Model </Link>
                <b> { ">" } </b>
                <Link to="/admin/trainingmodel/nerviewtraining"> Ner Training</Link>
              </Col>
            </Row>
            <GeneralSettings handleFileSelect={(e) => handleFileSelect(e)}
            LoadDefaultExamples={(e) => LoadDefaultExamples(e)} text_num={ count } package_name={ package_name }
            loaddefaultexamples={ loaddefaultexamples } packagedefaultexample={ packagedefaultexample }
            changeCounter = {(e) => changeCounter(e) }>
            </GeneralSettings>
            <Row>
              <Col md="4">
                <NERField  style={{'userSelect': 'none'}} nerfields = { dataner }/>
                <Button  style={{'userSelect': 'none'}} onClick = { RunTrainingButton }>Run Training</Button>
              </Col>
                <Col>
                  { getData() }
                  { dataresp !== null ? getDataTest(dataresp, count) : null }
                </Col>
            </Row>
          </>
        );
      }

      export default NerViewTraining;
```

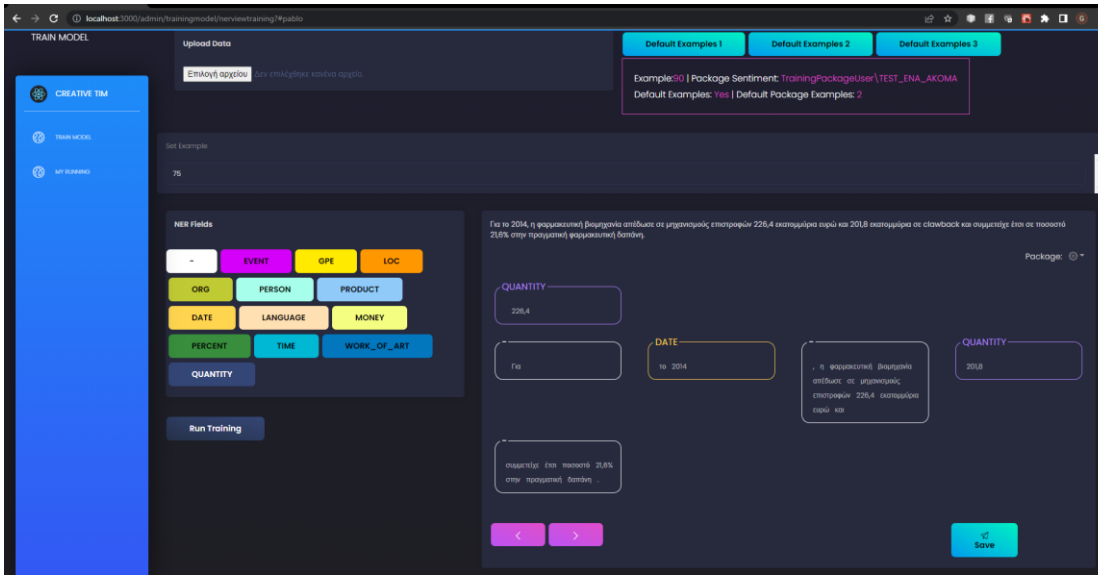
Εικόνα NerViewTraining.js

Τα αποτελέσματα του παραπάνω κώδικα εμφανίζονται στην συνέχεια.



Εικόνα «Αποτελέσματα NerViewTraining.js»





Εικόνα «Αποτελέσματα NerViewTraining.js (εμπλουτισμός με νέο ner-token)»

Αμέσως μετά έχουμε το αρχείο «EmotiontrainingView.js», όπου έχει να κάνει την λειτουργεία του εμπλουτισμού ενός συναισθηματικού λεξικού. Παρακάτω παραθέτουμε τον κώδικας:

```
import React, { Component } from "react";

import {
  Button,
  Card,
  CardHeader,
  CardBody,
  CardTitle,
  Table,
  CardFooter,
  Row,
  Col,
  Input,
  FormGroup
} from "reactstrap";

import { Link,
} from "react-router-dom";

import
{
  Bar
} from "react-chartjs-2";

import axios from "axios";

import "assets/css/mystyle.css";
import SettingSentimentLexicons from "components/SettingsPackageSentiment/SettingSentimentLexicon";
import SentimentDataFields from "SentimentDataFields";
import GeneralSettings from "components/GeneralOptions/GeneralSettings";
```

```
class EmotiontrainingView extends Component {
    constructor(props){ //καταστευαστής όπου όριζονται και τα states
        super(props);
        this.state = {
            text: null, //το κειμενο
            text_num: 1, //ο αριθμός του παραδειγματος
            subjectivity: null, //η αντικειμενικότητα της ανάλυσης
            emotion_score: null, //το σκορ του συναισθήματος
            result: null, //τα αποτελέσματα
            emotions: null, //τα συνασιθήματα
            list_matched_token: null, //η λίστα τα tokens που εντοπίστηκαν στο συνασιθηματικό λεξικό
            matched_token: null, //τα tokes που συμμετέχουν στο συναίσθημα
            matched_tokens_pos: null, //τα pos tag των tokens που συμμετέχουν
            matched_tokens_pos_col: null,//η θέση των pos tag των tokens που συμμετέχουν
            main_emotion: null, //το κύριο συναίσθημα
            file_name_sentiment: null, //το όνομα του συναισθηματικου λεξικου που δουλεύουμε
            loaddefaultexamples:false, //αν τα παραδείγματα είναι τα default
            packagedefaultexample: null //το default πακέτο παραδειμγάτων που έχουμε επιλέξει
        }
```

```
        this.chartExample3 = { //η δομή του γραφήματος
            data: (canvas) => {
              let ctx = canvas.getContext("2d");

              let gradientStroke = ctx.createLinearGradient(0, 230, 0, 60);

              gradientStroke.addColorStop(1, "rgba(72,72,176,0.1)");
              gradientStroke.addColorStop(0.4, "rgba(72,72,176,0.0)");
              gradientStroke.addColorStop(0, "rgba(119,52,169,0)"); //purple colors

              return {
                labels: ["Anger", "Disgust", "Fear", "Happiness", "Sadness", "Surprise"],
                datasets: [
                  {
                    label: "Emotion",
                    fill: true,
                    backgroundColor: gradientStroke,
                    hoverBackgroundColor: gradientStroke,
                    borderColor: "#d048b6",
                    borderWidth: 2,
                    borderDash: [],
                    borderDashOffset: 0.0,
                    data: this.state.emotions, //τα δεδομενα για το γράφημα
                  },
                ],
              };
            },
```

```
78          options: {
79            maintainAspectRatio: false,
80            legend: {
81              display: false,
82            },
83            tooltips: {
84              backgroundColor: "#f5f5f5",
85              titleFontColor: "#333",
86              bodyFontColor: "#666",
87              bodySpacing: 4,
88              xPadding: 12,
89              mode: "nearest",
90              intersect: 0,
91              position: "nearest",
92            },
93            responsive: true,
94            scales: {
95              yAxes: [
96                {
97                  gridLines: {
98                    drawBorder: false,
99                    color: "rgba(225,78,202,0.1)",
100                   zeroLineColor: "transparent",
101                 },
102                 ticks: {
103                   suggestedMin: 10,
104                   suggestedMax: 100,
105                   padding: 10,
106                   fontColor: "#9e9e9e",
107                 },
108               },
109             ],
110             xAxes: [
111               {
112                 gridLines: {
113                   drawBorder: false,
114                   color: "rgba(225,78,202,0.1)",
115                   zeroLineColor: "transparent",
116                 },
117                 ticks: {
118                   padding: 20,
119                   fontColor: "#9e9e9e",
120                 },
121               },
122             ],
123           },
```

```
            },
          };
    } //end constructor

    componentDidMount = () =>{ //καλείται μετά το πρώτο render
        this.getGetEmotionAnalysis(); //κλήση συνάρτησης
    }
```

```
getGetEmotionAnalysis = () => { //συνάρτηση κλησης προς το back-end, για να πάρουμε τα
                                //αποτελέσματα της συναισθηματικής ανάλυσης
    let formdata = {count_text: this.state.text_num, //το αριθμό του παραδείματος
        file_name: this.state.file_name_sentiment, //το όνομα του συναισθηματικου λεξικου
        default_example: this.state.loaddefaultexamples, //αν πρόκειται από τα default παραδειγματα
        package_example: this.state.packagedefaultexample} //ποιο default πακέτο παραδειγμάτων

    axios.get("http://127.0.0.1:5000/nertraining/emotionalysistraining", {params: formdata},
        {
            headers: {
                'Content-Type': 'application/json',
                "Access-Control-Allow-Origin": "*"
            }
        }
        ).then(response => { //μόλις λάβουμε απάντηση ενημερώνουμε το state

        // console.log(response.data)
        this.setState({
            subjectivity: response.data.Resp.subjectivity,
            emotion_score: response.data.Resp.emotion_score,
            result: response.data.Resp.result,
            emotions: this.getEmotions(response.data.Resp.emotions),
            matched_token: response.data.Resp.matched_tokens_num,
            list_matched_token: response.data.Resp.matched_tokens_list,
            matched_tokens_pos: response.data.Resp.matched_tokens_pos,
            matched_tokens_pos_col: response.data.Resp.matched_tokens_pos_col,
            text: response.data.Resp.text,
            main_emotion: response.data.Resp.main_score
        })
    })
    .catch(error => {
        console.error(error)
    });
}
```

```
inMatchedListPos = (cell) => { //συνάρτηση για το αν το token συμμετέχει στο συναισθημα
    if(this.state.matched_tokens_pos != null)
        return this.state.matched_tokens_pos.some(x => x === cell);
    return false;
}

findPosMatched = (cell, key, count_pos_matched) => { //εντοπισμός της θέσεις του pos ενός token
                                                     //που συμμετέχει στο συναισθημα
    const findIndexElement = (element) => element === cell[0];
    const pos = this.state.matched_tokens_pos_col[this.state.matched_tokens_pos.findIndex(findIndexElement)];
    if( key === 0 || key === Number(pos) || key === count_pos_matched + 5)
    {
        return true;
    }
    return false;
}
```

```
    getMatchedList = (list) => { //συνάρτηση που επιστρέφει όλες τις εγγραφές,
                                 //σχεδιάζοντας τις γραμμές ενος πίνακα
        //console.log(list);
        // console.log(list[0][0]);
        var count_pos_matched = 0;
        return list.map((ele, key) => (
            this.inMatchedListPos(ele[0]) ?
            <tr key= {ele[0] + key} id={count_pos_matched = key}>
                { ele.map((cell, keycell) => (
                    this.findPosMatched(ele, keycell, count_pos_matched) ?
                        <th className="color_matched_token_pos" key={ele[0]+key+keycell} id={count_pos_matched = keycell-1}
                            { cell }
                        </th>
                    :
                    <th key={ele[0]+key+keycell} >
                        { cell }
                    </th>
                ))
                }
            </tr>
            : <tr key= {key} /*contentEditable={ 'true' }*/>
                { ele.map((cell, keycell) => (
                    <th key={ele[0]+key+keycell}>{cell}</th>
                ))}
            </tr>
        ));
    }

    getEmotions = (emotions) => { //επιστρέφει τις τιμές τον συναισθημάτων *%
        return emotions != null ? [
            emotions['anger'] * 100,
            emotions['disgust'] * 100,
            emotions['fear'] * 100,
            emotions['happiness'] * 100,
            emotions['sadness'] * 100,
            emotions['surprise'] * 100,
        ]
        : []
    }
```

```
    saveButton = Event => { //συνάρτηση για την αποθήκευση των νέων token στο συναισθηματικό λεξικο
        let name_package = prompt('Name Package');
        // console.log(name_package);
        let elem = document.getElementsByTagName('table')[0].getElementsByTagName("tbody")[0].getElementsByTagName("tr");
        let datalist = [];
        for (var el = 0;  el < elem.length; el++)
        {
            datalist[el] = elem[el].innerText; //για κάθε κελι του πίνακα, τοποθέτησε τα σε μια λίστα
        }
        let dataform = {filename: name_package, data: datalist} //τα δεδομένα που θα στείλουμε
        axios.post("http://127.0.0.1:5000/nertraining/emotionalysistrainingsavedata", dataform).then(response => { //μόλις λάβουμε απάντηση
        alert("The change succes saved...");
        this.setState({
            file_name_sentiment: name_package, //επέλεξε το νεό λεξικο
        }, () => { //μόλις αλλάξει το state
            this.getGetEmotionAnalysis(); //υπολόγισε ξανά το συναίσθημα
        })
        })
        .catch(error => {
        console.error(error)
        });
    }
```

```
246
247         nextButton = () => { //συνάρτηση, μετάβασης στο επόμενο παράδειγμα
248             this.setState( prevState => {
249                 return {text_num: prevState.text_num + 1}
250             }, () => {
251                 this.getGetEmotionAnalysis()
252             })
253         }
254
255         backButton = () => { //συνάρτηση, μετάβαση στο προηγούμενο παράδειγμα
256             if(this.state.text_num === 0) return;
257             this.setState( prevState => {
258                 return {text_num: prevState.text_num - 1}
259             }, () => {
260                 this.getGetEmotionAnalysis()
261             })
262         }
263
264         changeCounter = (e) =>{ //συνάρτη, όπου ο χρήστης ορίζει τον αριθμό του παράδειγμα
265             var elemval = parseInt(document.getElementById("change-counter-user-not-selected").value);
266             console.log(elemval);
267             this.setState({
268                 text_num: elemval
269             }, () => {
270                 this.getGetEmotionAnalysis()
271             })
272         }
```

```
274         handleFileSelect = (e) => { //συνάρτηση όπου χρήστη ανεβάζει το αρχείο με δικά του παραδείγματα
275             let files = e.target.files;
276             let reader = new FileReader();
277             reader.readAsDataURL(files[0]);
278             reader.onload = (e) => {
279             // console.log(e.target.result);
280             let formData = new FormData();
281             formData.append("files", e.target.result); //αποστολή του αρχείου
282             axios.post("http://127.0.0.1:5000/exespacy/uploadfile", formData).then(response => {
283                 // console.log(response)
284                 this.setState({
285                     text_num: 1
286                 }, () => {
287                     this.getGetEmotionAnalysis();
288                 })
289             })
290             .catch(error => {
291                 console.error(error)
292             });
293             }
294         }
295
296         buttonchangefileSentiment = input_file => { // συνάρτηση, κάθε φορά που αλλάζει λεξικό ο χρήστης
297             this.setState({
298                 file_name_sentiment: input_file //η επιλογή του νέου λεξικού
299             }, () => {
300                 this.getGetEmotionAnalysis(); //εκ νέου κλήση στο back-end για την παραλαβεί νέων αποτελεσματων
301             })
302         }
```

```
setSentimentDataFields = () => { //συνάρτηση για κάθε πεδίο του συναισθήματικου λεξικού
                                 //επέστρεψε τα παρακατω πεδία POS1...POS4
    return SentimentDataFields.map((sentiment, key) => {
        return (
            <>
                <FormGroup key = { sentiment.name }>
                    <label style={{color: sentiment.color}}>{ sentiment.name }</label>
                    <Row key = { sentiment.name+key }>
                        <Col md="2">
                            <FormGroup>
                                <label>POS 1</label>
                                <Input id={'sentiment1'+sentiment.name} type="text" placeholder="N/A"/>
                            </FormGroup>
                        </Col>
                        <Col md="2">
                            <FormGroup>
                                <label>POS 2</label>
                                <Input id={'sentiment2'+sentiment.name} type="text" placeholder="N/A"/>
                            </FormGroup>
                        </Col>
                        <Col md="2">
                            <FormGroup>
                                <label>POS 3</label>
                                <Input id={'sentiment3'+sentiment.name} type="text" placeholder="N/A"/>
                            </FormGroup>
                        </Col>
                        <Col md="2">
                            <FormGroup>
                                <label>POS 4</label>
                                <Input id={'sentiment4'+sentiment.name} type="text" placeholder="N/A"/>
                            </FormGroup>
                        </Col>
                    </Row>
                </FormGroup>
            </>
        )
    })
}
```

```
LoadDefaultExamples = (value) => { //αν επιλεγή default πακέτο παραδειγματων
    this.setState({ //ενήμερωση του state
        loaddefaultexamples: true,
        packagedefaultexample: value,
        text_num: 1
    },  () => {
        this.getGetEmotionAnalysis()
    });
}

ButtonAddWordSentiment = () => { //συνάρτηση, προσθήκης νέου token
    var inputword = document.getElementById('input_word').value;
    document.getElementById('input_word').value = "";
    let datasentiment = [];
    datasentiment.push(inputword);
    SentimentDataFields.forEach(sentiment => {
        for(var j=0; j<4; j++)
        {
            var val = document.getElementById('sentiment'+(j+1)+sentiment.name).value;
            datasentiment.push( val == null || val === '' ? 'N/A' : isNaN(Number(val))? val : val+'.0' );
            document.getElementById('sentiment'+(j+1)+sentiment.name).value = "";
        }
    });
    let temp_list_matched_token = this.state.list_matched_token;
    if (temp_list_matched_token == null)
    {
        //console.log(temp_list_matched_token);
        temp_list_matched_token = [];
    }
    temp_list_matched_token.push(datasentiment);
    this.setState({ //ενημέρωση της λίστας με matched tokens
        list_matched_token: temp_list_matched_token
    });
}
```

```
ButtonCalculate = Event => { //συνάρτηση, εκ νέου υπολογισμού κάνοντας χρήση του πίνακα της εφαρμογής
    let elem = document.getElementsByTagName('table')[0].getElementsByTagName("tbody")[0].getElementsByTagName("tr")
    let datalist = [];
    for (var el = 0;  el < elem.length; el++)
    {
        datalist[el] = elem[el].innerText;
    }
    //console.log(datalist);
    let formdata = {text: this.state.text, package: this.state.file_name_sentiment, matched_token: datalist}
    axios.post("http://127.0.0.1:5000/nertraining/emotionalysiscalculated", formdata,
        {
            headers: {
                'Content-Type': 'application/json',
                "Access-Control-Allow-Origin": "*"
            }
        }
        ).then(response => {

        // console.log(response.data)
        this.setState({
            subjectivity: response.data.Resp.subjectivity,
            emotion_score: response.data.Resp.emotion_score,
            result: response.data.Resp.result,
            emotions: this.getEmotions(response.data.Resp.emotions),
            matched_token: response.data.Resp.matched_tokens_num,
            list_matched_token: response.data.Resp.matched_tokens_list,
            matched_tokens_pos: response.data.Resp.matched_tokens_pos,
            matched_tokens_pos_col: response.data.Resp.matched_tokens_pos_col,
            text: response.data.Resp.text,
            main_emotion: response.data.Resp.main_score
        })
    })
    .catch(error => {
        console.error(error)
    });
}
```

```
render(){
  return (
      <>
          <Row>
              <Col>
              <Link to="/admin/trainingmodel"> Train Model </Link>
              <b> { ">" } </b>
              <Link to="/admin/trainingmodel/viewemotiontraining">Emotion Training</Link>
              </Col>
          </Row>
          <GeneralSettings handleFileSelect={(e) => this.handleFileSelect(e)}
          LoadDefaultExamples={(e) => this.LoadDefaultExamples(e)}
          text_num={ this.state.text_num } package_name={  this.state.file_name_sentiment }
          loaddefaultexamples={ this.state.loaddefaultexamples }
          packagedefaultexample={ this.state.packagedefaultexample }
          changeCounter = {(e) => this.changeCounter(e) }>
          </GeneralSettings>
          <Row>
          <Card className="card-tasks my-card-tasks">
              <CardHeader>
                  <h5 className="title d-inline">Example: { this.state.text_num }</h5>
                  <SettingSentimentLexicons onClick={ this.buttonchangefileSentiment }/>
              <CardTitle tag="h3">
                  { this.state.text }
              </CardTitle>
              </CardHeader>
```

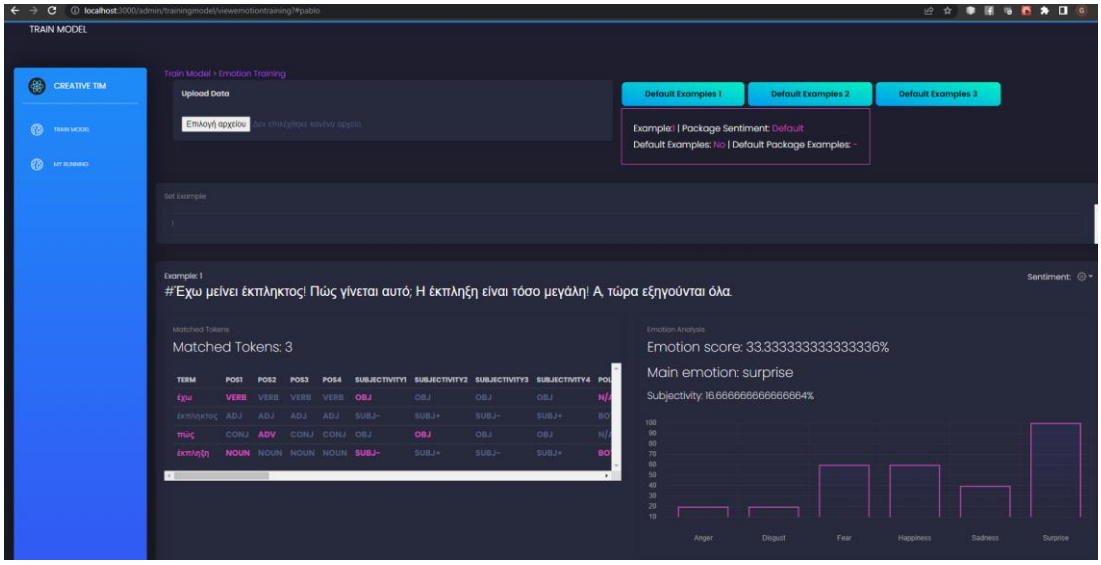
```
441                 <CardBody>
442                     {this.state.emotion_score != null || this.state.list_matched_token != null ?
443                     <Row>
444                         <Col md="6">
445                             <Card>
446                                 <CardHeader>
447                                     <h5 className="card-category">Matched Tokens</h5>
448                                 <CardTitle tag="h3">
449                                     { this.state.matched_token }
450                                 </CardTitle>
451                                 </CardHeader>
452                                 <CardBody style={{ 'maxHeight': "600px", 'overflowY': 'scroll' }}>
453                                 <Table className="tablesorter" > {/**κεφαλίδες πίνακα Term, POS1...POS4, Subjectivity1...Subjectivity4 κτλ*/
506                                 </Table>
507                                 </CardBody>
508                             </Card>
509                         </Col>
510                         <Col md="6">
511                             <Card className="card-chart">
512                                 <CardHeader>
513                                 <h5 className="card-category">Emotion Analysis</h5>
514                                 <CardTitle tag="h3">
515                                     { this.state.emotion_score }
516                                 </CardTitle>
517                                 <CardTitle tag="h3">
518                                     { this.state.main_emotion }
519                                 </CardTitle>
520                                 <CardTitle tag="h4">
521                                     { this.state.subjectivity }
522                                 </CardTitle>
523                                 </CardHeader>
524                                 <CardBody>
525                                 <div className="chart-area">
526                                     <Bar
527                                     data={this.chartExample3.data}
528                                     options={ this.chartExample3.options}
529                                     />
530                                 </div>
531                                 </CardBody>
```

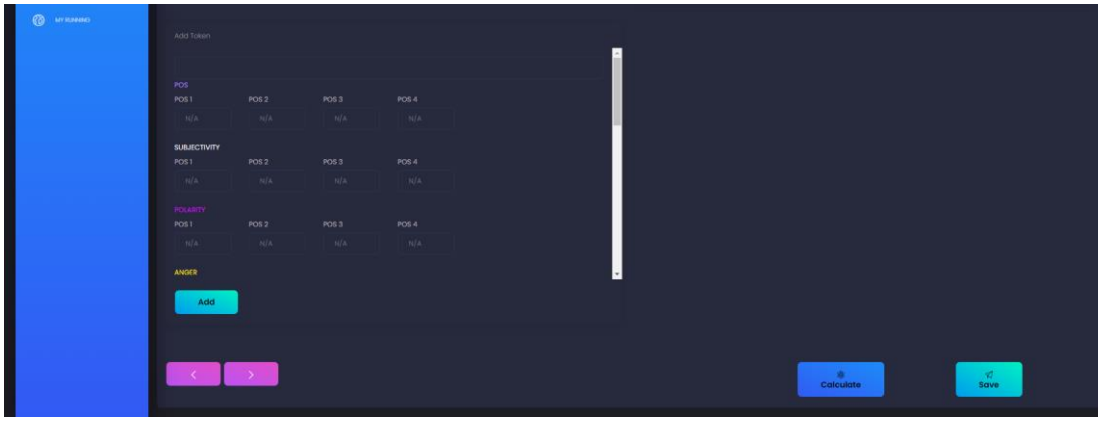
```
532                             </Card>
533                         </Col>
534                     </Row>
535                     : <Card className="card-chart">
536                             <CardHeader>
537                             <h5 className="card-category">Emotion Analysis</h5>
538                             </CardHeader>
539                             <CardBody>
540                                 { this.state.result }
541                             </CardBody>
542                         </Card>
543                     }
544                     <Row>
545                         <Col md="6">
546                             <Card>
547                                 <CardHeader>
548                                     <h5 className="card-category">Add Token</h5>
549                                 <CardTitle tag="h3">
550                                 </CardTitle>
551                                 </CardHeader>
552                                 <CardBody key="table_for_add_word_sentiment" style={{ 'maxHeight': "400px", 'overflowY': 'scroll' }}>
553                                     <Input id="input_word"></Input>
554                                     { this.setSentimentDataFields() }
555                                 </CardBody>
556                                 <CardFooter>
557                                     <Row>
558                                         <Col md="2" key="button_for_add">
559                                             <Button key="buttonAdd" className="btn-fill" color="success" style={{ color:"black" }} type="submit
560                                             onClick={(e) => this.ButtonAddWordSentiment()}>
561                                                 Add
562                                             </Button>
563                                         </Col>
564                                     </Row>
565                                 </CardFooter>
566                             </Card>
567                         </Col>
568                     </Row>
569                 </CardBody>
```

```
570                 <CardFooter>
571                 <Row>
572                     <Col key="butons_for_excamples">
573                     <Button id="backButton" className="btn-fill" color="primary" type="submit"
574                             onClick={ this.backButton }>
575                                 <i className="tim-icons icon-minimal-left"></i>
576                     </Button>
577                     <Button id="nextButton" className="btn-fill" color="primary" type="submit"
578                             onClick={ this.nextButton }>
579                                 <i className="tim-icons icon-minimal-right" />
580                     </Button>
581                     </Col>
582                     <Col md="2" key="buttons_calculate">
583                     <Button id="saveButton" className="btn-fill" color="info" style={{ color:"black" }} type="submit"
584                             onClick={(e) => this.ButtonCalculate(e) }>
585                                 <i className="tim-icons icon-atom"/> <br/>
586                                 Calculate
587                     </Button>
588                     </Col>
589                     <Col md="2" key="buton_for_save_changes">
590                     <Button id="saveButton" className="btn-fill" color="success" style={{ color:"black" }} type="submit"
591                             onClick={ this.saveButton }>
592                                 <i className="tim-icons icon-send"/> <br/>
593                                 Save
594                     </Button>
595                     </Col>
596                 </Row>
597                 </CardFooter>
598             </Card>
599             </Row>
600         </>
601     );
602   }
603 }
604
605 export default EmotiontrainingView;
```

Εικόνα «EmotiontrainingView.js»

Το αποτέλεσμα του παραπάνω κώδικα, φαίνεται στις παρακάτω εικόνες:





Εικόνα «Αποτελέσματα EmotiontrainingView.js»

Τέλος, προσθέσαμε μια επιπλέων λειτουργία, πιο γενική για την απλή εικόνα της ανάλυσης κείμενου. Αυτή δυνατότητα προστέθηκε, ώστε να δοκιμάσουμε και να επαληθεύσουμε την σωστή λειτουργία του εκπαιδευμένου πακέτου spacy. Παρακάτω παραθέτουμε τον κώδικα και τα αποτελέσματα:

```
import React from "react";

import {
  Form,
  Row,
  Col,
  Card,
  CardTitle,
  CardHeader,
  CardBody,
  Input,
  Table,
  Label,
} from "reactstrap";

import axios from "axios";
import "assets/css/mystyle.css";
import EmotionsChart from "components/Emotions/EmotionsChart";
import SettingSpacyPackage from "components/SettingsPackageSentiment/SettingSpacyPackage";
import SettingSentimentLexicons from "components/SettingsPackageSentiment/SettingSentimentLexicon";
import PopupWindow from "components/GeneralOptions/PopupWindow";
```

```
function SpacyProfile(props) {
  const [tokens, setTokens] = React.useState(null); //μια λιστα με ολα τα tokens
  const [alltext, setAllText] = React.useState(null); //όλο το κείμενο
  const [depdata, setDepData] = React.useState(null); //συναντικτηκες εξαρτήσεις
  const [package_name, setPackageName] = React.useState(null); //το πακέτο spacy που επιλέγουμε για την ανάλυση κειμένου
  const [sentiment_file, setSentimentFile] = React.useState(null); //το συναισθηματικό λεξικο που θα χρεισιμοποηθεί
  const [openGraphs, setOpenPopupGraphs] = React.useState(null); //άνοιγμα του γραφήματος, για τις συντακτικές εξαρτήσεις
```

```
30
31  const clearData = () => { //συνάρτηση καθαρισμού δεδομένων
32    setTokens(null);
33    setAllText(null);
34    setDepData(null);
35  }
36
37  const handleFileSelect = (e) => { //συνάρτη όπου ο χρήστη ανεβάσει το αρχείο με τα δεδομένα του
38    console.log(e.target.files);
39    let files = e.target.files;
40    let reader = new FileReader();
41    reader.readAsDataURL(files[0]);
42    reader.onload = (e) => {
43      console.log(e.target.result);
44      let formData = new FormData();
45      formData.append("files", e.target.result);
46      axios.post("http://127.0.0.1:5000/exespacy/uploadfile", formData).then(response => {
47        console.log(response)
48        getTokens();
49        getGetDep();
50      })
51      .catch(error => {
52        console.error(error)
53      });
54    }
55  }
```

```
56
57  const getTokens = (package_name, sentiment_file) => { //συνάρτη όπου καλεί το back-end, στέλνουμε το όνομα
58                                                         //του πακέτου spacy και το όνομα του συναισθήματικο λεξικο
59                                                         //και μας απαντάει με τα αποτελέσμα της ανάλυσεις του κειμένου
60    let formdata = {package: package_name, sentiment_file: sentiment_file}
61    axios.get("http://127.0.0.1:5000/exespacy/gettokens", {params: formdata}).then(response => {
62      let resp = JSON.parse(response.data.Resp);
63      console.log(resp)
64      setAllText(resp.alltext); //παίρνουμε το κείμενο
65      setTokens(resp.nlplistdoc); //παίρνουμε την λίστα με τα doc
66    })
67    .catch(error => {
68      console.error(error)
69    });
70  }
71
72  const getGetDep = (package_name, sentiment_file) => { //συνάρτη όπου καλεί το back-end, στέλνουμε το όνομα
73                                                        //του πακέτου spacy και το όνομα του συναισθήματικο λεξικο
74                                                        //και μας απαντάει με τα αποτελέσμα των συντακτικών εξαρτήσεων
75    let formdata = {package: package_name, sentiment_file: sentiment_file}
76    axios.get("http://127.0.0.1:5000/exespacy/getdep", {params: formdata}).then(response => {
77      let resp = JSON.parse(response.data.Resp);
78      console.log(resp)
79      setDepData(resp)
80    })
81    .catch(error => {
82      console.error(error)
83    });
84  }
```

```
85
86  const buttonchangefilepackage = input_file => { //συνάρτη όταν αλλάζει το όνομα του πακέτου spacy
87                                                  //και παίρνουμε εκ νεου τα δεδομένα
88    setPackageName(input_file);
89    clearData();
90    getTokens(input_file, sentiment_file);
91    getGetDep(input_file, sentiment_file);
92  }
93
94  const buttonchangefilesenitment = input_file => { //συνάρτη όταν αλλάζει το όνομα του συναισθηματικου λεξικου
95                                                    //και παίρνουμε εκ νεου τα δεδομένα
96    setSentimentFile(input_file);
97    clearData();
98    getTokens(package_name, input_file);
99    getGetDep(package_name, input_file);
100 }
101
102 const buttonOpenGraphs = (keyindex) => { //το button για το άνοιγμα του γράφήματος συντακτικών εξαρτήσεων,
103                                          //δέχετε στην είσοδο το counter του παραδείγματος
104     setOpenPopupGraphs(keyindex);
105 }
106
107 const depdatagraphs = (keyparagraph) => { //συνάρτηση για κάθε κείμενο που αναλύθηκε,
108                                           //δημιουργείται ένα component popupWindow,
109                                           //για το άνοιγμα του γραφήματος
110   return depdata.map((data, key) => {
111     if(key === keyparagraph)
112     {
113       return(
114           <PopupWindow indexKey = { key } openPopup={ openGraphs === key }
115           onClick={buttonOpenGraphs} handleClose={buttonOpenGraphs}>
116             <div className="showscrolling" dangerouslySetInnerHTML={{ __html: data.graphs }} />
117           </PopupWindow>
118       )
119     }
120     else{
121       return null;
122     }
123   })
124 }
```

```
126   const getRowAllText = () => { //δώσαμε την δυνατότητα να μπορεί ένα κείμενο να σπάσει
127                                 //στις διαθέσημες προτάσεις του και να αναλυθεί κάθε μια ξεχωρηστά
128                                 //η συναρτη επιστρέχει όλο το κείμενο.
129     return(
130       alltext !== null ?
131       <Row>
132         <Col>
133           <Card className="card-plain">
134             <CardHeader>
135               <h5 className="title">A text has been detected</h5>
136             </CardHeader>
137             <CardBody>
138               <Row>
139                 <Col>
140                   <Label>{ alltext }</Label>
141                 </Col>
142               </Row>
143             </CardBody>
144           </Card>
145         </Col>
146       </Row>
147       : null
148     );
149   }
150
151   const getlemmapostagsdep = (obj, tokenkey) => {//συνάρτηση όπου επιστρέφει το δεδομένο
152                                                  //στην σωστή θεση του πίνακα
153     return obj.map((ob, key) => {
154       if(key === tokenkey)
155       {
156         return(
157           <td key={ key+tokenkey }>{ ob }</td>
158         )
159       }
160       else{
161         return null;
162       }
163     })
164   }
```

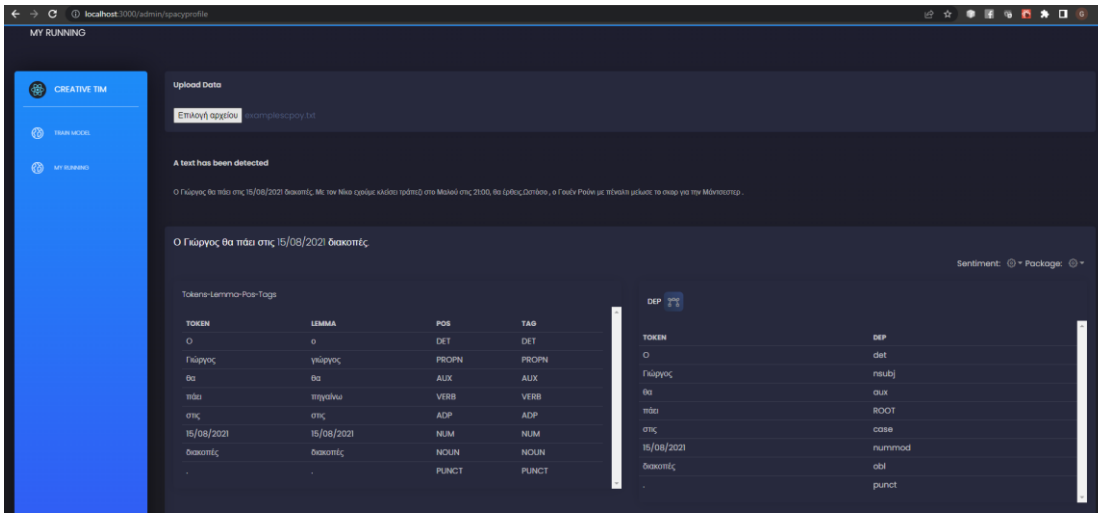
```
165
166   const getDataToken = (tokens, lemma, pos, tags, dep) => { //για κάθε tokens, επιστρέφουμε το αποτέλεσμα
167                                                             //token, lemma, pos, tags, dep
168                                                             //Έχω έχω VERB  VERB  ROOT
169     return tokens.map((token, key) => {
170       return (
171         tokens != null ?
172         <tr key = { key }>
173           <td>{ token }</td>
174           { lemma != null ? getlemmapostagsdep(lemma, key) : null }
175           { pos != null ? getlemmapostagsdep(pos, key) : null }
176           { tags != null ? getlemmapostagsdep(tags, key) : null }
177           { dep != null ? getlemmapostagsdep(dep, key) : null }
178         </tr>
179         : null
180       );
181     })
182   }
183
184   const getDepdataTable = (keyparent) => { //συνάρτηση όπου θέλουμε για κάθε token
185                                            //μονο την συντακτική του εξάρτηση
186     return depdata.map((data, key) => {
187       return (
188         key === keyparent ?
189         getDataToken(data.tokens, null, null, null, data.dep)
190         : null
191       );
192     })
193   }
```

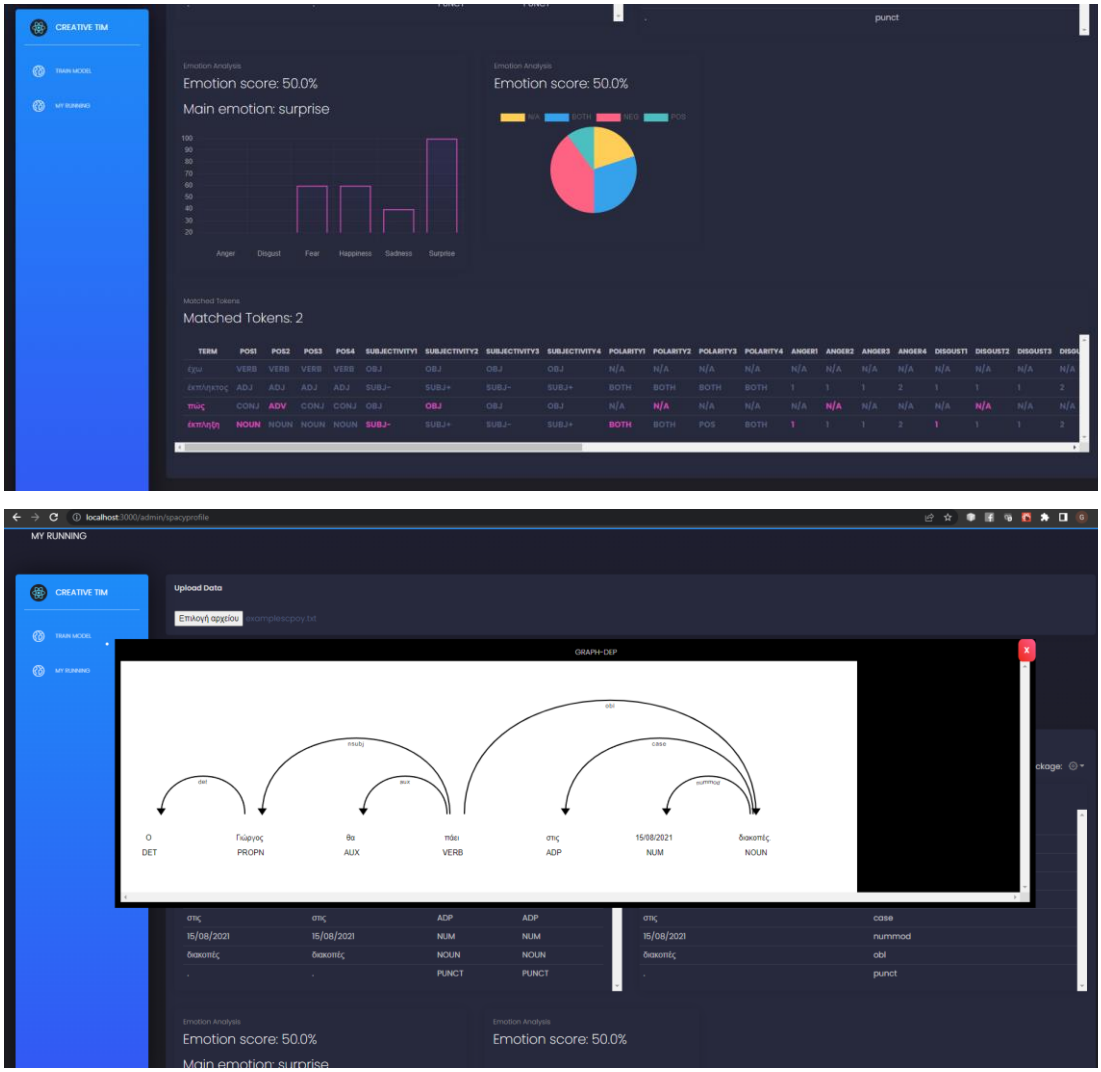
```
194
195   const getDataCardToken = (datatest) => { //συνάρτηση για κάθε δεδομένο που αναλύθηκε
196                                            //φτιάχουμε δύο πίνακες:
197                                            //1) ένα με τα token, lemma, pos, tags
198                                            //2) ενα με τα token, dep
199                                            //τέλος γίνεται χρήση του component EmotionsChart, για την
200                                            //συναισθηματική ανάλυση του κειμένου.
201
202     return datatest.map((prop, key) => {
203       return (
204         <>
205         <Row key = {key}>
206           <Col>
207             <Card className="card-tasks my-card-tasks">
208               <CardHeader>
209                 <CardTitle tag="h4"> { prop.paragraph } </CardTitle>
210                 <SettingSpacyPackage onClick={ buttonchangefilepackage }/>
211                 <SettingSentimentLexicons onClick={ buttonchangefilesenitment }/>
212               </CardHeader>
213               <CardBody>
214                 <Row md="2">
215                   <Col md="6">
216                     <Card>
217                       <CardHeader>
218                         <CardTitle tag="h5">Tokens-Lemma-Pos-Tags</CardTitle>
219                       </CardHeader>
220                       <CardBody style={{ 'maxHeight': "500px", 'overflowY': 'scroll' }}>
221                         <Table className="tablesorter" response>
222                         <thead className="text-primary">
223                           <tr>
224                             <th>TOKEN</th>
225                             <th>LEMMA</th>
226                             <th>POS</th>
227                             <th>TAG</th>
228                           </tr>
229                         </thead>
230                         <tbody>
231                             { getDataToken(prop.tokens, prop.lemma, prop.pos, prop.tags, null) }
232                         </tbody>
233                         </Table>
234                       </CardBody>
```

```
                    </Card>
                  </Col>
                  <Col md="6">
                    <Card>
                      <CardHeader>
                        <CardTitle tag="h5" className="title">
                          DEP       { depdata != null ? depdatagraphs(key) : null}
                        </CardTitle>
                      </CardHeader>
                      <CardBody style={{ 'maxHeight': "500px", 'overflowY': 'scroll' }}>
                        <Row>
                          <Table className="tablesorter" response>
                            <thead className="text-primary">
                              <tr>
                                <th>TOKEN</th>
                                <th>Dep</th>
                              </tr>
                            </thead>
                            <tbody>
                              { depdata != null ? getDepdataTable(key) : null }
                            </tbody>
                          </Table>
                        </Row>
                      </CardBody>
                    </Card>
                  </Col>
                </Row>
                <Row>
                  <EmotionsChart text={ prop.paragraph }
                  package_name={ package_name }
                  sentiment_file={ sentiment_file }/>
                </Row>
              </CardBody>
            </Card>
          </Col>
        </Row>
        </>
      )
    });
  };
```

```
276    return ( //
277      <>
278        <div className="content">
279          <Row>
280            <Col>
281              <Card>
282                <CardHeader>
283                  <h5 className="title">Upload Data</h5>
284                </CardHeader>
285                <CardBody>
286                  <Form>
287                    <Row>
288                      <Col>
289                        <Input type="file" onChange={(e) => handleFileSelect(e)}/>
290                      </Col>
291                    </Row>
292                  </Form>
293                </CardBody>
294              </Card>
295            </Col>
296          </Row>
297          { alltext === null ? null : getRowAllText() }
298          { tokens !== null ? getDataCardToken(tokens) : null }
299          <Row>
300          </Row>
301        </div>
302      </>
303    );
304  }
305
306  export default SpacyProfile;
```

Εικόνα SpacyProfile.js





Εικόνα SpacyProfile.js (Αποτελέσματα)