ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

Ανάλυση και οπτικοποίηση ρευμάτων δεδομένων που αφορούν αντιδράσεις πολιτών στα μέσα κοινωνικής δικτύωσης

Πτυχιακή εργασία

Μαρία Βουλιέρη

Αθήνα, 2022

HAROKOPIO UNIVERSITY

SCHOOL OF DIGITAL TECHNOLOGY

DEPARTMENT INFORMATICS & TELEMATICS

Analysis and visualization of data streams related to citizens' reactions on social media

Bachelor Thesis

Maria Voulieri

Athens, 2022

ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

Τριμελής Εξεταστική Επιτροπή

Ηρακλής Βαρλάμης
Αναπληρωτής καθηγητής,
Τμήμα Πληροφορικής και Τηλεματικής, Χαροκόπειο Πανεπιστήμιο

Χρήστος Δίου
Επίκουρος καθηγητής,
Τμήμα Πληροφορικής και Τηλεματικής, Χαροκόπειο Πανεπιστήμιο

Δημήτριος Μιχαήλ
Αναπληρωτής καθηγητής,
Τμήμα Πληροφορικής και Τηλεματικής, Χαροκόπειο Πανεπιστήμιο

Η Μαρία Βουλιέρη

δηλώνω υπεύθυνα ότι:

1) Είμαι ο κάτοχος των πνευματικών δικαιωμάτων της πρωτότυπης αυτής εργασίας και από όσο γνωρίζω η εργασία μου δε συκοφαντεί πρόσωπα, ούτε προσβάλει τα πνευματικά δικαιώματα τρίτων.

2) Αποδέχομαι ότι η ΒΚΠ μπορεί, χωρίς να αλλάξει το περιεχόμενο της εργασίας μου, να τη διαθέσει σε ηλεκτρονική μορφή μέσα από τη ψηφιακή Βιβλιοθήκη της, να την αντιγράψει σε οποιοδήποτε μέσο ή/και σε οποιοδήποτε μορφότυπο καθώς και να κρατά περισσότερα από ένα αντίγραφα για λόγους συντήρησης και ασφάλειας.

''Machine intelligence is the last invention that humanity will ever need to make.''

- **Nick Bostrom**

(Swedish philosopher)

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα αρχικά να ευχαριστήσω τον κύριο Βαρλάμη για όλη την πολύτιμη βοήθεια και άμεση ανταπόκριση του καθ΄ όλη τη διάρκεια της εκπόνησης της πτυχιακής μου.

Θα ήθελα επίσης να ευχαριστήσω όλους τους καθηγητές του τμήματος Πληροφορικής και Τηλεματικής για τη θερμή υποδοχή κατά την έναρξη της φοίτησης μου και τη συνεχή υποστήριξη τους καθ’ όλη τη διάρκεια της. Σας ευχαριστώ πολύ για όλες τις φορές που με βοηθήσατε και με συμβουλεύσατε.

Θα ήθελα επίσης να ευχαριστήσω την οικογένεια μου και τους φίλους μου που με στήριξαν και πίστεψαν σε μένα από την πρώτη στιγμή.

Πίνακας Περιεχομένων

Περίληψη

Η εργασία αυτή πραγματεύεται την ανάπτυξη μιας διαδικτυακής εφαρμογής, που σαν σκοπό έχει να προσφέρει σε πραγματικό χρόνο τη δυνατότητα παρατήρησης ρευμάτων δεδομένων από το Twitter και επιδιώκει με εργαλεία μηχανικής μάθησης, να τα κατανοήσει και να βγάλει στατιστικά συμπεράσματα μέσα από αυτά.

Για να επιτευχθεί αυτό, υλοποιούνται με τη βοήθεια του Django framework το front end, το back end, η σύνδεση με το API του Twitter και η διαλογή και προ-επεξεργασία των εισερχόμενων δεδομένων. Παράλληλα, ερευνείται και εντοπίζεται ο βέλτιστος τρόπος κατηγοριοποίησης των δεδομένων σε τρεις κλάσεις: θετικό, ουδέτερο και αρνητικό.

Οι τελικές επιδόσεις των μοντέλων κατηγοριοποίησης ξεπερνούν σε μέση επιτυχία το 70%.

Λέξεις κλειδιά: Μηχανική Μάθηση, Κατηγοριοποίηση, Ρεύματα Δεδομένων, Διαδίκτυο

Abstract

This essay deals with the development of a web application, which aims to offer in real time the ability to observe data streams from Twitter and tries to draw statistical conclusions through them using machine learning tools.

To achieve this, the implementation of the front end, the back end, the connection to the Twitter API and the sorting and pre-processing of incoming data is done with the help of the Django framework. At the same time, it is being investigated and found the best way to categorize the data into three classes: positive, neutral and negative.

The final performance of the categorization models exceeds 70% in average success.

Keywords: Machine Learning, Classification, Data Streams, Web

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

NLP	Natural Language Processing
API	Application Programming Interface
HTTP	Hyper Text Transfer Protocol
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheets
AJAX	Asynchronous JavaScript And XML

ΕΙΣΑΓΩΓΗ

Στη σημερινή εποχή το διαδίκτυο θεωρείται από πολλούς το πιο δημοκρατικό μέσο. Δεν έχει κόστος, όλοι ανεξαιρέτως μπορούν να έχουν πρόσβαση σε αυτό και ο καθένας μπορεί να εκφράσει τη γνώμη του στις διάφορες πλατφόρμες που υπάρχουν. Σαν αποτέλεσμα, προκύπτει μια ‘’θάλασσα’’ γνώσης και πληροφορίας, που δεν είναι πάντα εύκολο να ερευνηθεί ή να βγουν συμπεράσματα μέσα από αυτή. Υπάρχει έντονο επιστημονικό ενδιαφέρον ως προς την έρευνα και ανάλυση μέσω των αναρτήσεων των πολιτών στα μέσα κοινωνικής δικτύωσης. Επί παραδείγματι, η επεξεργασία κριτικών απόψεων των πελατών ως προς τα προϊόντα ενός καταστήματος, δημιουργεί στους καταστηματάρχες πλήρη εικόνα για την απόδοση της επιχείρησης τους. Διαδικασίες σαν και αυτή αξιοποιούν εργαλεία της μηχανικής μάθησης.

Η μηχανική μάθηση είναι υποπεδίο της τεχνητής νοημοσύνης, που εστιάζει στην ανάλυση και ερμηνεία μοτίβων και δομών σε σύνολα δεδομένων με σκοπό τη μάθηση, εκλογίκευση και λήψη αποφάσεων, χωρίς τη βοήθεια ανθρώπου. Με άλλα λόγια η μηχανική μάθηση επιτρέπει στους χρήστες να παρέχουν δεδομένα σε αλγορίθμους, οι οποίοι θα τα επεξεργαστούν και θα προτείνουν αποφάσεις βασισμένοι σε αυτά. Εφαρμογές της τεχνητής νοημοσύνης, όπως η επεξεργασία φυσικής γλώσσας (NLP) και το computer vision, έχουν ενισχύσει κλάδους όπως η μηχανική, οι πωλήσεις, η ιατρική, ο τουρισμός κ.α..

Η εργασία αυτή αποτελεί μια εφαρμογή επεξεργασίας φυσικής γλώσσας και βασίζεται πάνω στη μελέτη και ανάπτυξη μιας διαδικτυακής εφαρμογής προορισμένη να κατηγοριοποιεί τα συναισθήματα των πολιτών, με χρήση εργαλείων μηχανικής μάθησης. Συγκεκριμένα, θα εντοπίζει σε πραγματικό χρόνο επιλεγμένες αναρτήσεις, μέσω λέξεων-κλειδιών και θα αναλύει τα συναισθήματα των χρηστών που τις δημοσίευσαν. Μια τέτοια εφαρμογή θα μπορέσει να

χρησιμοποιηθεί από οποιονδήποτε φορέα ή μεμονωμένο άτομο, των οποίων οι δραστηριότητες απασχολούν τη γνώμη του κοινού.

Δομή της Εργασίας

Στο κεφάλαιο 1 γίνεται μια συνοπτική περιγραφή της λειτουργικότητας της εφαρμογής, έτσι ώστε ο αναγνώστης να οικειοποιηθεί με την ιδέα της εργασίας.

Στο κεφάλαιο 2 αναφέρονται οι τεχνολογίες και τα εργαλεία ανάπτυξης της εφαρμογής.

Στο κεφάλαιο 3 παρουσιάζεται η δομή των αρχείων της εφαρμογής και μια συνοπτική περιγραφή του κάθε αρχείου.

Στο κεφάλαιο 4 αναλύονται σε τέσσερις ενότητες οι κύριες υλοποιήσεις της εργασίας.

Στο κεφάλαιο 5 παρατίθενται παραδείγματα χρήσης της εφαρμογής με screenshots από την εκτέλεση, περιγραφές και πίνακες αποτελεσμάτων.

ΚΕΦΑΛΑΙΟ 1: Περιγραφή της Λειτουργικότητας της Εφαρμογής

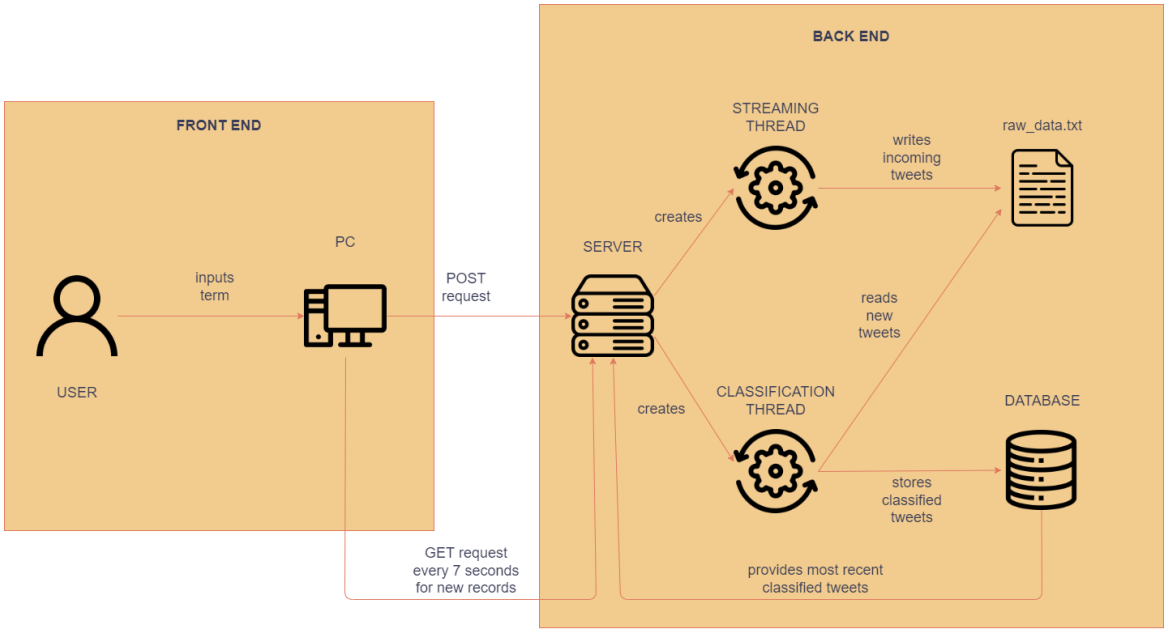
Η εφαρμογή βασίζεται σε 3 συστατικά: νήματα για streaming και κατηγοριοποίηση, ένα αρχείο ‘’raw_data.txt’’, στο οποίο το νήμα του streaming γράφει και το νήμα της κατηγοριοποίησης διαβάζει και το table ‘’Tweet’’ στη βάση δεδομένων.

Από τη στιγμή που η εφαρμογή δεχτεί ένα POST request αναλαμβάνει να δημιουργήσει δυο νήματα για να τρέχουν παράλληλα οι διαδικασίες του streaming και της κατηγοριοποίησης:

- Το πρώτο νήμα αναλαμβάνει τη σύνδεση με το Twitter και την εγγραφή των εισερχόμενων δεδομένων στο αρχείο ‘’raw_data.txt’’.
- Το δεύτερο νήμα κρατάει καθ’ όλη τη διάρκεια της ζωής του το αρχείο ‘’raw_data.txt’’ ανοιχτό για να διαβάζει κάθε νέα γραμμή που γράφει το πρώτο νήμα. Για κάθε νέο tweet αναλαμβάνει την κατηγοριοποίηση και αποθήκευση του στη βάση.

Ταυτόχρονα, ένα script γραμμένο σε AJAX στον client ελέγχει ανά χρονικά διαστήματα εάν προστέθηκαν νέα tweets στη βάση για να τα εμφανίσει στον χρήστη.

Εάν ο χρήστης στείλει ένα καινούριο POST request οι διαδικασίες αυτές θα σταματήσουν για να ξεκινήσουν καινούριες.



1. Η λειτουργικότητα της εφαρμογής

Παράδειγμα χρήσης: Ο χρήστης εισέρχεται στο http://127.0.0.1:8000/, συμπληρώνει τη φόρμα και πατάει 'Εκκίνηση' για να σταλθούν οι λέξεις-κλειδιά, που εισήγαγε, στο backend. Από τη στιγμή που ληφθεί ένα POST request γίνονται τα εξής:

1. Καθάρισμα του table Tweets από προηγούμενα δεδομένα
2. Join των active threads εάν υπάρχουν
3. Διαγραφή του αρχείου ''raw_data.txt''
4. Δημιουργία καινούριου thread για streaming με τις λέξεις-κλειδιά που έδωσε ο χρήστης
5. Δημιουργία καινούριου thread για classification

Το νήμα του streaming βρίσκει ένα tweet που περιέχει μια από τις λέξεις- κλειδιά που έδωσε ο χρήστης. Το γράφει σε μια καινούρια γραμμή στο raw_data.txt.
Το νήμα του classification διαβάζει τη νέα γραμμή, κατηγοριοποιεί το tweet και το αποθηκεύει στη βάση.

Καθ’ όλη τη διάρκεια, ο browser του χρήστη στέλνει GET requests ανά 2 δευτερόλεπτα σε άλλο endpoint για να εντοπίσει νέες εγγραφές στη βάση δεδομένων, έτσι ώστε να τις οπτικοποιήσει.

Ο τρόπος λειτουργίας των δυο νημάτων και η οπτικοποίηση περιγράφονται σε επόμενες ενότητες λεπτομερώς.

ΚΕΦΑΛΑΙΟ 2: Τεχνολογίες που χρησιμοποιήθηκαν

2.1 Python

Η Python είναι διερμηνευόμενη, υψηλού επιπέδου, γλώσσα προγραμματισμού. Είναι μια από τις πιο δημοφιλείς γλώσσες λόγω της απλής δομής της και της ευκολίας στη χρήση της [1] .

Ένα από τα χαρακτηριστικά της που την καθιστούν μια από τις πιο δημοφιλείς γλώσσες προγραμματισμού είναι η υποστήριξη πολλαπλών modules, libraries και frameworks. Τα πιο σημαντικά εργαλεία που χρησιμοποιήθηκαν γι' αυτή την εργασία φαίνονται παρακάτω:

- Django – framework για δημιουργία διαδικτυακών εφαρμογών
- Tweepy – library που παρέχει εύκολη πρόσβαση στο API του Twitter
- Threading – module για δημιουργία παράλληλων εργασιών
- Requests – library για αλληλεπίδραση με το HTTP
- Sklearn – library με εργαλεία μηχανικής μάθησης
- Torch – framework για μηχανική μάθηση
- BeautifulSoup – library για web scraping

2.2 Greek Bert

Ο Greek Bert αποτελεί την ελληνική εκδοχή του μοντέλου BERT της Google [2] . Είναι ένα transformer-based μοντέλο, εκπαιδευμένο σε 29GB ελληνικών κειμένων, που φέρει εις πέρας τρεις NLP σκοπούς [3]:

1. *Part-of-Speech (PoS) tagging* - ονομάζεται επίσης γραμματική σήμανση και είναι η διαδικασία επισήμανσης μιας λέξης σε ένα κείμενο, *π.χ.* ουσιαστικό, επίθετο, ρήμα. [4]

2. *Named Entity Recognition (NER)* - είναι μια δευτερεύουσα εργασία εξαγωγής πληροφοριών που επιδιώκει να εντοπίσει και να ταξινομήσει οντότητες σε προκαθορισμένες κατηγορίες, όπως άτομα ονόματα, οργανισμοί, τοποθεσίες, ιατρικοί κώδικες, χρονικές εκφράσεις, ποσότητες, χρηματικές αξίες, ποσοστά κ.λπ. [5]

3. *Natural Language Inference (NLI)* – είναι μια διαδικασία που αποφασίζει εάν μια πρόταση είναι επακόλουθη κάποιας άλλης. [6]

Για τους σκοπούς αυτής της εργασίας, ο Greek Bert χρησιμοποιήθηκε για τη δημιουργία word vectors, τόσο στα training data, όσο και στα εισερχόμενα tweets.

2.3 SQLite

Η SQLite είναι μια βιβλιοθήκη που υλοποιεί μια μικρή, αλλά γρήγορη βάση δεδομένων SQL [7], που αποτελεί τη default βάση δεδομένων στα project του Django [8].

Είναι ιδανική για τη συγκεκριμένη εφαρμογή, αφού τα στοιχεία που αποθηκεύονται είναι λίγα και ανά διαστήματα διαγράφονται εξ ολοκλήρου για να αποθηκευτούν νέα.

2.4 HTML

Η HTML είναι η γλώσσα που προσφέρει δομικά στοιχεία για την κατασκευή ιστοσελίδων [9] . Σε συνδυασμό με τα templates του Django αποτελεί τη μέθοδο απεικόνισης των αποτελεσμάτων της εργασίας στον χρήστη.

Η εργασία περιλαμβάνει μόνο μια σελίδα HTML, η οποία υποστηρίζει τις μεθόδους GET και POST. Τα βασικά της στοιχεία είναι ένα header, που περιέχει έναν τίτλο, έναν υπότιτλο και μια φόρμα εισόδου λέξεων-κλειδιών για να ξεκινήσει η λειτουργία του streaming και της κατηγοριοποίησης των ευρεθέντων tweets. Στο υπόλοιπο body, βρίσκεται ένα table και ένα στοιχείο canvas, τα οποίο ανανεώνονται δυναμικά με χρήση Javascript, όπως περιγράφεται στην επόμενη ενότητα.

2.5 CSS

Η CSS (Cascading Style Sheets – διαδοχικά φύλλα ύφους) χρησιμοποιείται για λόγους βελτιστοποίησης της εμφάνισης σελίδων HTML. [10].

Για τους σκοπούς της εργασίας, εκτός από pure CSS, χρησιμοποιήθηκε εκτενώς η Bootstrap, το πιο δημοφιλές framework της CSS. Συγκεκριμένα το περιεχόμενο της σελίδας οργανώνεται από έννοιες, όπως containers, rows και columns.

2.6 Javascript

Η JavaScript είναι μια γλώσσα προγραμματισμού με πολλές χρήσεις. Η πιο κοινή είναι η συγγραφή client side scripts για την προσαρμογή της συμπεριφοράς μια ιστοσελίδας [11].

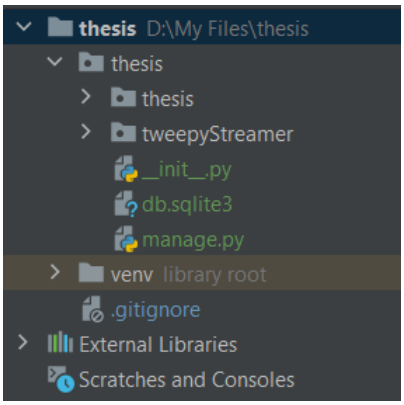
Αποτελεί, ίσως, το πιο σημαντικό κομμάτι για την οπτικοποίηση των αποτελεσμάτων της εργασίας, αφού με χρήση της τεχνικής AJAX, ανανεώνει δυναμικά το στοιχείο table με τα πιο πρόσφατα ευρεθέντα tweets. Επίσης, ενσωματώνει στο στοιχείο canvas ένα δυναμικό ‘’doughnut’’ chart για μια ολοκληρωμένη επίβλεψη των αποτελεσμάτων.

Για τους σκοπούς της εργασίας έχουν συμπεριληφθεί τα παρακάτω javascript scripts:

- bootstrap.min.js: επέκταση της Bootstrap
- jquery-3.6.0.js: open source library για χρήση AJAX
- popper.min.js: βοηθητικό script για στοιχεία tooltips
- chart.min.js: open source library για charts

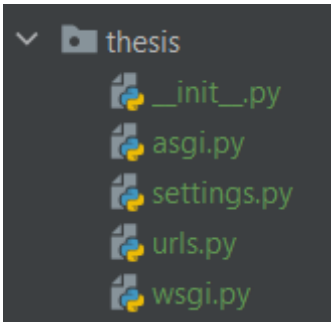
ΚΕΦΑΛΑΙΟ 3: Αρχεία και Οργάνωση της Εργασίας

Η εφαρμογή αναπτύχθηκε με τη βοήθεια του Django framework, συνεπώς η οργάνωση των αρχείων της είναι προκαθορισμένη.



1. Τα directories του project και της εφαρμογής

Το directory ''thesis" αποτελεί το directory του project, ενώ το ''tweepySteamer'' της εφαρμογής. Από τα αρχεία του ''thesis'', τα μόνα που έχουν τροποποιηθεί είναι τα *urls.py*, για να χρησιμοποιεί το project τα urls της εφαρμογής ''tweepyStreamer'', και *settings.py*. Στο τελευταίο έχουν προστεθεί ρυθμίσεις απαραίτητες για τη λειτουργία της εφαρμογής και επίσης είναι υπεύθυνο για την φόρτωση του Greek Bert, των μοντέλων κατηγοριοποίησης και λιστών με χρήσιμες πληροφορίες, που αξιοποιεί το πρόγραμμα. Με αυτόν τον τρόπο τα μοντέλα φορτώνονται μόνο μια φορά με την εκκίνηση του server και αποφεύγονται περιττές καθυστερήσεις.



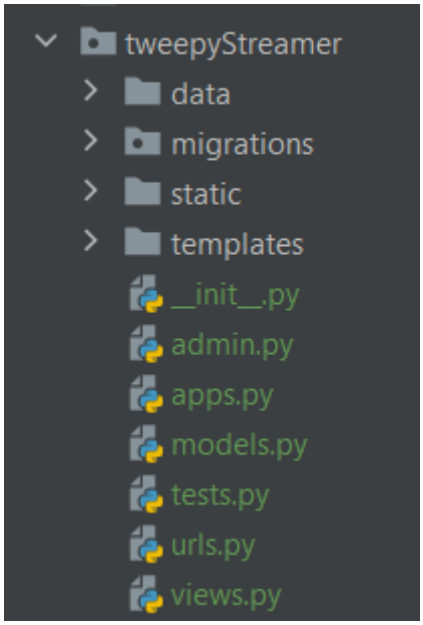
2. Το directory του project

Στο directory της εφαρμογής, δηλαδή το ''tweepyStreamer'', περιλαμβάνεται όλη η λογική της εργασίας:

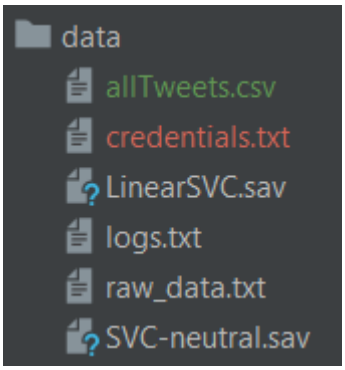
- *models.py*: περιγράφεται πως τα αντικείμενα tweets αποθηκεύονται στη

βάση στο table ‘’Tweet’’

- *urls.py*: δηλώνονται τα δυο endpoints της εφαρμογής, το ‘’index’’ και το ‘’ajax/getTweets’’. Το πρώτο δέχεται GET και POST requests από τον χρήστη και το δεύτερο δέχεται GET requests από τον browser εν αγνοία του χρήστη για εντοπισμό νέων tweets στη βάση δεδομένων.
- *views.py*: διαχείριση των requests στα δυο endpoints και ο κώδικας των νημάτων
- *data* (directory): custom directory που περιλαμβάνει:



3. Το directory της εφαρμογής

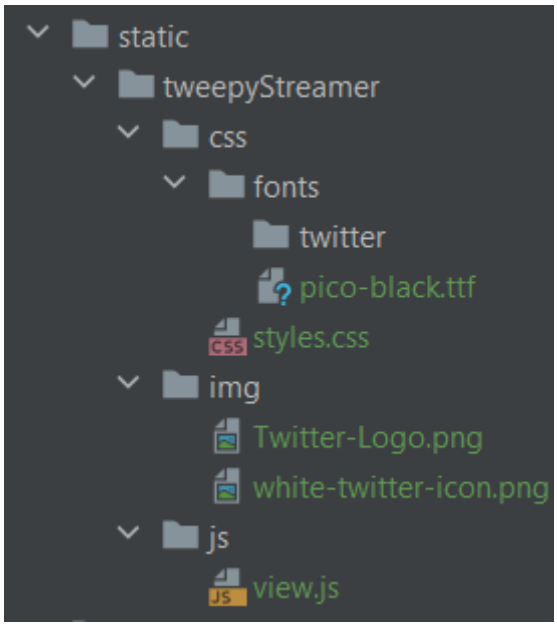


4. Το directory data

  - *credentials.txt*: ένα αρχείο txt με τα credentials που χρειάζονται για τη σύνδεση με το API του Twitter
  - *logs.txt*: ένα αρχείο txt στο οποίο σημειώνονται τυχόν προβλήματα που προκύπτουν κατά την λειτουργία της εφαρμογής
  - *LinearSVC.sav*: pickled αρχείο που περιλαμβάνει το trained μοντέλο κατηγοριοποίησης πάνω σε θετικές και αρνητικές προτάσεις
  - *SVC-neutral.sav*: pickled αρχείο που περιλαμβάνει το trained μοντέλο κατηγοριοποίησης πάνω σε ουδέτερα και μη ουδέτερα tweets

- *raw_data.txt*: ένα αρχείο txt που δημιουργείται μέτα από POST request, στο οποίο σημειώνονται όλα τα εισερχόμενα tweets
- *allTweets.csv*: ένα αρχείο csv που συγκεντρώνει όλα τα tweets που η εφαρμογή κατηγοριοποιεί

- *static* (directory): directory που συγκεντρώνει σε ξεχωριστά directories όλα τα στατικά αρχεία:



5. To directory static

- *tweepyStreamer/img*: περιλαμβάνει τα logo του Twitter σε δυο χρώματα με transparent background. Ανακτήθηκαν από το http://www.stickpng.com/ και είναι ελεύθερα για προσωπική χρήση.
- *tweepyStreamer/js*: περιλαμβάνει ένα javascript αρχείο ‘’view.js’’, το οποίο επηρεάζει τη συμπεριφορά της σελίδας.
- *tweepyStreamer/css*: περιλαμβάνει ένα css αρχείο ‘’styles.css’’ με όλους τους custom css κανόνες της σελίδας και το directory *tweepyStreamer/fonts* που περιλαμβάνει τη γραμματοσειρά *pico-black.ttf*. Ανακτήθηκε από το http://famfonts.com και είναι ελεύθερο για προσωπική χρήση.

- *templates* (directory): περιλαμβάνει το αρχείο html ‘’tweepyStreamer/index.html’’, που αποτελεί τη μοναδική σελίδα της εργασίας.

6. Το directory templates

Για την εκκίνηση της εφαρμογής στον server αρκεί να εκτελεστεί η εντολή:

python manage.py runserver

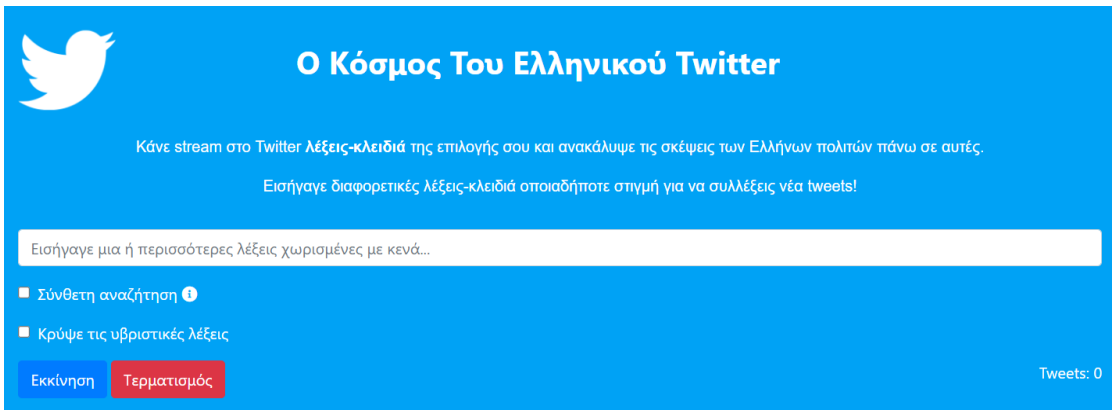
,στο directory ‘thesis/thesis’ και να εισέρθει ο χρήστης στον σύνδεσμο http://127.0.0.1:8000/ στον browser της επιλογής του. Προτείνεται ο chrome πάνω στον οποίο αναπτύχθηκε και δοκιμάστηκε η εφαρμογή.

ΚΕΦΑΛΑΙΟ 4: Υλοποίηση

Η υλοποίηση της εφαρμογής θα περιγραφεί σε τέσσερις μεγάλες ενότητες που αποτελούν και τα θεμέλια της εργασίας: GET & POST Requests στο endpoint ‘’index’’, Streaming, Κατηγοριοποίηση και GET Requests στο endpoint ‘’ajax/getTweets’’.

4.1 GET & POST Requests στο Endpoint ‘’index’’ & Στατικά Στοιχεία του User Interface

Το endpoint ‘’index’’ αποτελεί το πρωτεύον μέσο αλληλεπίδρασης του χρήστη με την εφαρμογή. Αναλαμβάνει το ‘’σερβίρισμα’’ της μοναδικής HTML σελίδας στο url http://127.0.0.1:8000/ ,καθώς και την αποδοχή και την επεξεργασία των POST requests.



7. Η σελίδα που βλέπει ο χρήστης μετά από ένα GET request στο index

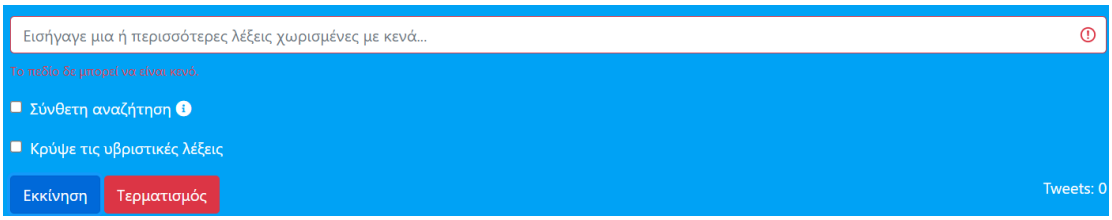
4.1.1 Συμπλήρωση και υποβολή φόρμας

Για να ξεκινήσει η εφαρμογή, ο χρήστης καλείται να συμπληρώσει τη φόρμα με όσες λέξεις επιθυμεί χωρισμένες με κενά και να κάνει check τα παρακάτω checkboxes εάν του είναι χρήσιμα:

- *Σύνθετη Αναζήτηση*: επιδιώκει να εντοπίσει κλίσεις / πτώσεις / γένη / πληθυντικούς αριθμούς των εισαγόμενων λέξεων για να εμπλουτίσει τα κριτήρια αναζήτησης των tweets. Είναι εργασία του νήματος ‘’stream’’ και αναλύεται στην ενότητα 4.2.

- *Κρύψε τις υβριστικές λέξεις*: επιδιώκει να εντοπίσει και να κρύψει λέξεις που σύμφωνα με το **Greek Wictionary**[1] θεωρούνται υβριστικές ή χυδαίες. Είναι εργασία του νήματος ‘’classifier’’ και αναλύεται στην ενότητα 4.3.

Ο χρήστης είναι υποχρεωμένος να εισάγει έστω και μια λέξη, αλλιώς το front end θα τον ενημερώσει πως δεν μπορεί να υποβάλλει το αίτημα του. Τα checkboxes είναι προαιρετικά και έχουν σκοπό να ενισχύσουν την εμπειρία του χρήστη.



8. Ενημερωτικό μήνυμα σε περίπτωση άδειας εισόδου στη φόρμα

Στη συνέχεια περιγράφεται πως το endpoint ‘’index’’ διαχειρίζεται τα requests που λαμβάνει.

[1] https://el.wiktionary.org/ [Online]

4.1.2 GET Requests

Το index δεν προσφέρει κάποιο context στη σελίδα, δηλαδή δεν περνάει κάποια μεταβλητή του Django project στο front end. Όλη η δυναμικότητα της σελίδας οφείλεται σε ένα ajax script που αλληλοεπιδρά με το endpoint ‘’ajax/getTweets’’, το οποίο θα περιγραφεί εκτενώς στην ενότητα 4.4. Κάθε GET request στο index επιστρέφει μόνο τη σελίδα **index.html** και τα στατικά της στοιχεία.

4.1.3 POST Requests

Εάν το index δεχθεί ένα POST request θα ελέγξει πρώτα ποιο στοιχείο input με τύπο ‘’submit’’ πατήθηκε από τον χρήστη, δηλαδή θα εντοπίσει ποια από τις ακόλουθες δυο τιμές είναι παρούσα:

- ‘’start’’: σημαίνει πως ο χρήστης πάτησε το κουμπί ‘’Εκκίνηση’’ στο user interface. Σε αυτήν την περίπτωση το index θα καλέσει τη συνάρτηση ‘’startProcesses’’ με arguments:
    - term: string που περιέχει τις λέξεις που εισήγαγε ο χρήστης στη φόρμα
    - advancedSearch: λίστα που περιέχει την τιμή ‘’on’’ εάν ο χρήστης έκανε κλικ στο checkbox ‘’Σύνθετη Αναζήτηση’’, διαφορετικά είναι άδεια
    - hideBadWords: λίστα που περιέχει την τιμή ‘’on’’ εάν ο χρήστης έκανε κλικ στο checkbox ‘’Κρύψε τις υβριστικές λέξεις’’, διαφορετικά είναι άδεια
- ‘’stop’’: σημαίνει πως ο χρήστης πάτησε το κουμπί ‘’Τερματισμός’’ στο user interface. Σε αυτήν την περίπτωση το index θα καλέσει τη συνάρτηση ‘’stopProcesses’’.

Η ‘’**startProcesses**’’ εκτελεί με την σειρά τις παρακάτω εργασίες:

1. Διαγράφει όλες τις εγγραφές από το table Tweets στη βάση δεδομένων. Έτσι αφήνει τον χώρο ελεύθερο για τα καινούρια tweets που θα αποθηκευτούν με βάση τις νέες λέξεις-κλειδιά που έδωσε ο χρήστης.

2. Σταματάει τα παλιά νήματα εάν υπάρχουν. Τα νήματα είναι αποθηκευμένα σε μια global λίστα ‘’threads’’ και για κάθε νήμα της λίστας γίνεται ξεχωριστά raise exception και join. Η μέθοδος raise_exception είναι custom μέθοδος της κλάσης Thread, όπου δίνει σήμα SystemExit για να σταματήσουν τα νήματα τις διεργασίες τους.

3. Διαγράφει το αρχείο *raw_data.txt* εάν υπάρχει για να δημιουργηθεί εκ νέου αργότερα από το νήμα ‘’stream’’ και να αποκτήσει καινούριες εγγραφές, σύμφωνα με τις λέξεις-κλειδιά που παρείχε ο χρήστης.

4. Δημιουργεί καινούρια νήματα ‘’streamer’’ και ‘’classifier’’. Με αυτόν τον τρόπο τα νήματα αρχικοποιούνται με τα arguments από το πιο πρόσφατο POST request. Συγκεκριμένα, στο νήμα ‘’streamer’’ δίνεται η είσοδος term, δηλαδή το string με τις λέξεις-κλειδιά και η λίστα advancedSearch, ενώ στο νήμα ‘’classification’’ δίνεται η λίστα hideBadWords.

Η ‘’**stopProcesses**’’ αναλαμβάνει μόνο να τερματίσει τα νήματα με τον ίδιο τρόπο που τα τερματίζει η ‘’startProcesses’’.

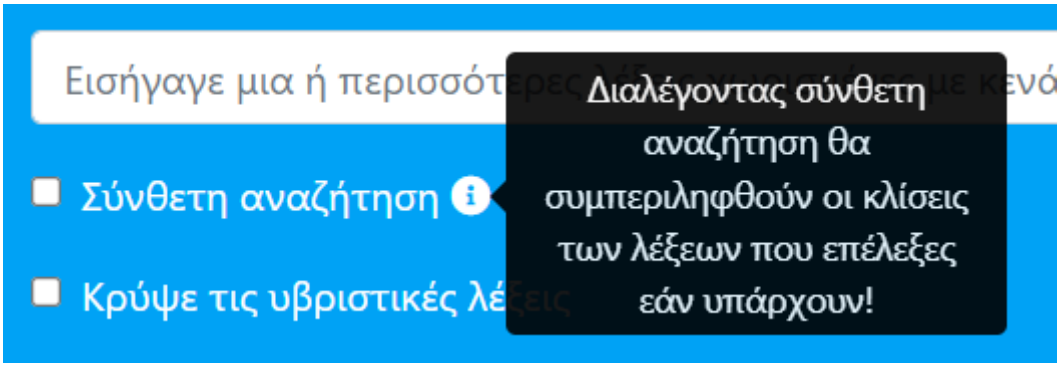
Από τη στιγμή που πατηθεί το κουμπί ‘’Εκκίνηση’’ η σκυτάλη περνάει στα επόμενα τρία λειτουργικά κομμάτια της εφαρμογής: την συλλογή tweets με τη μέθοδο streaming, την κατηγοριοποίηση τους σε ‘’θετικό’’, ‘’ουδέτερο’’ και ‘’αρνητικό’’ και την οπτικοποίηση τους. Οι τρεις αυτές διαδικασίες περιγράφονται στις επόμενες τρεις ενότητες αντίστοιχα.

4.2 Streaming

Η διαδικασία του streaming εστιάζει στη συλλογή tweets, που περιλαμβάνουν τις λέξεις-κλειδιά που επέλεξε ο χρήστης και εκτελείται υπό τη μορφή νήματος, το οποίο δημιουργείται κάθε φορά που λαμβάνεται ένα POST request στο endpoint ‘’index’’. Με άλλα λόγια, μόλις ο χρήστης εισάγει στη φόρμα όσες λέξεις-κλειδιά επιθυμεί χωρισμένες με ένα κενό και πατήσει ‘’Εκκίνηση’’, θα δημιουργηθεί το νήμα ‘’streamer’’, με arguments τις λέξεις-κλειδιά που εισήγαγε ο χρήστης και την τιμή ‘’advancedSearch’’ που αναφέρεται στο checkbox ‘’Σύνθετη Αναζήτηση’’.

Το νήμα streamer είναι υπεύθυνο:

1. Για τη δημιουργία ενός instance της κλάσης ‘’StreamListener’’ με arguments τα απαραίτητα credentials για τη σύνδεση στο API του Twitter, τα οποία είναι διαθέσιμα σε ξεχωριστές μεταβλητές στο settings.py.
2. Να ελέγξει εάν στο user interface, ο χρήστης πάτησε το checkbox ‘’*Συνθετη Αναζήτηση*’’, ελέγχοντας την τιμή της μεταβλητής ‘’advancedSearch’’.
3. Να εντοπίσει με web scraping, εάν υπάρχουν, τις κλίσεις / πτώσεις / γένη / πληθυντικούς αριθμούς όλων των λέξεων-κλειδιών που δόθηκαν στην περίπτωση που ο χρήστης επέλεξε ‘’Σύνθετη Αναζήτηση’’.
4. Να καλέσει τη μέθοδο filter του instance, δίνοντας σαν argument την τελική λίστα λέξεων.



9. Ενημέρωση του χρήστη για την λειτουργία της "Σύνθετης Αναζήτησης"

4.2.1 Σύνθετη Αναζήτηση

Στην περίπτωση που ο χρήστης το επέλεξε, θα αναλάβει να βρει, εάν υπάρχουν, τις κλίσεις, πτώσεις, τα γένη και τους πληθυντικούς των λέξεων-κλειδιών που εισήχθησαν, με web scraping στο **Greek Wictionary** και θα τις προσθέσει στη λίστα λέξεων-κλειδιών για να συμπεριληφθούν και αυτές στο streaming. Με αυτόν τον τρόπο η μέθοδος *filter* του StreamListener, που ‘’φιλτράρει’’ τη ροή δεδομένων από το Twitter, θα έχει επιπλέον πόρους για να αναζητήσει και συνεπώς θα υπάρχουν περισσότερα εισερχόμενα tweets.

Πιο συγκεκριμένα, η Σύνθετη Αναζήτηση είναι ικανή να εντοπίσει εάν υπάρχουν:

- Τις κλίσεις των άρθρων
    - Οριστικών (ο / η / το)
    - Αόριστων (ένας / μια / ένα)
- Τις κλίσεις των αντωνυμιών
    - Προσωπικών (πχ. εγώ, εσύ, εμένα …)
    - Οριστικών (πχ. ίδιος, μόνος)
    - Δεικτικών (πχ. αυτός, τούτος, εκείνος)
    - Αναφορικών (πχ. όποιος)
    - Αορίστων (πχ. άλλος)
    - Επιμεριστικές (πχ. εκάτερος)

- Τις πτώσεις, τα γένη και τους πληθυντικούς αριθμούς ουσιαστικών (πχ. άλογο)
- Τις πτώσεις, τα γένη και τους πληθυντικούς αριθμούς επιθέτων (πχ. καλός)

Ο τρόπος εύρεσης των στοιχείων αυτών, έχει υλοποιηθεί γενικευμένα, έτσι ώστε να εφαρμόζεται και να λειτουργεί στην πλειοψηφία των σελίδων του wictionary. Δηλαδή, ο τρόπος δόμησης των στοιχείων «table», που περιέχουν τις κλίσεις των λέξεων κάνουν χρήση συγκεκριμένων κλάσεων και έτσι εντοπίζονται και απομονώνονται εύκολα.

Εξαίρεση αποτελούν οι σελίδες με τις κλίσεις των οριστικών άρθρων και των προσωπικών αντωνυμιών, που τα tables δομούνται λίγο διαφορετικά. Γι’ αυτό το λόγο πρώτα ελέγχεται εάν μια λέξη ανήκει σε μια από τις παραπάνω δυο κατηγορίες, αλλιώς χρησιμοποιείται η γενικότερη αναζήτηση, που περιγράφηκε στην προηγούμενη παράγραφο.

Παρακάτω φαίνεται ο ψευδοκώδικας της Σύνθετης Αναζήτησης:

```
For term in list_of_terms:
      content = request to 'https://el.wiktionary.org/wiki/' + term
      soup = BeautifulSoup(content)
      if span with id ‘’κλίσεις_των_άρθρων’’ in soup:
            term is an article
      elif 'Προσωπικές αντωνυμίες' in soup:
            term is personal pronoun
      elif td with certain classes in soup:
            term is either adjective or substantive
      else:
            term has no table with related words
```

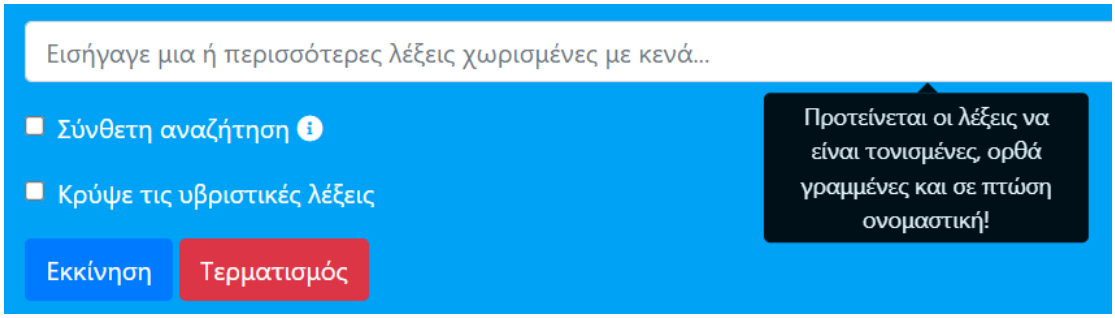
1. Ο ψευδοκώδικας της σύνθετης αναζήτησης

Στον παρακάτω πίνακα φαίνονται μερικά παραδείγματα χρήσης της *Σύνθετης Αναζήτησης* από τα logs της εφαρμογής:

Είσοδος Χρήστη	**Η τελική λίστα αναζήτησης του νήματος streamer από τα logs**
Ο	Starting streamer with term(s) ['ο', 'το', 'σε', 'του', 'στου', 'της', 'στης', 'τον', 'στον', 'τη(ν)', 'στη(ν)', 'στο', 'οι', 'τα', 'των', 'στων', 'τους', 'στους', 'τις', 'στις', 'στα']
Όποιος	Starting streamer with term(s) ['όποιος', 'όποια', 'όποιο', 'όποιου-', 'όποιας-', 'όποιο(ν)', 'όποιοι', 'όποιες', 'όποιων-', 'όποιους-']
Εμβόλιο	Starting streamer with term(s) ['εμβόλιο', 'εμβόλια', 'εμβολίου', 'εμβόλιου', 'εμβολίων']
Όμορφος	Starting streamer with term(s) ['όμορφος', 'όμορφη', 'όμορφο', 'όμορφου', 'όμορφης', 'όμορφε', 'όμορφοι', 'όμορφες', 'όμορφα', 'όμορφων', 'όμορφους']
Ένας	Starting streamer with term(s) ['ένας', 'μία', 'ένα', 'ενός', 'μίας', 'έναν']
Εκείνος	Starting streamer with term(s) ['εκείνος', 'εκείνη', 'εκείνο', 'εκείνου', 'εκείνης', 'εκείνοι', 'εκείνες', 'εκείνα', 'εκείνων', 'εκείνους']

1. Παραδείγματα χρήσης της Σύνθετης Αναζήτησης

Πρέπει να σημειωθεί πως η λειτουργία αυτή θα είναι περισσότερο αποτελεσματική, εάν ο χρήστης ακολουθήσει την προτροπή του user interface και παρέχει λέξεις τονισμένες, ορθά γραμμένες και σε ονομαστική πτώση.



10. Παρότρυνση του χρήστη για βελτιωμένη εισαγωγή λέξεων

Εάν ο χρήστης δεν επιλέξει τη ‘’Σύνθετη Αναζήτηση’’, στη μέθοδο *filter* θα δοθεί σαν argument μόνο η αρχική λίστα λέξεων-κλειδιών, έτσι ώστε να ξεκινήσει το streaming.

4.2.2 Διαχείριση του stream

Αφού έχουν βρεθεί οι λέξεις-κλειδιά και το νήμα έχει καλέσει τη μέθοδο filter, το instance της κλάσης StreamListener ξεκινάει να φιλτράρει τη ροή δεδομένων του Twitter. Η διαχείριση του stream γίνεται με βάση τρεις μεθόδους:

1. on_data: διαχείριση των εισερχόμενων δεδομένων
2. on_error: διαχείριση των error που μπορεί να προκύψουν
3. on_limit: διαχείριση της περίπτωσης υπέρβασης του ορίου του Twitter

on_data

Η on_data επιδιώκει να εντοπίσει χρήσιμο υλικό για να το αποθηκεύσει, ώστε να κατηγοριοποιηθεί αργότερα. Το μοναδικό είδος δημοσίευσης που δε δέχεται είναι εκείνων που ξεκινούν με ‘’RT’’ , δηλαδή τα retweets, αφού αποτελεί δημοσίευση

άλλου ατόμου και μπορεί να μη βρίσκει σύμφωνο τον κοινοποιητή του. Επίσης, το tweet πρέπει να περιλαμβάνει το λιγότερο 5 ελληνικούς χαρακτήρες για να θεωρηθεί ελληνικό. Με αυτόν τον τρόπο ο χρήστης είναι ελεύθερος να εισάγει και λέξεις σε άλλες γλώσσες και να λάβει ελληνικά αποτελέσματα.

Για κάθε tweet που εντοπίζεται και που πληροί τις παραπάνω προϋποθέσεις θα αποθηκεύει το id, το userId και το text του σε μια γραμμή του **raw_data.txt** σε μορφή dictionary. Εάν καθ' όλη τη διάρκεια του on_data προκύψει κάποιο exception θα εγγραφεί στο logs.txt.

Πρέπει να σημειωθεί πως αν το raw_data.txt δεν υπάρχει ακόμα, επειδή διαγράφτηκε από την συνάρτηση ''startProcesses'', όπως περιγράφηκε στην ενότητα 4.1, το δημιουργεί με το πρώτο εισερχόμενο tweet.

on_error & on_limit

Οι on_error και on_limit λειτουργούν ενημερωτικά και εκτυπώνουν ένα μήνυμα στην οθόνη εάν ενεργοποιηθούν.

Κάθε tweet που αποθηκεύεται στο raw_data.txt , περιμένει να γίνει αντιληπτό από το νήμα classifier, έτσι ώστε να περάσει από επεξεργασία, να ταξινομηθεί και να αποθηκευτεί στη βάση.

4.3 Κατηγοριοποίηση

Η κατηγοριοποίηση, όμοια με το streaming, είναι μια διεργασία που εκτελείται υπό τη μορφή νήματος και δημιουργείται μετά από κάθε POST request του χρήστη. Εάν υπάρχει ήδη νήμα κατηγοριοποίησης, ενώ έχει ζητηθεί καινούριο τότε πρώτα θα τερματιστεί με ασφάλεια κάνοντας raise exception και μετά θα γίνει join, ώστε να φτιαχτεί το επόμενο. Επίσης, έχει μόνο ένα argument, το hideBadWords, που περιέχει την τιμή ‘’on’’ εάν επιλέχθηκε το checkbox ‘’Κρύψε τις υβριστικές λέξεις’’ στο user interface.

Το νήμα classifier είναι υπεύθυνο:

1. Να περιμένει τη δημιουργία του ‘’raw_data.txt’’ από το νήμα streamer. Όσο δεν το βρίσκει στο αναμενόμενο path, καλεί τη μέθοδο sleep() για ένα δευτερόλεπτο.

2. Να ανοίξει το αρχείο ‘’raw_data.txt’’, όταν το εντοπίσει στο αναμενόμενο path σε read mode και να το κρατήσει ανοιχτό μέχρι κάποια συνάρτηση να το τερματίσει

3. Να επιδιώκει να διαβάζει τις γραμμές του αρχείου και αν αυτές δεν έχουν ακόμα περιεχόμενο, δηλαδή το νήμα streamer δεν έχει προλάβει να γράψει κάτι καινούριο, να καλεί τη μέθοδο sleep() για ένα δευτερόλεπτο και να ξανα-προσπαθεί

4. Για κάθε καινούρια γραμμή που εντοπίζει στο αρχείο να προ-επεξεργάζεται το κείμενο του tweet, να το κατηγοριοποιεί και να το αποθηκεύει στη βάση δεδομένων

4.3.1 Επεξεργασία των tweets

Κάθε tweet, που διαβάζει το νήμα classifier, ξεκαθαρίζεται από stopwords, αριθμούς, σημεία στίξης και ειδικούς χαρακτήρες και ελέγχεται ως προς τους ειδικούς χαρακτήρες του και το πλήθος άγνωστων λέξεων του. Επίσης, σε περίπτωση που ο χρήστης επέλεξε τη λειτουργία ‘’Κρύψε τις υβριστικές λέξεις’’, η κάθε λέξη ξεχωριστά ελέγχεται ως προς το ύφος της και εάν θεωρηθεί υβριστική, αντικαθίσταται με ‘*’.

4.3.1.1 Καθάρισμα των tweets

Πριν δοθεί για κατηγοριοποίηση, κάθε tweet καθαρίζεται από:

- Stopwords: ορίζονται ως εκείνες οι λέξεις που δεν προσφέρουν τίποτα στην κατηγοριοποίηση, όπως για παράδειγμα τα άρθρα και οι αντωνυμίες. Το settings.py κατά την έναρξη του server, φορτώνει μια λίστα ‘’GREEK_STOPWORDS”, που περιέχει 847 ελληνικές stopwords τονισμένες και με πεζούς χαρακτήρες. Η λίστα ανακτήθηκε από: https://github.com/stopwords-iso/stopwords-el/blob/master/stopwords-el.json

- Αριθμούς , σημεία στίξης, ειδικούς χαρακτήρες. Όλες αυτές οι κατηγορίες απομακρύνονται με τη μέθοδο isalpha() των string, που επιστρέφει True, εάν το string περιέχει μόνο γράμματα.

Στον παρακάτω πίνακα φαίνονται μερικά παραδείγματα σύγκρισης των πρωτότυπων tweets με την καθαρισμένη εκδοχή τους:

Tweet	Tweet μετά το καθάρισμα
Το σημερινο επεισοδιο επρεπε να παιξει αυριο #survivorGR \nΘα εσβηνε και σασμους και μελισσες και τα παντα στο διαβα του!! \n Καθαρο 30αρι θα κανει\n\n#sasmos #gitiselias #agriesmelisses #masterchefgr	σημερινο επεισοδιο επρεπε παιξει εσβηνε σασμους μελισσες παντα διαβα Καθαρο κανει
Αλαλούμ με τη μετάλλαξη Deltacron: «Βιάστηκε ο Μαγιορκίνης – Δεν είναι τεχνικό λάθος» #Deltacron #κορονοιος #thesspress\nhttps://t.co/AR1ndobHxt	Αλαλούμ μετάλλαξη Μαγιορκίνης είναι τεχνικό
Το κέντρο επιχειρήσεων στην Αθήνα θα μπορεί να στέλνει ασθενείς με Covid στα ιδιωτικά νοσοκομεία που έχουν προσφέρει τις 300 κλίνες\n#κορονοιος #ΕΣΥ\nhttps://t.co/QOxsqdnZ5w	Κέντρο επιχειρήσεων Αθήνα μπορεί στέλνει ασθενείς Covid ιδιωτικά νοσοκομεία έχουν προσφέρει κλίνες

2. Παραδείγματα αφαίρεσης stopwords & περιττών χαρακτήρων από tweets

4.3.1.2 Ειδικοί χαρακτήρες του Twitter

Αφού σε μια λίστα αποθηκεύτηκαν οι πραγματικά χρήσιμες λέξεις του tweet, η επόμενη κίνηση είναι να δημιουργηθούν άλλες δυο που θα αποφασίσουν εάν έχει νόημα να περάσει το tweet για κατηγοριοποίηση. Η πρώτη λίστα περιλαμβάνει όλες τις λέξεις που ξεκινούν με:

- http:// ή https:// , δηλαδή links
- @ , δηλαδή αναφορά σε άλλους χρήστες
- # , δηλαδή κάποιο hashtag

και η δεύτερη περιέχει όλες τις υπόλοιπες λέξεις. Στη συνέχεια, πραγματοποιούνται δυο έλεγχοι:

1. Συγκρίνονται τα πλήθη των στοιχείων των δυο πιο πρόσφατων λιστών και εάν η λίστα με τους ειδικούς χαρακτήρες ξεπερνάει σε μήκος τη λίστα με τις υπόλοιπες λέξεις

2. Ξεκαθαρίζεται εάν υπάρχουν χρήσιμες λέξεις για κατηγοριοποίηση

Εάν κάποιος από τους δυο ελέγχους ή και οι δυο αποτύχουν το νήμα προσπερνάει το συγκεκριμένο tweet.

Ο λόγος για τον οποίο γίνεται αυτός ο έλεγχος είναι διότι στο Twitter είναι συχνό φαινόμενο, κάποιος να κάνει μια δημοσίευση μόνο με έναν σύνδεσμο ή μόνο με hashtags, τα οποία δεν είναι χρήσιμα στοιχεία για την κατηγοριοποίηση.

Στον παρακάτω πίνακα φαίνονται μερικά παραδείγματα σύγκρισης των δυο προαναφερθέντων λιστών και εάν θα πέρναγαν για κατηγοριοποίηση.

Tweet	**len(usefulWords)**	**len(words)**	**len(special)**	**Accepted**
Το κέντρο επιχειρήσεων στην Αθήνα θα μπορεί να στέλνει ασθενείς με Covid στα ιδιωτικά νοσοκομεία που έχουν προσφέρει τις 300 κλίνες\n#κορονοιος #ΕΣΥ\nhttps://t.co/QOxsqdnZ5w	12	21	3	True
#ΠαιδωνΑγιαΣοφια11 #ΠαιδοκαρδιοχειρουργικοΑγιαΣοφια #Παιδοκαρδιοχειρουργικο	0	0	3	False
ο η το	0	3	0	False

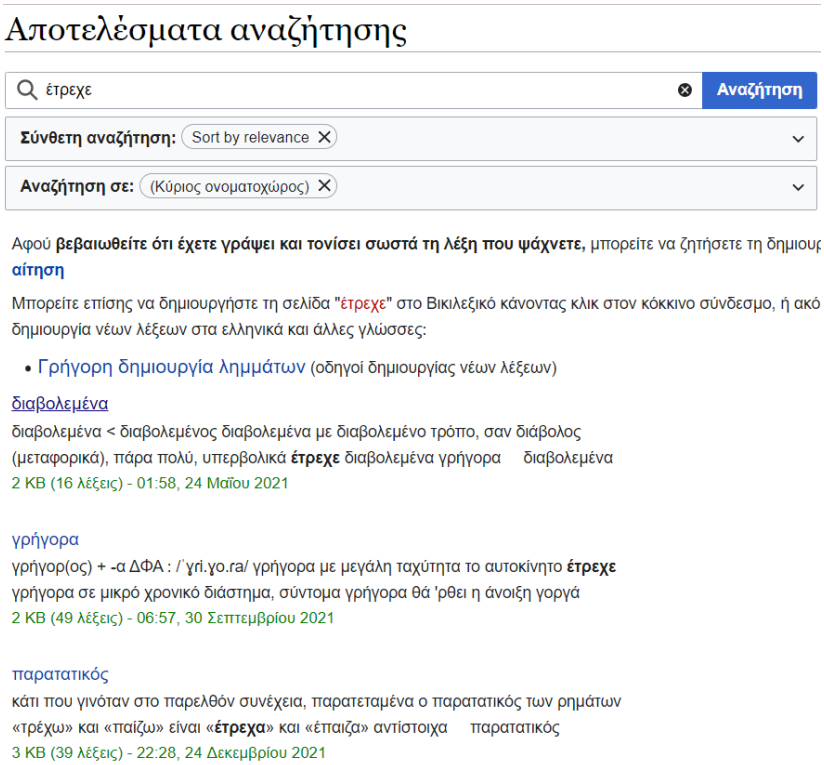
3. Παραδείγματα ελέγχων που καθορίζουν την πορεία του tweet

4.3.1.3 Πλήθος άγνωστων λέξεων

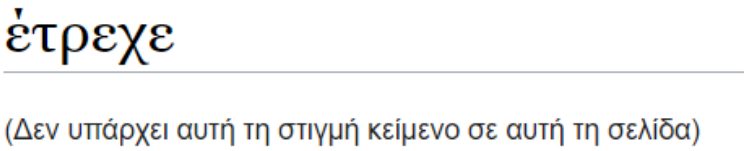
Αφού το νήμα ξεκαθαρίσει πως σε ένα tweet υπάρχουν περισσότερες λέξεις από ειδικούς χαρακτήρες και αφού το καθαρίσει από stopwords και άλλα μη χρήσιμα στοιχεία, οι εναπομείναντες λέξεις θα περάσουν μια-μια από τη φόρμα αναζήτησης του **Greek Wictionary**.

Συγκεκριμένα, για κάθε λέξη στέλνεται ένα request στο https://el.wiktionary.org/w/index.php?search= και με beautifulSoup εξερευνούνται τα αποτελέσματα της αναζήτησης. Εάν δεν υπάρχουν αποτελέσματα, η λέξη σημειώνεται ως άγνωστη, αλλιώς ως γνωστή.

Πρέπει να σημειωθεί πως δεν είναι απαραίτητο οι λέξεις να έχουν δική τους αναλυτική σελίδα και γι' αυτό αρκεί να εμφανιστεί σε μια λίστα αποτελεσμάτων αναζήτησης. Στις παρακάτω δυο εικόνες φαίνονται τα αποτελέσματα της λέξης ''έτρεχε'' ως μέρος μια λίστας αποτελεσμάτων και ως αυτούσια σελίδα:



11. Αποτελέσματα της: https://el.wiktionary.org/w/index.php?search=έτρεχε



12. Αποτελέσματα της: https://el.wiktionary.org/wiki/έτρεχε

Συνεπώς, αυτή η μέθοδος είναι πιο φιλική ως προς την ευρεία ελληνική γραμματική.

Εάν μια λέξη σημειωθεί ως άγνωστη, αυτό μπορεί να οφείλεται σε γραμματικό λάθος, σε λαϊκό λεξιλόγιο ή αν είναι πραγματικά άγνωστη.

Αφού καταμετρηθούν οι άγνωστες και γνωστές λέξεις ενός tweet, θα ελεγχθεί εάν οι άγνωστες είναι περισσότερες από τις γνωστές και σε μια τέτοια περίπτωση το tweet θα προσπεραστεί.

4.3.1.4 Απόκρυψη υβριστικών λέξεων

Η λειτουργία αυτή θα εκτελεστεί, εάν ο χρήστης είχε επιλέξει το checkbox ‘’Κρύψε τις υβριστικές λέξεις’’ και βασίζεται επίσης σε μεθόδους web scraping πάνω στο **Greek Wictionary**. Εφαρμόζεται στις λέξεις του καθαρισμένου tweet, δηλαδή σε εκείνες που θα δοθούν για κατηγοριοποίηση, οι οποίες επίσης σημειώθηκαν ως γνωστές.

Για να θεωρηθεί μια λέξη ως χυδαία ή υβριστική πραγματοποιούνται 3 έλεγχοι:

1. Στα πρώτα τρία αποτελέσματα της σελίδας αναζήτησης, υπάρχει μια από τις ακόλουθες λέξεις: *χυδαίο*, *υβριστικό*, *βρισιά*. Με αυτόν τον τρόπο καταλαβαίνουμε πως κάποιο από τα τρία πιο σχετικά αποτελέσματα της λέξης, την χρησιμοποιούν με χυδαίο τρόπο.

2. Ή στα πρώτα τρία αποτελέσματα της σελίδας αναζήτησης, υπάρχει κάποια από τις υβριστικές λέξεις στη λίστα SWEAR_WORDS, που είναι αρχικοποιημένη στο settings.py και περιέχει τις λέξεις της σελίδας ‘’*Χυδαιολογίες*’’ του Greek Wictionary. Έτσι, μπορούν να εντοπιστούν χυδαίες λέξεις με πολλά συνθετικά, των οποίων η ετυμολογία θα προδώσει, αφού η λίστα περιέχει τις πιο συχνά χρησιμοποιούμενες υβριστικές λέξεις.

3. Και η ίδια η λέξη να μην ανήκει στη λίστα NON_SWEAR_WORDS ,που επίσης αρχικοποιείται στο settings.py. Αυτή η λίστα έχει δημιουργηθεί χειροκίνητα και περιέχει λέξεις που λανθασμένα έχουν χαρακτηριστεί ή μπορούν να χρησιμοποιηθούν λανθασμένα σαν υβριστικές λέξεις.

Οι υβριστικές λέξεις , πέρα του πρώτου γράμματος τους, αντικαθιστούνται με ‘*’ και τα ‘’καθαρά’’ tweets, αποθηκεύονται στο column ‘’cleanText’’, το οποίο χρησιμοποιείται και για οπτικοποίηση.

4.3.2 Κατηγοριοποίηση

Μετά το πέρας της επεξεργασίας του κάθε tweet, το σύνολο των χρήσιμων λέξεων του θα περάσουν για κατηγοριοποίηση, όπου οι διαθέσιμες κλάσεις είναι *θετικό*, *αρνητικό*, και *ουδέτερο*. Στο σύνολο υπάρχουν δυο ταξινομητές, από τους οποίους ο πρώτος αναλαμβάνει να αποφασίσει εάν ένα tweet είναι ουδέτερο ή όχι, αν δηλαδή περιέχει άποψη και μόνο στην περίπτωση που το θεωρήσει μη ουδέτερο, ο δεύτερος ταξινομητής θα διαπιστώσει εάν είναι θετικό ή ουδέτερο.

Οι χαρακτηρισμοί είναι: [0] – αρνητικό. [1] – ουδέτερο, [2] – θετικό και το αποτέλεσμα αποθηκεύεται στο column ‘’*classPredicted*’’.

Στις επόμενες υποενότητες περιγράφονται τα μοντέλα που χρησιμοποιήθηκαν, τα datasets που παρείχαν τα training δεδομένα, τη διαδικασία πριν το training και τον τρόπο προετοιμασίας των tweets για κατηγοριοποίηση.

4.3.2.1 Datasets

Η εφαρμογή στηρίζεται σε 8 διαφορετικά datasets, από τα οποία διαφορετικές τιμές χρησιμοποιήθηκαν για κάθε training. Στον παρακάτω πίνακα φαίνονται τα στοιχεία αυτών των datasets:

Όνομα Dataset	**Χρησιμοποιήθηκε στην ουδέτερη -μη ουδέτερη κατηγοριοποίηση**	**Χρησιμοποιήθηκε στην θετική – αρνητική κατηγοριοποίηση**
Greek movies [12]	Όχι	Ναι
Greek labeled Covid-19 tweets [13]	Ναι	Ναι
Skroutz shop reviews sentiment analysis [14]	Όχι	Ναι
Sentiment ratings for greek tweets [15]	Ναι	Ναι
TIFF53 [16]	Ναι	Ναι
TDF14 [16]	Ναι	Ναι
Offensive Greek Tweet Dataset [17]	Όχι	Ναι
Sentiment Analysis Greek (Political Tweets) [18]	Ναι	Ναι

4. Τα datasets που αξιοποιήθηκαν για το training

Μερικές διευκρινίσεις:

- Το dataset *Greek movies* περιέχει κριτικές ταινιών από χρήστες στο athinorama.gr. Για να αξιοποιηθεί θεωρήθηκε πως με άριστα το 5, βαθμολογίες από 4 και πάνω είναι θετικές, ενώ βαθμολογίες μικρότερες του 3 είναι αρνητικές. Οι κριτικές με βαθμολογία 3 και 3.5 εξαιρέθηκαν γιατί μπορεί να μην είναι ξεκάθαρο προς ποια κατεύθυνση τείνει ο χρήστης.

- Το dataset *Sentiment ratings for greek tweets* περιέχει αξιολογήσεις από δυο ερευνητές πάνω σε έξι βασικά συναισθήματα: θυμός, αποστροφή, λύπη, χαρά, φόβος, έκπληξη. Για να αξιοποιηθεί θεωρήθηκε πως εάν υπάρχει βαθμολογία στα πεδία *χαρά* και *έκπληξη* είναι θετικό, ενώ εάν υπάρχει βαθμολογία στα υπόλοιπα πεδία τότε είναι αρνητικό.

- Το *Offensive Greek Tweet Dataset* τοποθετεί τις ετικέτες *OFF* και *NOT*, ανάλογα με το αν ο ερευνητής θεώρησε το tweet προσβλητικό ή όχι αντίστοιχα. Για να αξιοποιηθεί, όλα τα προσβλητικά tweets θεωρήθηκαν αρνητικά, ενώ τα μη προσβλητικά δεν χρησιμοποιήθηκαν καθόλου, αφού θα μπορούσαν να είναι οποιαδήποτε από τις τρεις κλάσεις – θετικό, αρνητικό, ουδέτερο.

Επισκόπηση των τελικών datasets για κάθε περίπτωση:

- Για ουδέτερη – μη ουδέτερη κατηγοριοποίηση:

  Συνολικό μέγεθος: 34.182
  Σύνολο ουδέτερων: 12.577
  Σύνολο μη ουδέτερων: 21.605

- Για θετική – αρνητική κατηγοριοποίηση:

  Συνολικό μέγεθος: 152.197
  Σύνολο θετικών: 82.558
  Σύνολο αρνητικών: 69.639

4.3.2.2 Προετοιμασία των training δεδομένων

Η προετοιμασία των δυο τελικών datasets για την εκπαίδευση, έγινε με τη βοήθεια του μοντέλου *Greek Bert*. Συγκεκριμένα,

1. Κάθε κείμενο/πρόταση από κάθε dataset δόθηκε στον tokenizer του Greek Bert για να κωδικοποιηθεί. Ο tokenizer αναλαμβάνει για κάθε λέξη, αριθμό και σημείο στίξης της πρότασης, που δεν είναι ειδικός χαρακτήρας του Twitter (π.χ. hashtag, link, user mention):

   a. να αφαιρέσει τους τόνους, εάν υπάρχουν
   b. να μετατρέψει τα γράμματα σε πεζά
   c. να τα αντιστοιχήσει σε ids και να τα αποθηκεύσει σε ένα αντικείμενο tensor
   d. να αποθηκεύσει την attention mask, επίσης σαν αντικείμενο tensor. Η attention mask είναι απλά μια λίστα από 0 και 1 που δείχνει ποια tokens είναι padding και ποια όχι αντίστοιχα.
   e. να προσθέσει όσες ετικέτες padding χρειάζεται έτσι ώστε όλες οι προτάσεις να είναι ίδιου μήκους

   Για παράδειγμα η πρόταση:

   Έχω 1 μεγάλο πρόβλημα.

   Θα κωδικοποιηθεί ως:

   [101, 576, 124, 552, 649, 121, 102, 0, 0, 0, 0, …]

   , όπου 0 το padding

   Και εάν μετατρέψουμε τα ids σε tokens:

   ['[CLS]', 'εχω', '1', 'μεγαλο', 'προβλημα', '.', '[SEP]', '[PAD]', '[PAD]', …]

   ,όπου [CLS] η αρχή της πρότασης, [SEP] διαχωριστικό για την επόμενη πρόταση και [PAD] το padding που προστέθηκε

   Επίσης η attention mask θα μοιάζει κάπως έτσι:

   [1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0,…]

2. Τα ids και η attention mask, που προέκυψαν από τον tokenizer, δόθηκαν στο μοντέλο Greek Bert με σκοπό να επιστρέψει το last_hidden_state tensor του μοντέλου. Με τη βοήθεια του last_hidden_state tensor, μπορούμε να

μετατρέψουμε τις προτάσεις ενός κειμένου σε vectors και να τις φορτώσουμε σε ένα μοντέλο κατηγοριοποίησης για να εκπαιδευτεί.

4.3.2.3 Μοντέλα Κατηγοριοποίησης

Η κατηγοριοποίηση των tweets βασίζεται σε δυο μοντέλα, ένα SVC, που προβλέπει εάν το tweet είναι ουδέτερο ή όχι και ένα Linear SVC, που προβλέπει εάν είναι αρνητικό ή θετικό.

Τα μοντέλα SVC και LinearSVC αποτελούν modules του **sklearn.SVM** και μπορούν να πραγματοποιήσουν κατηγοριοποιήσεις πολλαπλών κλάσεων. Ενώ και τα δυο αναλαμβάνουν την επίλυση προβλημάτων με vectors, η μεγαλύτερη διαφορά τους είναι στην ταχύτητα εκτέλεσης, αφού ο Linear SVC είναι αρκετά πιο γρήγορος και γι' αυτό χρησιμοποιείται στην κατηγοριοποίηση με το μεγαλύτερο training dataset. Από την άλλη ο SVC φαίνεται να παρουσιάζει καλές επιδόσεις με σχετικά μικρότερου μεγέθους datasets.

Η υλοποίηση του μοντέλου SVC βασίζεται στο libsvm, μια βιβλιοθήκη μηχανικής μάθησης. Ο χρόνος εκτέλεσης κλιμακώνεται σχεδόν τετραγωνικά με τον αριθμό δειγμάτων, το οποίο καθιστά το μοντέλο μη πρακτικό για datasets με πάνω από 10.000 δείγματα [19].

Το μοντέλο LinearSVC είναι παρόμοιο με το μοντέλο SVC σε συνδυασμό με την υπερπαράμετρο ''kernel = linear'', αλλά η υλοποίηση του βασίζεται στη βιβλιοθήκη liblinear, δίνοντας του μεγαλύτερη ευελιξία στην επιλογή των penalties και των loss functions και καλύτερους χρόνους εκτέλεσης σε μεγάλα datasets [20].

Και τα δυο μοντέλα, αφού εκπαιδεύτηκαν, έγιναν save με τη βοήθεια του pickle σε δυο ξεχωριστά αρχεία και προστέθηκαν στον φάκελο ''tweepyStreamer/data''. Για να χρησιμοποιηθούν από την εφαρμογή, γίνονται load από το settings.py κάθε φορά που ξεκινάει ο server.

4.3.2.4 Προετοιμασία των tweets για την πρόβλεψη στην εφαρμογή

Τα tweets που εισέρχονται στην εφαρμογή σε αληθινό χρόνο, μετά την προ επεξεργασία τους, πρέπει να περάσουν από την ίδια ακριβώς διαδικασία που πέρασαν τα training data, δηλαδή να περάσουν από τον tokenizer και το μοντέλο Greek Bert και έπειτα να δοθούν στη μέθοδο predict του κατάλληλου μοντέλου.

Αφού έχει βρεθεί και η τιμή του πεδίου classPredicted, το tweet πρέπει να αποθηκευτεί στη βάση. Για το σκοπό αυτό δημιουργείται ένα instance της κλάσης *Tweet* του αρχείου *models.py,* στο οποίο δίνονται:

- Το *id* του tweet (string)
- Το *userId* του χρήστη που το δημοσίευσε (string)
- Το αρχικό κείμενο του tweet (string)
- Το επεξεργασμένο κείμενο του tweet, στην περίπτωση που ο χρήστης επέλεξε τη λειτουργία ‘’*Κρύψε τις υβριστικές λέξεις*’’ (string)
- Η κλάση που προβλέφθηκε (int)

Επίσης, το αρχικό κείμενο του tweet και η προβλεπόμενη κλάση του, αποθηκεύονται στο αρχείο allTweets.csv, μαζί με τα παλιότερα tweets, που έχουν περάσει από την εφαρμογή.

Έπειτα καλείται η μέθοδος save() και στη συνέχεια περιμένει να ανακαλυφθεί από το GET request στο endpoint ‘’ajax/getTweets’’ για να εμφανιστεί στο user interface.

4.4 GET Requests στο endpoint ‘’ajax/getTweets’’ & Δυναμικά Στοιχεία του User Interface

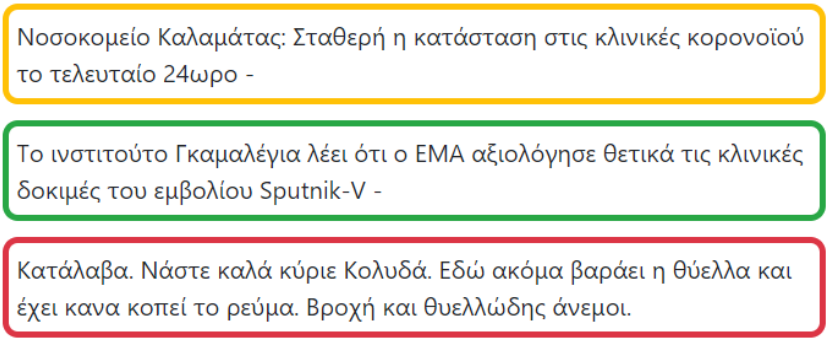
Το endpoint ‘’ajax/getTweets’’ δέχεται GET requests ανά δυο δευτερόλεπτα εν αγνοία του χρήστη, από τη στιγμή που εκείνος ανοίξει το σύνδεσμο http://127.0.0.1:8000/ σε έναν browser. Τα requests αυτά στέλνονται αυτόματα από ένα ajax script, το οποίο σε περίπτωση που το request επιστρέψει επιτυχία, θα διαβάσει το JsonResponse με όλα τα tweets και θα τα προσθέτει ένα-ένα στο table της σελίδας. Με αυτόν τον τρόπο αποφεύγεται το reload όλης της σελίδας κάθε φορά που αποθηκεύονται τα εισερχόμενα tweets και αντ’ αυτού ξαναχτίζεται το table από την αρχή, προσθέτοντας επιπλέον περιεχόμενο. Την ίδια στιγμή, ένα doughnut chart ανανεώνεται δυναμικά, έτσι ώστε ο χρήστης να μπορεί να παρακολουθεί τις ποσότητες των θετικών, αρνητικών και ουδέτερων tweet.

4.4.1 Δυναμικό table

Κάθε δυο δευτερόλεπτα και με κάθε επιτυχημένο GET request στο ‘’ajax/getTweets’’, τα tweets προσθέτονται στην πρώτη σειρά του πίνακα. Με αυτόν τον τρόπο, στην κορυφή του πίνακα φαίνονται πάντα τα πιο πρόσφατα. Στα στοιχεία ‘’div’’ που περιέχουν το περιεχόμενο του πεδίου *cleanText*, δίνεται το κατάλληλο χρώμα στο border και η κατάλληλη κλάση. Δηλαδή:

- Θετικό tweet -> πράσινο border, ‘’positive’’ κλάση
- Ουδέτερο tweet -> κίτρινο border, ‘’neutral’’ κλάση
- Αρνητικό tweet -> κόκκινο border, ‘’negative’’ κλάση

Παρακάτω φαίνονται μερικά σχετικά παραδείγματα:

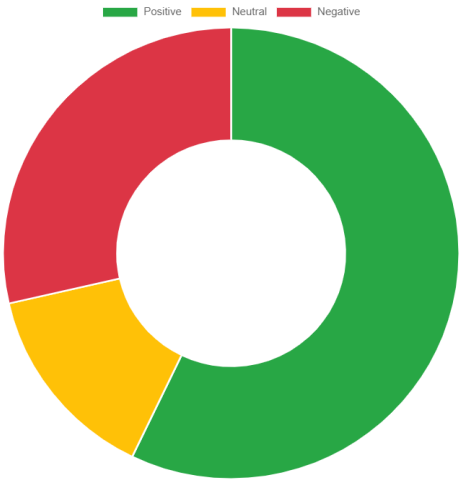


13. Παραδείγματα οπτικοποίησης κατηγοριοποιημένων tweet

4.4.2 Δυναμικό chart

Δίπλα στη στήλη που περιέχει τον δυναμικό πίνακα, τοποθετείται ένα στοιχείο ‘’canvas’’, στο οποίο θα ενσωματωθεί ένα doughnut chart της open source javascript βιβλιοθήκης **chart.js**. Το chart λαμβάνει αρχικές τιμές 0 για όλα τα labels και με τη βοήθεια ενός interval, ανανεώνει τις τιμές του με βάση το πλήθος των στοιχείων κάθε κλάσης. Για την καταμέτρηση των κλάσεων αξιοποιούνται οι κλάσεις που δόθηκαν στα στοιχεία του table όταν αυτά προστέθηκαν στον πίνακα.

Παρακάτω φαίνεται ένα παράδειγμα του chart:



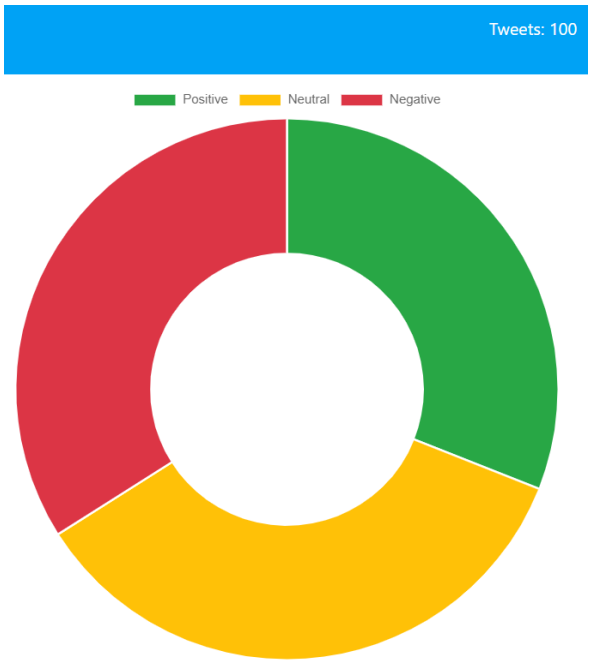
14. Ένα παράδειγμα του chart

ΚΕΦΑΛΑΙΟ 5: Παραδείγματα Χρήσης της Εφαρμογής και Αξιολόγηση Αποτελεσμάτων

Σε αυτό το κεφάλαιο θα παρουσιαστούν τα αποτελέσματα τριών πειραμάτων με την εφαρμογή. Σε κάθε πείραμα θα δοθούν λέξεις, στοχευμένες να φέρουν συγκεκριμένου είδους tweets και αυτά θα τερματίζουν όταν πλέον έχουν μαζευτεί 100 tweets, έτσι ώστε να γίνει ένα manual review των αποτελεσμάτων.

5.1 Πείραμα 1ο: Tweets γενικού περιεχομένου

Για το πρώτο πείραμα εισήχθησαν στην αναζήτηση τα γράμματα «ο» και «η», πράγμα που σημαίνει πως κάθε tweet, που τα περιέχει εντοπίστηκαν από το νήμα streamer.



15. Το chart που προέκυψε από το πρώτο πείραμα

Τα αποτελέσματα της εφαρμογής αξιολογήθηκαν χειροκίνητα και σε κάθε tweet δόθηκε ένα γνώρισμα ‘’Real Label’’. Συγκρίνοντας την αληθινή ετικέτα κάθε tweet με την προβλεπόμενη προέκυψε το παρακάτω confusion matrix:

		Real			
		Neutral	Positive	Negative	Total Predicted
Predicted	Neutral	29	0	6	35
	Positive	3	17	11	31
	Negative	4	4	26	34
Total Real		36	21	43	100

5. Το confusion matrix του 1ου πειράματος

Από το παραπάνω matrix παρατηρείται πως η επιτυχία του πειράματος ήταν (29+17+26) / 100 -> **72%**. Στον παρακάτω πίνακα παρουσιάζονται τα precision, recall και f1 scores της κάθε κλάσης:

		Metrics		
		Precision	Recall	F1
Class	Neutral	80%	80%	80%
	Positive	54%	80%	64%
	Negative	76%	60%	67%

6. Precision, Recall και F1 του πρώτου πειράματος

Από τον πίνακα φαίνεται πως το μοντέλο ουδέτερης – μη ουδέτερης κατηγοριοποίησης λειτουργεί πολύ καλύτερα σε σχέση με το μοντέλο θετικής – αρνητικής κατηγοριοποίησης, αν και ένα μέρος των σφαλμάτων του δεύτερου οφείλονται στο πρώτο. Εάν δεν συνυπολογιστούν τα σφάλματα της ουδέτερης – μη ουδέτερης κατηγοριοποίησης, τότε οι κλάσεις Positive και Negative, πετυχαίνουν τα παρακάτω scores:

		Metrics		
		Precision	Recall	F1
Class	Positive	60%	80%	68%
	Negative	86%	70%	77%

7. Οι πραγματικές επιδόσεις της θετικής-αρνητικής κατηγοριοποίησης στο πρώτο πείραμα

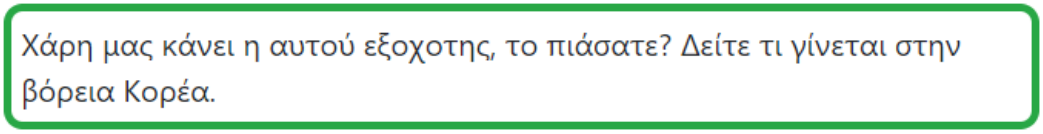
Επίσης, αξίζει να σημειωθεί πως πολλές φορές ο λόγος του Twitter είναι ειρωνικός, με αποτέλεσμα πολλά πραγματικά αρνητικά tweets να θεωρούνται θετικά. Σε αυτό το πείραμα φαίνεται πως 11 αρνητικά tweets κατηγοριοποιήθηκαν ως θετικά. Παρακάτω φαίνονται μερικές αξιοσημείωτες περιπτώσεις σφαλμάτων:

[Γράφει η Στέλλα Σωτήρκου] Κι εσύ εκεί, χαμένος στους δαίδαλους του μυαλού, ψάχνεις να βρεις το φως. Και μπλέκεσαι και σε πνίγει το άγχος. Και οι ιδέες φέρνουν λόγια. Και τα λόγια παρεξηγήσεις... https://t.co/eVG8DjsYj3

16. Ουδέτερο tweet που θεωρήθηκε αρνητικό

Στην εικόνα 16 ο χρήστης που έγραψε το παραπάνω tweet, είχε ως στόχο να παραθέσει ένα κείμενο κάποιας συγγραφέα. Ενώ το tweet θα έπρεπε να κατηγοριοποιηθεί ως ουδέτερο, μιας και δεν εκφέρει άποψη, η εφαρμογή το

θεώρησε αρνητικό, πιθανότατα λόγω των πολλαπλών αρνητικών λέξεων και εκφράσεων.



17. Αρνητικό tweet που κατηγοριοποιήθηκε ως θετικό

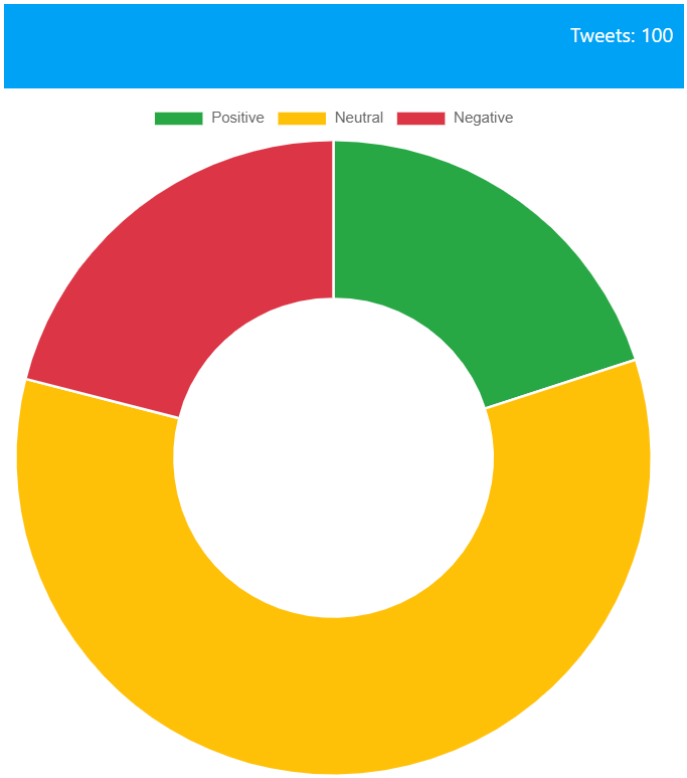
Στην εικόνα 17 ο χρήστης είναι φανερό πως ειρωνεύεται μια κατάσταση στη Βόρεια Κορέα. Το θετικό λεξιλόγιο όμως επηρέασε την κατηγοριοποίηση και το tweet θεωρήθηκε θετικό.

5.2 Πείραμα 2ο: Tweets που σχετίζονται με την επικαιρότητα

Για το δεύτερο πείραμα εισήχθησαν οι λέξεις «εμβόλιο», «Μητσοτάκης», «Τσίπρας», «κακοκαιρία» και «κακοκαιρια». Επιλέγοντας ‘’Σύνθετη Αναζήτηση’’ προέκυψαν επίσης οι όροι:

'εμβόλια', 'εμβολίου', 'εμβόλιου', 'εμβολίων', 'Μητσοτάκηδες', 'Μητσοτάκη', 'Μητσοτάκηδων', 'Τσίπρες', 'Τσιπραίοι', 'Τσίπρα', 'Τσιπραίων', 'Τσιπραίους', 'κακοκαιρίες', 'κακοκαιρίας', 'κακοκαιριών'

Οι όροι αυτοί αφορούν την επικαιρότητα και τα περισσότερα tweets που σχετίζονται με αυτούς είναι ενημερωτικής φύσεως. Όπως και στο πρώτο πείραμα, σε όλα τα tweets δόθηκαν χειροκίνητα οι αληθινές ετικέτες τους, έτσι ώστε να συγκριθούν με τις προβλεπόμενες.



18. Το chart που προέκυψε από το δεύτερο πείραμα

Όπως και στο πρώτο πείραμα, προκύπτει το ακόλουθο confusion matrix:

		Real			
		Neutral	Positive	Negative	Total Predicted
Predicted	Neutral	56	0	3	59
	Positive	2	12	6	20
	Negative	9	1	11	21
Total Real		67	13	20	100

8. Το confusion matrix του δεύτερου πειράματος

Από το παραπάνω matrix παρατηρείται πως η επιτυχία του πειράματος ήταν (56+12+11) / 100 -> **79%**, το οποίο είναι λογικό, αφού στο πρώτο πείραμα φάνηκε η ουδέτερη – μη ουδέτερη κατηγοριοποίηση να έχει καλύτερες επιδόσεις. Στον παρακάτω πίνακα παρουσιάζονται τα precision, recall και f1 scores της κάθε κλάσης:

		Metrics		
		Precision	Recall	F1
Class	Neutral	94%	82%	87%
	Positive	60%	92%	72%
	Negative	52%	55%	53%

9. Precision, Recall και F1 για το δεύτερο πείραμα

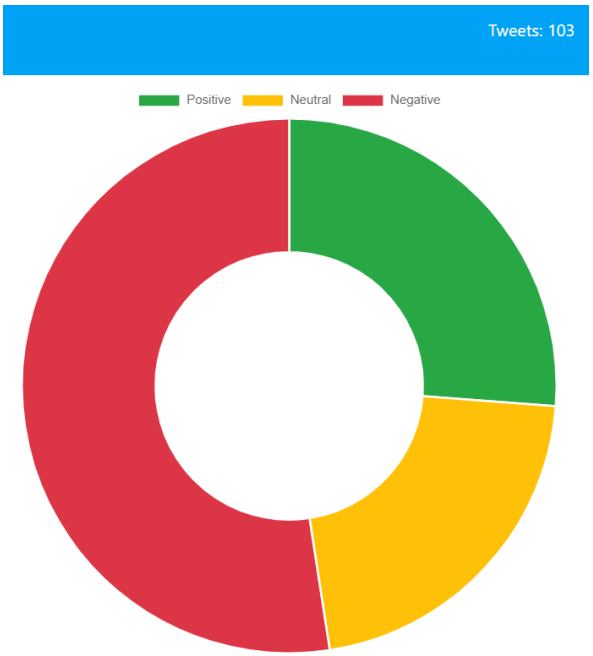
Όμοια με το πρώτο πείραμα, φαίνεται από τους 2 πίνακες, πως τα σφάλματα της ουδέτερης – μη ουδέτερης κατηγοριοποίησης επηρέασαν τις επιδόσεις της θετικής – αρνητικής κατηγοριοποίησης. Γι' αυτό το λόγο στον παρακάτω πίνακα φαίνονται οι πραγματικές επιδόσεις των κλάσεων Positive και Negative, χωρίς να συνυπολογίζονται τα λάθη της πρώτης κατηγοριοποίησης:

		Metrics		
		Precision	Recall	F1
Class	Positive	66%	92%	76%
	Negative	91%	64%	75%

10. Οι πραγματικές επιδόσεις της θετικής - αρνητικής κατηγοριοποίησης στο δεύτερο πείραμα

5.3 Πείραμα 3ο: Tweets που σχετίζονται κυρίως με αρνητικές ειδήσεις

Για το τρίτο πείραμα δόθηκαν οι όροι «Με_τη_Γεωργια», «Γεωργια», «Μυρτω», «xamogelakaipali» και «metoo». Όλοι οι όροι αποτελούν trending όρους του Twitter και συνδέονται με κυρίως αρνητικές απόψεις, αφού αφορούν ένα πολύ δυσάρεστο και σοκαριστικό θέμα.



19. Το chart που προέκυψε από το τρίτο πείραμα

Όμοια με τα προηγούμενα πειράματα, προέκυψε το παρακάτω confusion matrix:

		Real			
		Neutral	Positive	Negative	Total Predicted
Predicted	Neutral	8	4	10	22
	Positive	1	8	18	27
	Negative	1	1	52	54
Total Real		10	13	80	103

11. Το confusion matrix του τρίτου πειράματος

Από το παραπάνω matrix παρατηρείται πως η επιτυχία του πειράματος ήταν (8+8+52) / 103 -> **65%**. Στον παρακάτω πίνακα παρουσιάζονται τα precision, recall και f1 scores της κάθε κλάσης:

		Metrics		
		Precision	Recall	F1
Class	Neutral	36%	80%	49%
	Positive	29%	61%	39%
	Negative	96%	65%	77%

12. Precision, Recall και F1 του τρίτου πειράματος

Σε αυτό το πείραμα φαίνεται να μη βρέθηκαν πολλά αρνητικά tweets, λόγω ειρωνικού λόγου. Οι λέξεις-κλειδιά που δόθηκαν συνδέονται με πολύ αρνητικές ειδήσεις, καθώς και διαμάχες χρηστών. Παρακάτω φαίνονται οι ελαφρώς

αυξημένες επιδόσεις των κλάσεων Positive και Negative, χωρίς τα σφάλματα της Neutral κλάσης:

		Metrics		
		Precision	Recall	F1
Class	Positive	30%	88%	44%
	Negative	98%	74%	84%

13. Οι πραγματικές επιδόσεις της θετικής – αρνητικής κατηγοριοποίησης στο τρίτο πείραμα

ΚΕΦΑΛΑΙΟ 6: Μελλοντικά Βήματα

Μετά από την ενασχόληση με την εφαρμογή, φαίνεται πως υπάρχει χώρος για βελτιστοποιήσεις, αλλά και για επεκτάσεις.

Οι κύριες και πιο σημαντικές βελτιστοποιήσεις για την εμπειρία του χρήστη θα ήταν:

1) **Χρήση offline λεξικού** ή **caching των αποτελεσμάτων** των λέξεων που αναζητούνται στο Greek wictionary.

Ο τρόπος με τον όποιο υλοποιήθηκε η αναζήτηση των λέξεων στο online λεξικό, είναι αρκετά χρονοβόρος, αφού για κάθε λέξη στέλνεται και ένα καινούριο request. Θα ήταν πιο αποδοτική η χρήση ενός offline λεξικού ή να γίνεται caching των παλαιότερων αποτελεσμάτων, έτσι ώστε να γλιτώνονται κάποιες περιττές καθυστερήσεις.

2) **Ενδυνάμωση του κώδικα εύρεσης υβριστικών λέξεων**

Τα κριτήρια που ορίζουν εάν μια λέξη είναι υβριστική ή όχι φαίνεται να μην επαρκούν και επίσης, προς το παρόν, τα hashtags, που πολλές φορές περιέχουν υβριστικές φράσεις, δεν ελέγχονται καθόλου. Αυτή η μικρο-εφαρμογή θα μπορούσε να ενδυναμωθεί με πολλαπλασιασμό των κριτηρίων απόφασης ή με χρήση μιας έτοιμης λίστας υβριστικών λέξεων και των ρίζων τους.

3) **Αύξηση της απόδοσης των μοντέλων κατηγοριοποίησης**

Ένα μεγάλο πρόβλημα που προέκυψε κατά την έρευνα για τον τρόπο κατηγοριοποίησης ήταν η μη ύπαρξη αρκετών ελληνικών datasets για σκοπούς sentiment analysis. Με μεγαλύτερα datasets θα μπορούσαν να αυξηθούν σημαντικά οι αποδόσεις των κατηγοριοποιητών. Ένας τρόπος θα

ήταν η χειροκίνητη συλλογή και ετικετοποίηση tweets, πράγμα όμως που θα χρειαζόταν πολύ χρόνο

Ταυτόχρονα, παρουσιάζονται κάποιες επεκτάσεις που θα μπορούσαν να προσθέσουν επιπλέον δυνατότητες στην εφαρμογή:

1) **Εξαγωγή των αποτελεσμάτων της εφαρμογής σε αρχείο**

   Θα ήταν χρήσιμο για ερευνητές και άλλους ενδιαφερόμενους, να υπάρχει ένα κουμπί εξαγωγής των δεδομένων σε μορφή csv για εύκολη συλλογή δεδομένων.

2) **Δυναμική εκπαίδευση του μοντέλου**

   Υποθέτοντας πως η συνολική απόδοση της κατηγοριοποίησης είναι υψηλότερη, θα είχε νόημα η δυναμική εκπαίδευση των μοντέλων. Αλλά ακόμα και με χαμηλότερες επιδόσεις, χρησιμοποιώντας το feedback του χρήστη, θα μπορούσε να επιτευχθεί αυτό. Για παράδειγμα, στο user interface, πάνω από κάθε εμφανιζόμενο tweet, θα μπορούσαν να υπάρχουν δυο κουμπιά, που το ένα βρίσκει το χρήστη σύμφωνο με το αποτέλεσμα της κατηγοριοποίησης και το άλλο όχι.

ΚΕΦΑΛΑΙΟ 7: Αποθετήριο Κώδικα

Ο κώδικας της εφαρμογής βρίσκεται στον παρακάτω σύνδεσμο:

https://github.com/MaryVou/Analysis-and-Visualization-of-Twitter-Data-Streams-in-Greek

Σημείωση: στον φάκελο *datasets* μπορούν να βρεθούν τα τελικά datasets που χρησιμοποιήθηκαν για τους δυο κατηγοριοποιητές, με την ελπίδα ότι θα βοηθήσουν άλλους ερευνητές σε έρευνες για sentiment analysis.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1 *Python*. (2022). Retrieved from wikipedia.org: https://el.wikipedia.org/wiki/Python

2 *Greek Bert*. (2022). Retrieved from github.com: https://github.com/nlpaueb/greek-bert

3 Koutsikakis, J., Chalkidis, I., Malakasiotis, P., & Androutsopoulos I. (2020). *GREEK-BERT: The Greeks visiting Sesame Street.* Retrieved from doi.org: https://doi.org/10.1145/3411408.3411440

4 *Part-Of-Speech Tagging*. (2022). Retrieved from wikipedia.org: https://en.wikipedia.org/wiki/Part-of-speech_tagging

5 *Named-entity-recognition*. (2022). Retrieved from wikipedia.org: https://en.wikipedia.org/wiki/Named-entity_recognition

6 *Natural Language Inference*. (2022). Retrieved from paperswithcode.com: https://paperswithcode.com/task/natural-language-inference

7 *What is SQLite?* (2022). Retrieved from sqlite.org: https://www.sqlite.org/index.html

8 *Database Setup*. (2022). Retrieved from docs.djangoproject.com: https://docs.djangoproject.com/el/4.0/intro/tutorial02/

9 *HTML*. (2022). Retrieved from wikipedia.org: https://el.wikipedia.org/wiki/HTML

10 *CSS*. (2022). Retrieved from wikipedia.org: https://el.wikipedia.org/wiki/CSS

11 *Javascript*. (2022). Retrieved from wikipedia.org: https://el.wikipedia.org/wiki/JavaScript

12 Fragkis, N. (2021). *Athinorama Greek Movie Reviews, Sentiment Analysis*. Retrieved from kaggle.com: https://www.kaggle.com/nikosfragkis/greek-movies-dataset

13 Tzana, M. (2021). *Greek labeled COVID-19 vaccine tweets*. Retrieved from kaggle.com: https://www.kaggle.com/minatzana/greek-labeled-covid19-tweets

14 Fragkis, N. (2021). *Skroutz Shops Greek Reviews, Sentiment Analysis*. Retrieved from kaggle.com: https://www.kaggle.com/nikosfragkis/skroutz-shop-reviews-sentiment-analysis

15 Kalamatianos, G., Mallis, D., Symeonidis, S., & Arampatzis, A. (2015). *Sentiment Ratings for Greek Tweets Dataset*. Retrieved from researchgate.net: https://www.researchgate.net/publication/301888705_Sentiment_Ratings_for_Greek_Tweets_Dataset

16 Schinas, E., Papadopoulos, S., Diplaris, S., Kompatsiaris, Y., Mass, Y., Herzig, J., & Boudakidis, L. (2013). *Event-oriented sentiment-annotated Twitter datasets*. Retrieved from mklab.iti.gr: https://mklab.iti.gr/results/event-oriented-sentiment-annotated-twitter-datasets/

17 Pitenis, Z. (2020). *Offensive Greek Tweet Dataset - OGTD*. Retrieved from zpitenis.com: https://zpitenis.com/resources/ogtd/

18 Beleveslis, D., Tjortjis, C., Psaradelis, D., & Nikoglou, D. (2020). *Sentiment Analysis Greek*. Retrieved from github.com: https://github.com/dimosbele/sentiment_analysis_greek

19 *SVC*. (2022). Retrieved from scikit-learn.org: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

20 Scikit-learn.org (Ιανουάριος 2022) LinearSVC [Online] - https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

ΠΑΡΑΡΤΗΜΑ: Οι κλάσεις Thread, StreamListener, Tweet

```
class StreamListener(Stream):
    def on_data(self, data):
        try:
            data = json.loads(data)

            if not data['text'].startswith('RT'):
                greek_chars = [char for char in data['text'] if
char in settings.GREEK_CHARACTERS]
                if len(greek_chars) >= 5:
                    tweet = {}
                    tweet['id'] = data['id_str']
                    tweet['userId'] = data['user']['id_str']
                    if 'extended_tweet' in data:
                        tweet['text'] =
data['extended_tweet']['full_text']
                    else:
                        tweet['text'] = data['text']

                    with
open('tweepyStreamer\\data\\raw_data.txt', 'a', encoding='utf-8')
as f:
                        f.write(str(tweet) + '\n')
            return True
        except Exception as e:
            print('[!] Error: ' + str(e))
            with open('tweepyStreamer\\data\\logs.txt', 'a',
encoding='utf-8') as f:
                f.write('[' + str(datetime.datetime.now()) + ']
[Streamer] ' + str(e) + '\n')

    def on_limit(self, track):
        print('[!] Limit: ' + track)
        sleep(10)

    def on_error(self, status):
        print('[!] Error: ' + str(status))
        return False
```

2. Η κλάση StreamListener

```
class Tweet(models.Model):
    tweetId = models.CharField(max_length=100, primary_key=True)
    userId = models.CharField(max_length=100)
    text = models.CharField(max_length=4000)
    cleanText = models.CharField(max_length=4000)
    classPredicted = models.IntegerField()
```

3. Η κλάση Tweet

```
class Thread(threading.Thread):

    def __init__(self, *args, **kwargs):
        threading.Thread.__init__(self, *args, **kwargs)

    def get_id(self):
        if hasattr(self, '_thread_id'):
            return self._thread_id
        for id, thread in threading._active.items():
            if thread is self:
                return id

    def raise_exception(self):
        thread_id = self.get_id()
        res =
ctypes.pythonapi.PyThreadState_SetAsyncExc(thread_id,

ctypes.py_object(SystemExit))
        if res > 1:
            ctypes.pythonapi.PyThreadState_SetAsyncExc(thread_id,
0)
            print('Killing thread ', self.get_id())
```

4. Η κλάση Thread