Γλωσσική Τεχνολογία

PROJECT 2019-2020

Ελευθεριάδης Πέτρος 6044

ΜΕΡΟΣ Α

Προσκομιστής Ιστοσελίδων.

Για τον προσκομιστη χρησιμοποίησα Scrapy όπως προτείνεται στην εκφώνηση. Ο τρόπος λειτουργίας του είναι ο εξής. Έχω πάρει τις αρχικές σελίδες 5 ειδησεογραφικές ιστοσελίδες. Κάθε μια από αυτές, στις αρχικές τους, έχουν 5-10 αρθρα. Οπότε ακολουθάω το link από αυτά τα 5-10 άρθρα για κάθε μια ιστοσελίδα και παίρνω το title, author και text από αυτά. Έτσι, στο τέλος έχω γύρω στα 50 documents στη βάση.

Για τη βάση δεδομένων χρησιμοποίησα τη MongoDB και το pymongo πακέτο. Δημιουργείται στο pipeline του scrapy και περιέχει title, author, text και website. Το website είναι ένα στοιχείο που απλα περιγράφει σε ποια από τις 5 σελίδες ανήκει το άρθρο. Όσο για το id, το αφήνω στη MongoDB να φτιάξει μόνη της.

Προεπεξεργασία δεδομένων

Αρχικά δημιουργώ ένα πεδίο “myid” στη βάση το οποίο με βοηθάει να παίρνω κάθε φορά διαφορετικό document αλλάζοντας το κάθε φορά που επεξεργάζεται ένα document.

Παίρνω το πλήθος των documents στη βάση και φτιάχνω ένα loop μέσα στο οποίο γίνεται το εξής:

(Τα 2 παραπάνω βήματα τα κάνω σε κάθε υποερωτημα, οπότε αντί να επαναλαμβάνω θα περιγράφω τι γίνεται μέσα στο loop)

Παίρνω το text και το id από ένα document από τη βάση.

Μετατρέπω τη λίστα του text σε string

Καθαρίζω τα html tags από το text

Καθαρίζω τα σημεία στίξης επειδή είναι άχρηστα

Με το id στέλνω το ανανεωμένο text στη βαση

Για το καθαρισμό των html tagsχρησιμοποίησα το BeautifulSoup με parser lxml

Μορφοσυντακτική Ανάλυση

Παίρνω το text και το id από ένα document από τη βαση.

Κάνω tokenize το text με τη χρήση του NLTK

Προσθέτω τα postags

Γυρνάω στη βαση ένα καινούριο πεδίο το οποίο περιέχει 2 στοιχεία αν θέση. Το token και το postag του

Αναπαράσταση ιστοσελίδων στο Μοντέλο Διανυσματικού Χώρου.

Το συγκεκριμένο υποερωτημα εγινε σε 2 κώδικες. Το πρώτο αφαιρεί τα stopwords και το δεύτερο βρίσκει τις συχνότητες των λέξεων.

Stopwords// stopwords.py

Παίρνω τα tokens από τη βάση.

Έχω μια λίστα από stopwords από την NLTK συνάρτηση `stopwords.words('english')` και συγκρίνω κάθε token (χωρις κεφαλαια) με τη λίστα αυτή. Αν είναι ίδια τότε τη διαγράφω, αν δεν είναι σημαίνει πως δεν είναι stopword, άρα παραμένει.

Έπειτα αντικαθιστώ κάθε token που έμεινε, με το lemma του με μικρά γράμματα, παίρνοντας βοήθεια από το `wordnet` για τα σωστά postags.

Τέλος ανανεώνω τη βάση

Συχνότητα λήμματος // lemmatize.py

Φτιάχνω 2 λίστες. Στη πρώτη θα μπούνε τα μοναδικά λήμματα και στη δεύτερη θα μπούνε οι συχνότητες τους

Παίρνω το πρώτο token. Το βάζω στη λίστα. Μετα κάνω άλλο ένα loop στο οποίο συγκρίνω το token με όλα τα αλλα. Όσες φορές η σύγκριση βγει true, τόση είναι η συχνότητα του και τη βάζω στη δεύτερη λίστα. Παρόμοια για όλα τα υπόλοιπα tokens

Στο τέλος φτιάχνω ένα dict που ενώνω τις δυο λίστες και το ανεβάζω στη βάση ως Word Frequency

Δημιουργία του ευρετηρίου

Αρχικά φτιάχνω ένα ενιαίο dict Total Word Frequency. Δηλαδή ένα dict που περιέχει κάθε μοναδική λέξη από όλα τα documents. Οι τιμες συχντοτητων δε μας ενδιαφέρουν ακόμα. Θα τις φτιάξουμε στη συνέχεια

Φτιάχνω ένα loop για κάθε λέξη σε αυτό το ενιαίο dict.

Έπειτα ένα loop για κάθε document

Αν η λέξη είναι στο document, παίρνουμε τη συχνότητα του από το Word Frequency του document. Επίσης κρατάμε το id του document σε μια λιστα ώστε να μετρήσουμε πόσα documents περιλαμβάνουν τη λέξη για βρούμε το idf value

Μετα το τέλος του loop αυτού, έχουμε τη συχνότητα και τον αριθμό των κειμένων οπότε εύκολα βρίσκουμε το tfidf

Συνεχίζουμε όμως και είμαστε στο 1ο loop και μέσα σε αυτό θα φτιάξουμε και το xml. Αφού έχουμε σε λίστες τα tfidf και τα id, φτιάχνουμε ένα lemma element και με μια for loop για τη συγκεκριμένη λέξη βάζουμε τα tfidf και τα id.

Έτσι φτιάχνεται και όλο το ανεστραμμένο ευρετήριο

Τέλος το εκτυπώνουμε σε ένα xml αρχείο.

Για το xml χρησιμοποίησα το `xml.etree.ElementTree και το minidom για prettyxml`

Αποθήκευση και επαναφόρτωση ευρετηρίου.

Στο προηγούμενο υποερωτημα έγινε και η αποθήκευση του ευρετηρίου σε xml αρχείο με όνομα `inverted_index.xml`

Εδώ θα περιγράψω το κώδικα που κάνει append ένα xml του ιδιου τύπου στο υπάρχον.

Με τη βοήθεια του `xml.etree.ElementTree παλι, κάνω parse το inverted_index.xml` που είναι η βάση και επίσης το `input.xml` που θα είναι το νέο xml το οποίο θα γίνει append στο 1ο

Με μια απλή for loop για κάθε παιδί τη ρίζας του νέου xml, κάνω append στη ριζα του βασικού xml και αποθηκεύεται στο `inverted_index.xml`

Σημείωση: Για να κάνετε append ένα νέο xml, πρέπει να το ονομάσετε `input.xml.` Επίσης ως βάση θεωρείται το `inverted_index.xml`

Αξιολόγηση ευρετηρίου.

Το σύστημα δέχεται ως είσοδο μια λίστα από λέξεις.

Για κάθε λέξη στη λίστα ψάχνει κάθε παιδί της ρίζας του xml και όταν βρει attribute lemma = name, το όνομα της λέξης που ψάχνουμε, τότε βάζουμε σε μια λίστα το id και σε άλλη το weight.

Σε περίπτωση που ψάχνουμε πάνω από μια λέξη, και οι 2 η παραπάνω λέξεις βρίσκονται σε ίδιο document, τότε αντί να ξαναβάλουμε το id kai το weight στις λίστες, βρίσκουμε τη θέση του id στην υπάρχουσα λίστα και προσθέτουμε σε αυτή τη θέση το weight που βρήκαμε τώρα.

Τέλος κάνουμε sort τα weights και τα ids(ανάλογα με τα weights) και τα ενώνουμε σε ένα dict

Μετα από μετρήσεις βρήκα ότι ο συνολικός χρόνος σε 100 ερωτήματα ήταν *4.034247159957886 sec*

Αν διαιρέσω προς 100, τότε ο μέσος χρόνος απόκρισης είναι **0.04034247159957886 se**c

Με περισσότερη λεπτομέρεια τα αποτελέσματα κάθε ερώτησης βρίσκονται στο αρχείο results_of_querries.txt

Σημείωση*: Για να δουλέψει το σύστημα σωστά οι κώδικες πρέπει να τρέξουν με τη παρακάτω σειρά*

1. Clear_html.py
2. Postag.py
3. Stopwords.py
4. Lemmatize.py
5. Inverted.py

Έπειτα κάνουμε όσα queries ή appends θέλουμε