ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

Ανάκτηση Πληροφορίας

Εργαστηριακή Άσκηση

Χειμερινό Εξάμηνο 2023

Διδάσκων: Χ. Μακρής

Συντάκτης :

Ον/μο	Μίρα Ισλαμάϊ
ΑΜ	1070736
Email	miraslamaj@gmail.com ή st1070736@ceid.webmail.gr

Τμήμα Μηχανικών Η/Υ & Πληροφορικής

Πανεπιστήμιο Πατρών

Περιβάλλον Υλοποίησης – Εγκατάσταση – Υλικό

ΠΕΡΙΒΑΛΛΟΝ ΥΛΟΠΟΙΗΣΗΣ

Η εργασία αυτή δημιουργήθηκε , σε περιβάλλον που τρέχει το λειτουργικό σύστημα Ubuntu 22.04.3 LTS (64-bit). Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε είναι η Python 3.10.12 και η υλοποίηση πραγματοποιήθηκε σε ένα εικονικό περιβάλλον που ονομάστηκε "ir". Το εργαλείο διαχείρισης εικονικού περιβάλλοντος είναι το venv. Τέλος, ο κώδικας αναπτύχθηκε μέσω του επεξεργαστή κειμένου (IDE) Visual studio Code.

ΕΓΚΑΤΑΣΤΑΣΗ

Δημιουργία εικονικού περιβάλλοντος:

Δημιουργούμε αρχικά το εικονικό περιβάλλον με το ενσωματωμένο εργαλείο venv, ενώ βρισκόμαστε μέσα στο φάκελο της εργασίας της εργασίας μας. (Μπορείτε προφανώς να αντικαταστήσετε την ονομασία “ir” , με οποιαδήποτε άλλη θελήσετε να δώσετε στο εικονικό περιβάλλον)

```
python3 -m venv ir          //δημιουργία
source ir/bin/activate      //ενεργοποίηση
```

Εγκατάσταση Εξαρτημένων Βιβλιοθηκών python:

εγκαταστάθηκαν βιβλιοθήκες python “pandas”, “nltk”, “word2number”, ”num2words” και “matplotlib”

```
pip install pandas nltk word2number num2words matplotlib
```

Υλοποίηση ColBERT:

Για την χρήση του ColBERT, ακολουθήθηκαν η ίδια λογική των οδηγιών από το Google colab: ColBERT Intro

Συγκεκριμένα:

```
git clone https://github.com/stanford-futuredata/ColBERT.git
pip install -e ColBERT/['faiss-gpu','torch']
```

ΠΛΗΡΟΦΟΡΊΕΣ ΣΥΣΚΕΥΉΣ (HARDWARE INFO)

- RAM: 16,0 GB
- Επεξεργαστής: 12th Gen Intel® Core™ i5-1235U × 12

- Γραφικά: Mesa Intel® Graphics (ADL GT2)
- Χωρητικότητα δίσκου: 512,1 GB (SSD)

ΛΟΙΠΕΣ ΠΛΗΡΟΦΟΡΙΕΣ

Η εργασία επίσης υπάρχει και στο GitHub :
https://github.com/miraq3bnkc/Info_retrieval

Οι φάκελοι που αγνοούνται από το .gitignore και δεν υπάρχουν στο GitHub είναι ο "ColBERT/" που περιέχει αντίγραφο αυτού το κώδικα : https://github.com/stanford-futuredata/ColBERT και ο "ir/" που περιλαμβάνει τα πακέτα του περιβάλλοντος υλοποίησης.

Ερώτημα 1 - Ανάγνωση και Επεξεργασία της Συλλογής

Για την ανάγνωση και την επεξεργασία της συλλογής , με την βιβλιοθήκη της python "os" διαβάζουμε όλα τα αρχεία στον φάκελο "docs", τον φάκελο αυτόν βρίσκεται η συλλογή των κειμένων. Γνωρίζοντας ότι αυτός ο φάκελος έχει μόνο .txt αρχεία και το πως είναι γραμμένα μέσα αυτά τα κείμενα, με ένα split() σε κάθε αρχείο πήραμε όλες τις λέξεις που υπάρχουν στην συλλογή (λεξιλογική ανάλυση). Με αυτή την βάση μπορούμε στην συνέχεια να φτιάξουμε το ανεστραμμένο ευρετήριο.

Η βιβλιοθήκη pandas της python χρησιμοποιήθηκε για την αποθήκευση αρχικά του πλήρους ανεστραμμένου αρχείου και στην συνέχεια για την εξαγωγή του σε .csv αρχείο , ονομαζόμενο inverted_index.csv. Το πλήρες ανεστραμμένο ευρετήριο , έχει σχεδιαστεί με τέτοιο τρόπο ούτως ώστε να έχουμε την θέση της λέξης σε κάθε κείμενο (word position) , αντί για την θέση του χαρακτήρα. Το αρχείο έχει την μορφή ,που βλέπουμε στο παρακάτω παράδειγμα για την λέξη "ANGLES" που εμφανίζεται στην συλλογή C.F. ,όπως διδαχτήκαμε στις διαλέξεις.

ΠΑΡΑΔΕΙΓΜΑ 1

ANGLES	[['00486', 3, [37, 90, 108]], ['00499', 1, [77]]]

Θέση της λέξης στο κείμενο

πλήθος εμφανίσεων στο ίδιο αρχείο

Όνομα αρχείου εμφάνισης της λέξης "angles"

Σύμφωνα με τις διαφάνειες του τρίτου φροντιστηρίου 2023-2024 , η προ-επεξεργασία εγγράφων μπορεί να διαχωριστεί σε τέσσερις κατηγορίες. Μία από αυτές είναι η λεξιλογική ανάλυση , που ήδη αναφέρθηκε παραπάνω, που εφόσον δεν υπήρχαν σημεία στίξης , σύμβολα και ήταν ήδη όλα σε κεφαλαία , δεν χρειάστηκε ιδιαίτερη επεξεργασία. Η κατηγορία της εξάλειψης stop words , στην οποία υπάρχουν περιπτώσεις όπου η αγνόησή των "stop words" μπορεί να οδηγήσει σε απώλεια σημασίας, όπως σε αναζητήσεις φράσεων ή σε ειδικού τύπου ερωτήματα ([2] σελίδα 27). Στην συνέχεια, δόθηκε μεγαλύτερη σημασία στην χρήση ενός stemmer, συγκεκριμένα του Porter2 ή αλλιώς English stemmer.

Ο λόγος που θελήσαμε να χρησιμοποιήσουμε έναν stemmer , είναι διότι παρατηρήθηκε στις λέξεις να υπάρχουν πολλές ίδιες λέξεις και σε ενικό αριθμό αλλά και σε πληθυντικό αριθμό, για παράδειγμα “angles” και “angle”. Ένας stemmer όπως ο Porter2 , μπορεί να θέσει έναν όρο δεικτοδότησης για τέτοιες λέξεις και τελικά να μειώσει τον χώρο πολυπλοκότητας. Ο Porter2 είναι μια βελτίωση του διαδεδομένου αλγορίθμου Porter ([1]). Για την εκτέλεση του stemmer , χρησιμοποιήθηκε η βιβλιοθήκη nltk, συγκεκριμένα η nltk.stem όπου έχει μια έτοιμη συνάρτηση. [3]

Με την χρήση του Porter2 stemmer , το ανεστραμμένο αρχείο μειώθηκε κατά περίπου 26%. Πριν δώσουμε σημασία στις καταλήξεις, το inverted_index.csv αρχείο περιείχε 11.368 όρους , ενώ μετά την χρήση του stemmer έχουμε 8.373 όρους δεικτοδότησης (με την δοκιμή του απλού Porter αλγορίθμου είχαμε 8.436 όρους, 63 περισσοτέρους από τον Porter2).

Παρατηρήθηκε κατά την επεξεργασία των συλλογών ότι υπάρχει ορθογραφικό λάθος , όπως για παράδειγμα στο αρχείο “00211” υπάρχει η λέξη “0RONCHIAL” , που δεν σημαίνει τίποτα, αντί για την λέξη “BRONCHIAL”. Δεν διαπιστώθηκε κάπου αλλού ορθογραφικό, και για αυτό τον λόγο δεν έγινε κάποια ενέργεια για διόρθωση ορθογραφικών, αλλά διαφορετικά θα μπορούσε να είχε χρησιμοποιηθεί η βιβλιοθήκη “pyspellchecker” [4].

Παρατηρήθηκε, επίσης, πως υπάρχουν ακόμα διαφορετικές αναπαραστάσεις σε λέξεις που δεν μπόρεσε ο αλγόριθμος του Porter να επεξεργαστεί , για παράδειγμα “1H” με “1HOUR” ή “FIFTYTWO” με “52”, καθώς δεν διαχειρίζεται λέξεις με αριθμούς. Ένας τρόπος για να λυθούν μερικά από αυτά τα περιστατικά είναι να μετατρέψουμε όλους τους αριθμούς στην ίδια αναπαράσταση. Οπότε λέξεις όπως “ONE”, “TWO”, “FIFTYTWO” και άλλες πολλές θα μετατραπούν στο αντίστοιχο τους νούμερο “1”, “2”, “52”. Αυτό επιτυγχάνεται στο αρχείο “word_conversion.py”, όπου γίνεται χρήση των βιβλιοθηκών “word2number” και “num2words”. Άλλη μία αλλαγή που γίνεται στο αρχείο word_conversion.py είναι οι λέξεις “1st”, “2nd” και άλλα να αντικαθιστηθούν από “first”, “second” και ούτω καθεξής.

Η παραπάνω λογική μείωσε τους όρους δεικτοδότησης κατά 156 (-1,8%) όρους δεικτοδότησης, και τώρα έχουμε συνολικά 8217 όρους. Ομολογουμένως , δεν είναι κάποια αισθητή μείωση , αλλά είναι μια αρκετά σημαντική αλλαγή.

Ενδιαφέρον είναι να αναφερθεί πως σε αυτό το σημείο έγινε μία δόκιμη για την εξάλειψη των stop word. Η διεργασία αυτή εκτελέστηκε με την submodule nltk.corpus [5]. Καταγράφηκε μείωση μόνο κατά 110 (-1,3%) λέξεις , οπότε έχουμε σύνολο 8107 όρους. Η πιθανότητα απώλειας σημασίας δεν μας φοβίζει, καθώς το ανεστραμμένο ευρετήριο θα χρησιμοποιηθεί για το μοντέλο διανυσματικού χώρου.

Όπως αναφέρθηκε προηγουμένως , υπάρχουν προβλήματα διαφορετικής αναπαράστασης ίδιων λέξεων. Στις περιπτώσεις που είχαμε αριθμούς και ολογράφως αλλά και με αριθμητική αναπαράσταση , το πρόβλημα λύθηκε , όμως, για τις άλλες περιπτώσεις είναι πιο πολύπλοκα τα πράγματα. Έχουμε λέξεις “1HOUR”, “1H” ,”ONEHOUR” και”1SEC”, “ONESEC” ,“ONESECOND” κλπ όπου προφανώς θα έπρεπε να αντιστοιχίζονται στην ίδια λέξη στο ανεστραμμένο ευρετήριό μας. Μία πρώτη σκέψη ήταν να χωρίσουμε τους αριθμούς από τα γράμματα , δηλαδή να χωρίζαμε το “1HOUR” σε “1” και “HOUR” , όμως υπάρχουν όροι δεικτοδότησης όπως “111IN” , “14CCHOLYGLYCINE”, “23F”, “11A”, “1ACYL2N4NITROBENZO2OXA13DIAZOLEAMINOCAPROYL” και άλλα ,που καθώς η βιολογία δεν είναι η κατάρτισή μου , δεν ξέρω αν θα ήταν καλό να χωριστούν.

ΠΑΡΑΤΗΡΗΣΕΙΣ

Θα μπορούσε να είχε ληφθεί υπόψιν η μορφή των ερωτήσεων , για την προ-επεξεργασία της συλλογής κειμένων. Οι ερωτήσεις που θα γίνουν είναι αυτές που υπάρχουν στο αρχείο “Queries_20”. Παρατηρείτε ότι δεν υπάρχουν αριθμοί στις ερωτήσεις, ούτε γραμμένες ολογραφώς ούτε με αριθμητική αναπαράσταση (πέρα από το “one” στην 2η ερώτηση , μόνο που εδώ έχει την σημασία του “κάποιο άτομο”). Συμπεραίνουμε, λοιπόν, πως θα μπορούσαμε να αφαιρέσουμε όλες τις λέξεις με αριθμούς από το ανεστραμμένο ευρετήριό μας. Αυτό θα έλυνε αρκετά προβλήματα διαφορετικής αναπαράστασης αλλά και θα μείωνε τους όρους ευρετηρίασης. Ωστόσο , για την δημιουργία του ανεστραμμένου ευρετηρίου δεν έγινε κάποια τέτοια επεξεργασία.

Γενικότερα η συλλογή μας περιέχει πολλές ιδιαιτερότητες. Έχουμε ορθογραφικά, πολυσημίες, συνωνυμίες, πολλές λέξεις είναι ενωμένες μαζί (πχ “512monthold” αντί για “5-12-month-old”). Η συλλογή , λοιπόν θα μπορούσε να είχε επεξεργαστεί πολύ παραπάνω για να λυθούν αυτά τα προβλήματα. Με μία πολύ καλύτερη και λεπτομερή προ-επεξεργασία θα είχαμε καλύτερα αποτελέσματα με το Vector Space Model.

ΥΛΟΠΟΙΗΣΗ ΕΡΩΤΗΜΑΤΟΣ 1

Για την δημιουργία του ανεστραμμένου αρχείου όπως ζητείτε από το 1ο ερώτημα , έχουν δημιουργηθεί τα εξής αρχεία κώδικα:

- docs_inverted_index.py
- inverted_index.py
- word_conversion.py

Το αρχείο που εκτελούμε για την δημιουργία του ανεστραμμένου ευρετηρίου (inverted_index.csv) είναι το docs_inverted_index.py. Τα άλλα δύο αρχεία περιέχουν

απλά συναρτήσεις που χρησιμοποιούνται από την docs_inverted_index.py. Συγκεκριμένα, το inverted_index.py περιέχει συναρτήσεις για την δημιουργία ανεστραμμένων ευρετηρίων, καθώς θα χρειαστεί στην συνέχεια να φτιαχτεί ενα ανεστραμμένο αρχείο και για τα query. Το word_conversion αναφέρθηκε και παραπάνω.

Ερώτημα 2 – Δημιουργία Διανυσματικού Χώρου (VSM)

Για την δημιουργία Διανυσματικού χώρου (Vector Space Model) , θα πρέπει να υπολογιστούν τα κατάλληλα βάρη , term frequency (TF) και inverse document frequency (IDF).

Τα τρία συνιστώμενα TF-IDF σχήματα που προτείνει ο Salton, παρουσιάζονται στο Figure 1 . Από αυτά το 1ο προτεινόμενο σχήμα βαρών για την επίλυση της εργασίας. Με άλλα λόγια, έχουμε απλή συχνότητα εμφάνισης και απλή ανάστροφη συχνότητα εμφάνισης για τα βάρη ορών εγγράφων, ενώ για τα βάρη ορών ερωτημάτων έχουμε διπλή 0,5 κανονικοποίηση στο TF και απλή ανάστροφη συχνότητα εμφάνισης, όπως προηγουμένως.(σελίδα 79-81 [7])

weighting scheme	document term weight	query term weight
1	$f_{i,j} * \log \frac{N}{n_i}$	$(0.5 + 0.5\frac{f_{i,q}}{max_i\ f_{i,q}}) * \log \frac{N}{n_i}$
2	$1 + \log f_{i,j}$	$\log(1 + \frac{N}{n_i})$
3	$(1 + \log f_{i,j}) * \log \frac{N}{n_i}$	$(1 + \log f_{i,q}) * \log \frac{N}{n_i}$

Figure 1:Προτεινόμενα συστήματα στάθμισης TF-IDF βαρών (από διαφάνειες:[8] σελ. 46)

Επιλέχθηκε αυτός ο υπολογισμός της συχνότητας εμφάνισης ορών στα έγγραφα, λόγο της ευκολίας υπολογισμού του. Παρόλ'αυτά για τον υπολογισμό της συχνότητας εμφάνισης ορών στα έγγραφα, θα ήταν καλό να είχε γίνει μία κανονικοποίηση, καθώς σε έγγραφα μεγάλου μήκους, οι όροι που εμφανίζονται μπορεί να έχουν μεγαλύτερο βάρος, διότι αυξάνεται η πιθανότητα ύπαρξής τους. Η κανονικοποίηση αυτή θα μπορούσε να είναι η διαίρεση του f_{ij} με την μέγιστη συχνότητα όρου στο ίδιο έγγραφο :
f(i,j) = freq(i,j) / max(freq(l,j)) . ([6] σελίδα 77-78)

Άλλη μια κανονικοποίηση που θα μπορούσε να γίνει θα μπορούσε να είναι με η διαίρεση με το μήκος του κάθε εγγράφου : f(i,j) = freq(i,j) / number of terms in doc . [9]

Ο πίνακας TF-IDF που θα δημιουργηθεί για τα έγγραφα θα είναι τεράστιος, (1209 έγγραφα) * (8107 όρους) ! Οι υπολογισμοί που θα χρειαστούν για την δημιουργία αυτού το πίνακα θα είναι χιλιάδες και πολλοί από αυτοί μάλιστα θα είναι απλά μηδενικά. Η πολυπλοκότητα λοιπόν είναι αρκετά μεγάλη. Μία ιδέα για την μείωση του χώρου και χρόνου υλοποίησης ήταν να χρησιμοποιηθούν αραιά μητρώα , η ιδέα όμως που τελικώς επικράτησε ήταν να δημιουργήσουμε ένα TF-IDF πίνακα μόνο για τους όρους που μας ενδιαφέρουν. Αυτό που εννοούμε με αυτό είναι να υπολογίσουμε τις τιμές TF και IDF μόνο για τους όρους που υπάρχουν και στα έγγραφα και στα ερωτήματα, δηλαδή φιλτράρουμε το ανεστραμμένο ευρετήριο των εγγράφων όπου κρατάμε μόνο την πληροφορία για τους όρους που υπάρχουν στα ερωτήματα (και αντιστρόφως για το ανεστραμμένο ευρετήριο ερωτημάτων). Αυτή η υλοποίηση μειώνει κατά πολύ τους υπολογισμούς μας καθώς πλέον ο πίνακας TF-IDF που θα δημιουργηθεί θα είναι το πολύ 1209 εγγραφα * 84 όρους , εφόσον 84 είναι όλοι οι όροι σύμφωνα με το ανεστραμμένο ευρετήριο των ερωτημάτων “queries.csv” (το TF-IDF των ερωτημάτων θα είναι το πολύ 20 ερωτήματα * 84 όρους).

Θα ήταν πάλι καλή ιδέα να χρησιμοποιηθούν και σε αυτό το σημείο τα αραιά μητρώα, καθώς και πάλι θα υπάρχουν αρκετά μηδενικά στους υπολογισμούς μας και η ύπαρξή τους θα μείωνε την πολυπλοκότητά του κωδικά μας.

Ο λόγος που έχουμε την δυνατότητα να αφαιρέσουμε έναν τόσο τεράστιο όγκο ορών , είναι επειδή στο Μοντέλο Διανυσματικού Χώρου (VSM) οι όροι ευρετηρίασης είναι αμοιβαία ανεξάρτητοι μεταξύ τους. ([7] σελίδα 85) και ούτε χρησιμοποιήθηκε κάποιος θησαυρός για τα συνώνυμα.

ΥΛΟΠΟΙΗΣΗ

Για την υλοποίηση αυτού του ερωτήματος έχουμε ως είσοδο το πλήρη ανεστραμμένο ευρετήριο των εγγράφων που δημιουργήθηκε στο 1ο ερώτημα (“inverted_index.py”), όμως θα χρειαστεί ένα ανεστραμμένο ευρετήριο και για τα ερωτήματα για να μπορέσουμε να υπολογίσουμε το βαθμό ομοιότητας εγγράφου-ερωτήματος. Το ανεστραμμένο ευρετήριο αυτό δημιουργείται με την εκτέλεση του queries_proprocess.py όπου εξάγει το αρχείο που θα φέρουμε ως είσοδο “queries.csv”. Στο queries_preprocess.py εκτελούνται οι ίδιες συναρτήσεις όπως και στο docs_inverted_index.py. Η μόνη διαφορά είναι πως για τα ερωτήματα δεν δημιουργήθηκε ένα πλήρες ανεστραμμένο ευρετήριο, καθώς δεν μας ενδιαφέρει η θέση των όρων σε αυτό το ερώτημα.

Στην συνέχεια εκτελούμε το κώδικα στο αρχείο vsm.py όπου μας δημιουργεί το Διανυσματικό χώρο και εξάγεται το αρχείο “vsm_rankings.csv” που περιέχει σε κατάταξη τους βαθμούς ομοιότητας κάθε ερωτήματος με έγγραφο. Συγκεκριμένα για κάθε ερώτημα έχουν έχουν αποθηκευτεί στο αρχείο “vsm_rankings.csv” τα 100 πιο όμοια έγγραφα σύμφωνα με τους υπολογισμούς μας.

Ερώτημα 3 – Υλοποίηση ColBERT Μοντέλου

Όπως αναφέρθηκε και στην παράγραφο “Περιβάλλον Υλοποίησης- Εγκατάσταση – Υλικό”, για την υλοποίηση του ColBERT μοντέλου ακολουθήθηκαν οδηγίες που δίνονται από την υλοποίηση του Stanford [10][11].

Για να χρησιμοποιήσουμε το προ-εκπαιδευμένο μοντέλο , αρχικά θα πρέπει να μετατρέψουμε σε μία δομή που μπορεί να διαβάσει, τα ερωτήματα και την συλλογή ως εισόδους. Έχουμε αποθηκεύσει λοιπόν το περιεχόμενο όλων των εγγράφων σε μία λίστα , και τα ερωτήματα σε μία άλλη λίστα. Έτσι μπορούμε στην συνέχεια να κάνουμε “Indexing” την συλλογή μας και “Searching” με βάση τα ερωτήματά μας.

ΠΑΡΑΤΗΡΗΣΕΙΣ

Καθώς το προ-εκπαιδευμένο μοντέλο που χρησιμοποιούμε έχει εκπαιδευτεί με μία συλλογή (MS MACRO) η οποία περιέχει διαφορετικά δεδομένα από αυτά της συλλογής μας, που είναι εξειδικευμένα για Cystic Fibrosis , προφανώς το μοντέλο αυτό δεν είναι τόσο αποτελεσματικό όσο θα ήταν άμα το είχαμε εκπαιδεύσει από την αρχή , ή άμα ήταν εκπαιδευμένο με μία συλλογή που περιέχει βιολογικούς όρους.

ΥΛΟΠΟΙΗΣΗ

Για αυτό το ερώτημα εκτελούμε το κώδικα με όνομα “using_colbert.py”. Η εκτέλεση αυτού του αρχείου εξάγει τα αποτελέσματά της κάτω από το directory “experiments/notebook”. Υπάρχουν πολλές ενδιαφέροντες πληροφορίες για το indexing και για το searching που έγινε, η πιο σημαντική από όλες είναι το αρχείο “ranking.tsv” που μας δείχνει τα 100 πιο σχετικά έγγραφα κάτα φθίνουσα σειρά για κάθε ερώτημα σύμφωνα με το μοντέλο του ColBERT. Πρέπει να σημειωθεί όμως πως αυτή η κατάταξη έχει ως document id , όχι το όνομα του εγγράφου ,αλλά έναν δείκτη που δείχνει την σειρά με την οποία διαβάστηκε το έγγραφο.

Για την χρήση της κατάταξης αυτής , εκτελούνται συναρτήσεις από το αρχείο “find_and_change_ranking.py” μέσα από την εκτέλεση του κώδικα για το επόμενο ερώτημα, όπου αλλάζουν την μορφή του. Συγκεκριμένα αντιστοιχούν κάθε document id με το αντίστοιχο όνομα κάθε εγγράφου και προσθέτουν μία γραμμή για τους τίτλους των στηλών.

Ερώτημα 4 – Αξιολόγηση Μοντέλων Ανάκτησης

Σε αυτό το ερώτημα καλούμαστε να αναπτύξουμε τουλάχιστον 2 μετρικές αξιολόγησης μοντέλων ανάκτησης πληροφορίας. Έχουμε στην διάθεσή μας αρχεία που δείχνουν ποια είναι τα αληθινά σχετικά κείμενα και τα αρχεία κατάταξης που δημιουργήθηκαν από τα δύο διαφορετικά μοντέλα VSM και ColBERT.

Γνωρίζουμε πως οι δυο κατατάξεις των δύο μοντέλων (“vsm_rankings.csv” και “rankings.tsv”), δείχνουν τα πιο σχετικά κείμενα στην αρχή, δηλαδή έχουν φθίνουσα σειρά για τις τιμές της ομοιότητας ερωτήματος-εγγραφου. Εκμεταλλευόμενη αυτή την πληροφορία , σκεφτόμαστε να αξιολογήσουμε τα μοντέλα με μια μετρική που κοιτάει την σειρά των αποτελεσμάτων.

Μια άλλη πληροφορία που μας δίνεται βρίσκεται στο αρχείο “cfquery_detailed” , όπου παρουσιάζει για κάθε σχετικό έγγραφο ενός ερωτήματος τέσσερις βαθμούς σχετικότητας (0=καθόλου σχετικό , 1=λίγο σχετικό και 2= πολύ σχετικό) που έχουν αποδοθεί από ειδικούς του πεδίου. Μπορούμε να εκμεταλλευτούμε και αυτή την γνώση για να αναπτύξουμε μια μετρική αξιολόγησης που θα αναλογίζεται αυτούς τους βαθμούς σχετικότητας.

Η πρώτη μετρική που θα υλοποιήσουμε είναι η Mean Average Precision (MAP). Στην μετρική αυτή μας ενδιαφέρει η σειρά με την οποία ανακτήθηκαν τα έγγραφα. Συγκεκριμένα μας ενδιαφέρουν όλες οι θέσεις των σωστών απαντήσεων. Αυτή η μετρική χαρακτηρίζεται από τους παρακάτω τύπους [12] ([7] σελ.142-143) :

$$AP = \frac{\sum_k P@k \cdot l_k}{\#\{\text{relevant documents}\}} \quad , \quad MAP = \frac{\sum_{i=1}^{Q} AP(i)}{Q}$$

Η εύρεση του MAP έγινε μέσα από της συναρτήσεις mean_average_precision και precision που αναπτύξαμε στον κώδικά μας evaluation.py. Στην παρακάτω εικόνα (Figure 2) παραθέτουμε την εξέλιξη του MAP σε διαφορετικές τιμές του K (πλήθος ανακτήσεων) για το μοντέλο Vector Space και ColBERT. Παρατηρούμε ότι ενώ αρχικά τα δύο μοντέλα είχαν παρόμοια μέση ακρίβεια προς το τέλος το ColBERT φαίνεται να παραδίδει πιο σχετικά αποτελέσματα σε σχέση με το VSM. Επιπλέον η διαφορά των εμβαδών εκτυπώνεται από το κώδικα:

The MAP of Colbert is 1.9824392670420394 different from VSM

Παρατηρούμε λοιπόν ότι, ενώ το ColBERT είναι το καλύτερο σε σχέση με την ακρίβεια, δεν είναι τόσο μεγάλη η διαφορά των δύο μοντέλων.

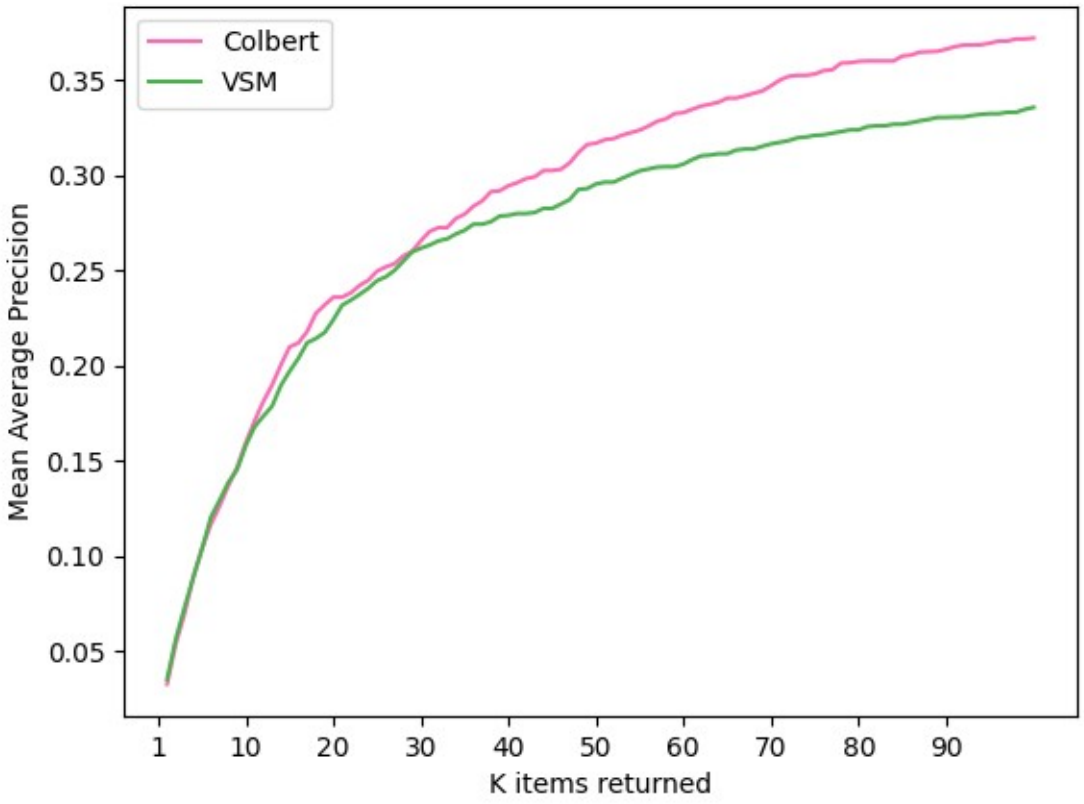

Figure 2: Mean Average Precision για διαφορετικά K

Η επόμενη μετρική που αναπτύξαμε ήταν η NDCG (Κανονικοποιημένο Εκπτωτικό Συσσωρευτικό Κέρδος), όπου σε σύγκριση με το MAP η σχετικότητα ενός εγγράφου δεν είναι δυαδικό (σχετικό ή μη-σχετικό), αλλά χαρακτηρίζεται από διαφορετικούς βαθμούς σχετικότητας. Ο λόγος που αναπτύξαμε αυτή την μετρική ήταν λόγο της πληροφορίας που μας δίνεται από το αρχείο “cfquery_detailed” , και έγιναν υπολογισμοί και για τους 4 ειδικούς βαθμούς σχετικότητας ξεχωριστά. (Οι τύποι αυτής της μετρικής υπάρχουν αντίστοιχα στις σελίδες 147-152 [7] και [12]). Η συνάρτηση που κάνει αυτές της μετρήσεις έχει ονομαστεί ndcg.

Παρακάτω παρατίθενται τα αποτελέσματα των μετρήσεών μας. Έχουμε διαφορετικό γράφημα για κάθε διαφορετικό ειδικό βαθμό σχετικότητας και την διαφορά των εμβαδών τους. Συγκεκριμένα εκτυπώθηκε:

The NDCG of Colbert is -1.0894340221813081 different from VSM (expert_ 1)

The NDCG of Colbert is 0.06721193420972327 different from VSM (expert_ 2)

The NDCG of Colbert is -1.3015160132996044 different from VSM (expert_ 3)

The NDCG of Colbert is -0.5310994617386768 different from VSM (expert_ 4)

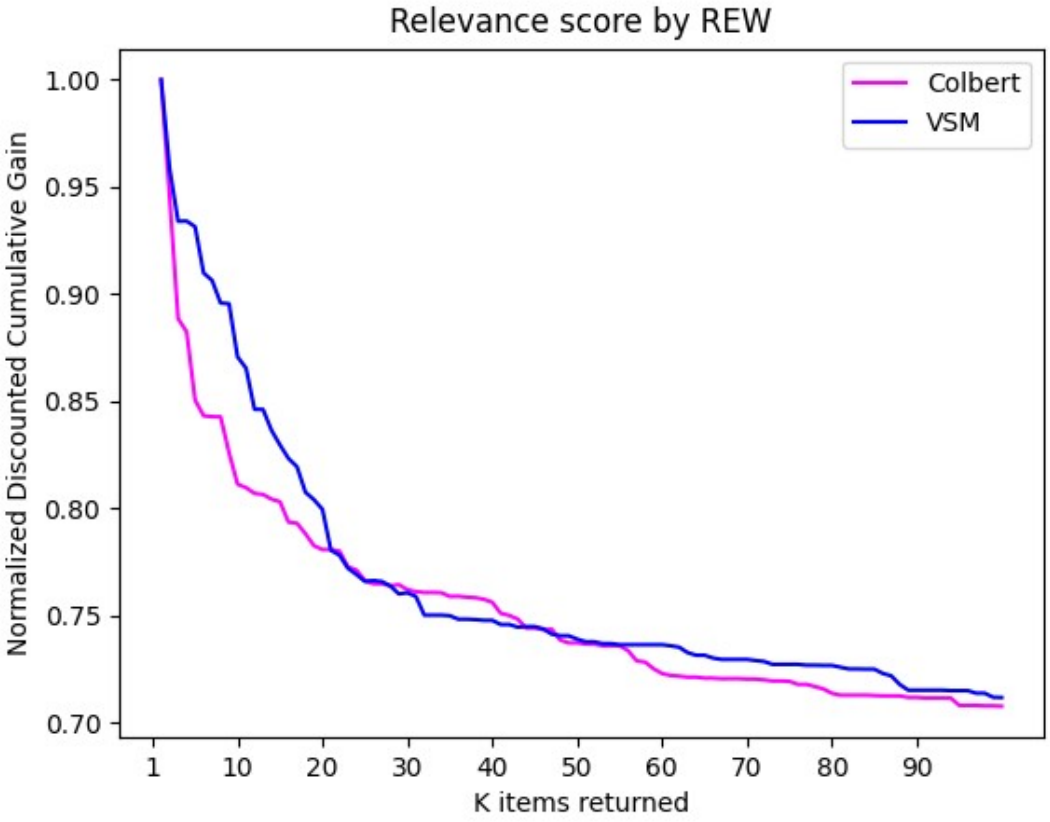

Figure 3: NDCG for expert 1

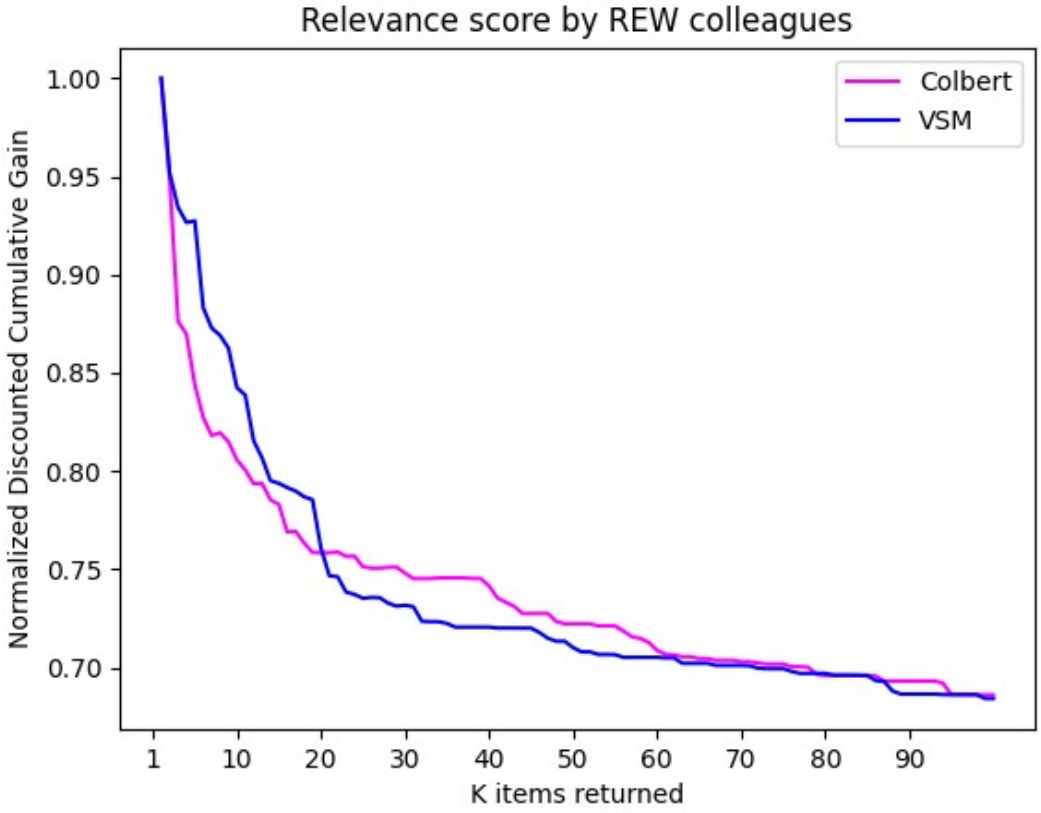

Figure 4: NDCG for expert 2

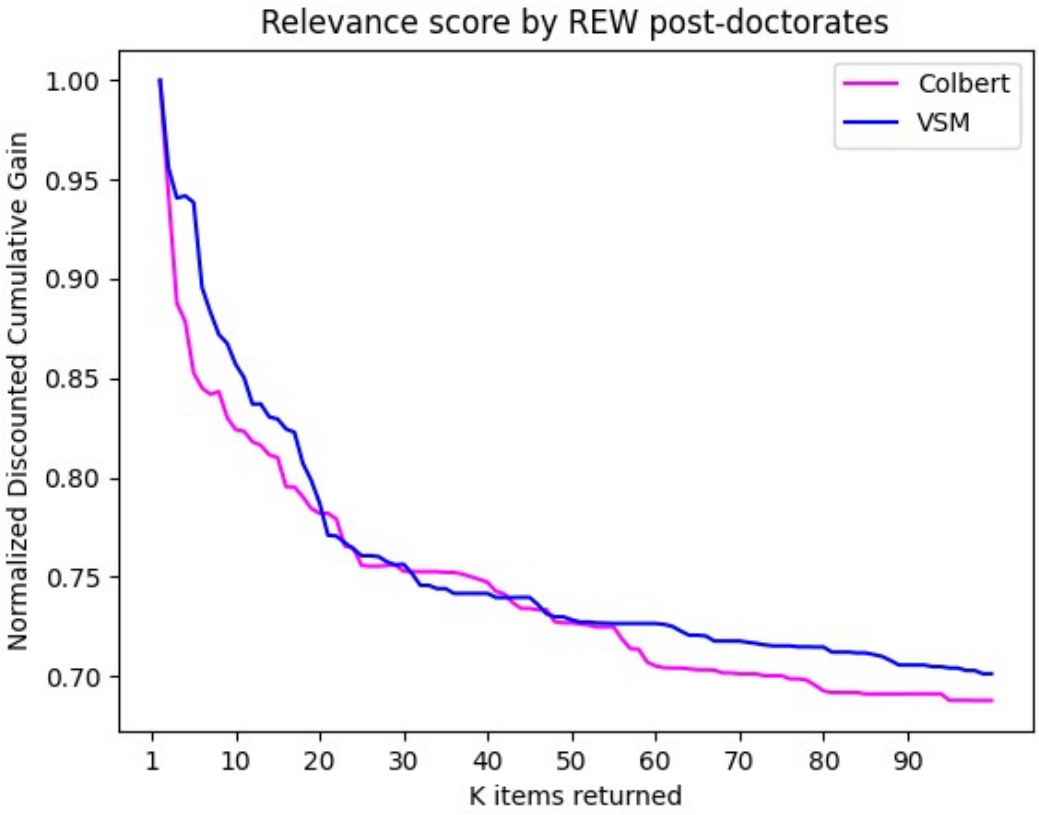

Figure 5: NDCG for expert 3

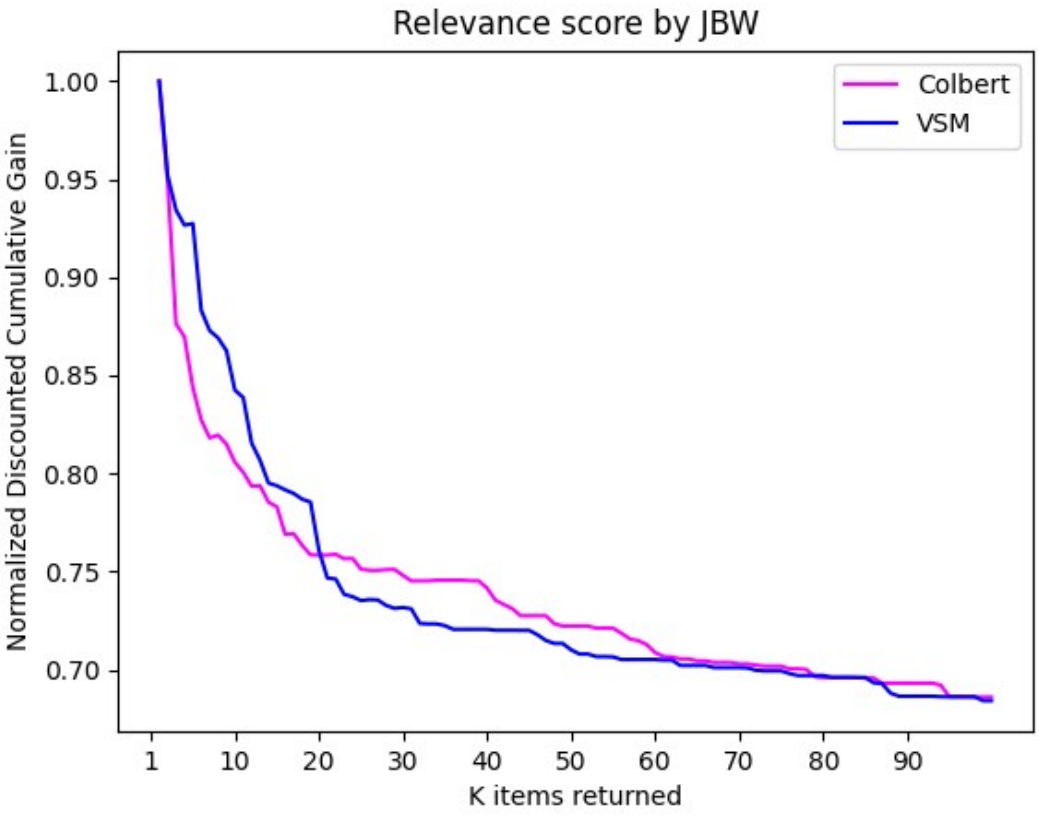

Figure 6: NDCG for expert 4

Διαπιστώνουμε , λοιπόν , πως για την περίπτωση του Κανονικοποιημένο Εκπτωτικό Συσσωρευτικό Κέρδος , δεν είναι τόσο ξεκάθαρα τα πράγματα για το πιο μοντέλο είναι καλύτερο. Θα περιμέναμε πως το ColBERT να είναι γενικά καλύτερο , εφόσον είναι ένα νευρωνικό μοντέλο, αλλά επειδή δεν είχε την κατάλληλη εκπαίδευση δεν μπορεί να είναι πολύ καλύτερο από ένα VSM. Συγκεκριμένα στα παραπάνω γραφήματα βλέπουμε οι γραμμές των δύο μοντέλων να "μπερδεύονται" μεταξύ τους. Σκεπτόμενοι και τα γραφήματα, αλλά και τα αποτελέσματα από την εκτύπωση , μόνο στην περίπτωση του expert 2 , μπορεί να θεωρηθεί το ColBERT να έχει οριακά καλύτερα αποτελέσματά από το VSM, σε όλες τις άλλες περιπτώσεις το VSM υπερτερεί.

Συνδυάζοντας τις δύο μετρικές που αναπτύξαμε μπορούμε να συμπεράνουμε πως ενώ το ColBERT εμφανίζει πιο πολλά σχετικά έγγραφα (σύμφωνα με το MAP) , τα σχετικά έγγραφα που εμφανίζει το VSM αν και λιγότερα έχουν μεγαλύτερο βαθμό σχετικότητας.

ΠΑΡΑΤΗΡΗΣΕΙΣ

Τα αποτελέσματα μας δεν είναι απόλυτα! Με διαφορετική προ-επεξεργασία της συλλογής C.F. μπορεί να είχαμε καλύτερη ή χειρότερη απόδοση από το VSM. Το ίδιο ισχύει και στο προ-εκπαιδευμένο ColBERT , αν χρησιμοποιούσαμε διαφορετικές παραμέτρους (όπως το doc_maxlen που βλέπουμε να χρησιμοποιείται στο google colab) θα είχαμε άλλα αποτελέσματα.

Όπως είχε αναφερθεί προηγουμένως, χωράει μεγάλη βελτίωση για το ανεστραμμένο ευρετήριο και κατά συνέπεια και στο Μοντέλο Διανυσματικού Χώρου. Με τις βελτιώσεις που θα μπορούσαν να γίνουν , πιθανότητα το VSM να είχε πολύ καλύτερα αποτελέσματα.

ΥΛΟΠΟΙΗΣΗ

Για αυτό το ερώτημα εκτελούμε το αρχείο κώδικα ονομαζόμενο "evaluation.py". Το αρχείο αυτό χρησιμοποιεί βοηθητικές συναρτήσεις που έχουν αναπτυχθεί στο αρχείο "find_and_change_ranking.py", όπου όπως έχει ξαναειπωθεί φέρνουν την κατάταξη από το ColBERT στην κατάλληλη μορφή που θέλουμε για επεξεργασία. Επιπλέον όμως αποσπούν την πληροφορία που χρειαζόμαστε από το "cfquery_detailed" για τον υπολογισμό του NDCG.

Η εκτέλεση του "evaluation.py" μας εκτυπώνει εκτυπώνει κάποια αποτελέσματα και επιπλέον σχεδιάζει 4 διαφορετικά γραφήματα (οι εικόνες :Figure 2,Figure 3,Figure 4,Figure 5 και Figure 6). Τα γραφήματα αυτά υπάρχουν και μέσα στον φάκελο "graph_figures"

ΣΗΜΕΙΩΣΗ Στο zip αρχείο που ανέβασα στο πεδίο εργασιών στο eclass περιέχεται μόνο ο κώδικας! Αναλυτικά τα αποτελέσματα που έβγαλα , χωρίς να τρέξετε τους κώδικες μπορείτε να δείτε στο github.**

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] "The English (Porter2) stemming algorithm." *Snowball*, https://snowballstem.org/algorithms/english/stemmer.html . Accessed 9 November 2023.

[2] Manning, Christopher D., et al. *Introduction to information retrieval*. Edited by Prabhakar Raghavan and Hinrich Schütze, Cambridge University Press, 2008, https://nlp.stanford.edu/IR-book/information-retrieval-book.html . Accessed 9 November 2023.

[3] "nltk.stem.snowball module." *NLTK*, https://www.nltk.org/api/nltk.stem.snowball.html . Accessed 11 November 2023.

[4] Barrus, Tyler. "pyspellchecker · PyPI." *PyPI*, https://pypi.org/project/pyspellchecker/ . Accessed 27 January 2024.

[5] "nltk.corpus package." *NLTK*, https://www.nltk.org/api/nltk.corpus.html#module-nltk.corpus . Accessed 27 January 2024.

[6] Παπαδόπουλος Α., Μανωλόπουλος Ι. & Τσίχλας, Κ. (2015). *Ανάκτηση Πληροφορίας*. kallipos. from https://repository.kallipos.gr/bitstream/11419/4191/3/irbook.pdf Accessed 27 January 2024.

[7] Modeling, Baeza-Yates & Ribeiro-Neto, *Modern Information Retrieval*, 2nd Edition ISBN 978-960-418-460-6 (greek translation)

[8] "Modern Information Retrieval." n.d. DCC UChile. Accessed January 28, 2024. https://users.dcc.uchile.cl/~rbaeza/mir2ed/pdf/slides_chap03.pdf.

[9] Martin, Brendan. n.d. "TF-IDF — Term Frequency-Inverse Document Frequency – LearnDataSci." LearnDataSci. Accessed January 28, 2024. https://www.learndatasci.com/glossary/tf-idf-term-frequency-inverse-document-frequency/

[10] intro2new.ipynb - Colaboratory. Accessed January 28, 2024. https://colab.research.google.com/github/stanford-futuredata/ColBERT/blob/main/docs/intro2new.ipynb#scrollTo=nl_YBBPTo5AZ

[11] "stanford-futuredata/ColBERT: ColBERT: state-of-the-art neural search (SIGIR'20, TACL'21, NeurIPS'21, NAACL'22, CIKM'22)." GitHub. Accessed January 28, 2024. https://github.com/stanford-futuredata/ColBERT

[12] Carnevali, Laura. 2023. "Evaluation Measures in Information Retrieval." Pinecone. https://www.pinecone.io/learn/offline-evaluation/