Πανεπιστήμιο Πατρών
Τμήμα Μηχανικών Η/Υ και Πληροφορικής

Ανάκτηση Πληροφορίας

Εργαστηριακή Άσκηση
Χειμερινό Εξάμηνο 2023

Μπάλλας Μιχαήλ *
Σούρλας Ζήσης †

30 Ιανουαρίου 2024

*AM: 1072599 Email: up1072599@ac.upatras.gr
†AM: 1072477 Email: sourlas.zisis@ac.upatras.gr

Περιεχόμενα

1 Εισαγωγή

1.1 Γενικά για τα μοντέλα

1.1.1 Μοντέλο Διανυσματικού Χώρου

Το μοντέλο διανυσματικού χώρου (vector space model) βασίζεται στην ιδέα της αναπαράστασης των κειμένων και των ερωτημάτων ως διανυσμάτων ενός πολυδιάστατου χώρου. Σε κάθε όρο του κειμένου/ερωτήματος ανατίθεται ένα βάρος ανάλογο με τη "σημαντικότητα" του μέσα σε αυτό. Τα διανύσματα αυτά θεωρούνται ορθοκανονικά καθώς υποτίθεται ότι σε κάθε κείμενο οι όροι εμφανίζονται ανεξάρτητα. Οι ομοιότητα μεταξύ ενός ερωτήματος και ενός κειμένου υπολογίζεται μέσω της ομοιότητας συνημιτόνου των διανυσμάτων που τα αναπαριστούν. Ιδανικά επιθυμούμε η αναπαράσταση να είναι τέτοια ώστε τα κείμενα που είναι σχετικά με ένα ερώτημα να παρουσιάζουν μεγάλη ομοιότητα μεταξύ τους ενώ τα άσχετα κείμενα να παρουσιάζουν μικρή ομοιότητα με τα υπόλοιπα. (Salton et al., 1975) [1].

Ο υπολογισμός των βαρών είναι ιδιαίτερα σημαντικός για την απόδοση του μοντέλου καθώς καθορίζει την αναπαράσταση κειμένων και ερωτημάτων. Κεντρικό ρόλο στον υπολογισμό του βάρους ενός όρου παίζουν οι έννοιες της συχνότητας εμφάνισης του όρου μέσα σε ένα κείμενο ή ερώτημα (term frequency) και η ανεστραμμένη συχνότητα εγγράφου (inverted document frequency) η οποία ορίζεται ως:

$$idf_k = \log \frac{N}{n_k}$$

Όπου N το πλήθος των κειμένων της συλλογής και n_k το πλήθος των κειμένων όπου εμφανίζεται ο όρος k. Ο συνδυασμός των δύο όρων $(tf \cdot idf)$ αναθέτει μεγαλύτερα βάρη σε όρους με μεγάλη συχνότητα εμφάνισης μέσα σε ένα κείμενο και ταυτοχρόνως σχετικά μικρή συχνότητα εμφάνισης στη συλλογή. Η παραπάνω θεωρείται από τις καλύτερες διαδικασίες υπολογισμού βαρών. (Salton et al., 1975) [1].

Υπάρχουν μια σειρά παραλλαγές για τον υπολογισμό των βαρών που αξιοποιούν την tf και την idf με διάφορους τρόπους. Εμείς επιλέξαμε την υλοποίηση και σύγκριση δύο μοντέλων που χρησιμοποιούν τις δύο παραλλαγές που έχουν αξιολογηθεί πειραματικά ως ανώτερες (Salton and Buckley, 1988) [2]:

- Για τα βάρη των όρων των ερωτημάτων και στα δύο μοντέλα:

$$\left(0.5 + \frac{0.5 * tf}{\max tf}\right) * idf$$

- Για τα βάρη του κειμένου στο πρώτο μοντέλο (VSM 1):

$$tf * idf$$

- Για τα βάρη του κειμένου στο δεύτερο μοντέλο (VSM 2):

$$tf$$

1.1.2 Μοντέλο colBERT

Το colBERT είναι ένα dense retrieval μοντέλο το οποίο αξιοποιεί το μοντέλο BERT προκειμένου να κωδικοποιήσει τα ερωτήματα και τα κείμενα της συλλογής. Κάθε κείμενο/ερώτημα αναπαρίσταται ως ένα διάνυσμα όρων όπου κάθε όρος αναπαρίσταται και ο ίδιος ως διάνυσμα συγκεκριμένου μήκους. Συνεπώς κάθε κείμενο/ερώτημα αναπαρίσταται ως ένα μητρώο. Τα διανύσματα των όρων κανονικοποιούνται με την ευκλείδεια νόρμα και αυτό έχει ως συνέπεια το εσωτερικό τους γινόμενο να είναι στην ουσία η ομοιότητα συνημιτόνου τους.

Ως βήμα προεπεξεργασίας το colBERT αξιοποιεί την βιβλιοθήκη Faiss της Facebook προκειμένου να παράξει μια πρώτη κατάταξη της συλλογής βασιζόμενη σε αναζήτηση nearest neighbours. Έτσι λαμβάνει υπόψιν του μόνο ένα κομμάτι της συλλογής το οποίο στην συνέχεια ανακατατάσει χρησιμοποιώντας τον τελεστή MaxSim. Πιο συγκεκριμένα:

$$s_{q,d} = \sum_{i \in \eta(q)} \max_{j \in \eta(d)} \eta(q)_i \cdot \eta(d)_j$$

Όπου:

- $s_{q,d}$ η ομοιότητα του ερωτήματος q με το κείμενο d.
- $\eta(\cdot)$ η κωδικοποιημένη αναπαράσταση του κειμένου/ερωτήματος.
- $\eta(\cdot)_i$ το διάνυσμα που αναπαριστά τον i-οστό όρο του κειμένου/ερωτήματος.

Κατ' ουσίαν πρόκειται για το άθροισμα των ομοιοτήτων συνημιτόνου μεταξύ του διανύσματος του κάθε όρου του ερωτήματος και του διανύσματος του βέλτιστου όρου του κειμένου για αυτόν τον όρο του ερωτήματος. (Lin et al., 2021) [3]

Η συνάρτηση απώλειας (loss function) που χρησιμοποιείται για την εκπαίδευση του μοντέλου είναι:

$$L(q, d^+, d^-) = -\log \frac{e^{s_{q,d^+}}}{e^{s_{q,d^+}} + e^{s_{q,d^-}}}$$

Όπου q το ερώτημα και d^+, d^- τα κείμενα που βρέθηκαν σχετικά και μη σχετικά αντίστοιχα.

1.2 Περιγραφή περιβάλλοντος υλοποίησης

Ως γλώσσα προγραμματισμού χρησιμοποιήθηκε η γλώσσα **python** μαζί με μια σειρά από βιβλιοθήκες της. Αναλυτικά:

- **pandas:** Για την ανάγνωση και επεξεργασία της συλλογής και των ερωτημάτων
- **nltk:** Για την αφαίρεση των stopwords και την εφαρμογή stemming.

- **pickle:** Για την αποθήκευση και την φόρτωση αρχείων.
- **json:** Για την αποθήκευση και την φόρτωση αρχείων.
- **numpy:** Για την εκτέλεση ορισμένων μαθηματικών πράξεων.
- **matplotlib:** Για την δημιουργία των διαγραμμάτων.

Για τη συγγραφή του κώδικα χρησιμοποιήθηκε το **Jupyter Notebook** για λόγους ευελιξίας και εύκολης εκτέλεσης τμημάτων κώδικα. Για την λειτουργία του απαιτούνται δύο πακέτα:

- **notebook:** Για την εγκατάσταση του Jupyter Notebook.
- **ipykernel:** Για την σύνδεση του virtual environment της python με το jupyter.

Όλες οι απαιτούμενες βιβλιοθήκες με τις συγκεκριμένες εκδόσεις που χρησιμοποιήθηκαν καταγράφονται αναλυτικά στο αρχείο *"requirements.txt"* που παρέχεται μαζί με τα αρχεία κώδικα. Για την εγκατάσταση τους το μόνο που απαιτείται είναι η εκτέλεση της εντολής:

```
pip install -r requirements.txt
```

Κατ' εξαίρεση η συγγραφή και εκτέλεση του αρχείου *"colBERT.ipynb"* έγινε σε περιβάλλον Google Colab για λόγους ευκολίας. Εφόσον το αρχείο φορτωθεί στο ίδιο περιβάλλον δεν απαιτείται η εγκατάσταση κάποιας βιβλιοθήκης ή άλλου περιβάλλοντος.

2 Υλοποίηση

2.1 Μοντέλο διανυσματικού χώρου

2.1.1 Προεπεξεργασία συλλογής

Για την καλύτερη απόδοση του Μοντέλου Διανυσματικού Χώρου προχωρήσαμε στην προεπεξεργασία των κειμένων της συλλογής και των ερωτημάτων. Οι παρεμβάσεις μας ήταν οι εξής:

- Αφαίρεση των stopwords
- Stemming

Για την αφαίρεση των stopwords χρησιμοποιήσαμε την λίστα με stopwords της αγγλικής γλώσσας που παρέχει η βιβλιοθήκη ntlk. Διατρέχοντας κάθε κείμενο και κάθε ερώτημα, διατηρήσαμε μόνο τις λέξεις που δεν περιέχονταν σε αυτή τη λίστα.

Για την εφαρμογή του stemming χρησιμοποιήθηκε ο Snowball Stemmer από τη

βιβλιοθήκη nltk. Ο συγκεκριμένος αλγόριθμος επιλέχθηκε γιατί πρόκειται για μια βελτιωμένη έκδοση του Porter Stemmer, ο οποίος χρησιμοποιείται πολύ συχνά στην ανάκτηση πληροφορίας και είναι υπολογιστικά φθηνός και αλγοριθμικά απλός.

2.1.2 Κατασκευή ανεστραμμένου αρχείου

Η υλοποίηση του απλού ανεστραμμένου αρχείου γίνεται μέσω της συνάρτησης build_inverted_index.

Η συνάρτηση αυτή παίρνει ως όρισμα μια συλλογή κειμένων. Αρχικοποιεί ένα λεξικό αποτελούμενο από λεξικά. Το εξωτερικό λεξικό είναι το λεξικό των όρων. Κάθε εσωτερικό λεξικό είναι λεξικό κειμένων όπου κάθε ζεύγος (κλειδί,τιμή) αναπαριστά το κείμενο και τη συχνότητα εμφάνισης του όρου μέσα στο κείμενο. Διατρέχοντας τη συλλογή, για κάθε μοναδικό όρο κάθε κειμένου αποθηκεύει στο λεξικό τη συχνότητα του όρου στο εκάστοτε κείμενο. Τέλος επιστρέφει το λεξικό το οποίο αποτελεί το ανεστραμμένο αρχείο.

2.1.3 Υλοποίηση μοντέλου

Η υλοποίηση του vector space model γίνεται μέσω της συνάρτησης vector_space.

Η συνάρτηση παίρνει ως ορίσματα το ερώτημα, την συλλογή κειμένων, το ανεστραμμένο αρχείο, τη συνάρτηση για τον υπολογισμό των βαρών του διανύσματος ερωτήματος, καθώς και τη συνάρτηση για τον υπολογισμό των βαρών του διανύσματος κειμένου. Αρχικά υπολογίζει το term frequency (tf) για κάθε μοναδικό όρο του ερωτήματος. Στη συνέχεια διατρέχει όλους τους μοναδικούς όρους του ερωτήματος. Για κάθε όρο εξετάζει αν ο αυτός υπάρχει στο ανεστραμμένο αρχείο. Στην περίπτωση που ο όρος υπάρχει, υπολογίζει το inverted document frequency (idf) του όρου (αξιοποιώντας το ανεστραμμένο αρχείο) και ακολούθως υπολογίζει το βάρος του όρου στο διάνυσμα ερωτήματος καλώντας την αντίστοιχη συνάρτηση. Μετά διατρέχει όλα τα κείμενα της συλλογής και βρίσκει το tf του όρου στο εκάστοτε κείμενο χρησιμοποιώντας το ανεστραμμένο αρχείο. Έπειτα υπολογίζει το βάρος του όρου στο διάνυσμα κειμένου. Τέλος, καλώντας την συνάρτηση cosine_similarity υπολογίζει την ομοιότητα συνημιτόνου των δύο διανυσμάτων (ερωτήματος και κειμένου) και επιστρέφει τα 500 κείμενα με την μεγαλύτερη ομοιότητα.

2.2 colBERT

2.2.1 Προεπεξεργασία

Για το παρόν μοντέλο δεν εφαρμόστηκε η προεπεξεργασία που εφαρμόστηκε στο μοντέλο διανυσματικού χώρου (αφαίρεση stopwords και stemming). Ο λόγος που κάναμε αυτή την επιλογή έχει να κάνει με το ότι το colBERT εν αντιθέσει με το μοντέλο διανυσματικού χώρου δεν είναι bag-of-words μοντέλο, αλλά κάνει σημασιολογική ανάλυση. Συνεπώς μια τέτοιου είδους προεπεξεργασία θα επιδρούσε αρνητικά στις επιδόσεις του, κάτι που διαπιστώθηκε και πειραματικά.

Η μόνη προεπεξεργασία στην οποία προβήκαμε κατά συνέπεια ήταν η μετατροπή των ερωτημάτων από πεζά σε κεφαλαία γράμματα και η αποθήκευση κειμένων και ερωτημάτων σε μορφή λεξικού με το ζευγάρι (κλειδί, τιμή) να αντιστοιχεί σε (id, περιεχόμενα).

2.2.2 Υλοποίηση μοντέλου

Για την εκτέλεση ερωτημάτων στο colBERT αξιοποιήθηκε η προτεινόμενη υλοποίηση του Standford και χρησιμοποιήθηκε ως βάση το σχετικό notebook. Αρχικά διαβάζονται τα κείμενα (και τα ερωτήματα αντίστοιχα) και διαχωρίζονται σε δύο πίνακες, ένας με τα περιεχόμενά τους και ένας με τα id τους ώστε να μπορούν να αντιστοιχιστούν με τα αποτελέσματα του coBERT. Στην συνέχεια δημιουργείται ο indexer του colBERT πάνω στα κείμενα της συλλογής. Έπειτα δημιουργείται ο searcher ο οποίος εκτελείται για κάθε query και επιστρέφει τα 500 πιο σχετικά κείμενα των οποίων τα id βρίσκουμε μέσω του προαναφερθέντα πίνακα. Τέλος τα αποτελέσματα αυτά αποθηκεύονται ώστε να χρησιμοποιηθούν για την αξιολόγηση του μοντέλου.

2.3 Μετρικές

Για την συγκριτική αξιολόγηση των μοντέλων χρησιμοποιήθηκαν το διάγραμμα ανάκλησης – ακρίβειας και η μετρική mean average precision (MAP).

2.3.1 Διάγραμμα ανάκλησης - ακρίβειας

Για την υλοποίηση του διαγράμματος ανάκλησης ακρίβειας δημιουργήθηκε η συνάρτηση recall_precision_curve.

Η συνάρτηση παίρνει ως όρισμα έναν πίνακα με τα κείμενα που επιστρέφει το μοντέλο και έναν πίνακα με τα σχετικά κείμενα για κάθε ερώτημα. Για κάθε ένα ερώτημα διατρέχονται τα κείμενα του μοντέλου και για κάθε κείμενο που υπάρχει στα σχετικά κείμενα αυξάνεται κατά μια μονάδα ένας μετρητής που χρησιμοποιείται για τον υπολογισμό των σωστών απαντήσεων (true positives). Βάσει αυτού του μετρητή υπολογίζονται η τιμή της ανάκλησης και της ακρίβειας και εισάγονται σε αντίστοιχες λίστες. Τέλος η συνάρτηση επιστρέφει μια λίστα με πλειάδες όπου κάθε πλειάδα περιέχει τις λίστες των τιμών ανάκλησης και ακρίβειας για το αντίστοιχο ερώτημα η οποίες μπορούν να χρησιμοποιηθούν για τη σχεδίαση της καμπύλης ανάκλησης – ακρίβειας και για τον υπολογισμός της MAP.

2.3.2 Mean average precision (MAP)

Για την υλοποίηση της MAP δημιουργήθηκε η συνάρτηση mean_average_precision.

Η συνάρτηση παίρνει ως όρισμα έναν πίνακα με τις τιμές του διαγράμματος ακρίβειας-ανάκλησης των ερωτημάτων (αυτές που επιστρέφει η racall_precision_curve). Για κάθε ερώτημα αποθηκεύονται σε αντίστοιχους πίνακες οι τιμές της ακρίβειας και της ανάκλησης και υπολογίζεται προσεγγιστικά το εμβαδόν του διαγράμματος

μέσω του τύπου:

$\sum_{i=1}^{i=k} (recall[i] - recall[i-1]) * precision[i]$ όπου $k =$ πλήθος τιμών ανάκλησης

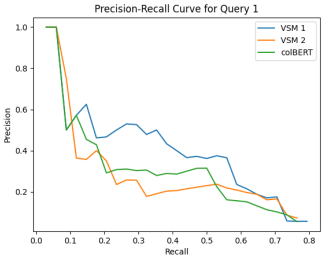
Τέλος επιστρέφεται ο μέσος όρος των εμβαδών όλων των ερωτημάτων.

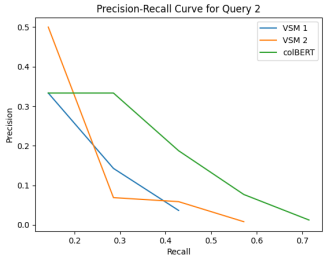
3 Αποτελέσματα - Παρατηρήσεις

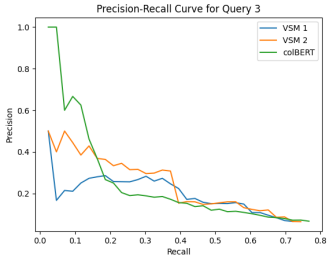
Για την πειραματική αξιολόγηση και σύγκριση των μοντέλων που υλοποιήθηκαν επιλέξαμε το σχεδιασμό καμπυλών ανάκλησης – ακρίβειας και τον υπολογισμό της mean average precision. Τα μοντέλα ρυθμίστηκαν έτσι ώστε να επιστρέφουν τα 500 πρώτα κείμενα της κατάταξης που δημιουργούν για κάθε ερώτημα και βάσει αυτών υπολογίστηκαν οι μετρικές.

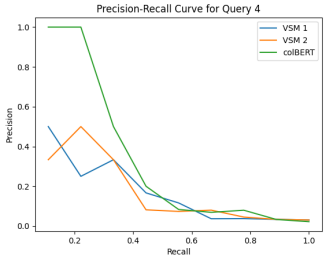
Σε σχέση με τα διαγράμματα ανάκλησης – ακρίβειας (Figure 1) δεν μπορεί να εξαχθεί κάποιο καθαρό συμπέρασμα για την σχετική απόδοση των μοντέλων. Υπάρχουν ερωτήματα στα οποία το μοντέλο διανυσματικού χώρου έχει σαφώς καλύτερες επιδόσεις από το colBERT και το αντίθετο. Αυτό που παρατηρείται με μεγαλύτερη σιγουριά είναι η υπεροχή του VSM 1 ($tf \cdot idf$) έναντι του VSM 2 (tf) στα περισσότερα ερωτήματα αν και όχι στο σύνολό τους. Οι τιμές της MAP (Table 1) επιβεβαιώνουν αυτήν την παρατήρηση ενώ δείχνουν και μια μικρή υπεροχή του colBERT έναντι του VSM 1.

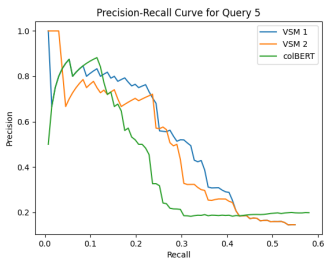
Οι παραπάνω παρατηρήσεις θεωρούμε πως ενδεχομένως οφείλονται σε ένα βαθμό στη συλλογή κειμένων που χρησιμοποιήθηκε. Η συλλογή cystic fibrosis περιέχει κείμενα που αφορούν την ασθένεια της κυστικής ίνωσης δηλαδή κείμενα που περιέχουν κυρίως ιατρική ορολογία. Τα δε ερωτήματα της συλλογής είναι επίσης ιατρικής φύσεως. Σε μια τέτοια περίπτωση η εμφάνιση περιπτώσεων συνωνυμίας ή αμφισημίας είναι προφανώς εξαιρετικά σπάνια καθώς οι επιστημονικοί όροι είναι αυστηροί και δεν συγχέονται εύκολα. Συνεπώς ένα σημαντικό πλεονέκτημα που έχει το colBERT έναντι των bag-of-words μοντέλων όπως το μοντέλο διανυσματικού χώρου, δηλαδή το να αντιλαμβάνεται το πλαίσιο μέσα στο οποίο χρησιμοποιείται μια λέξη και άρα την κατά περίπτωση σημασία της, δεν μπορεί να αξιοποιηθεί σε αυτή την περίπτωση. Επιπλέον θεωρούμε ότι η μη αξιοποίηση του *idf* στον υπολογισμό των βαρών (VSM 2) αν και γενικά επιζήμια, φαίνεται να οδηγεί σε καλύτερα αποτελέσματα σε ορισμένα ερωτήματα και πάλι πιθανώς λόγω της συλλογής. Φυσικά το μικρό μέγεθος της συλλογής αλλά και το μικρό πλήθος των ερωτημάτων δεν επιτρέπουν γενίκευση των συμπερασμάτων ώστε να κριθεί η γενικότερη επίδοση του colBERT έναντι του vector space.

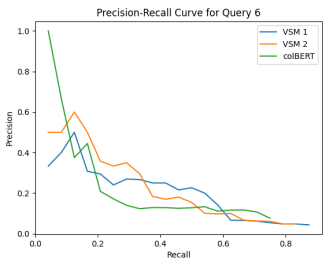

Precision-Recall Curve for Query 1
VSM 1
VSM 2
colBERT
Precision
Recall

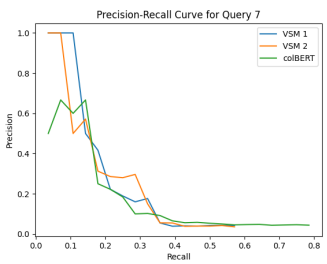

Precision-Recall Curve for Query 2
VSM 1
VSM 2
colBERT
Precision
Recall

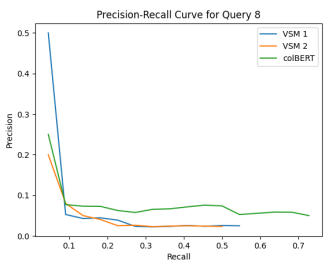

Precision-Recall Curve for Query 3
VSM 1
VSM 2
colBERT
Precision
Recall

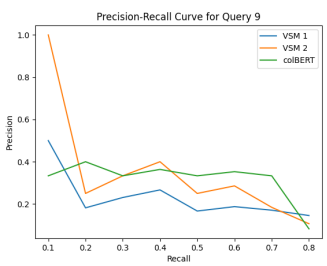

Precision-Recall Curve for Query 4
VSM 1
VSM 2
colBERT
Precision
Recall

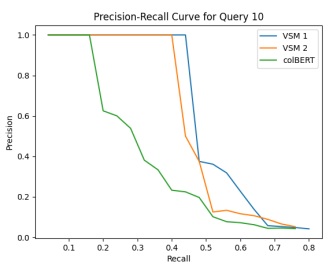

Precision-Recall Curve for Query 5
VSM 1
VSM 2
colBERT
Precision
Recall


Precision-Recall Curve for Query 6
VSM 1
VSM 2
colBERT
Precision
Recall


Precision-Recall Curve for Query 7
VSM 1
VSM 2
colBERT
Precision
Recall


Precision-Recall Curve for Query 8
VSM 1
VSM 2
colBERT
Precision
Recall


Precision-Recall Curve for Query 9
VSM 1
VSM 2
colBERT
Precision
Recall


Precision-Recall Curve for Query 10
VSM 1
VSM 2
colBERT
Precision
Recall

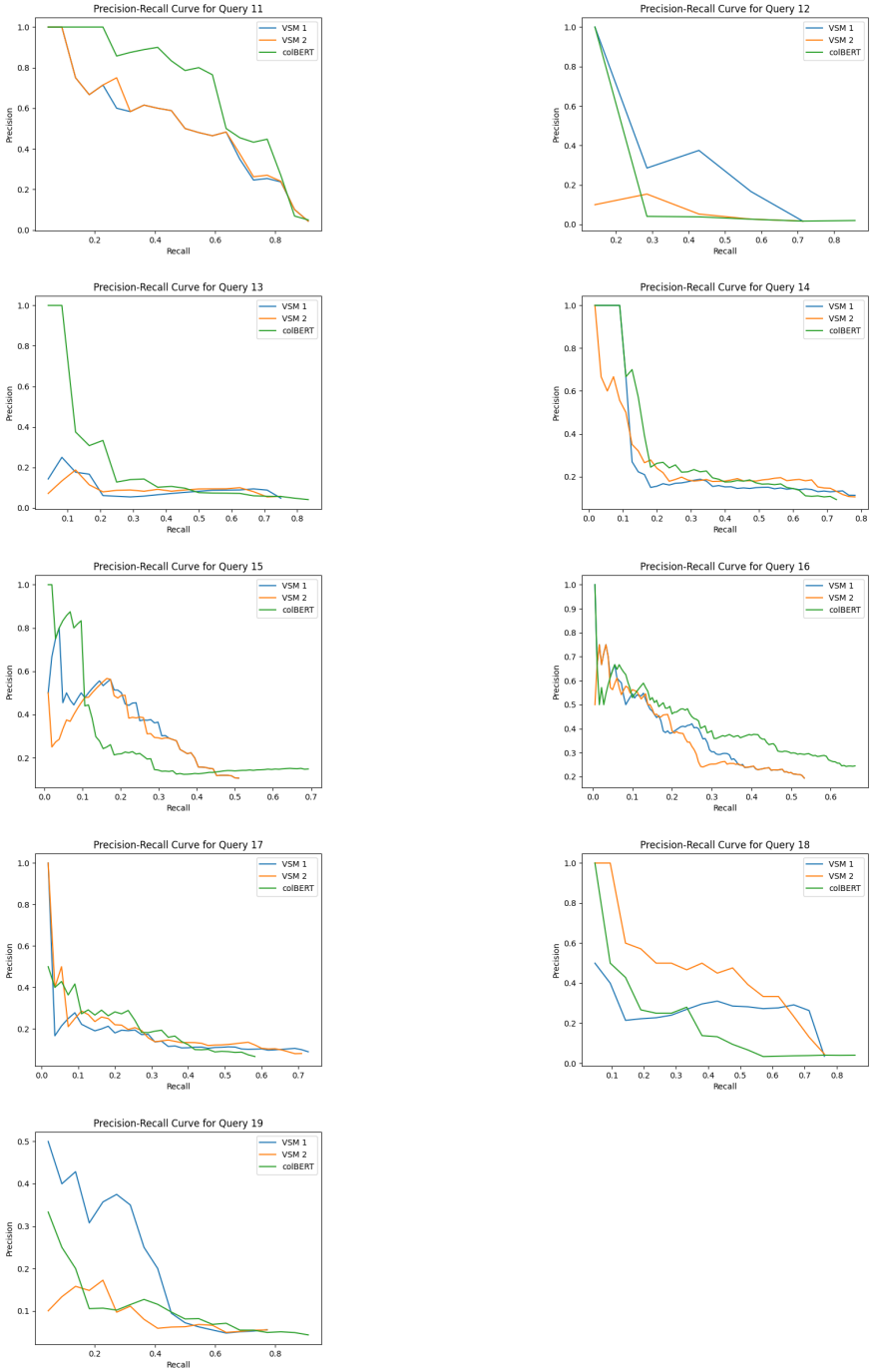


Figure 1: Διαγράμματα ανάκλησης ακρίβειας για κάθε ερώτημα της συλλογής

Mean Average Precision	
VSM 1	0.178
VSM 2	0.173
colBERT	0.181

Table 1: MAP των μοντέλων για απάντηση 500 κειμένων

4 Κώδικας

4.1 build_inverted_index

```
def build_inverted_index(documents):
    inverted_index = defaultdict(lambda:defaultdict(int))

    for doc_title, document in documents: # Unpack the tuple
        for term in set(document):
            inverted_index[term][doc_title] = document.count(term)

    return inverted_index
```

4.2 vector_space

```
def vector_space(query,docs, query_weight_func, doc_weight_func):

        tf={}

        for term in set(query):#Get query term frequency
                tf[term] = query.count(term)

        #Calculate query and doc weights
        query_weights = []
        document_weights = {}

        for doc in docs:
                document_weights[doc[0]] = []

        for term in set(query):

                if term in inverted_index and len(inverted_index[term]):

                        idf = math.log(len(docs)/len(inverted_index[term]))

                        query_weights.append(query_weight_func(tf,idf,term))

                        for doc in docs:
```

```
                        try:
                                tf_doc = inverted_index[term][doc[0]]
                        except:
                                tf_doc = 0

                        w = doc_weight_func(tf_doc,idf,term)
                        document_weights[doc[0]].append(w)


        #Calcualte cosine similarity for each doc
        sim = {}
        for doc in document_weights:
                sim[doc] =
                    cosine_similarity([query_weights],[document_weights[doc]])

        return sorted(sim.items(), key=lambda x:x[1])[-500:][::-1]
```

4.3 recall_precision_curve

```
def recall_precision_curve(result_docs, relevant):

    RP = []
    # result_docs is an array with vector space most relevant docs for
        each query and relevant is an array with the true relevant
    for i in range(len(result_docs)):

        # Initialize variables for precision, recall, and true positives
        precision_values = []
        recall_values = []
        true_positives = 0

        # Iterate over retrieved documents
        for j, doc in enumerate(result_docs[i]):
            if doc in relevant[i]:
                true_positives += 1

                # Calculate precision and recall at this point
                precision = true_positives /(j+1)
                recall = true_positives / len(relevant[i])

                precision_values.append(precision)
                recall_values.append(recall)

        RP.append((recall_values, precision_values))


    return RP
```

4.4 mean_average_precision

```
def mean_average_precision(rp):
    ap_values=[]
    for i in range(len(rp)):
        ap=0
        recall_values,precision_values=rp[i]

        for j in range(1,len(recall_values)):
            ap += (recall_values[j] - recall_values[j- 1]) *
                precision_values[j]
        ap_values.append(ap)
    return np.mean(ap_values)
```

5 Αναφορές

References

[1] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing", *Commun. ACM*, vol. 18, no. 11, pp. 613–620, Nov. 1975, ISSN: 0001-0782. DOI: 10.1145/361219.361220. [Online]. Available: https://doi.org/10.1145/361219.361220.

[2] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval", *Information Processing Management*, vol. 24, no. 5, pp. 513–523, 1988, ISSN: 0306-4573. DOI: https://doi.org/10.1016/0306-4573(88)90021-0. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0306457388900210.

[3] J. Lin, R. Nogueira, and A. Yates, *Pretrained transformers for text ranking: Bert and beyond*, 2021. arXiv: 2010.06467 [cs.IR].