# Προγραμματιστική Εργασία: Ανακτώντας και Προτείνοντας Σημασιολογικά Συναφές Περιεχόμενο χρησιμοποιώντας Ενσωματώσεις Κειμένων

Ο σκοπός της εργασίας είναι να εξασκηθείτε σε κλασικές μεθόδους και μοντέλα ανάκτησης πληροφορίας, αλλά και να εφαρμόσετε state-of-the-art τεχνικές για να βελτιώσετε τα αποτελέσματα ενός συστήματος ανάκτησης πάνω σε πραγματικά δεδομένα.

Μία από τις βασικές δυνατότητες ενός συστήματος ανάκτησης είναι η υπόδειξη στο χρήστη κατάλληλου σχετικού περιεχόμενου – προτεινόμενου περιεχομένου – με βάση το ερώτημά του.

Ας εξετάσουμε τη ροή ενεργειών που εκτελεί ένας χρήστης σε μια πλατφόρμα διαμοιρασμού βίντεο (όπως το YouTube). Η κύρια (ή ακόμα και η μόνη) διεπαφή είναι το πλαίσιο αναζήτησης (search box) όπου ο χρήστης πληκτρολογεί το ερώτημά του. Ας υποθέσουμε ότι ο χρήστης πληκτρολογεί το ερώτημα «Lucene tutorial» και κάνει κλικ στο πλήκτρο «Αναζήτηση». Το σύστημα ανάκτησης εμφανίζει μια λίστα αποτελεσμάτων αναζήτησης και ο χρήστης επιλέγει τελικά αυτό που θεωρεί ενδιαφέρον. Από εκεί και πέρα, είναι σύνηθες για τον χρήστη να σταματήσει την αναζήτηση και να επιλέξει κάποιο από τα βίντεο που εμφανίζονται ως Σχετικά ή Προτεινόμενα (Related/Recommended) με το αποτέλεσμα που επέλεξε (συνήθως τα βίντεο αυτά εμφανίζονται σε μία στήλη στα δεξιά της διεπαφής). Τυπικά προτεινόμενα βίντεο για ένα βίντεο με τίτλο "Lucene tutorial" θα μπορούσαν να είναι βίντεο όπως "Lucene for beginners", "Intro to search engines" και "Building recommender systems with Lucene". Ο χρήστης μπορεί να επιλέξει οποιαδήποτε από τις προτάσεις αυτές. Για παράδειγμα, αν έμαθε αρκετά από το βίντεο "Lucene tutorial", μπορεί να προχωρήσει στην παρακολούθηση ενός πιο προηγμένου βίντεο. Διαφορετικά, μπορεί να θέλει να παρακολουθήσει ένα άλλο εισαγωγικό βίντεο ή ένα εισαγωγικό βίντεο για μηχανές αναζήτησης, εάν συνειδητοποιήσει ότι απαιτούνται πρόσθετες προηγούμενες γνώσεις για να κατανοήσει πώς να χρησιμοποιήσει τη Lucene. Η διαδικασία αυτή «κατανάλωσης» ανακτημένου περιεχομένου και στη συνέχεια πλοήγησης σε σχετικό προτεινόμενο περιεχόμενο μπορεί να συνεχιστεί επ' αόριστον. Έτσι, η παροχή σχετικού προτεινόμενου περιεχομένου είναι υψίστης σημασίας για την καλύτερη ικανοποίηση των αναγκών του χρήστη.

Τα σχετικά προτεινόμενα αποτελέσματα μπορούν ακόμη και να μετατοπίσουν το ενδιαφέρον (την πρόθεση) του χρήστη σε θέματα που απέχουν πολύ από το αρχικό του ερώτημα. Στο προηγούμενο παράδειγμα, ο χρήστης ήθελε να μάθει πώς να χρησιμοποιεί τη Lucene. Το σύστημα ανάκτησης πρότεινε στο χρήστη ένα σχετικό βίντεο το κύριο θέμα του οποίου δεν είχε άμεση σχέση με τη Lucene: αφορούσε την κατασκευή ενός συστήματος συστάσεων με βάση τη Lucene. Αυτό είναι μια μεγάλη αλλαγή-μετατόπιση για το χρήστη, από την ανάγκη του για πληροφορίες σχετικά με τη χρήση της Lucene για αρχάριους στην εκμάθηση συστημάτων συστάσεων που βασίζονται στη Lucene (ένα πιο προχωρημένο θέμα).

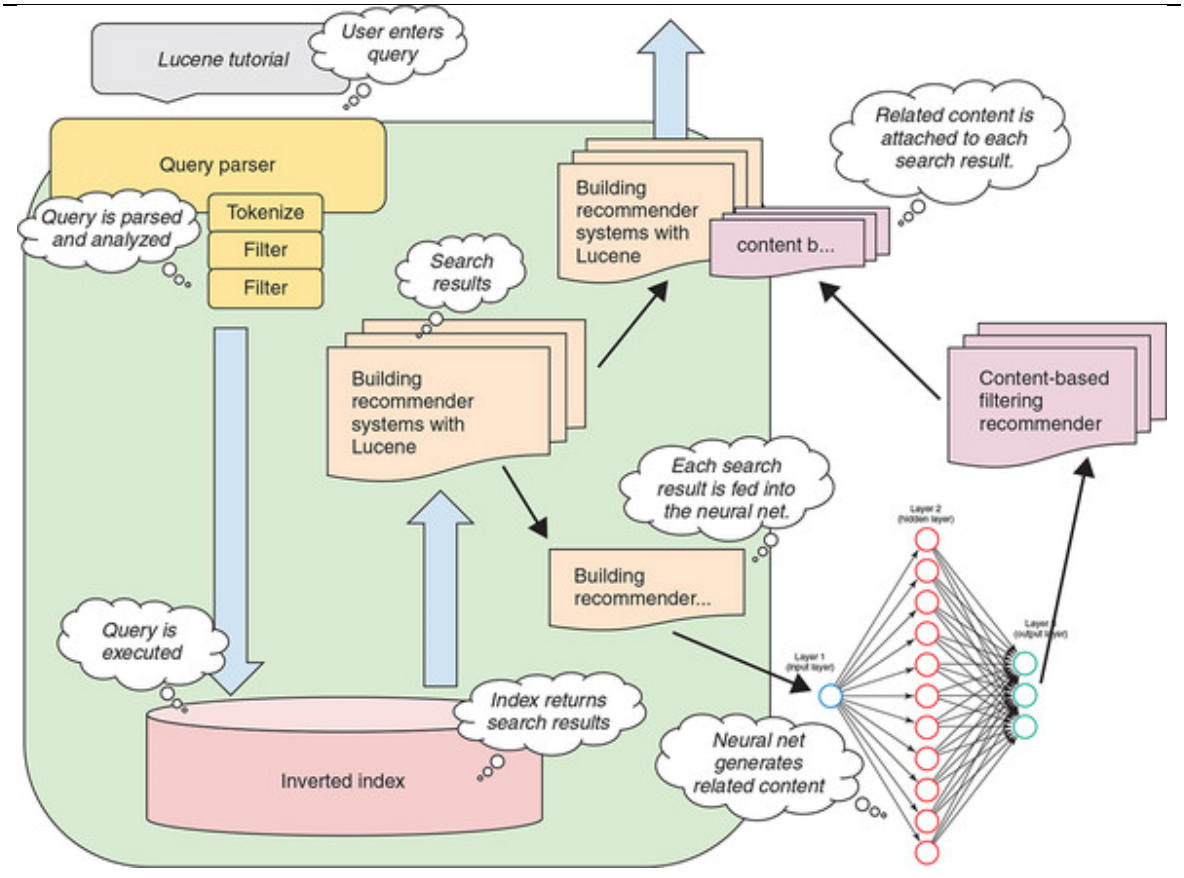
Το παράδειγμα αυτό ισχύει επίσης για ιστότοπους ηλεκτρονικού εμπορίου (e-commerce). Ο κύριος σκοπός των ιστότοπων αυτών είναι να πουλήσουν κάτι στο χρήστη. Έτσι, παρόλο που ο χρήστης ενθαρρύνεται να αναζητήσει το προϊόν που (μπορεί να) χρειάζεται, πλημμυρίζεται επίσης με πολλά «συνιστώμενα για εσάς» προϊόντα. Οι συστάσεις αυτές βασίζονται σε παράγοντες όπως:

- Ποια προϊόντα αναζήτησε ο χρήστης στο παρελθόν

- Ποια θέματα αναζήτησε περισσότερο ο χρήστης
- Νέα Προϊόντα
- Ποια προϊόντα είδε (περιηγήθηκε ή επέλεξε-έκανε κλικ) πρόσφατα ο χρήστης

Ένας από τους κύριους λόγους αυτής της πλημμύρας προτάσεων/συστάσεων συναφούς περιεχομένου είναι η διατήρηση του χρήστη (user retention): ένας ιστότοπος ηλεκτρονικού εμπορίου θέλει να κρατάει το χρήστη να περιηγείται και να αναζητεί προϊόντα όσο το δυνατόν περισσότερο, ελπίζοντας ότι κάποιο από τα προϊόντα που πουλάει θα είναι αρκετά ενδιαφέρον για να το αγοράσει τελικά.

Η πρόταση συναφούς περιεχομένου ξεπερνά βέβαια την αγορά και την πώληση. Τέτοιες δυνατότητες είναι πολύ σημαντικές και για άλλες εφαρμογές, όπως στον τομέα της υγειονομικής περίθαλψης: για παράδειγμα, ένας γιατρός που εξετάζει τα ιατρικά αρχεία ενός ασθενούς θα ωφεληθεί από το να μπορεί να εξετάσει παρόμοια ιατρικά αρχεία από άλλους ασθενείς (και το ιστορικό τους) που θα του προτείνει το σύστημα προκειμένου να πραγματοποιήσει μία καλύτερη διάγνωση.




**Εικόνα 1.0: Χρησιμοποιώντας νευρωνικά δίκτυα για ανάκτηση συναφούς περιεχομένου**

Στην εργασία αυτή, θα επικεντρωθείτε στην εφαρμογή αλγορίθμων για την ανάκτηση σχετικών ή παρόμοιων κειμένων σε ένα κείμενο-ερώτημα, με βάση το περιεχόμενό τους, προκειμένου να προταθούν στο χρήστη ως συνιστώμενα για αυτόν. Πρώτα, θα εξετάσετε πώς να κάνετε το σύστημα ανάκτησης να εξάγει σχετικό περιεχόμενο και στη συνέχεια, θα δείτε πώς να χρησιμοποιήσετε διαφορετικές προσεγγίσεις, όπως τη δημιουργία ενσωματώσεων κειμένων, για να ξεπεράσετε

ορισμένους περιορισμούς της πρώτης προσέγγισης και να προτείνετε καλύτερο περιεχόμενο στο χρήστη (βλέπε Εικόνα 1.0).

Θα εφαρμόσετε δύο τρόπους για να προτείνετε σχετικό περιεχόμενο στο χρήστη. Ο ένας είναι με χρήση των στατιστικών των κειμένων (`MoreLikeThis`) και ο άλλος με χρήση του δημοφιλή αλγόριθμου νευρωνικών δικτύων `Paragraph Vector`[3]. Κάθε μεθοδολογία έχει τα δικά της πλεονεκτήματα, αλλά και τους δικούς της περιορισμούς που θα τους εξετάσουμε στο μάθημα. Για το σκοπό αυτό θα χρησιμοποιήσετε τη `Lucene` και τη `DeepLearningForJava` (`DL4J`) σε μία `συλλογή κειμένων` καθώς και το εργαλείο αξιολόγησης `trec_eval`. Η `Lucene` είναι μία open source βιβλιοθήκη που παρέχει εργαλεία για τη δημιουργία μίας μηχανής αναζήτησης. Η `DeepLearningForJava` είναι μια βιβλιοθήκη που παρέχει υλοποιημένα μοντέλα νευρωνικών δικτύων για τη `java`. Η `συλλογή κειμένων` που θα χρησιμοποιήσετε είναι η `IR2022`, μία συλλογή από 18.316 κείμενα. Περιλαμβάνει ένα σύνολο από ερωτήματα μαζί με τις σωστές συναφείς απαντήσεις. Θα αξιολογήσετε το σύστημα ανάκτησής σας πάνω στη συλλογή `IR2022` χρησιμοποιώντας το εργαλείο αξιολόγησης `trec_eval`.

**Φάση 1 – Baseline – Ανάκτηση χωρίς εύρεση προτεινόμενων κείμενων**

Στην πρώτη φάση της εργασίας θα εφαρμόσετε ένα από τα κλασικά μοντέλα ανάκτησης που θα δούμε στο μάθημα (πχ. μοντέλο διανυσματικού χώρου, πιθανοτικό μοντέλο, κλπ.) στη συλλογή κειμένων `IR2022`. Το σύστημά σας απλά θα επιστρέφει τα πιο σχετικά κείμενα σε κάθε ερώτημα με βάση το επιλεγμένο μοντέλο, χωρίς να βρίσκει και να προτείνει άλλα συναφή κείμενα.

1. Προεπεξεργαστείτε τη συλλογή κειμένων `IR2022` (αρχείο `documents.txt`) προκειμένου να είναι σε κατάλληλη μορφή για να χρησιμοποιηθεί από τη μηχανή αναζήτησης `Lucene`.
2. Δημιουργήστε ένα ευρετήριο από τη συλλογή χρησιμοποιώντας τη μηχανή αναζήτησης `Lucene`. Επιλέξτε κατάλληλο `Analyzer` και συνάρτηση ομοιότητας που επιθυμείτε. Κάθε κείμενο θα πρέπει να αποθηκευτεί σε ένα `field` της `Lucene`.
3. Εκτελέστε τα ερωτήματα (αρχείο `queries.txt`) πάνω στο ευρετήριο και συλλέξτε τις απαντήσεις της μηχανής, τα `k` πρώτα ανακτηθέντα κείμενα, για `k = 20, 30, 50`. (Το 1ο κείμενο που ανακτάται δεν χρησιμοποιείται γιατί είναι ίδιο με το ερώτημα. Το αφαιρείτε.).
4. Αξιολογήστε τις απαντήσεις σας συγκρίνοντάς τες με τις σωστές απαντήσεις (αρχείο `qrels.txt`) χρησιμοποιώντας το εργαλείο αξιολόγησης `trec_eval` και τα μέτρα αξιολόγησης `MAP` (mean average precision) και `avgPre@k` (μέση ακρίβεια στα `k` πρώτα ανακτηθέντα κείμενα) για `k = 5, 10, 15, 20`.
5. Καταγράψτε τα πειράματά σας σε μια αναφορά. Περιγράψτε πώς υλοποιήσατε τα 4 παραπάνω βήματα, συμπεριλάβετε screenshots όπου θεωρείτε χρήσιμο (εγκατάσταση εργαλείων, εκτέλεση κώδικα, εκτέλεση `trec_eval`), και φτιάξτε έναν πίνακα με τα αποτελέσματα του `trec_eval` για τις διάφορες τιμές του `k`. Συζητήστε τα αποτελέσματα σας. Δημιουργήστε ένα αρχείο pdf με την αναφορά σας. Θα υποβάλετε την αναφορά σας, τον κώδικά σας και τα αποτελέσματα του `trec_eval` σε ένα αρχείο `zip` με ονομασία `αριθμός_μητρώου1_αριθμός_μητρώου2.zip` (πχ. 3170100.zip). Μη συμπεριλάβετε τη συλλογή κειμένων. Καταγράψτε τις πηγές σας.

Τα κλασικά μοντέλα ανάκτησης «αντιλαμβάνονται» την ομοιότητα κειμένων και ερωτημάτων ως μια διαδικασία *λεξικής σύγκρισης* (term matching). Η απλοποιημένη ιδέα είναι ότι ένα κείμενο θεωρείται συναφές με ένα ερώτημα εφόσον περιέχει ακριβώς τους όρους του ερωτήματος. Η ιδέα αυτή είναι

περιοριστική. Σε έναν ιδανικό κόσμο ένα σύστημα ανάκτησης θα προσπαθούσε να κατανοήσει την *πληροφοριακή ανάγκη* του χρήστη και να απαντήσει ανάλογα. Για παράδειγμα, θα μπορούσε να καταλάβει ότι όταν ο χρήστης υποβάλλει ένα ερώτημα "laptop" είναι σωστό και θεμιτό να του επιστρέψει ως συναφή, κείμενα που περιέχουν τη λέξη laptop, αλλά και κείμενα που περιέχουν τη λέξη notebook και όχι αναγκαστικά τη λέξη laptop. Κι αυτό γιατί οι λέξεις laptop και notebook είναι *σημασιολογικά* παρόμοιες, έχουν παρόμοια σημασία. Ένα τέτοιο σύστημα ανάκτησης προσπαθεί να καταλάβει τη σημασιολογία των λέξεων και των κειμένων και να αποφασίσει αν ένα κείμενο είναι συναφές με ένα ερώτημα ή όχι. Το σημασιολογικό αυτό κενό θα προσπαθήσουμε να το αντιμετωπίσουμε στις επόμενες 2 φάσεις της εργασίας.

**Φάση 2 – Χρήση όρων με μεγάλη συχνότητα εμφάνισης για την εύρεση παρόμοιου περιεχομένου**

Στη διάρκεια του μαθήματος θα μιλήσουμε για τα στατιστικά των κειμένων. Συγκεκριμένα, θα δούμε ότι η συχνότητα εμφάνισης ενός όρου μέσα σε ένα κείμενο μίας συλλογής, αλλά και σε ολόκληρη τη συλλογή, το επονομαζόμενο TFIDF βάρος ενός όρου, παρέχει ένα μέτρο της σημαντικότητας των κειμένων και μας βοηθάει στο να κατατάξουμε τα κείμενα με βάση ένα ερώτημα. Το σκεπτικό είναι ότι η σημαντικότητα ενός κειμένου αυξάνεται όταν ένας όρος του ερωτήματος εμφανίζεται πολλές φορές σε αυτό και επιπλέον ο όρος είναι σπάνιος μέσα στη συλλογή. Με βάση τις παραδοχές αυτές, μπορούμε να ορίσουμε έναν αλγόριθμο για να βρούμε κείμενα παρόμοια με ένα κείμενο εισόδου και να τα προτείνουμε στο χρήστη, βασισμένο αποκλειστικά στις δυνατότητες ανάκτησης του συστήματός μας. Ο αλγόριθμος ξεκινάει με την εκτέλεση του ερωτήματος του χρήστη στο ευρετήριο του συστήματος. Δεδομένων των αποτελεσμάτων αναζήτησης που επιστρέφονται για το ερώτημα, θέλουμε να φέρουμε (για παράδειγμα) τα πέντε πιο παρόμοια κείμενα και να τα παρουσιάσουμε στο χρήστη ως σχετικό, προτεινόμενο περιεχόμενο. Για να το κάνουμε αυτό, επιλέγουμε κάθε αποτέλεσμα αναζήτησης, εξάγουμε τους πιο σημαντικούς όρους από το περιεχόμενό του και εκτελούμε ένα άλλο ερώτημα χρησιμοποιώντας τους εξαγόμενους όρους. Τα πρώτα πέντε κείμενα που προκύπτουν μπορούν να χρησιμοποιηθούν ως σχετικό προτεινόμενο περιεχόμενο για τον χρήστη.

Η `Lucene` σάς επιτρέπει να το κάνετε αυτό χρησιμοποιώντας τη δυνατότητα που ονομάζεται `MoreLikeThis` (`MLT`), η οποία μπορεί να εξάγει τους πιο σημαντικούς όρους από ένα κείμενο και να δημιουργήσει ένα νέο ερώτημα για εκτέλεση.

```
Query simQuery = moreLikeThis.like(fieldName, new StringReader(text));
searcher.setSimilarity(new ClassicSimilarity());
TopDocs related = searcher.search(simQuery, 5);
for (ScoreDoc rd : related.scoreDocs) {
    Document document = reader.document(rd.doc);
    System.out.println(searcher.getSimilarity() +
                       " -> " + document.get("title"));
}
```

**Κώδικας 1.0: Εκτελώντας ένα MLT ερώτημα στη Lucene**

Επαναλάβετε τα βήματα 3 έως 5 της Φάσης 1 χρησιμοποιώντας την δυνατότητα `MoreLikeThis` (δείτε Κώδικα 1.0) της `Lucene`. Η αξιολόγηση του αλγορίθμου πραγματοποιείται συγκρίνοντας τα αποτελέσματά του με τις σωστές απαντήσεις, αναμένοντας ότι στα προτεινόμενα κείμενα θα υπάρχουν σωστά συναφή κείμενα που δεν είχαν ανακτηθεί στην προηγούμενη φάση, οδηγώντας σε βελτίωση στις διάφορες μετρικές αξιολόγησης. Στην αναφορά σας συγκρίνετε τα αποτελέσματα της Φάσης 2 με της Φάσης 1. Υπάρχει πραγματικά κάποια βελτίωση; Προσπαθήσετε να αιτιολογήσετε τα αποτελέσματά σας είτε αυτά είναι θετικά είτε αρνητικά.

Στις δύο πρώτες φάσεις, τα κείμενα και τα ερωτήματα αναπαρίστανται από διανύσματα που ονομάζονται bag-of-words (θα τα εξετάσουμε στο μάθημα). Η αναπαράσταση αυτή είναι πολύ δημοφιλής, παρουσιάζει όμως σημαντικά μειονεκτήματα: χάνεται η σειρά των λέξεων μέσα στο κείμενο και αγνοείται η σημασιολογία των λέξεων (μεταξύ άλλων). Για παράδειγμα, οι λέξεις «powerful», «strong» και «Athens» έχουν διανύσματα που απέχουν το ίδιο μεταξύ τους, παρόλο που οι δύο πρώτες λέξεις είναι πιο σχετικές και τα διανύσματά τους θα έπρεπε να είναι πιο κοντά. Επιπλέον, κείμενα που αποτελούνται από τις ίδιες λέξεις σε διαφορετική σειρά και είναι διαφορετικά, μπορεί να έχουν ίδιες αναπαραστάσεις επομένως να θεωρηθούν ίδια. Στην τρίτη φάση θα αντιμετωπίσετε τα προβλήματα αυτά χρησιμοποιώντας νευρωνικά δίκτυα και έναν νέο τρόπο αναπαράστασης των κειμένων.

**Φάση 3 – Χρήση ενσωματώσεων κειμένων για την εύρεση παρόμοιου περιεχομένου**

Στη φάση αυτή θα χτίσετε ένα σύστημα ανάκτησης, στο οποίο οι αναπαραστάσεις των κειμένων και των ερωτημάτων θα έχουν δημιουργηθεί με το μοντέλο `Paragraph Vector`[3]. Πρόκειται για έναν αλγόριθμο που χρησιμοποιεί νευρωνικά δίκτυα πρόσθιας τροφοδότησης για τη μάθηση διανυσματικών αναπαραστάσεων των κειμένων (γνωστές ως *ενσωματώσεις κειμένων*), που μπορούν να χρησιμοποιηθούν για την εύρεση κειμένων με παρόμοια σημασία ή που αφορούν παρόμοιο περιεχόμενο (context). Το μοντέλο αποτελεί προέκταση του `word2vec`[1,2] μοντέλου νευρωνικών δικτύων που θα εξετάσουμε στο μάθημα, το οποίο εξάγει τους κοντινότερους γείτονες μιας λέξης εξετάζοντας τα συμφραζόμενα, το περιβάλλον της λέξης, και καθορίζει πότε δύο λέξεις είναι σημασιολογικά συναφείς (όταν εμφανίζονται σε ίδιο ή παρόμοιο περιβάλλον-context). Υπό το πρίσμα αυτό μπορούμε να χρησιμοποιήσουμε το `Paragraph Vector` μοντέλο για να ανακαλύψουμε κείμενα που παρουσιάζουν παρόμοια σημασιολογία. Τα κείμενα που αποτελούνται από λέξεις παρόμοιας σημασιολογίας και γενικά έχουν παρόμοιο σημασιολογικά περιεχόμενο, μπορούν να θεωρηθούν συναφή και να ανακτηθούν ως προτεινόμενα.

Η τεχνική αυτή μπορεί να εφαρμοστεί τόσο στα κείμενα όσο και στα ερωτήματα αφού και αυτά μπορούν να θεωρηθούν ως μικρά κείμενα. Τα διανύσματα που θα προκύψουν θα είναι πυκνά, με περισσότερη σημασιολογική πληροφορία από τα διανύσματα των κλασικών μοντέλων των προηγούμενων φάσεων, αλλά ταυτόχρονα, θα απαιτούν λιγότερη μνήμη (ή χώρο στο δίσκο) για να αποθηκευτούν. Διαισθητικά, τα κείμενα που το `Paragraph Vector` μοντέλο θα έχει θεωρήσει ότι έχουν παρόμοια σημασιολογία θα έχουν παρόμοια διανύσματα, επομένως τα διανύσματα των κείμενων και των ερωτημάτων που θα προκύψουν από την παραπάνω διαδικασία και θα αποτελούνται από σημασιολογικά παρόμοιο περιεχόμενο θα θεωρούνται παρόμοια.

Αφού δημιουργηθούν τα διανύσματα των κειμένων και των ερωτημάτων, στη συνέχεια, υπολογίζεται η ομοιότητά τους με βάση το πόσο κοντά βρίσκονται στο χώρο. Τα βήματα που θα πρέπει να ακολουθήσετε είναι τα παρακάτω:

1. Εκπαιδεύστε ένα μοντέλο `Paragraph Vector` χρησιμοποιώντας τη συλλογή `IR2022` ως είσοδο και τη βιβλιοθήκη `DeepLearningForJava` (`DL4J`), η οποία παρέχει υλοποιημένα μοντέλα νευρωνικών δικτύων, όπως το `Paragraph Vector`, για τη `java`. Η διαδικασία της παραγωγής του μοντέλου μπορεί να παραμετροποιηθεί ως προς την αρχιτεκτονική που θα χρησιμοποιηθεί, τους όρους που θα λάβει υπόψη και τον αριθμό των διαστάσεων. Οι διαθέσιμες αρχιτεκτονικές είναι οι `PV-DM` και `PV-DBOW` που θα αναλυθούν στο μάθημα. Οι όροι που θα λαμβάνονται υπόψη εξαρτώνται από τη συχνότητα εμφάνισής τους. Όσον αφορά τον αριθμό των διαστάσεων, αυτός ορίζει την πολυπλοκότητα και το μέγεθος του μοντέλου (δείτε κώδικα 2.1 για παράδειγμα).

```
ParagraphVectors paragraphVectors = new ParagraphVectors.Builder()
  .iterate(iterator)
  .layerSize(50)
  .minWordFrequency(7)
  .sequenceLearningAlgorithm(new DM<>())
  .tokenizerFactory(new DefaultTokenizerFactory())
  .build();
paragraphVectors.fit();
```

**Κώδικας 2.1: Εκπαιδεύοντας το `paragraph vectors` μοντέλο στην DL4J**

2. Υπολογίστε τη συνημιτονοειδή ομοιότητα των ερωτημάτων με τα κείμενα στο νέο χώρο (δείτε κώδικα 2.2). Ταξινομήστε τα κείμενα σε φθίνουσα σειρά ομοιότητας και συλλέξτε τα `k` κείμενα με το μεγαλύτερο σκορ ομοιότητας, για `k = 20, 30, 50`.

```
Collection<String> strings = paragraphVectors.nearestLabels("your_query", 3);
for (String s : strings) {
  int docId = Integer.parseInt(s.substring(4));
  Document document = reader.document(docId);
  System.out.println(document.get(fieldName));
}
```

**Κώδικας 2.2: Ανακτώντας σημασιολογικά παρόμοια κείμενα στην DL4J**

3. Αξιολογήστε τα αποτελέσματά σας συγκρίνοντάς τα με τις σωστές απαντήσεις (αρχείο `qrels.txt`) χρησιμοποιώντας το εργαλείο αξιολόγησης `trec_eval` και τα μέτρα αξιολόγησης `MAP` (mean average precision) και `avgPre@k` (μέση ακρίβεια στα `k` πρώτα ανακτηθέντα κείμενα) για `k = 5, 10, 15, 20`.
4. Καταγράψτε τα πειράματά σας σε μια αναφορά. Περιγράψτε πώς υλοποιήσατε τα παραπάνω βήματα, συμπεριλάβετε screenshots όπου θεωρείτε χρήσιμο (εγκατάσταση εργαλείων, εκτέλεση κώδικα, εκτέλεση `trec_eval`), και φτιάξτε έναν πίνακα με τα αποτελέσματα του

`trec_eval` για τις διάφορες τιμές του `k`. Στην αναφορά σας συγκρίνετε τα αποτελέσματα της Φάσης 3 με τα αποτελέσματα των προηγούμενων φάσεων. Υπάρχει κάποια βελτίωση; Προσπαθήσετε να αιτιολογήσετε τα αποτελέσματά σας είτε αυτά είναι θετικά είτε αρνητικά. Δημιουργήστε ένα αρχείο pdf με την αναφορά σας όπως στις προηγούμενες φάσεις.

Οι ενσωματώσεις κειμένων (document embeddings) όπως αυτές που δημιουργούνται από το `paragraph vector` μοντέλο έχουν στόχο τη δημιουργία μιας καλής αναπαράστασης της σημασιολογίας ολόκληρου του κειμένου, με τη μορφή ενός διανύσματος. Μπορείτε να τα χρησιμοποιήσετε στο πλαίσιο της ανάκτησης για να αντιμετωπίσετε το πρόβλημα της σημασιολογικής κατανόησης στην κατάταξη (ranking) των κειμένων. Η ομοιότητα μεταξύ τέτοιων ενσωματώσεων εξαρτάται περισσότερο από την έννοια του κειμένου και λιγότερο από την απλή αντιστοίχιση όρων (term matching) όπως συμβαίνει στα κλασικά μοντέλα ανάκτησης στις προηγούμενες φάσεις της εργασίας.

**Φάση 4 – Χρήση συνδυασμού ενσωματώσεων κειμένων για την εύρεση παρόμοιου περιεχομένου – ΠΡΟΑΙΡΕΤΙΚΗ – BONUS**

Επαναλάβετε τη φάση 3, αλλά αυτή τη φορά στο 1ο βήμα εκπαιδεύστε δύο μοντέλα `Paragraph Vector` χρησιμοποιώντας και τις δύο διαθέσιμες αρχιτεκτονικές, `PV-DM` και `PV-DBOW`. Στη συνέχεια, για κάθε κείμενο της συλλογής συνενώστε (concat) τα διανύσματα κειμένων που προέκυψαν από τα δύο μοντέλα σε ένα διάνυσμα. Χρησιμοποιήστε το «διπλό» αυτό διάνυσμα στα επόμενα βήματά σας.

**Υλοποίηση**

Η υλοποίηση της μηχανής αναζήτησης *προτείνεται* να πραγματοποιηθεί με χρήση `Java`, `Lucene` και `DL4J`. Μπορείτε να δοκιμάσετε μια άλλη μηχανή αναζήτησης (πχ. `ElasticSearch`, `Solr`) και άλλη γλώσσα προγραμματισμού (πχ. `python`), αλλά πιθανόν να έχετε περιορισμένη υποστήριξη από τη διδάσκουσα.

**Εργαλεία:** τα εργαλεία που θα χρειαστείτε μπορείτε να τα βρείτε παρακάτω

| | |
|---|---|
| Lucene | https://lucene.apache.org/ |
| | https://lucene.apache.org/core/downloads.html - κατεβάστε από εδώ την τελευταία έκδοση της σειράς 7.x όχι της 8.x. |
| DL4J | https://deeplearning4j.org/ |
| Java 8++ | |
| trec_eval | https://trec.nist.gov/trec_eval/ (διαθέσιμο στο eclass) |

**Συλλογή `IR2022`:**

Διαθέσιμη στο eclass στο φάκελο «Προγραμματιστική εργασία\Συλλογή IR2022»

**Βιβλιογραφία - Πηγές:**

[1] Mikolov, T., Corrado, G., Chen, K., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. Proceedings of the International Conference on Learning Representations (ICLR 2013), 1–12.

[2] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. NIPS, 1–9.

[3] Mikolov, T., Le, Q., (2014). Distributed Representations of Sentences and Documents, Proceedings of the 31st International Conference on Machine Learning, PMLR 32(2):1188-1196.

**Ημερομηνίες υποβολής φάσεων εργασίας (ενδέχεται να τροποποιηθούν)**

Φάση 1: 10 Μαΐου 2022 (10 βαθμοί)

Φάση 2: 30 Μαΐου 2022 (10 βαθμοί)

Φάση 3: ημ/νία εξέτασης μαθήματος ή τέλος εξαμήνου (20 βαθμοί)

Φάση 4: ημ/νία εξέτασης μαθήματος ή τέλος εξαμήνου (5 βαθμοί)

Η προγραμματιστική εργασία είναι ομαδική (ομάδες 2 φοιτητών), είναι υποχρεωτική, πιάνει το 40% του τελικού βαθμού σας (+5% bonus), και θα προσμετρηθεί στον τελικό βαθμό σας αν ο βαθμός γραπτής εξέτασης είναι μεγαλύτερος ή ίσος του 4. Στο τέλος του εξαμήνου θα πραγματοποιηθεί <u>υποχρεωτική ατομική προφορική εξέταση</u>, από την οποία θα εξαρτηθεί ο τελικός βαθμός κάθε φοιτητή στην εργασία.