ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Τεχνικές ταξινόμησης αποτελεσμάτων μηχανών αναζήτησης με βάση την ιστορία του χρήστη

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΝΙΚΟΛΑΪΔΗ ΑΝΤΩΝΙΟΥ

Επιβλέπων : Σελλής Τιμολέων
Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2009

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Τεχνικές ταξινόμησης αποτελεσμάτων μηχανών αναζήτησης με βάση την ιστορία του χρήστη

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΝΙΚΟΛΑΪΔΗ ΑΝΤΩΝΙΟΥ

Επιβλέπων : Σελλής Τιμολέων
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 30η Μαρτίου 2009.

.....	.....	.....
Σελλής Τιμολέων	Βασιλείου Ιωάννης	Στάμου Γεώργιος
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Λέκτορας Ε.Μ.Π.

Αθήνα, Μάρτιος 2009

.....

ΝΙΚΟΛΑΪΔΗΣ ΑΝΤΩΝΙΟΣ

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον κ. Τίμο Σελλή και τον κ. Ιωάννη Βασιλείου, καθηγητές του Εθνικού Μετσόβιου Πολυτεχνείου, για την ανάθεση της διπλωματικής μου εργασίας και γιατί με ενέπνευσαν να ακολουθήσω τη συγκεκριμένη επιστημονική κατεύθυνση.

Ευχαριστώ επίσης θερμά τους συνεπιβλέποντες Θοδωρή Δαλαμάγκα και Γιώργο Γιαννόπουλο για την πολύτιμη καθοδήγηση και βοήθειά τους καθ' όλη τη διάρκεια της εκπόνησης της εργασίας αυτής.

Τέλος, θα ήθελα να ευχαριστήσω όλους όσους ήταν δίπλα μου και με στήριξαν όλα τα χρόνια των σπουδών μου.

Περίληψη

Σκοπός της εργασίας αυτής είναι ο σχεδιασμός και η ανάπτυξη τεχνικών που θα βελτιώσουν τον τρόπο ταξινόμησης των αποτελεσμάτων μιας μηχανής αναζήτησης, έτσι ώστε η κατάταξή τους να είναι εξατομικευμένη για το χρήστη.

Η χρησιμότητα μίας μηχανής αναζήτησης για ένα χρήστη εξαρτάται από την σχετικότητα των αποτελεσμάτων που του παρουσιάζονται σε σχέση με αυτά που περίμενε να δει. Οι περισσότερες μηχανές αναζήτησης χρησιμοποιούν μεθόδους για την κατάταξη των αποτελεσμάτων έτσι ώστε να εμφανίζουν στην κορυφή τα καλύτερα αποτελέσματα. Καθώς η χρήση του Διαδικτύου και συνεπώς και των χρηστών μηχανών αναζήτησης συνεχώς αυξάνεται, είναι φανερό ότι η παραδοσιακή μέθοδος της ενιαίας κατάταξης για όλους τους χρήστης δεν είναι αρκετά ικανοποιητική. Η εξατομίκευση των αποτελεσμάτων που πραγματοποιεί η εφαρμογή μας είναι μία μέθοδος που απευθύνεται σε αυτό το πρόβλημα των μηχανών αναζήτησης.

Η τροποποίηση της παρουσιαζόμενης κατάταξης γίνεται με βάση το ιστορικό της δραστηριότητας του χρήστη καθώς πραγματοποιεί αναζητήσεις στο διαδίκτυο. Το ιστορικό περιλαμβάνει μόνο δεδομένα τύπου clickstream, δηλαδή την ακολουθία των κλικ που ο χρήστης έκανε κατά τη διάρκεια χρήσης της μηχανής αναζήτησης. Αναλύοντας τα δεδομένα αυτά, εξάγονται κάποια συμπεράσματα για τις προτιμήσεις του χρήστη. Ο χρήστης δε χρειάζεται να δηλώσει ρητά τις προτιμήσεις του, αλλά τις εξάγουμε έμμεσα, βασιζόμενοι στο γεγονός ότι κάνει κλικ σε κάποια αποτελέσματα ενώ εσκεμμένα αγνοεί κάποια άλλα που εμφανίζονται υψηλότερα στην αρχική κατάταξη. Παράλληλα καταγράφονται και τα χαρακτηριστικά των αποτελεσμάτων που εμφανίζονται στο χρήστη, ανεξαρτήτως αν τα επισκέφτηκε ή όχι.

Αφού έχουμε μαζέψει έναν ικανοποιητικό αριθμό τέτοιων προτιμήσεων, τα τροφοδοτούμε μαζί τα χαρακτηριστικά των ιστοσελίδων στον αλγόριθμο εκπαίδευσης Support Vector Machine. Ο αλγόριθμος αυτός εκπαιδεύει ένα μοντέλο, το οποίο ουσιαστικά μαθαίνει τι χαρακτηριστικά έχουν τα αποτελέσματα που προτιμά ο χρήστης. Μπορούμε να χρησιμοποιήσουμε το μοντέλο αυτό για την αλλαγή της ταξινόμησης των αποτελεσμάτων που επιστρέφει η μηχανή αναζήτησης, ώστε να παρουσιάζονται με εξατομικευμένη κατάταξη για το χρήστη.

Λέξεις Κλειδιά: διαδίκτυο, μηχανή αναζήτησης, εξατομίκευση αποτελεσμάτων, ακολουθία κλικ, Support Vector Machine

Abstract

The scope of this thesis is to design and develop techniques that will improve the classification results of a search engine, so that the classification is personalized for that user.

The usefulness of a search engine for a user depends on the relevancy of the results presented in relation to those he was expecting to see. Most search engines employ methods to rank the results so that the most relevant results are at the top of the rankings. As the use of the Internet and the use of search engines are increasing, it is clear that the traditional method of a single classification for all users is not quite satisfactory. The personalization of the results makes our application a way to address this problem of search engines.

The change of classification is based on the history of the user's activity while he conducts searches on the Internet. The history includes only clickstream type of data, i.e. the sequence of clicks the user made during his use of the search engine. Analyzing this data, we draw several conclusions about the user's preferences. The user does not need to explicitly state his preferences, but they are exported implicitly, based on the fact that clicking on any results while deliberately ignoring others that appear higher in the initial ranking means the user prefers the clicked-on result. At the same moment, we are recording the characteristics of the results displayed to the user, regardless of whether or not they were clicked on.

After we collect a sufficient number of those preferences, we input them alongside with the characteristics of the results to a Support Vector Machine learning algorithm. The algorithm produces a trained model, which effectively learns the characteristics of the results that the user prefers. We can use this model to change the classification of the results returned by the search engine, so that they are presented personalized for that user.

Keywords: internet, search engine, personalized ranking, clickstream data, Support Vector Machine

Πίνακας περιεχομένων

1

Εισαγωγή

1.1 Μηχανές αναζήτησης

Μια διαδικτυακή μηχανή αναζήτησης είναι ένα εργαλείο σχεδιασμένο για την αναζήτηση στο Διαδίκτυο. Τα αποτελέσματα της αναζήτησης παρουσιάζονται συνήθως σε μια λίστα, που μπορεί να αποτελείται από ιστοσελίδες, εικόνες, πληροφορίες και άλλα είδη αρχείων. Αντίθετα με τους δικτυακούς καταλόγους (Web directories), οι οποίοι συντηρούνται από ανθρώπους, οι μηχανές αναζήτησης λειτουργούν αλγοριθμικά.

Μια μηχανή αναζήτησης λειτουργεί, με την ακόλουθη σειρά:

1. Ανίχνευση του διαδικτύου (web crawling)
2. Δημιουργία ευρετηρίου
3. Αναζήτηση

Οι μηχανές αναζήτησης δουλεύουν αποθηκεύοντας πληροφορίες σχετικά με πολλές ιστοσελίδες, τις οποίες ανακτούν από το Διαδίκτυο. Οι σελίδες αυτές έχουν ανακτηθεί από μια μηχανή ανίχνευσης ιστοσελίδων (γνωστή επίσης ως αράχνη) - ένα αυτοματοποιημένο πρόγραμμα περιήγησης στο Διαδίκτυο που ακολουθεί κάθε σύνδεσμο που βλέπει. Στη συνέχεια τα περιεχόμενα κάθε σελίδας αναλύονται για να καθοριστεί πώς θα πρέπει να

καταχωρηθούν στο ευρετήριο. Στοιχεία σχετικά με τις ιστοσελίδες αποθηκεύονται σε μια βάση δεδομένων ευρετηρίου για χρήση σε επόμενα ερωτήματα.

Υπάρχουν τρεις μεγάλες κατηγορίες που καλύπτουν τα περισσότερα ερωτήματα αναζήτησης στο διαδίκτυο:

- Πληροφοριακά ερωτήματα - ερωτήματα που καλύπτουν ένα ευρύ θέμα (π.χ., *Ελλάδα* ή *αυτοκίνητο*) για τις οποίες μπορεί να υπάρχουν χιλιάδες σχετικά αποτελέσματα.
- Ερωτήματα πλοήγησης - ερωτήματα που αναζητούν ένα μόνο δικτυακό τόπο ή ιστοσελίδα ενός και μόνο φορέα (π.χ., *YouTube* ή *Ολυμπιακές Αερογραμμές)*.
- Ερωτήματα συναλλαγών - ερωτήματα που αντικατοπτρίζουν την πρόθεση του χρήστη να εκτελέσει μια συγκεκριμένη ενέργεια, όπως η αγορά ενός αυτοκινήτου.

Όταν ένας χρήστης εισάγει ένα ερώτημα σε μια μηχανή αναζήτησης (συνήθως χρησιμοποιώντας λέξεις κλειδιά), η μηχανή εξετάζει το ευρετήριο και παρέχει μια λίστα με τις καλύτερες ιστοσελίδες που ταιριάζουν με τα κριτήρια, συνήθως με μια σύντομη περίληψη που περιέχει τον τίτλο του εγγράφου και ενίοτε μέρη του κειμένου. Οι διαδικτυακές αναζητήσεις είναι ξεχωριστές, δεδομένου ότι είναι ασαφείς και συχνά αδόμητες. Διαφέρουν σημαντικά από τις πρότυπες γλώσσες ερωτημάτων, οι οποίες διέπονται από αυστηρούς κανόνες σύνταξης. Οι περισσότερες μηχανές αναζήτησης υποστηρίζουν τη χρήση λογικών πράξεων ΚΑΙ, Ή, ΟΧΙ για περαιτέρω διευκρίνηση της αναζήτησης. Ορισμένες μηχανές αναζήτησης παρέχουν και μια προηγμένη δυνατότητα που ονομάζεται αναζήτηση εγγύτητας που επιτρέπει στους χρήστες να καθορίσουν την απόσταση μεταξύ των λέξεων – κλειδιών.

Η χρησιμότητα της μηχανής αναζήτησης εξαρτάται από την σχετικότητα του συνόλου των αποτελεσμάτων που δίνει πίσω. Ενώ μπορεί να υπάρχουν εκατομμύρια ιστοσελίδες που περιλαμβάνουν μια συγκεκριμένη λέξη ή φράση, μερικές σελίδες μπορεί να είναι πιο σχετικές, δημοφιλείς, ή έγκυρες από άλλες. Οι περισσότερες μηχανές αναζήτησης χρησιμοποιούν μεθόδους για την κατάταξη των αποτελεσμάτων έτσι ώστε να παρέχουν πρώτα τα "καλύτερα" αποτελέσματα.

Η αυξανόμενη διαθεσιμότητα πληροφορίας σε ψηφιακή μορφή αλλάζει τη φύση της αναζήτησης πληροφοριών. Οι χρήστες αντιμετωπίζουν δύο προβλήματα: το μεγάλο αριθμό εγγράφων και την ακόμα μεγαλύτερη διαφοροποίηση σε ποιότητα των εγγράφων αυτών. Ειδικά στις μέρες μας, που η ετερογένεια των εγγράφων (σε ποιότητα αλλά και σε μορφή), αλλά και η ανάγκη των χρηστών να βρουν γρήγορα σχετικές και αξιόπιστες απαντήσεις στα ερωτήματά τους ολοένα και αυξάνεται, γίνεται σαφής η ανάγκη για βελτιστοποίηση των αποτελεσμάτων των μηχανών αναζήτησης ώστε οι χρήστες να μπορούν να βρουν εύκολα τα έγγραφα που τους ενδιαφέρουν.

1.2 Αντικείμενο διπλωματικής

Σκοπός της εργασίας αυτής είναι ο σχεδιασμός και η ανάπτυξη τεχνικών που θα βελτιώσουν τον τρόπο ταξινόμησης των αποτελεσμάτων μιας μηχανής αναζήτησης, έτσι ώστε η κατάταξή τους να είναι εξατομικευμένη για το χρήστη. Συγκεκριμένα θα δουλέψουμε με τα αποτελέσματα που παρέχει η μηχανή αναζήτησης Google, αλλά αυτό εύκολα μπορεί να τροποποιηθεί και σε άλλη μηχανή αναζήτησης της επιθυμίας μας.

Η τροποποίηση της κατάταξης των αποτελεσμάτων θα γίνει εφαρμόζοντας μία συνάρτηση αναταξινόμησης των αποτελεσμάτων του Google, η οποία λειτουργεί με βάση ένα προηγουμένως εκπαιδευμένο μοντέλο. Το μοντέλο αυτό δημιουργείται από προτιμήσεις του χρήστη οι οποίες έχουν έμμεσα εξαχθεί από το ιστορικό της δραστηριότητάς του όταν χρησιμοποιεί τη μηχανή αναζήτησης. Έτσι τα αποτελέσματα προσαρμόζονται κάθε φορά στις συνήθειες και τις προτιμήσεις του χρήστη.

Το ιστορικό περιλαμβάνει μόνο δεδομένα τύπου clickstream, δηλαδή την ακολουθία των κλικ που ο χρήστης έκανε κατά τη διάρκεια χρήσης της μηχανής αναζήτησης. Έτσι ο χρήστης δεν επιβαρύνεται με επιπλέον χρονοβόρες διαδικασίες δημιουργίας προφίλ ή άμεσης αξιολόγησης των αποτελεσμάτων που του παρέχει η μηχανή αναζήτησης, αλλά τη χρησιμοποιεί με το συνηθισμένο και "φυσικό" του τρόπο.

Από τα κλικ αυτά του χρήστη, εξάγουμε κάποιες σχετικές προτιμήσεις για τις σελίδες στις οποίες προτίμησε να πλοηγηθεί έναντι άλλων που αγνόησε. Επίσης βρίσκουμε τα χαρακτηριστικά των σελίδων που του παρουσίασε η μηχανή αναζήτησης, ανεξαρτήτως αν τα επισκέφτηκε ή όχι. Τα χαρακτηριστικά αυτά περιλαμβάνουν το βαθμό ομοιότητας του ερωτήματος με το αποτέλεσμα, το domain της ιστοσελίδας καθώς και την κατάταξη την οποία τους δίνει το Google.

Τα παραπάνω δεδομένα εισάγονται σε έναν ειδικό αλγόριθμο ταξινόμησης, ο οποίος αναλύοντας τα χαρακτηριστικά των σελίδων που ο χρήστης προτίμησε να πλοηγηθεί, δημιουργεί ένα μοντέλο το οποίο μπορεί να χρησιμοποιηθεί για να "μαντέψει" τις προτιμήσεις του χρήστη για μελλοντικά ερωτήματα. Έτσι χρησιμοποιώντας το μοντέλο αυτό μπορούμε να ανακατατάξουμε τα αποτελέσματα που παρέχει το Google με ένα τρόπο προσωποποιημένο στις προτιμήσεις του χρήστη.

1.3 Οργάνωση κειμένου

Στο κεφάλαιο αυτό έγινε μια εισαγωγή στον τρόπο λειτουργίας των μηχανών αναζήτησης στο διαδίκτυο καθώς και στις απαιτήσεις των χρηστών από αυτές.

Στο κεφάλαιο 2 αναλύονται τα συμπεράσματα που μπορούμε να εξάγουμε από τον τρόπο συμπεριφοράς των χρηστών μηχανών αναζήτησης και γίνεται μια θεωρητική εισαγωγή στα συστήματα που θα χρησιμοποιήσουμε.

Το κεφάλαιο 3 εξηγεί τη διαδικασία της καταγραφής της δραστηριότητας του χρήστη της μηχανής αναζήτησης, καθώς και τη δημιουργία ευρετηρίου από τις πληροφορίες αυτές.

Στο κεφάλαιο 4 αναλύεται η διαδικασία εξαγωγής των προτιμήσεων του χρήστη και η εκπαίδευση του μοντέλου που θα χρησιμοποιηθεί για την ανακατάταξη των αποτελεσμάτων.

Στο κεφάλαιο 5 παρουσιάζεται η διαδικασία αναζήτησης με ανακατάταξη των αποτελεσμάτων χρησιμοποιώντας το μοντέλο που δημιουργήθηκε.

Στο κεφάλαιο 6 γίνεται έλεγχος του συστήματος με μερικά παραδείγματα χρήσης.

Το κεφάλαιο 7 συνοψίζει τα συμπεράσματα της εργασίας αυτής και περιγράφει μερικές από τις επεκτάσεις οι οποίες μπορούν να γίνουν στο σύστημα αυτό.

2

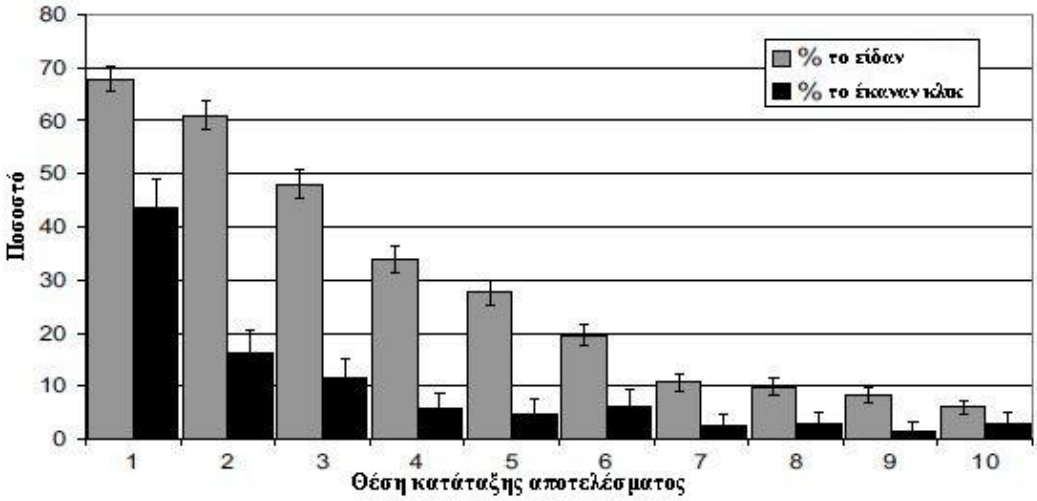
Θεωρητικό υπόβαθρο

Κατά κανόνα, υπάρχουν χιλιάδες ιστοσελίδες που περιέχουν τις λέξεις του ερωτήματος ενός χρήστη μηχανής αναζήτησης. Ο χρήστης όμως ενδιαφέρεται για ένα πολύ μικρότερο υποσύνολο από αυτές. Διαισθητικά, ένα καλό σύστημα Ανάκτησης Πληροφοριών (Information Retrieval) πρέπει να παρουσιάζει τα σχετικά έγγραφα υψηλά στην κατάταξη, με τα λιγότερο σχετικά να ακολουθούν πιο κάτω. Καθώς ο πληθυσμός των χρηστών μηχανής αναζήτησης ολοένα μεγαλώνει και γίνεται πιο ετερογενής, η εξατομίκευση των αποτελεσμάτων είναι κρίσιμη για να βοηθήσει την αναζήτηση να προοδεύσει πέρα από την μέχρι τώρα προσέγγιση της ενιαίας κατάταξης για όλους.

Οι περισσότερες προσεγγίσεις στο θέμα της εκπαίδευσης συναρτήσεων ανάκτησης από παραδείγματα στηρίζονται σε ρητές κρίσεις σχετικότητας των εγγράφων από ειδικούς ή στην επερώτηση του ίδιου του χρήστη για το ποια αποτελέσματα προτιμάει. Αυτό τις κάνει δύσκολες και ακριβές στη χρήση. Στόχος μας είναι η ανάπτυξη μιας μεθόδου που εκπαιδεύεται από το αρχείο καταγραφής των κλικ που κάνει ο χρήστης της μηχανής αναζήτησης. Αυτά τα δεδομένα καταγράφονται με πολύ χαμηλό κόστος, και σε πολύ μεγάλες ποσότητες. Η εκπαίδευση του μοντέλου προτιμήσεων γίνεται χρησιμοποιώντας την μέθοδο των Support Vector Machines, η οποία δουλεύει καλά ακόμα και με πολύ μεγάλα σύνολα δεδομένων.

2.1 Συμπεριφορά χρηστών μηχανών αναζήτησης

Οι περισσότεροι χρήστες όταν κάνουν ένα ερώτημα σε μία μηχανή αναζήτησης τείνουν να αξιολογούν μόνο λίγα από τα αποτελέσματα πριν να κάνουν κλικ σε κάποιο. Ο τρόπος με τον οποίο η μηχανή αναζήτησης παρουσιάζει τα αποτελέσματα στο χρήστη έχει ισχυρή επιρροή στο πώς οι χρήστες συμπεριφέρονται. Η πρώτη θέση προσδίδει αξιοπιστία στο αποτέλεσμα και επηρεάζει έντονα τη συμπεριφορά των χρηστών.



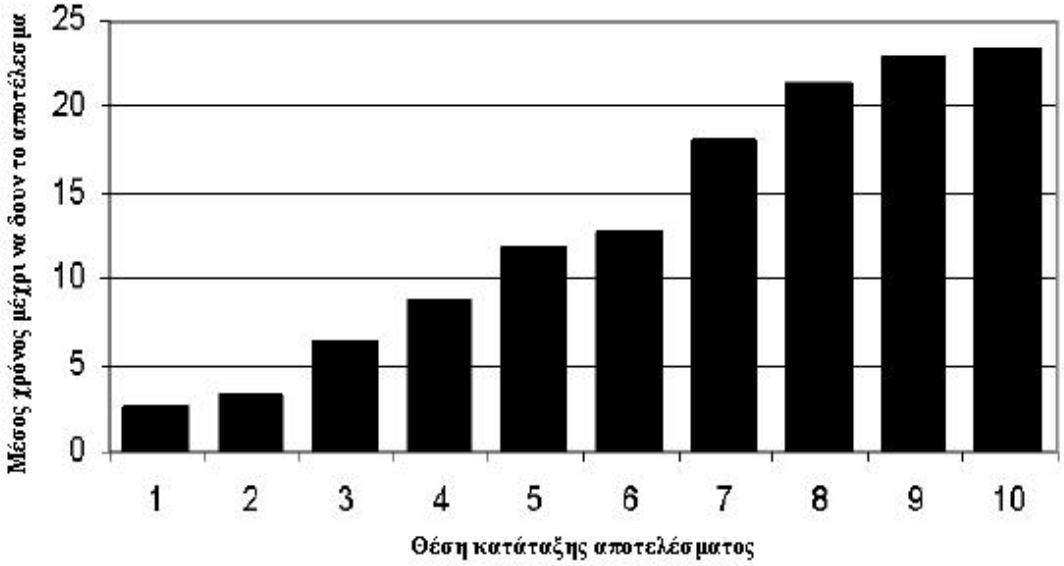
Εικόνα 1: Ποσοστό των φορών που έγινε κλικ σε ένα έγγραφο, με βάση τη θέση κατάταξής του

Στο παραπάνω σχήμα [JGP+05] οι γκρι μπάρες δείχνουν το ποσοστό των χρηστών που είδαν ένα αποτέλεσμα στη συγκεκριμένη θέση, ενώ οι μαύρες μπάρες δείχνουν το ποσοστό των χρηστών που το έκαναν κλικ. Από το παραπάνω σχήμα μπορούμε να βγάλουμε χρήσιμα συμπεράσματα σχετικά με το ποιους συνδέσμους βλέπουν οι χρήστες και ποιους πατάνε.

- Τα δύο πρώτα αποτελέσματα λαμβάνουν την περισσότερη προσοχή από τους χρήστες, ενώ τα στα πιο κάτω η προσοχή πέφτει γρήγορα. Παρ' όλα αυτά, πολλοί περισσότεροι χρήστες κάνουν κλικ στο πρώτο αποτέλεσμα παρά στο δεύτερο, ενώ συνήθως έχουν δει και τα δύο, δηλαδή φαίνεται να έχουν ένα επίπεδο εμπιστοσύνης στη μηχανή αναζήτησης.
- Στη θέση 7 παρατηρείται μια αλλαγή στον τρόπο συμπεριφοράς των χρηστών, τόσο στο πως βλέπουν τα αποτελέσματα όσο και στον αριθμό των κλικ. Κατ' αρχάς, οι σύνδεσμοι κάτω από αυτή τη θέση λαμβάνουν σημαντικά λιγότερη προσοχή από τους παραπάνω. Επιπλέον, αντίθετα με τις θέσεις 2 ως 6, τα αποτελέσματα 7 ως 10 λαμβάνουν περίπου ίδια σημασία. Αυτό μπορεί να εξηγηθεί από το γεγονός ότι συνήθως μόνο τα 5-6 πρώτα αποτελέσματα εμφανίζονται χωρίς να χρειάζεται κύλιση

πιο κάτω (scrolling). Αφού ο χρήστης αρχίσει το scrolling, η θέση του αποτελέσματος φαίνεται να έχει λιγότερη σημασία.

- Οι χρήστες συνήθως κοιτάζουν ένα αποτέλεσμα πιο κάτω από όποιο κάνουν κλικ.



Εικόνα 2: Μέσος χρόνος άφιξης (σε αριθμό καθηλώσεων του ματιού), με βάση τη θέση κατάταξης του αποτελέσματος

Το παραπάνω σχήμα [JGP+05] δείχνει πόσο χρόνο περίπου κάνει ο χρήστης μέχρι να δει το κάθε αποτέλεσμα. Ο χρόνος μετριέται σε καθηλώσεις του ματιού, δηλαδή σε ποια καθήλωση είδε πρώτη φορά ο χρήστης το ν-οστό αποτέλεσμα. Τα συμπεράσματα που βγάζουμε είναι:

- Οι χρήστες φαίνεται να σαρώνουν τα αποτελέσματα με τη σειρά από την αρχή μέχρι το τέλος.
- Οι χρήστες βλέπουν τα δύο πρώτα αποτελέσματα σχεδόν αμέσως ενώ υπάρχει ένα κενό για να προχωρήσουν στο τρίτο αποτέλεσμα και κάτω.
- Εμφανίζεται και εδώ το σημείο όπου οι χρήστες αρχίζουν την κύλιση, καθώς περνάει αρκετός χρόνος μέχρι να δουν τα αποτελέσματα 7 ως 10. Φαίνεται ότι οι χρήστες σαρώνουν αρκετά προσεκτικά τα πρώτα αποτελέσματα μέχρι να αναγκαστούν να καταφύγουν στο scrolling.

Από τα παραπάνω συμπεραίνουμε ότι οι χρήστες έχουν σημαντική εμπιστοσύνη στη δυνατότητα της μηχανής αναζήτησης να υπολογίσει την σχετικότητα μιας σελίδας, το οποίο επηρεάζει τη συμπεριφορά τους. Η ποιότητα, όμως, των αποτελεσμάτων εξακολουθεί να είναι σημαντικός παράγοντας της συμπεριφοράς τους, καθώς όσο μειώνεται η σχετικότητα των αποτελεσμάτων τόσο μειώνονται και τα κλικ στα αποτελέσματα αυτά.

2.2 Έμμεση ανατροφοδότηση (Implicit feedback)

2.2.1 Ρητή αξιολόγηση

Η χρήση μεθόδων ρητής αξιολόγησης είναι συχνή στην καθημερινή ζωή, π.χ. η βαθμολόγηση των εργασιών των μαθητών, ή η αξιολόγηση ανταγωνιστικών καταναλωτικών αγαθών. Αν και μερικές μορφές αξιολόγησης γίνονται σε ελεύθερη μορφή κειμένου (π.χ. κριτικές βιβλίων), είναι συχνές οι περιπτώσεις βαθμολογιών που γίνονται σε μια συμφωνημένη διακριτή κλίμακα (π.χ. βαθμολογία με αστέρια για ξενοδοχεία). Οι βαθμολογίες σε αυτές τις κλίμακες επιτρέπουν την στατιστική επεξεργασία τους για να βρεθούν μέσοι όροι, εύρη τιμών, κλπ. Η υλοποίηση αξιολογήσεων για υπολογιστικά συστήματα έχει ακολουθήσει κυρίως αυτήν την προσέγγιση.

Ένα κεντρικό χαρακτηριστικό των ρητών αξιολογήσεων είναι ότι ο κριτής πρέπει να εξετάσει το αντικείμενο και να του εκχωρήσει μια τιμή από την προκαθορισμένη κλίμακα βαθμολογίας. Αυτό επιβάλλει ένα γνωστικό κόστος για τον αξιολογητή, το οποίο δεν είναι απαραίτητα κάτι κακό (π.χ. η κοινωνία αναμένει οι καθηγητές να σκέφτονται για τους βαθμούς που δίνουν στους μαθητές). Η αξία πολλών μορφών βαθμολόγησης προκύπτει από αυτή την πνευματική προσπάθεια και παρέχει την αιτιολογία για την αμοιβή που συνοδεύει πολλές από αυτές.

2.2.2 Ιδιότητες έμμεσης ανατροφοδότησης

Κάθε φορά που ένας χρήστης διατυπώνει ένα ερώτημα ή κάνει κλικ σε ένα αποτέλεσμα αναζήτησης, παρέχεται ανατροφοδότηση (feedback) στη μηχανή αναζήτησης. Σε αντίθεση με έρευνες ή άλλες μορφές ρητής ανατροφοδότησης, η έμμεση ανατροφοδότηση αυτή είναι ουσιαστικά ελεύθερη, αντικατοπτρίζει τη φυσική χρήση της μηχανή αναζήτησης, και είναι συγκεκριμένη σε ένα χρήστη και μία συλλογή. Μια έξυπνη μηχανή αναζήτησης θα μπορούσε να χρησιμοποιήσει αυτή την έμμεση ανατροφοδότηση για να μάθει λειτουργίες εξατομικευμένης κατάταξης.

Το βασικό κίνητρο για τη χρήση της έμμεσης ανατροφοδότησης είναι ότι καταργεί το κόστος για τον χρήστη της εξέτασης και βαθμολόγησης κάθε στοιχείου ρητά. Ενώ εξακολουθεί να υπάρχει ένα υπολογιστικό κόστος αποθήκευσης και επεξεργασίας των δεδομένων, αυτό μπορεί να είναι αόρατο από το χρήστη. Σε ένα δικτυωμένο περιβάλλον συνήθως είναι δύσκολο για το χρήστη να διαχωρίσει την καθυστέρηση του δικτύου από την επιπλέον επεξεργασία της εφαρμογής. Αν και υπάρχουν σαφώς όρια στις ανοχές του χρήστη, η

αποθήκευση και μεταφορά των δεδομένων έμμεσης ανατροφοδότησης δεν είναι υπολογιστικά έντονο έργο.

Επίσης σε συστήματα ρητής βαθμολόγησης των στοιχείων, αν ο χρήστης δεν αντιλαμβάνεται κάποιο άμεσο όφελος, τότε μπορεί να συνεχίσει να χρησιμοποιεί το σύστημα χωρίς όμως να κάνει βαθμολόγηση. Έτσι το σύστημα αυτό θα μπορούσε να οδηγηθεί σε έλλειψη οποιασδήποτε διαβάθμισης.

Το πλεονέκτημα της έμμεσης ανατροφοδότησης σε σχέση με τη ρητή είναι ότι μπορούν να συλλέγονται δεδομένα σε πολύ χαμηλότερο κόστος, σε πολύ μεγαλύτερες ποσότητες, και χωρίς επιβάρυνση για το χρήστη του συστήματος ανάκτησης τους. Ωστόσο, η έμμεση ανατροφοδότηση είναι πιο δύσκολο να ερμηνευθεί και δυνητικά θορυβώδης.

Οι συμπεριφορές των χρηστών που δηλώνουν ένα έμμεσο ενδιαφέρον για την ιστοσελίδα μπορούν να κατηγοριοποιηθούν ως εξής [CLW+01]:

- *Δείκτες ενδιαφέροντος σήμανσης*. Διάφορες ενέργειες του χρήστη μπορούν να θεωρηθούν σαν μια μορφή σήμανσης, που μπορεί να ερμηνευτεί σαν δείκτης ενδιαφέροντος. Μερικές από αυτές είναι η αποθήκευση σε αρχείο ή προσθήκη στα αγαπημένα, η εκτύπωση, ή η προώθηση του εγγράφου μέσω ηλεκτρονικού ταχυδρομείου.
- *Δείκτες ενδιαφέροντος χειρισμού*. Μερικές ενέργειες, όπως η αντιγραφή και η επικόλληση, μπορούν να θεωρηθούν σαν δείκτες ενδιαφέροντος. Άλλες περιλαμβάνουν το άνοιγμα ενός νέου παραθύρου του περιηγητή (π.χ. πιθανώς ο χρήστης να θέλει να κρατήσει ανοιχτό το υπάρχον παράθυρο γιατί η σελίδα είναι ενδιαφέρουσα), η αναζήτηση στη σελίδα για κάποιο κείμενο, ή η κύλιση (scrolling) μέσα στο έγγραφο
- *Δείκτες ενδιαφέροντος πλοήγησης*. Αν ο χρήστης ξοδεύει χρόνο με τη σελίδα ανοιχτή, ακολουθεί ή δεν ακολουθεί ένα σύνδεσμο, τότε μπορούμε να θεωρήσουμε αυτές τις ενέργειες σαν δείκτες πλοήγησης.
- *Δείκτες ενδιαφέροντος επανάληψης*. Είναι λογικό να υποθέσουμε ότι όταν κάνουμε κάτι σε μεγάλες ποσότητες, αυτό φανερώνει μεγαλύτερο ενδιαφέρον. Έτσι όταν ο χρήστης επισκέπτεται πολλές φορές την ίδια σελίδα ή ξοδεύει περισσότερο χρόνο σε μια σελίδα, μπορούμε να συμπεράνουμε ότι είναι ενδιαφέρουσα για αυτόν.

Κάποιες από αυτές τις πηγές δεδομένων έχουν και πρόσθετη πληροφορία (π.χ. σε μια ενέργεια αγοράς ενός αντικειμένου αντιστοιχεί και μια τιμή, εκτός από την πληροφορία ότι ο χρήστης ενδιαφέρεται για το αντικείμενο αυτό). Είναι λογικό να υποθέσουμε περισσότερα από την αγορά του αντικειμένου, παρά από μια απλή επιθεώρησή του. Καθώς το διαδίκτυο γίνεται ένα ολοένα και πιο εμπορικό περιβάλλον, αυξάνονται και οι πληροφορίες αυτού του

τύπου. Οι πληροφορίες αυτές μπορούν να χρησιμοποιηθούν για να προτείνουν στο χρήστη άλλα προϊόντα που πιθανώς να του αρέσουν.

Στην εργασία αυτή θα χρησιμοποιήσουμε δεδομένα feedback μόνο από τις ακολουθίες των κλικ που κάνουν οι χρήστες της μηχανής αναζήτησης.

2.2.3 Clickstream δεδομένα

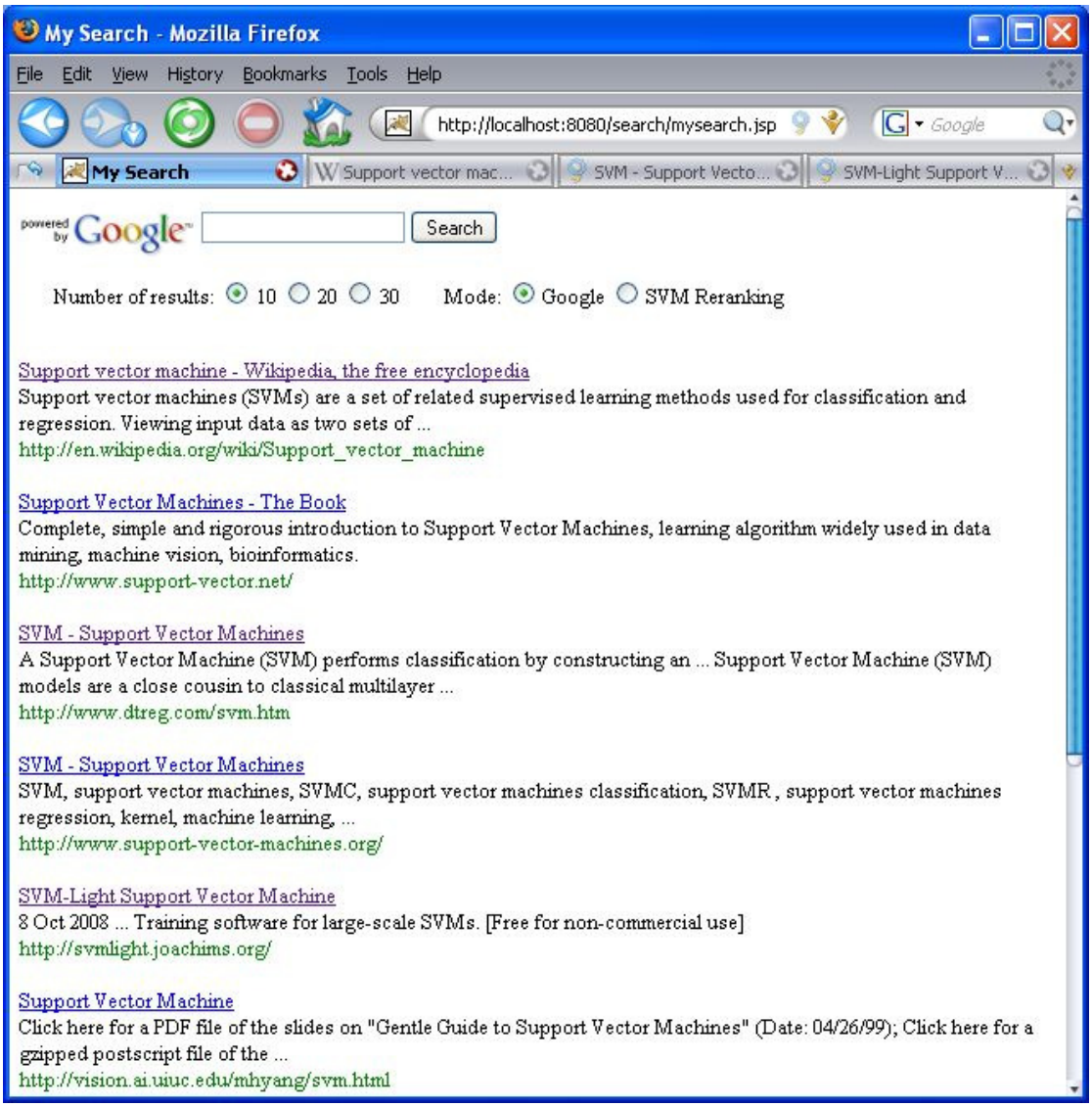
Με τον όρο clickstream εννοούμε το εικονικό μονοπάτι που αφήνει ένας χρήστης καθώς περιηγείται στο διαδίκτυο. Το εικονικό αυτό μονοπάτι, που το συνθέτουν όλα τα κλικ που κάνει ο χρήστης, καταγράφει όλη τη δραστηριότητά του στο διαδίκτυο, συμπεριλαμβανομένων των ιστοσελίδων που επισκέπτεται, για πόσο χρονικό διάστημα ήταν σε κάθε σελίδα καθώς και με ποια σειρά τις επισκέφθηκε. Τα clickstream δεδομένα είναι πολύ χρήσιμα για την ανάλυση της δραστηριότητας στο διαδίκτυο, τη δοκιμή λογισμικού, την έρευνα αγοράς, καθώς και για την ανάλυση της παραγωγικότητας των εργαζομένων.

Η μη εγκεκριμένη από το χρήστη συλλογή των clickstream δεδομένων μπορεί να εγείρει ανησυχίες για την ιδιωτική ζωή, ιδίως εφόσον ορισμένοι πάροχοι υπηρεσιών Internet, προκειμένου να ενισχύσουν τα έσοδά τους, πωλούν τα clickstream δεδομένα των χρηστών τους. Αν και η πρακτική αυτή δεν μπορεί να εντοπίσει μεμονωμένους χρήστες άμεσα, είναι όμως δυνατόν να εντοπιστούν έμμεσα συγκεκριμένοι χρήστες, με βάση το ιστορικό πλοήγησής τους. Οι περισσότεροι χρήστες δε γνωρίζουν την πρακτική αυτή, καθώς και την πιθανότητα έκθεσης της ιδιωτικής τους ζωής.

Άλλοι οργανισμοί χρησιμοποιούν τη συλλογή δεδομένων κλικ με την άδεια του χρήστη για έρευνες, ή για να επιτρέψουν στο χρήστη να επιστρέψει εύκολα σε μια σελίδα που έχει ήδη επισκεφθεί. Παρόμοια, και στη δική μας περίπτωση, ο χρήστης εν γνώσει του χρησιμοποιεί τη μηχανή αναζήτησης, προκειμένου να λάβει προσωποποιημένα αποτελέσματα βασισμένα στο ιστορικό χρήσης του.

Τα clickstream δεδομένα στης μηχανές αναζήτησης μπορούν να θεωρηθούν ως τριπλέτες (q, r, c), όπου q το ερώτημα, r η κατάταξη που παρουσιάστηκε στο χρήστη και c το σύνολο των συνδέσμων στους οποίους έκανε κλικ ο χρήστης.

Παρακάτω φαίνεται ένα παράδειγμα που ο χρήστης έκανε το ερώτημα “support vector machine”, πήρε την κατάταξη που φαίνεται, και έκανε κλικ στα αποτελέσματα 1, 3 και 5. Όλα αυτά τα δεδομένα καταγράφονται στο αρχείο ιστορικού για περαιτέρω επεξεργασία. Έτσι, βλέπουμε ότι η ποσότητα των δεδομένων που είναι διαθέσιμη είναι σχεδόν απεριόριστη, αφού με τόσο εύκολο τρόπο παίρνουμε τόσο πολλή πληροφορία.



Εικόνα 3: Κατάταξη για το ερώτημα "support vector machine"

Τα clickstream δεδομένα μπορούν να αποθηκευτούν με μικρό επιπλέον κόστος και χωρίς να διακινδυνεύουν τη λειτουργικότητα και χρησιμότητα της μηχανής αναζήτησης. Συγκεκριμένα, σε σχέση με μεθόδους ρητής ανάδρασης (explicit feedback), δεν προσθέτει καθόλου επιπλέον κόστος στο χρήστη. Το ερώτημα και η κατάταξη που επιστρέφεται μπορεί εύκολα να αποθηκευτεί όταν παρουσιάζεται στο χρήστη. Για την καταγραφή των κλικ, ένας απλός διακομιστής μεσολάβησης (proxy server) μπορεί να κρατάει ένα αρχείο ιστορικού (log file).

Σε κάθε ερώτημα εκχωρείται ένα μοναδικό αναγνωριστικό ID, το οποίο αποθηκεύεται στο log αρχείο μαζί με τις λέξεις – κλειδιά του ερωτήματος και την κατάταξη που επιστρέφεται στο χρήστη. Οι σύνδεσμοι στην σελίδα των αποτελεσμάτων δεν οδηγούν απευθείας στο έγγραφο, αλλά δείχνουν σε έναν proxy server. Οι σύνδεσμοι αυτοί ενσωματώνουν το αναγνωριστικό ID του ερωτήματος και την URL διεύθυνση του εγγράφου. Όταν ο χρήστης

κάνει κλικ στο σύνδεσμο, ο proxy server καταγράφει το URL και το αναγνωριστικό ID του ερωτήματος στο αρχείο ιστορικού. Κατόπιν ο proxy server προωθεί τον χρήστη στη διεύθυνση URL. Αυτή η διαδικασία μπορεί να γίνει διαφανής για το χρήστη και να μην επηρεάσει την επίδοση του συστήματος.

2.2.4 Σχετικές προτιμήσεις που εξάγονται από τα clickstream δεδομένα

Όπως εξηγήσαμε και παραπάνω στην παράγραφο 2.1, είναι δύσκολο να ερμηνεύσουμε τα κλικ των χρηστών σε απόλυτη κλίμακα. Έτσι θα προσπαθήσουμε να εξάγουμε έμμεσα κάποιες προτιμήσεις από ζευγάρια αποτελεσμάτων. Η στρατηγική που θα χρησιμοποιήσουμε είναι βασισμένη στην ιδέα ότι δεν πρέπει να χρησιμοποιούνται σαν ανάδραση (feedback) μόνο τα κλικ του χρήστη, αλλά και το γεγονός ότι σε κάποιους συνδέσμους *δεν* έκανε κλικ.

Ας πάρουμε την υποθετική κατάταξη των αποτελεσμάτων l_1 ως l_7 και ας υποθέσουμε ότι ο χρήστης έκανε κλικ στους συνδέσμους $\boldsymbol{l_1}$, $\boldsymbol{l_3}$, και $\boldsymbol{l_5}$.

$$\boldsymbol{l_1} \quad l_2 \quad \boldsymbol{l_3} \quad l_4 \quad \boldsymbol{l_5} \quad l_6 \quad l_7$$

Ενώ είναι δύσκολο να υποθέσουμε ότι τα αποτελέσματα $\boldsymbol{l_1}$, $\boldsymbol{l_3}$, και $\boldsymbol{l_5}$ είναι σχετικά σε απόλυτη κλίμακα, είναι πολύ πιο λογικό να υποθέσουμε ότι το αποτέλεσμα $\boldsymbol{l_3}$ είναι πιο σχετικό από το l_2. Όπως είπαμε οι χρήστες σαρώνουν τα αποτελέσματα από πάνω προς τα κάτω, και άρα στο παράδειγμά μας, ο χρήστης, πριν κάνει κλικ στο $\boldsymbol{l_3}$, είδε το l_2 και πήρε την απόφαση να μην πατήσει πάνω του. Αυτό δείχνει την προτίμηση του για το $\boldsymbol{l_3}$ σε σχέση με το l_2. Παρόμοια μπορούμε να εξάγουμε ότι το $\boldsymbol{l_5}$ είναι πιο σχετικό από τα l_2 και l_4.

Σημειώνοντας με rel() την αξιολόγηση σχετικότητας του χρήστη αυτού, παίρνουμε την παρακάτω πληροφορία

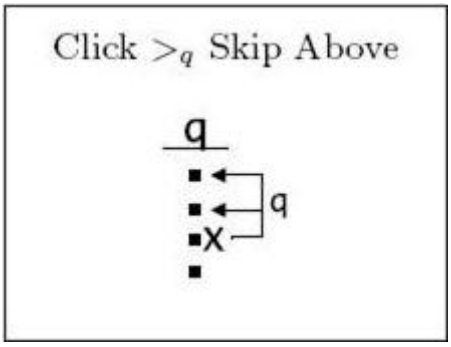
$$\text{rel}(\boldsymbol{l_3}) > \text{rel}(l_2), \quad \text{rel}(\boldsymbol{l_5}) > \text{rel}(l_2), \quad \text{rel}(\boldsymbol{l_5}) > \text{rel}(l_4)$$

Οι σχετικές προτιμήσεις που εξάγουμε από κάθε ζευγάρι αποτελεσμάτων αντιτίθεται στη μεροληψία παρουσίασης των αποτελεσμάτων, όπου το υψηλότερο στην κατάταξη αποτέλεσμα φαίνεται σαν πιο σχετικό.

Η στρατηγική που δείξαμε στο παραπάνω παράδειγμα και που θα χρησιμοποιήσουμε για την εξαγωγή σχετικών προτιμήσεων του χρήστη περιγράφεται παρακάτω [JGP+05]:

Στρατηγική 1 – "Click > Skip above"

Για μία κατάταξη (l_1, l_2, l_3, ...) και ένα σύνολο C που περιλαμβάνει τις θέσεις κατάταξης των αποτελεσμάτων στα οποία έγινε κλικ, εξάγεται η προτίμηση $rel(l_i) > rel(l_j)$ για όλα τα ζευγάρια $1 \leq j < 1$, με $i \in C$ και $j \notin C$.



Εικόνα 4: Στρατηγική "Click > Skip above". Δεδομένου ενός ερωτήματος q, οι τελείες αναπαριστούν τα αποτελέσματα και τα X εκείνα που έγιναν κλικ.

Δηλαδή δεδομένου ενός αποτελέσματος στο οποίο έγινε κλικ, κάθε αποτέλεσμα υψηλότερης κατάταξης στο οποίο δεν έγινε κλικ έχει μικρότερη σημασία για το χρήστη. Αυτό το συμπέρασμα προέρχεται από το ότι οι χρήστες βλέπουν τα αποτελέσματα σε σειρά, και ένας χρήστης είναι απίθανο να κάνει κλικ σε ένα έγγραφο που θεωρεί λιγότερο σημαντικό από ένα άλλο πιο σχετικό έγγραφο που έχει δει.

2.2.5 Προσθέτοντας υπάρχουσα γνώση

Εκτός από την πληροφορία που παίρνουμε από τις σχετικές προτιμήσεις που δείξαμε στην προηγούμενη παράγραφο, υπάρχει επιπλέον προηγούμενη γνώση η οποία πρέπει να ενσωματωθεί στο πρόβλημα. Αν δεν υπάρχει καμία άλλη πληροφορία, τα έγγραφα με υψηλότερη θέση στην αρχική κατάταξη θα πρέπει να έχουν και υψηλότερη θέση στην κατάταξη του εκπαιδευόμενου συστήματος. Αυτό είναι διαισθητικό επειδή κατά μέσο όρο θα περιμέναμε ότι η σχετικότητα του εγγράφου να είναι μια φθίνουσα συνάρτηση της αρχικής κατάταξης των εγγράφων, εκτός αν η αρχική συνάρτηση κατάταξης είναι πολύ φτωχή ποιοτικά.

Υπάρχει και πρακτική σημασία στο να προστεθούν τέτοιοι επιπλέον περιορισμοί προηγούμενης γνώσης. Στα δεδομένα μας της εκπαίδευσης, σχεδόν όλες οι σχετικές προτιμήσεις λένε ότι ένα χαμηλότερο σε κατάταξη έγγραφο προτιμάται από ένα άλλο υψηλότερης κατάταξης. Χωρίς επιπλέον περιορισμούς, μια απλή και ανεπιθύμητη λύση στο πρόβλημα θα ήταν να αντιστρέψουμε την αρχική κατάταξη. Αυτό φαίνεται και στο παράδειγμα που δείξαμε πριν, όπου θα πληρούνταν οι προτιμήσεις “Click > Skip above” αν αντιστρεφόταν η κατάταξη των αποτελεσμάτων.

Οι περιορισμοί προηγούμενης γνώσης πρέπει λοιπόν να είναι κάποιοι που να επιβεβαιώνουν την αρχική κατάταξη της μηχανής αναζήτησης, καθώς αλλιώς θα έχουμε μόνο προτιμήσεις

που δείχνουν αντιστροφή της σειράς της κατάταξης. Συγκεκριμένα εμείς θα προσθέσουμε περιορισμούς που επιβεβαιώνουν την αρχική κατάταξη της μηχανής αναζήτησης για τα αποτελέσματα στα οποία έκανε κλικ ο χρήστης. Δηλαδή για το παράδειγμα που δείξαμε και πριν, όπου ο χρήστης έκανε κλικ στους συνδέσμους $\boldsymbol{l_1}$, $\boldsymbol{l_3}$, και $\boldsymbol{l_5}$

$$\boldsymbol{l_1} \quad l_2 \quad \boldsymbol{l_3} \quad l_4 \quad \boldsymbol{l_5} \quad l_6 \quad l_7$$

εξάγουμε τους παρακάτω περιορισμούς:

$$\text{rel}(\boldsymbol{l_1}) > \text{rel}(\boldsymbol{l_3}), \quad \text{rel}(\boldsymbol{l_1}) > \text{rel}(\boldsymbol{l_5}), \quad \text{rel}(\boldsymbol{l_3}) > \text{rel}(\boldsymbol{l_5})$$

2.2.6 Αλυσίδες ερωτημάτων

Καθώς οι χρήστες αναζητούν, είναι διαπιστωμένο ότι συχνά αναδιατυπώνουν τα ερωτήματά τους, προσθέτοντας ή αφαιρώντας λέξεις – κλειδιά ή αναδιατυπώνοντας ολικά το ερώτημα, προκειμένου να επιλύσουν κάποια προηγούμενη ασάφεια του ερωτήματός τους. Θα αναφερόμαστε σε μια σειρά από αναδιατυπωμένα ερωτήματα ως μία αλυσίδα ερωτημάτων.

Όταν τα ερωτήματα θεωρούνται ανεξάρτητα, τα αρχεία καταγραφής παρέχουν συνήθως πληροφορίες έμμεσου feedback μόνο για λίγα αποτελέσματα στο πάνω μέρος του συνόλου των αποτελεσμάτων, αφού οι χρήστες πολύ σπάνια κοιτάζουν πιο κάτω στη λίστα. Το πλεονέκτημα της χρήσης των αλυσίδων ερωτημάτων είναι ότι μπορούμε να εξάγουμε επιπλέουν σχετικές προτιμήσεις για πολλά περισσότερα έγγραφα που έχει δει ο χρήστης κατά τη διάρκεια μιας ολόκληρης συνόδου (session) αναζήτησης.

Παρακάτω περιγράφονται δύο στρατηγικές για εξαγωγή έμμεσων προτιμήσεων από αλυσίδες ερωτημάτων [RJ05]:

Στρατηγική 2 – "Click > Skip Earlier QC"

Για μία κατάταξη (l_1, l_2, l_3, ...) που ακολουθείται από την κατάταξη (l_1', l_2', l_3', ...) στην ίδια αλυσίδα ερωτημάτων και σύνολα C και C' που περιλαμβάνουν τις θέσεις κατάταξης των αποτελεσμάτων στα οποία έγινε κλικ στην αντίστοιχη κατάταξη, εξάγεται η προτίμηση $rel(l_i')>rel(l_j)$ για όλα τα ζευγάρια $i \in C$ και $j < max(C)$, με $j \notin C$.

Η στρατηγική αυτή είναι ουσιαστικά μια επέκταση της στρατηγικής 1 (Click > Skip above) για πολλά σύνολα αποτελεσμάτων. Ακολουθεί ένα παράδειγμα μιας αλυσίδας ερωτημάτων με τρία ερωτήματα.

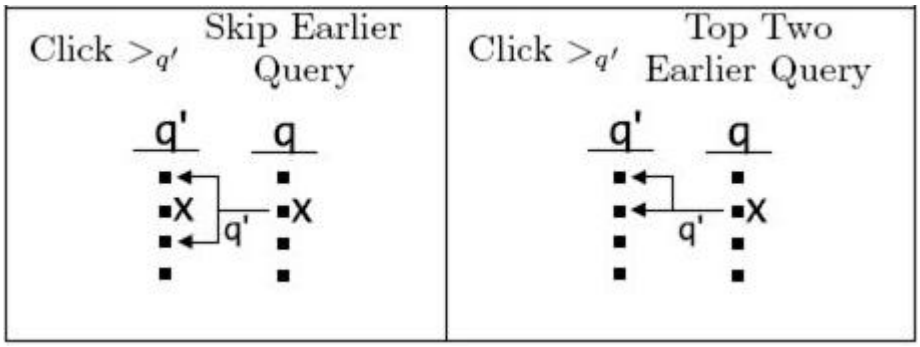
$$q_1: \quad l_{11} \quad l_{12} \quad l_{13} \quad l_{14} \quad l_{15} \quad l_{16} \quad l_{17}$$

$$q_2: \quad \boldsymbol{l_{21}} \quad l_{22} \quad \boldsymbol{l_{23}} \quad l_{24} \quad \boldsymbol{l_{25}} \quad l_{26} \quad l_{27}$$

$$q_3: \quad l_{31} \quad \boldsymbol{l_{32}} \quad l_{33} \quad l_{34} \quad l_{35} \quad l_{36} \quad l_{37}$$

Για το παράδειγμα αυτό, η στρατηγική “Click > Skip Earlier QC” παράγει τις εξής προτιμήσεις:

$$\text{rel}(\boldsymbol{l_{32}}) > \text{rel}(l_{22}),\ \ \text{rel}(\boldsymbol{l_{32}}) > \text{rel}(l_{24})$$



Εικόνα 5: Στρατηγικές εξαγωγής προτιμήσεων από αλυσίδες ερωτημάτων

Στην περίπτωση που κανένα αποτέλεσμα δεν έγινε κλικ στο προηγούμενο ερώτημα, όπως στο q_1 στο παράδειγμά μας, χρησιμοποιούμε την παρακάτω τεχνική για εξαγωγή προτιμήσεων [RJ05]:

Στρατηγική 3 – “Click > Top Two No Click Earlier QC”

Για μία κατάταξη (l_1, l_2, l_3, ...) όπου κανένα αποτέλεσμα δεν έλαβε κλικ και ακολουθήθηκε από την κατάταξη (l_1', l_2', l_3', ...) στην ίδια αλυσίδα ερωτημάτων και σύνολο C' που περιλαμβάνει τις θέσεις κατάταξης των αποτελεσμάτων στα οποία έγινε κλικ στην αντίστοιχη κατάταξη, εξάγεται η προτίμηση rel(l_i') > rel(l_1) και rel(l_i') > rel(l_2) για όλα τα $i \in C'$.

Η στρατηγική αυτή εξηγείται από το γεγονός ότι οι χρήστες συνήθως κοιτάζουν τουλάχιστον τα δύο πρώτα αποτελέσματα πριν αναδιατυπώσουν το ερώτημα τους. Για το παραπάνω παράδειγμα, η στρατηγική “Click > Top Two No Click Earlier QC” παράγει τις εξής προτιμήσεις:

$$\text{rel}(\boldsymbol{l_{21}}) > \text{rel}(l_{11}),\quad \text{rel}(\boldsymbol{l_{23}}) > \text{rel}(l_{11}),\quad \text{rel}(\boldsymbol{l_{25}}) > \text{rel}(l_{11}),\quad \text{rel}(\boldsymbol{l_{32}}) > \text{rel}(l_{11})$$

$$\text{rel}(\boldsymbol{l_{21}}) > \text{rel}(l_{12}),\quad \text{rel}(\boldsymbol{l_{23}}) > \text{rel}(l_{12}),\quad \text{rel}(\boldsymbol{l_{25}}) > \text{rel}(l_{12}),\quad \text{rel}(\boldsymbol{l_{32}}) > \text{rel}(l_{12})$$

Έρευνες έχουν δείξει [JGP+07] ότι οι παραπάνω στρατηγικές είναι εξαιρετικά ακριβείς και το επίπεδο του θορύβου (λανθασμένες προτιμήσεις) κυμαίνεται ικανοποιητικά χαμηλά. Συγκεκριμένα, παραθέτουμε τα παρακάτω στατιστικά στοιχεία για τις τεχνικές που αναλύσαμε. Η σύγκριση έγινε με ρητές κρίσεις ειδικών στο θέμα και δείχνει το ποσοστό συμφωνίας τους με τα αποτελέσματα των έμμεσων τεχνικών.

Στρατηγική Εξαγωγής Έμμεσων Προτιμήσεων	*Ποσοστό Συμφωνίας*
Click > Skip Above	83.1 ± 4.4
Click > Skip Earlier QC	84.5 ± 16.4
Click > Top Two No Click Earlier QC	88.9 ± 12.9

Πίνακας 1: Ακρίβεια των στρατηγικών εξαγωγής σχετικών προτιμήσεων

2.3 Ανάκτηση πληροφορίας (Information retrieval)

2.3.1 Αυτόματη ανάλυση κειμένου

Ένα σημαντικό πρόβλημα στα συστήματα ανάκτησης πληροφοριών είναι η αναπαράσταση του περιεχομένου των εγγράφων. Πρέπει να εφαρμοστεί ανάλυση κειμένου για να εκχωρηθεί σε κάθε έγγραφο ένα σύνολο περιγραφητών (descriptors) ικανών να εκπροσωπήσουν το περιεχόμενό του. Η ανάθεση των descriptors θα πρέπει να πληροί τρεις σκοπούς:

1. Να επιτρέπει την εύρεση των αντικειμένων που ασχολούνται με θέματα που ενδιαφέρουν το χρήστη.
2. Να συσχετίζουν αντικείμενα μεταξύ τους, και έτσι να συσχετίζουν θεματικές περιοχές, με τον εντοπισμό διακριτών αντικειμένων που ασχολούνται με παρόμοιες ή σχετικές θεματικές περιοχές.
3. Να προβλέπουν τη σχετικότητα των επιμέρους στοιχείων πληροφοριών με συγκεκριμένες απαιτήσεις πληροφόρησης, μέσω της χρήσης όρων ευρετηρίου με σαφώς καθορισμένο νόημα.

Η διαδικασία της εκχώρησης descriptors μπορεί να πραγματοποιείται αυτόματα, χρησιμοποιώντας ένα σύστημα ηλεκτρονικού υπολογιστή, ή χειροκίνητα, χρησιμοποιώντας ειδικούς στο θέμα. Εμείς θα ασχοληθούμε μόνο με την πρώτη, δηλαδή την αυτόματη ευρετηρίαση (*automatic indexing)*. Εκτός από την εκχώρηση descriptors, η ανάλυση του κειμένου θα πρέπει να παρέχει έναν τρόπο μέτρησης της σημασίας κάθε descriptor για σκοπούς αναγνώρισης περιεχομένου, το οποίο είναι γνωστό ως στάθμιση (*weighting)*. Ακολουθεί μια περιγραφή με βάση στατιστικές μεθόδους και των δύο διαδικασιών ανάλυσης κειμένου και ανάκτησης πληροφοριών, της αυτόματης ευρετηρίασης και της στάθμισης.

2.3.2 Αυτόματη ευρετηρίαση (Automatic indexing)

Στο επίκεντρο όλων των μηχανών αναζήτησης είναι η έννοια του ευρετηρίου: η επεξεργασία των αρχικών δεδομένων σε μια ιδιαίτερα αποδοτική δομή δεδομένων που επιτρέπει την ταχεία πρόσβαση σε τυχαίες λέξεις αποθηκευμένες στο εσωτερικό του και διευκολύνει τη γρήγορη αναζήτηση.

Η διαδικασία του indexing αρχίζει με την αναγνώριση κάθε ξεχωριστής λέξης από το έγγραφο ως πιθανός περιγραφέας (descriptor). Μετά την αναγνώριση των λέξεων, πρέπει να εξαλειφθούν οι stop-words δηλαδή λέξεις υψηλής συχνότητας που είναι φτωχοί descriptors (π.χ. άρθρα κλπ). Τα stop-words δεν μόνο είναι άχρηστα για την αναγνώριση περιεχομένου, αλλά καταλαμβάνουν και περίπου το 50% του κειμένου του εγγράφου.

Το επόμενο βήμα είναι η κατάργηση των προθεμάτων και επιθεμάτων των λέξεων, έτσι ώστε κάθε λέξη να είναι μειωμένη στη ρίζα της (stem). Η διαδικασία αυτή καλείται stemming και χρησιμοποιείται για τη βελτίωση της αποτελεσματικότητας της ανάκτησης και τη μείωση του μεγέθους του ευρετηρίου.

Το stemming γενικά είτε βελτιώνει την αποτελεσματικότητα της ανάκτησης ή δεν έχει καμία επίδραση, αλλά γενικά οι αντίστοιχοι αλγόριθμοι κάνουν συχνά λάθη. Για παράδειγμα ένας αλγόριθμος μειώνει τις λέξεις «public» και «publication» στη ρίζα «public», αν και οι δύο λέξεις είναι διαφορετικές και πρέπει να διακρίνονται. Η υπόθεση που γίνεται είναι ότι το ποσοστό αυτών των σφαλμάτων δεν έχει πραγματική επίδραση στην απόδοση της ανάκτησης.

Κάθε ρίζα που έχει ανιχνευθεί από ένα έγγραφο μπορεί να είναι ένας από τους descriptors του, που είναι γνωστοί ως όροι ευρετηρίου. Ωστόσο οι όροι ευρετηρίου περιλαμβάνουν και άλλους όρους πέρα από τις ρίζες που έχουν εξαχθεί από τα έγγραφα. Χρησιμοποιούνται λεξικά, έτσι ώστε να περιλαμβάνονται και οι έννοιες που σχετίζονται με τους όρους ευρετηρίου με την ελπίδα της διεύρυνσης της ερμηνείας τους. Ευρεία ερμηνεία μπορεί επίσης να επιτευχθεί με τη χρήση ενός θησαυρού (*thesaurus)*, που παρέχει μια ομαδοποίηση σε κατηγορίες των όρων που χρησιμοποιούνται σε μια δεδομένη θεματική περιοχή. Η διαδικασία ονομάζεται αυτόματη ταξινόμηση των λέξεων – κλειδιών (*automatic keyword classification)* και μπορεί να αξιοποιηθεί είτε με την αντικατάσταση κάθε όρου του συνόλου των descriptors από το όνομα της κατηγορίας που ανήκει, ή με την αντικατάσταση κάθε όρου του συνόλου των descriptors από όλες τις λέξεις – κλειδιά της κατηγορία που ανήκει.

Αφού παραχθούν όσο το δυνατόν περισσότεροι όροι ευρετηρίου, ακολουθεί η διαδικασία της στάθμισης, προκειμένου να προσδιοριστούν εκείνοι οι όροι που έχουν τη μεγαλύτερη σημασία για την αναγνώριση του περιεχομένου.

2.3.3 Στάθμιση (Weighting)

Η διαδικασία της στάθμισης εκχωρεί ένα βάρος σε κάθε όρο του ευρετηρίου αναλόγως τη σημασία του για την αναγνώριση περιεχομένου. Οι περισσότερες μέθοδοι στάθμισης είναι βασισμένοι στην παρατήρηση ότι η συχνότητα της εμφάνισης μια λέξης σε ένα κείμενο σχετίζεται με τη σημασία της για την αναπαράσταση περιεχομένου. Αν οι ξεχωριστές λέξεις του εγγράφου παρουσιαστούν με φθίνουσα σειρά συχνότητας εμφάνισής τους στο έγγραφο αυτό, τότε συνήθως παρατηρείται ο νόμος σταθερής κατάταξης-συχνότητας του Zipf (*constant rank-frequency law).* Ο νόμος αυτός λέει ότι η συχνότητα μιας λέξης πολλαπλασιασμένης με την σειρά κατάταξής της ισούται με τη συχνότητα μιας άλλης λέξης πολλαπλασιασμένης με την σειρά κατάταξής της. Ο νόμος αυτός εξηγείται από το γεγονός ότι συνήθως οι άνθρωποι προτιμούν να επαναλαμβάνουν λέξεις που έχουν χρησιμοποιήσει ήδη παρά να χρησιμοποιούν νέες,

Οι ιδιότητες που χαρακτηρίζουν μια λέξη ως χρήσιμο όρο ευρετηρίου είναι οι παρακάτω:

- Πρέπει να σχετίζεται με το περιεχόμενο του εγγράφου ώστε να μπορεί να είναι ανακτήσιμο όταν χρειαστεί, έχοντας έτσι ανακτήσει ένα μεγάλο μέρος των σχετικών εγγράφων. Το ποσοστό των σχετικών εγγράφων που έχουν ανακτηθεί είναι γνωστό ως ανάκληση *(recall)*.
- Πρέπει να ξεχωρίζει το έγγραφο της από το υπόλοιπα, για να αποτρέψει την ανάκτηση και αντικειμένων που δε θέλουμε, έχοντας έτσι σχετικό ένα μεγάλο μέρος των εγγράφων που έχουν ανακτηθεί. Το ποσοστό των ανακτημένων εγγράφων που είναι σχετικά είναι γνωστό ως ακρίβεια (*precision).*

Όροι με μεγάλη συχνότητα εμφάνισης στο έγγραφο φαίνεται να είναι χρήσιμοι για την πρώτη απαίτηση. Αυτό υποδηλώνει τη χρήση ενός παράγοντα *συχνότητας όρου (term frequency – tf*) σαν πρώτο κομμάτι του συστήματος στάθμισης

Όροι με χαμηλή συχνότητα εμφάνισης σε όλη τη συλλογή εγγράφων φαίνεται να είναι χρήσιμοι για τη δεύτερη απαίτηση. Αυτό υποδηλώνει τη χρήση ενός παράγοντα αντίστροφης συχνότητας εγγράφου (*inverse document frequency – idf)* σαν δεύτερο κομμάτι του συστήματος στάθμισης.

Χρησιμοποιώντας το γινόμενο του *term frequency* tf_{ij} και του *inverse document frequency* idf_j ενός όρου *j* ενός εγγράφου *i*, μπορούμε να αποκτήσουμε ένα καλό μέτρο της σημασίας αυτού του όρου για την αναγνώριση περιεχομένου αυτού του εγγράφου χρησιμοποιώντας το ακόλουθο βάρος w_{ij} για τον όρο αυτό:

$w_{ij} = tf_{ij} \cdot idf_j$

Το *term frequency* tf_{ij} υπολογίζεται ως ο αριθμός των φορών που εμφανίζεται ο όρος *j* στο έγγραφο *i*.

Το *inverse document frequency* idf_j υπολογίζεται ως:

$$idf_j = \log\left(\frac{N}{f_j}\right)$$

όπου f_j είναι ο συνολικός αριθμός εμφανίσεων του όρου j στη συλλογή εγγράφων και N ο αριθμός εγγράφων στη συλλογή.

Εκτός από τα *tf* και *idf*, είναι χρήσιμος και ένας παράγοντας κανονικοποίησης, ειδικά σε συλλογές εγγράφων με πολύ διαφορετικά μήκη. Τα πιο μεγάλα έγγραφα τείνουν να έχουν μεγαλύτερη πιθανότητα να ανακτηθούν σαν σχετικά, αν και όλα τα σχετικά έγγραφα πρέπει να θεωρούνται εξίσου σημαντικά ανεξαρτήτως μεγέθους. Το κανονικοποιημένο βάρος $tf_{ij} \cdot idf_j$ μπορεί να οριστεί ως εξής:

$$w_{ij} = \frac{tf_{ij} \cdot idf_j}{\sqrt{\sum_j \left(w_{ij}^2\right)}}$$

Το μοντέλο στάθμισης *tf· idf* δεν παρουσιάζει σημαντικές θεωρητικές ιδιότητες, σε αντίθεση με τη πιθανοτική στάθμιση (*probabilistic weighting).* Σύμφωνα με αυτήν, ένα κατάλληλο βάρος w_j για έναν όρο j δίνεται από την ακόλουθη έκφραση:

$$w_j = \frac{r/(R-r)}{(n-r)/[-n-(R-r)]}$$

όπου R είναι ο αριθμός των σχετικών εγγράφων, r ο αριθμός των σχετικών εγγράφων που περιέχουν τον όρο j, N ο αριθμός όλων των εγγράφων στη συλλογή και n ο αριθμός των εγγράφων που περιέχουν τον όρο j. Η παραπάνω έκφραση είναι γνωστή ως βάρος σχετικότητας *(relevance weight)* και ορίζει τη σημασία ενός όρου χρησιμοποιώντας το ποσοστό των σχετικών εγγράφων στα οποία εμφανίζεται ο όρος διαιρούμενου με το ποσοστό των μη-σχετικών εγγράφων στα οποία εμφανίζεται ο όρος.

Υποθέτοντας ανεξαρτησία των όρων (οι όροι εμφανίζονται ανεξάρτητα από τον κάθε άλλο) και δυαδική μορφή της συχνότητας όρου (δηλαδή 1 αν υπάρχει ο όρος στο έγγραφο και 0 αν δεν υπάρχει), έχει αποδειχτεί ότι η πιθανοτική στάθμιση σχετικότητας καταλήγει να είναι αρκετά παρόμοια με τη στάθμιση *tf· idf*.

2.3.4 Ανάκτηση διανυσματικού χώρου (Vector space retrieval)

Στην περίπτωση ενός συστήματος ανάκτησης πληροφορίας, ανακτώνται τα έγγραφα που θεωρούνται σχετικά με το ερώτημα ενός χρήστη. Όλες οι στρατηγικές ανάκτησης είναι βασισμένες σε μία σύγκριση μεταξύ του ερωτήματος και των εγγράφων, που αναγνωρίζει τα πιθανά σχετικά έγγραφα για το συγκεκριμένο ερώτημα.

Η ανάκτηση διανυσματικού χώρου θεωρεί ένα χώρο εγγράφων αποτελούμενο από έγγραφα. Ο τρι-διάστατος χώρος επεκτείνεται σε *u* διαστάσεις όταν είναι παρόντες u όροι ευρετηρίου. Για κάθε έγγραφο *i*, *u*-διάστατα διανύσματα εγγράφου D_i κατασκευάζονται από ένα σύνολο από u όρους ευρετηρίου $t_1, t_2, \ldots t_u$:

$D_i = (d_{i1}, d_{i2}, \ldots, d_{iu})$

όπου d_{ij} είναι το βάρος που έχει εκχωρηθεί στον όρο *j* του εγγράφου *i*.

Παρομοίως, ένα *u*-διάστατο διάνυσμα Q κατασκευάζεται για κάθε ερώτημα που εισάγει ένας χρήστης:

$Q = (q_1, q_2, \ldots, q_u)$

όπου q_j είναι το βάρος που έχει εκχωρηθεί στον όρο *j* του ερωτήματος *Q*.

Χρησιμοποιώντας τις παραπάνω διανυσματικές αναπαραστάσεις, υπολογίζονται αξίες ομοιότητας για κάθε ζευγάρι εγγράφου - ερωτήματος:

$$Sim(D_i, Q) = \frac{\sum_{j-1}^{u}(d_{ij} \cdot q_j)}{\sqrt{\sum_{j-1}^{u} d_{ij}^2 \cdot \sum_{j-1}^{u} q_j^2}}$$

Έγγραφα που έχουν αξία ομοιότητας με το ερώτημα παραπάνω από ένα προκαθορισμένο όριο θεωρούνται σχετικά με το ερώτημα αυτό. Έτσι, το τελικό αποτέλεσμα ενός IR (Information Retrieval) συστήματος βασισμένο σε διανυσματικό χώρο, είναι ένα σύνολο από έγγραφα που συνήθως κατατάσσονται σε φθίνουσα σειρά αξίας ομοιότητας με το ερώτημα του χρήστη.

2.3.5 Πιθανοτική ανάκτηση (Probabilistic retrieval)

Η πιθανοτική ανάκτηση παίρνει υπόψη τις ιδιότητες σχετικότητας των εγγράφων. Σύμφωνα με το μοντέλο ανάκτησης δυαδικής ανεξαρτησίας *(binary independence retrieval model)*, κάθε έγγραφο (ερώτημα) αναπαριστάται από ένα *u*-διάστατο δυαδικό διάνυσμα x (r) :

$x = (x_1, x_2, \ldots, x_u)$

$r = (r_1, r_2, \ldots, r_u)$

όπου x_j (r_j) υποδηλώνει την απουσία ή παρουσία του *j* όρου στο έγγραφο (ερώτημα) όταν είναι 0 ή 1 αντίστοιχα.

Ένα έγγραφο είναι σχετικό με ένα συγκεκριμένο ερώτημα αν η πιθανότητα του εγγράφου να είναι σχετικό, δεδομένου του διανύσματος εγγράφου x, είναι μεγαλύτερη από την πιθανότητα να μην είναι σχετικό το έγγραφο:

$P(Relevant|x) > P(Non\text{-}relevant|x)$

Από τον παραπάνω κανόνα απόφασης προκύπτει η επόμενη συνάρτηση αντιστοίχησης, από την οποία υπολογίζονται αξίες κατάστασης ανάκτησης *g (retrieval status values)* για κάθε ζευγάρι εγγράφου – ερωτήματος:

$$g(x,r) = \sum_{j-1}^{u} \left(r_j \cdot x_j \cdot \log \frac{p_j (1 - j_i)}{(1 - p_j) q_j} \right) + C$$

Όπου p_j η πιθανότητα να υπάρχει ο όρος ευρετηρίου j αν το έγγραφο είναι σχετικό, q_j η πιθανότητα να υπάρχει ο όρος ευρετηρίου j αν το έγγραφο δεν είναι σχετικό. Το C είναι σταθερά για ένα δεδομένο ερώτημα και δεν επηρεάζει την κατάταξη των εγγράφων. Τα έγγραφα κατατάσσονται με φθίνουσα σειρά των αξιών κατάστασης ανάκτησης.

Ένας τρόπος να εκτιμήσουμε τις πιθανότητες p_j και q_j είναι κάνοντας μια αρχική αναζήτηση βασισμένη σε άλλες στρατηγικές ανάκτησης και χρησιμοποιώντας τα κορυφαία έγγραφα σαν σχετικά, ή εφαρμόζοντας ανάδραση σχετικότητας των χρηστών *(user relevance feedback)*.

Υποθέτοντας ότι όλα τα p_j είναι τα ίδια και ότι τα q_j υπολογίζονται ως n_j/N, όπου n_j είναι ο αριθμός των εγγράφων στα οποία εμφανίζεται ο όρος j και N το μέγεθος της συλλογής, τότε η συνάρτηση πιθανοτικής αντιστοίχησης γίνεται πολύ παρόμοια με τη συνάρτηση αντιστοίχησης του διανυσματικού χώρου με δυαδική στάθμιση.

2.3.6 Άλλες τεχνικές ανάκτησης

Άλλες στρατηγικές ανάκτησης περιλαμβάνουν το μοντέλο λογικής ανάκτησης *(boolean retrieval model)* και το *cluster-based* μοντέλο. Στο πρώτο μοντέλο, κάθε έγγραφο σχετίζεται με ένα σύνολο λέξεων – κλειδιών και κάθε ερώτημα έχει τη μορφή μιας λογικής έκφρασης με τελεστές and, or και not. Τα ανακτηθέντα έγγραφα είναι αυτά που περιέχουν όρους ευρετηρίου στο συνδυασμό που ορίζεται από το ερώτημα. Στα μοντέλα, τα έγγραφα ομαδοποιούνται σε συστάδες (clusters). Τα clusters προμηθεύουν ένα ακόμη μηχανισμό για επιπλέον αντιστοιχήσεις μεταξύ όρων ερωτημάτων και συστάδων εγγράφων.

Η ανάδραση σχετικότητας είναι μια γνωστή τεχνική που χρησιμοποιείται σε πολλές στρατηγικές ανάκτησης για να αυξήσουν την αποτελεσματικότητα της ανάκτησης. Μπορεί να επιτευχθεί είτε με την επαναστάθμιση των όρων ερωτημάτων βασισμένη στην κατανομή αυτών των όρων στο σύνολο των σχετικών και μη-σχετικών εγγράφων που ανακτήθηκαν σε απάντηση του ερωτήματος, είτε αλλάζοντας τους πραγματικούς όρους στο ερώτημα. Οι χρήστες κρίνουν τη σχετικότητα των ανακτηθέντων εγγράφων αφού έχει γίνει μια αρχική αναζήτηση.

2.3.7 Αξιολόγηση ανάκτησης

Η αξιολόγηση της ανάκτησης πληροφορίας έχει κυρίως επικεντρωθεί στην αποτελεσματικότητα της ανάκτησης και βασιστεί στη χρήση συλλογών εγγράφων, ερωτημάτων και κρίσεων σχετικότητας που έχουν δοθεί από ειδικούς στο θέμα των ερωτημάτων ανάκτησης, αν και αυτές οι κρίσεις είναι συχνά υποκειμενικές. Τα έγγραφα που ανακτώνται σε απάντηση ενός ερωτήματος κρίνονται από αυτούς τους χρήστες σαν σχετικά ή όχι.

Τα πιο συνηθισμένα μέτρα αποτελεσματικότητας ανάκτησης είναι η ανάκληση (recall - RE) και η ακρίβεια (precision – PR), που παρουσιάστηκαν και νωρίτερα.

Για τη θέση κατάταξης i κάθε σχετικού εγγράφου ενός ερωτήματος, η ανάκληση PR_i και η ακρίβεια RE_i υπολογίζονται ως εξής:

$$PR_i = \frac{r_i}{n_i}, \qquad RE_i = \frac{r_i}{R}$$

όπου P ο αριθμός των σχετικών εγγράφων, r_i ο αριθμός των σχετικών εγγράφων που επιστρέφονται στη θέση κατάταξης i και n_i ο αριθμός των εγγράφων που επιστρέφονται στη θέση κατάταξης i. Μετά υπολογίζονται οι μέσες τιμές για όλα τα ερωτήματα έτσι ώστε να πάρουμε ένα σύνολο από τιμές ακρίβειας στα σημεία ανάκλησης 0.1, 0.2, …, 1, από το οποία κατασκευάζονται αντίστοιχα γραφήματα.

2.4 Support Vector Machines (SVM)

2.4.1 Ορισμός και ιδιότητες των SVMs

Τα Support Vector Machines (SVMs) είναι ένα σύνολο μεθόδων εκμάθησης που χρησιμοποιούνται για προβλήματα ταξινόμησης και παλινδρομικής ανάλυσης. Η κύρια ιδέα των SVM είναι να κατασκευαστεί ένα υπερεπίπεδο, έτσι ώστε η απόσταση του διαχωρισμού μεταξύ των θετικών και αρνητικών παραδειγμάτων να μεγιστοποιείται. Τα διανύσματα των πιο κοντινών στοιχείων στο υπερεπίπεδο αυτό είναι τα υποστηρικτικά διανύσματα (support vectors).

Αυτή η επιθυμητή ιδιότητα επιτυγχάνεται ακολουθώντας την αρχή της Ελαχιστοποίηση του Δομικού Ρίσκου *(Structural Risk Minimization)* από τη θεωρία της μηχανικής μάθησης. Η ιδέα της ελαχιστοποίησης του δομικού ρίσκου είναι να βρεθεί μια υπόθεση h για την οποία

μπορούμε να εγγυηθούμε το χαμηλότερο πραγματικό σφάλμα. Το πραγματικό σφάλμα της *h* είναι η πιθανότητα της *h* να κάνει λάθος σε ένα τυχαία επιλεγμένο παράδειγμα το οποίο δεν έχει δει στο παρελθόν. Το πλεονέκτημα της τεχνικής αυτής είναι ότι επιτυγχάνονται καλές επιδόσεις στα προβλήματα ταξινόμησης χωρίς να ενσωματώνεται γνώση από τον τομέα του προβλήματος.

Βλέποντας τα δεδομένα εισόδου σαν δύο σύνολα διανυσμάτων σε ένα *ν*-διάστατο χώρο, το SVM θα κατασκευάσει ένα διαχωριστικό υπερεπίπεδο σε αυτόν το χώρο, που θα μεγιστοποιεί την απόσταση μεταξύ των δύο συνόλων. Για τον υπολογισμό της απόστασης αυτής, κατασκευάζονται δύο παράλληλα υπερεπίπεδα, ένα σε κάθε πλευρά του διαχωριστικού υπερεπιπέδου, τα οποία "σπρώχνονται" πάνω στα δύο σύνολα δεδομένων. Διαισθητικά, ένας καλός διαχωρισμός επιτυγχάνεται από το υπερεπίπεδο που έχει τη μεγαλύτερη απόσταση από τα γειτονικά σημεία δεδομένων και των δύο συνόλων, δεδομένου ότι σε γενικές γραμμές όσο μεγαλύτερη είναι η απόσταση τόσο καλύτερο είναι το λάθος γενίκευσης του ταξινομητή.

Η ταξινόμηση των δεδομένων είναι μια κοινή ανάγκη στο πεδίο της μηχανικής μάθησης. Ας υποθέσουμε ότι δίνονται κάποια σημεία δεδομένων που ανήκουν στα δύο σύνολα, και ο στόχος είναι να αποφασίσουμε σε ποιο σύνολο θα μπει ένα νέο σημείο δεδομένων. Στην περίπτωση των SVM, ένα σημείο δεδομένων θεωρείται σαν ένα διάνυσμα *p*-διαστάσεων, και θέλουμε να ξέρουμε αν μπορούμε να χωρίσουμε αυτά τα σημεία με ένα *p*–1-διάστατο υπερεπίπεδο. Αυτό ονομάζεται γραμμικός ταξινομητής. Υπάρχουν πολλά υπερεπίπεδα που θα μπορούσαν να ταξινομήσουν τα δεδομένα. Ωστόσο, ενδιαφερόμαστε επιπλέον να διαπιστώσουμε εάν μπορούμε να πετύχουμε το μέγιστο διαχωρισμό (απόσταση) μεταξύ των δύο κλάσεων. Με αυτό εννοούμε ότι διαλέγουμε το υπερεπίπεδο, έτσι ώστε η απόσταση από το υπερεπίπεδο στο πλησιέστερο σημείο δεδομένων να μεγιστοποιείται. Αυτό σημαίνει ότι η κοντινότερη απόσταση ανάμεσα σε ένα σημείο στο ένα διαχωρισμένο υπερεπίπεδο και σε ένα σημείο στο άλλο διαχωρισμένο υπερεπίπεδο μεγιστοποιείται. Αν υπάρχει ένα τέτοιο υπερεπίπεδο, είναι γνωστό ως το υπερεπίπεδο μέγιστου-διαχωρισμού, και ένας τέτοιος γραμμικός ταξινομητής είναι γνωστός ως ένας ταξινομητής μέγιστου-διαχωρισμού.

Τα SVMs ανήκουν στην κατηγορία των γενικευμένων γραμμικών ταξινομητών. Μια ειδική ιδιότητά τους είναι ότι ταυτόχρονα ελαχιστοποιούν το εμπειρικό σφάλμα ταξινόμησης και μεγιστοποιούν τη γεωμετρική απόσταση. Ως εκ τούτου, είναι ταξινομητές μέγιστου-διαχωρισμού.

Μία αξιοσημείωτη ιδιότητα των SVMs είναι ότι η ικανότητά τους να μαθαίνουν είναι ανεξάρτητη από τις διαστάσεις του χώρου χαρακτηριστικών. Τα SVMs μετράνε την πολυπλοκότητα των υποθέσεων με βάση την απόσταση που μπορούν να διαχωρίσουν τα στοιχεία, και όχι με βάση τον αριθμό των χαρακτηριστικών. Αυτό σημαίνει ότι μπορούμε να γενικεύσουμε ακόμη και με την παρουσία πάρα πολλών χαρακτηριστικών, αν τα στοιχεία μας

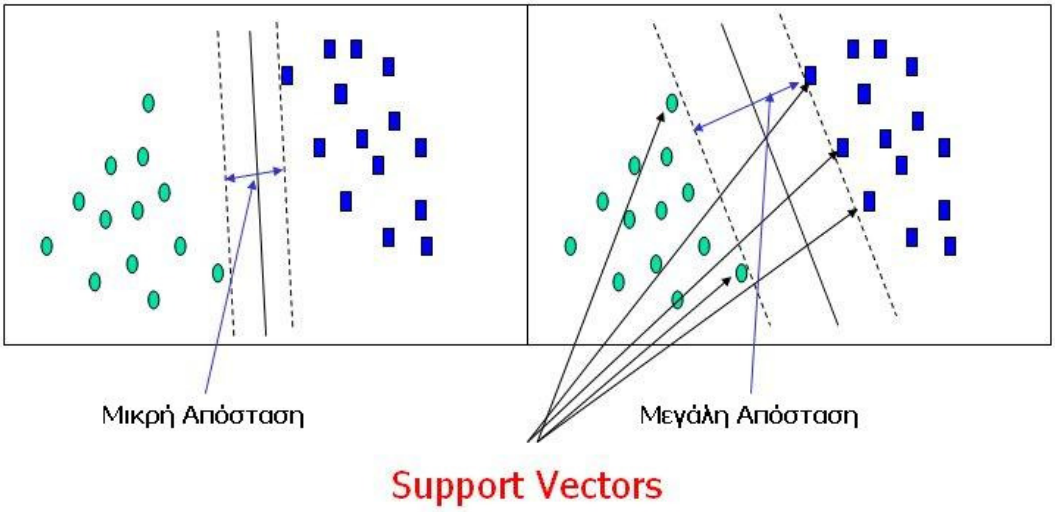
μπορούν να διαχωριστούν με ένα ευρύ περιθώριο χρησιμοποιώντας συναρτήσεις από το χώρο υποθέσεων.

Τα Support Vector Machines έχουν πολλά ελκυστικά χαρακτηριστικά. Είναι ένα σπάνιο παράδειγμα μεθοδολογίας όπου συνδυάζονται η γεωμετρική διαίσθηση, τα κομψά μαθηματικά, οι θεωρητικές εγγυήσεις και οι πρακτικοί αλγόριθμοι. Μπορούν να εφαρμοστούν αποτελεσματικά σε ένα ευρύ φάσμα προβλημάτων ταξινόμησης. Κλιμακώνονται σε τεράστια σύνολα δεδομένων και είναι ανεξάρτητα του τομέα του προβλήματος. Επιπλέον, μπορούν να αναπτυχθούν αποτελεσματικές συναρτήσεις πυρήνα για κάθε συγκεκριμένο πρόβλημα, προκειμένου να επιτευχθούν ακόμα καλύτερα αποτελέσματα. Τα SVMs έχουν πολλές επιτυχημένες εφαρμογές στον τομέα της βιοπληροφορικής (ταξινόμηση δεδομένων μικρό-συστοιχιών), της ανίχνευσης προσώπου και αναγνώρισης χειρογράφου κειμένου. Είναι επίσης πολύ καλά για την κατηγοριοποίηση κειμένου.

2.4.2 Δι-διάστατο παράδειγμα SVM

Σε ένα ιδεατό δι-διάστατο παράδειγμα, τα στοιχεία της μίας κατηγορίας βρίσκονται στο κάτω αριστερό άκρο ενώ τα στοιχεία της άλλης κατηγορίας βρίσκονται στο πάνω δεξιό άκρο, και έτσι είναι τελείως διαχωρισμένα. Προσπαθούμε να βρούμε ένα υπερεπίπεδο 1-διάστασης (δηλαδή μια γραμμή) που να χωρίζει τα στοιχεία των δύο κατηγοριών. Οι πιθανές γραμμές που το επιτυγχάνουν αυτό είναι άπειρες, ο στόχος είναι όμως να προσδιορίσουμε την καλύτερη δυνατή. Ο SVM αλγόριθμος βρίσκει τη γραμμή (ή το υπερεπίπεδο – στη γενική περίπτωση) έτσι ώστε η απόσταση μεταξύ των στοιχείων των δύο κατηγοριών είναι η μέγιστη.

Στο παρακάτω σχήμα, οι διακεκομμένες γραμμές που είναι σχεδιασμένες παράλληλα στην διαχωριστική γραμμή δείχνουν την απόσταση μεταξύ της διαχωριστικής γραμμής και των πλησιέστερων διανυσμάτων στη γραμμή. Η απόσταση μεταξύ των διακεκομμένων γραμμών ονομάζεται περιθώριο. Τα διανύσματα (σημεία) που περιορίζουν το πλάτος του περιθωρίου είναι τα υποστηρικτικά διανύσματα (support vectors). Είναι φανερό ότι η γραμμή στο δεξί σχήμα διαχωρίζει πολύ καλύτερα τα στοιχεία από αυτήν στο αριστερό σχήμα.



Εικόνα 6: SVM Margins & Support Vectors

2.4.3 Αλγόριθμος SVM

Με δεδομένα αρχεία καταγραφής της συμπεριφοράς των χρηστών σε μια δικτυακή μηχανή αναζήτησης, δείξαμε παραπάνω με ποια στρατηγική μετατρέπουμε τις εγγραφές του log αρχείου σε κρίσεις προτίμησης. Θα παρουσιάσουμε τώρα τον αλγόριθμο που χρησιμοποιεί το SVM για να μάθει από αυτές τις προτιμήσεις.

Θεωρούμε σαν είσοδο του αλγορίθμου προτιμήσεις της μορφής

$d_i >_q d_j$ (1)

όπου d_i, d_j έγγραφα για ένα δεδομένο ερώτημα q. Η παραπάνω σχέση δείχνει ότι το d_i προτιμάται σε σχέση με το d_j για ένα δεδομένο q. Για το μοντέλο ανάκτησης χρησιμοποιούμε μια γραμμική συνάρτηση ανάκτησης:

$rel(d_i,q) = w \cdot \Phi(d_i,q)$ (2)

όπου $\Phi(d_i,q)$ είναι μια συνάρτηση που αντιστοιχίζει έγγραφα και ερωτήματα σε ένα διάνυσμα χαρακτηριστικών (feature vector). Διαισθητικά, μπορεί να θεωρηθεί σαν ένα διάνυσμα χαρακτηριστικών που περιγράφει την ποιότητα της αντιστοίχησης μεταξύ ενός εγγράφου d_i και του ερωτήματος q. Το w είναι ένα διάνυσμα βάρους που αναθέτει βάρη σε καθένα από τα χαρακτηριστικά στο Φ, και άρα δίνοντάς μας μια συνάρτηση ανάκτησης πραγματικής αξίας, όπου ένα μεγαλύτερο σκορ δηλώνει ότι ένα έγγραφο d_i είναι πιο σχετικό για το ερώτημα q. Το έργο της εκμάθησης μιας συνάρτησης κατάταξης ισοδυναμεί με την εκμάθηση ενός βέλτιστου w.

Ξαναγράφουμε τη σχέση (1) ως εξής:

$w \cdot \Phi(d_i,q) > w \cdot \Phi(d_j,q)$

Στη συνέχεια προσθέτουμε ένα περιθώριο και μη-αρνητικές slack μεταβλητές ώστε να επιτρέψουμε κάποιους από τους περιορισμούς των προτιμήσεων να παραβιαστούν, όπως γίνεται και στα SVMs ταξινόμησης. Αυτό δείχνει ένα περιορισμό προτίμησης πάνω από το w

$w \cdot \Phi(d_i,q) \geq w \cdot \Phi(d_j,q) + 1 - \xi_{ij}$

Αν και δεν μπορούμε αποτελεσματικά να βρούμε ένα που να ελαχιστοποιεί τον αριθμό των παραβιασμένων περιορισμών, μπορούμε να ελαχιστοποιήσουμε ένα άνω όριο στον αριθμό των παραβιασμένων περιορισμών, $\sum \xi_{i,j}$. Η ταυτόχρονη μεγιστοποίηση του περιθωρίου οδηγεί στο παρακάτω δευτεροβάθμιο κυρτό πρόβλημα βελτιστοποίησης:

$$\min_{w,\xi_{ij}} \frac{1}{2} w \cdot w + C \sum_{ij} \xi_{ij}$$

subject to $\forall (q,i,j): \; w \cdot \Phi(d_i,q) \geq w \cdot \Phi(d_j,q) + 1 - \xi_{ij}$

$$\forall i,j: \; \xi_{ij} \geq 0$$

Στην εργασία μας θα χρησιμοποιήσουμε τον αλγόριθμο SVM^{light} για την επίλυση του παραπάνω προβλήματος βελτιστοποίησης.

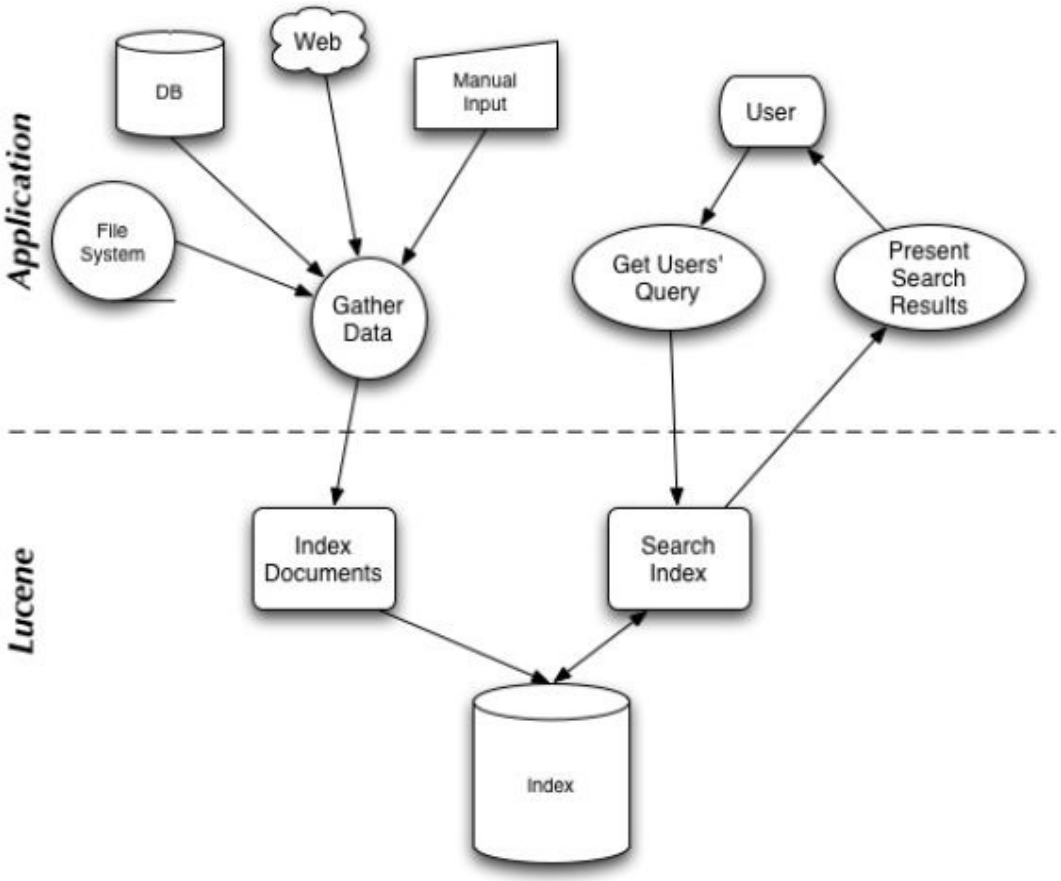
2.5 Πλατφόρμες και προγραμματιστικά εργαλεία

2.5.1 Lucene

Το Lucene είναι μία υψηλής απόδοσης βιβλιοθήκη Ανάκτησης Πληροφοριών (Information Retrieval). Αυτό επιτρέπει την προσθήκη δυνατοτήτων ευρετηρίου και αναζήτησης στις εφαρμογές στις οποίες χρησιμοποιείται. Το Lucene είναι λογισμικό ανοιχτού κώδικα που αρχικά υλοποιήθηκε σε Java και υποστηρίζεται από την Apache Software Foundation.

Το Lucene μπορεί να φτιάξει ευρετήριο και να δώσει τη δυνατότητα αναζήτησης σε οτιδήποτε δεδομένα που μπορούν να μετατραπούν σε μορφή κειμένου. Το Lucene δεν ενδιαφέρεται για την πηγή των δεδομένων, τη μορφή, ή ακόμη και τη γλώσσα, αρκεί να είναι κείμενο. Αυτό σημαίνει ότι μπορεί να χρησιμοποιηθεί για δημιουργία ευρετηρίου και αναζήτηση σε δεδομένα που αποθηκεύονται σε αρχεία: ιστοσελίδες σε απομακρυσμένους

δικτυακούς διακομιστές, έγγραφα που είναι αποθηκευμένα στο τοπικό σύστημα, δεδομένα αποθηκευμένα σε βάσεις δεδομένων, αρχεία απλού κειμένου, έγγραφα του Microsoft Word, HTML ή PDF αρχεία, ή σε οποιαδήποτε άλλη μορφή από την οποία μπορεί να εξαχθεί κείμενο.



Εικόνα 7: Διαδικασία indexing και αναζήτησης με το Lucene

2.5.2 Osmot

Το Osmot είναι μια ανοιχτού κώδικα μηχανή αναζήτησης για την εκμάθηση λειτουργιών ανάκτησης ταξινομημένων αποτελεσμάτων και την αξιολόγηση των κατατάξεων αυτών. Η μηχανή αναζήτησης Osmot υλοποιεί την καταγραφή σε αρχείο, την ανάλυση του αρχείου, την εκμάθηση, την αλλαγή κατάταξης και την αξιολόγηση της λειτουργικότητας. Το Osmot έχει σχεδιαστεί για να μπορεί να χρησιμοποιήσει οποιαδήποτε συνάρτηση κατάταξης και μηχανή αναζήτησης και μπορεί στη συνέχεια να μάθει να βελτιώνει την κατάταξη των αποτελεσμάτων της μηχανής αναζήτησης.

2.5.3 SVM^{light}

Το SVM^{light} είναι μια υλοποίηση των Support Vector Machines (SVM) στη γλώσσα C. Τα κύρια χαρακτηριστικά του προγράμματος που μας ενδιαφέρουν είναι ότι μπορεί να επιλύσει προβλήματα κατάταξης και ταξινόμησης, έχει γρήγορο αλγόριθμο βελτιστοποίησης και κάνει εκτιμήσεις του ποσοστού σφάλματος, της ακρίβειας, και της ανάκλησης.

Για να εκπαιδεύσουμε ένα SVM μοντέλο, πρέπει να τροφοδοτήσουμε τον αλγόριθμο SVM^{light} με ένα αρχείο που περιλαμβάνει τον βαθμό της προτίμησης του χρήστη για τα αποτελέσματα κάθε ερωτήματος, καθώς και κάποια χαρακτηριστικά αυτών των αποτελεσμάτων. Η ακρίβεια του μοντέλου μεγαλώνει αναλόγως με την ποσότητα των δεδομένων εκπαίδευσης καθώς και με τα χαρακτηριστικά που έχουν υλοποιηθεί.

Έχοντας πλέον δημιουργήσει ένα μοντέλο από τις προτιμήσεις του χρήστη, μπορούμε να εισάγουμε στον αλγόριθμο SVM^{light} τα αποτελέσματα μιας αναζήτησης του χρήστη και εκείνο θα τα ανακατατάξει με βάση το μοντέλο αυτό.

2.5.4 Apache Tomcat

Ο Apache Tomcat υλοποιεί τις προδιαγραφές για Java Servlet και JavaServer Pages (JSP) και παρέχει ένα περιβάλλον HTTP web server για να μπορεί να τρέχει κώδικας σε γλώσσα Java. Τα Servlets είναι αντικείμενα της γλώσσας προγραμματισμού Java που επεξεργάζονται με δυναμικό τρόπο αιτήματα (requests) και κατασκευάζουν απαντήσεις (responses). Αυτό επιτρέπει να έχουμε δυναμικό περιεχόμενο στο server χρησιμοποιώντας την πλατφόρμα της Java. Το περιεχόμενο που κατασκευάζεται και επιστρέφεται συνήθως είναι HTML. Οι JavaServer Pages επίσης δίνουν τη δυνατότητα για δημιουργία δυναμικών ιστοσελίδων, ενσωματώνοντας κώδικα Java μαζί με την HTML. Οι JSP σελίδες μεταγλωττίζονται σε Java Servlets όταν καλούνται για πρώτη φορά.

3

Καταγραφή δραστηριότητας χρήστη και δημιουργία ευρετηρίου

3.1 Καταγραφή δραστηριότητας αναζήτησης στα log αρχεία

Η δημιουργία του μοντέλου, που θα χρησιμοποιήσουμε για την ανακατάταξη των αποτελεσμάτων που δίνει το Google, προϋποθέτει την εκπαίδευση του με κάποια δεδομένα εκπαίδευσης. Ακολουθεί η περιγραφή της διαδικασίας για τη συλλογή αυτών των δεδομένων.

Τα δεδομένα που χρειαζόμαστε για την εκπαίδευση θα προκύψουν από την καταγραφή της δραστηριότητας του χρήστη όταν κάνει αναζητήσεις χρησιμοποιώντας την μηχανή αναζήτησής μας. Η καταγραφή αυτή περιλαμβάνει τα ερωτήματα που κάνει, τα αποτελέσματα που του επιστρέφει η μηχανή αναζήτησης και τα χαρακτηριστικά τους, καθώς και τα κλικ που κάνει στα αποτελέσματα αυτά. Η καταγραφή της δραστηριότητας γίνεται στο παρασκήνιο και ουσιαστικά είναι αόρατη στο χρήστη, ο οποίος μπορεί απλά να προσέξει μια μικρή καθυστέρηση, η οποία όμως κυμαίνεται σε λογικά πλαίσια και θα μπορούσε να οφείλεται και σε καθυστερήσεις του δικτύου.

3.1.1 Web Interface της εφαρμογής

Για να αρχίσει η διαδικασία της καταγραφής της δραστηριότητας του χρήστη, πρέπει ο χρήστης να χρησιμοποιήσει το web interface της εφαρμογής μας. Η εφαρμογή τρέχει σε κάποιον υπολογιστή που έχει το ρόλο του server, και χρειάζεται τον Apache Tomcat για να λειτουργήσει.

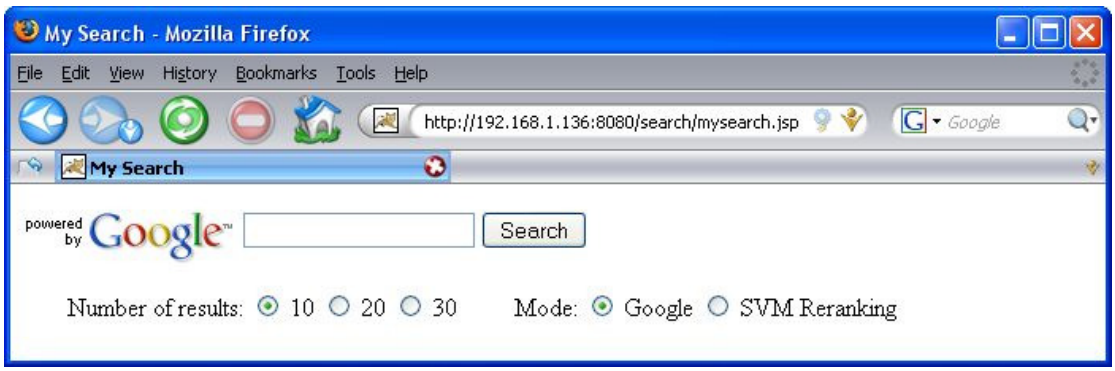
Η διεύθυνση της εφαρμογής έχει την εξής μορφή:

```
http://<Server IP>:<Tomcat Port>/search/mysearch.jsp
```

όπου <Server IP> η IP διεύθυνση του υπολογιστή στον οποίο τρέχει ο Tomcat και έχει εγκατεστημένη την εφαρμογή, και <Tomcat Port> ο αριθμός της θύρας στην οποία "ακούει" ο Tomcat (συνήθως είναι 8080).

Βλέπουμε ότι είναι μια JSP σελίδα, γεγονός το οποίο μας επιτρέπει να έχουμε δυναμικό περιεχόμενο με χρησιμοποίηση κώδικα Java. Την πρώτη φορά που κάποιος καλεί τη σελίδα από το server, υπάρχει μια μικρή επιπλέον καθυστέρηση μέχρι να γίνει η μεταγλώττισή της σε Java Servlet. Όλες οι επόμενες φορές όμως θα βρουν τη σελίδα ήδη μεταγλωττισμένη και άρα θα είναι γρήγορη η φόρτωσή της.

Αφού πληκτρολογήσει ο χρήστης τη διεύθυνση της εφαρμογής, θα δει το περιβάλλον αναζήτησης που φαίνεται στην παρακάτω εικόνα.



Εικόνα 8: Αρχική εικόνα του Web Interface της εφαρμογής

Όπως βλέπουμε, πρόκειται για ένα απλό και λιτό περιβάλλον, σε αντιστοιχία με τη φιλοσοφία του Google. Οι επιλογές που παρέχονται στο χρήστη είναι μια περιοχή κειμένου όπου πληκτρολογεί το ερώτημα του, πόσα αποτελέσματα θέλει να του επιστραφούν (10, 20 ή 30), και με τι τρόπο κατάταξης – απλή κατάταξη του Google ή ανακατάταξη των αποτελεσμάτων σύμφωνα με το SVM μοντέλο.

3.1.2 Περιεχόμενο αρχείων καταγραφής

Τα βασικά δεδομένα καταγραφής της δραστηριότητας του χρήστη αποθηκεύονται σε δύο αρχεία καταγραφής. Στο πρώτο αρχείο (out.log), που θα χρησιμοποιηθεί για την εξαγωγή των σχετικών προτιμήσεων του χρήστη, αποθηκεύονται:

- Για κάθε ερώτημα:
    - Η ημερομηνία και ώρα που έγινε το ερώτημα.
    - Ένα μοναδικό αναγνωριστικό του ερωτήματος (query id).
    - Οι λέξεις – κλειδιά του ερωτήματος.
    - Η IP διεύθυνση του χρήστη που έκανε το ερώτημα.
    - Οι URL διευθύνσεις των αποτελεσμάτων που παρουσιάστηκαν στο χρήστη.
- Για κάθε κλικ σε κάποιο αποτέλεσμα:
    - Η ημερομηνία και ώρα που έγινε το κλικ.
    - Το μοναδικό αναγνωριστικό του ερωτήματος (query id).
    - Η IP διεύθυνση του χρήστη που έκανε το κλικ..
    - Η URL διεύθυνση του αποτελέσματος.

Στο δεύτερο αρχείο (out2.log), με του οποίου τα περιεχόμενα θα δημιουργηθεί το ευρετήριο, καταγράφονται τα εξής:

- τα ερωτήματα του χρήστη,
- οι τίτλοι των αποτελεσμάτων
- οι περιλήψεις των αποτελεσμάτων,
- οι URL διευθύνσεις των αποτελεσμάτων.

3.1.3 Αλγόριθμος αναζήτησης

Όταν ο χρήστης πληκτρολογήσει το ερώτημα του προς αναζήτηση, εκτελούνται οι παρακάτω ενέργειες:

1. Αναζήτηση στο Google για τις λέξεις – κλειδιά που εισήγαγε ο χρήστης και με ζητούμενο αριθμό αποτελεσμάτων τον επιλεγμένο.
2. Το Google επιστρέφει ένα string το οποίο περιλαμβάνει ουσιαστικά όλο τον HTML κώδικα που θα εμφάνιζε αν κάποιος είχε κάνει το ίδιο ερώτημα από την κανονική ιστοσελίδα της Google. Μέσα από το string αυτό πρέπει να βρούμε και να

απομονώσουμε τους τίτλους, τις περιλήψεις και τις URL διευθύνσεις των αποτελεσμάτων.

3. Άνοιγμα των δύο log αρχείων που θα χρησιμοποιήσουμε.
4. Εγγραφή στο πρώτο log αρχείο (out.log) των παρακάτω στοιχείων:
    - Ημερομηνία και ώρα
    - Λέξεις – κλειδιά του ερωτήματος
    - Αναγνωριστικός αύξων αριθμός του ερωτήματος (query id)
    - Διεύθυνση IP του χρήστη που πραγματοποίησε το ερώτημα
5. Εκτέλεση του παρακάτω επαναληπτικού βρόχου:
    - Εύρεση και αποθήκευση σε έναν πίνακα της πρώτης URL διεύθυνσης που βρίσκεται μέσα στο string που επέστρεψε το Google.
    - Αν δεν είναι κανονικό αποτέλεσμα κειμένου, αλλά αποτέλεσμα που παραπέμπει σε άλλες σελίδες αναζήτησης του Google, όπως είναι οι σελίδες του Google για εικόνες, βίντεο, βιβλία ή ειδήσεις, παραβλέπουμε αυτό το αποτέλεσμα και συνεχίζουμε στην επόμενη επανάληψη του βήματος 5.
    - Εύρεση στο string και αποθήκευση σε έναν πίνακα του τίτλου του αποτελέσματος.
    - Εύρεση στο string και αποθήκευση σε έναν πίνακα της περίληψης (abstract) του αποτελέσματος.
    - Αφαίρεση των HTML tags από τη διεύθυνση, τον τίτλο και την περίληψη, ώστε να μείνει μόνο καθαρό κείμενο, για να μπορεί να καταχωρηθεί σωστά στο ευρετήριο.
    - Εγγραφή στο δεύτερο log αρχείο (out2.log) των παρακάτω στοιχείων:
        - Ερώτημα χρήστη
        - Τίτλος αποτελέσματος
        - Περίληψη αποτελέσματος
        - URL διεύθυνση αποτελέσματος
    - Αφαίρεση από το αρχικό string των στοιχείων του αποτελέσματος που βρήκαμε, ώστε η επόμενη επανάληψη να συνεχίσει στα επόμενα αποτελέσματα.
6. Οι επαναλήψεις τελειώνουν όταν έχουν απομονωθεί τα στοιχεία όλων των αποτελεσμάτων που επέστρεψε το Google.

7. Εγγραφή στο πρώτο log αρχείο (out.log) των URL διευθύνσεων όλων των αποτελεσμάτων χωρισμένων μεταξύ τους με έναν αστερίσκο *.
8. Εμφάνιση των αποτελεσμάτων στο χρήστη.

3.1.4 Παράδειγμα εκτέλεσης αναζήτησης

Θα δείξουμε ένα παράδειγμα εκτέλεσης του παραπάνω αλγορίθμου βήμα προς βήμα.

1. Πληκτρολογούμε στη μηχανή αναζήτησης το ερώτημα “information system” και πατάμε το κουμπί Search, ζητώντας 10 αποτελέσματα από το Google.
2. Το Google επιστρέφει στην εφαρμογή ένα string με τα αποτελέσματα, όπως θα τα εμφάνιζε στη σελίδα του. Ακολουθεί ένα μικρό κομμάτι του string αυτού με την πληροφορία για το πρώτο αποτέλεσμα:

Google string

```
<h2 class=hd>Search Results</h2><div><ol><li class=g><h3 class=r><a
href="http://en.wikipedia.org/wiki/Information_systems" class=l>
<em>Information systems</em> - Wikipedia, the free encyclopedia</a>
</h3><div class="s">16 Mar 2009 <b>...</b> In a general sense, the term
<em>information system</em> (IS) refers to a <em>system</em> of people,
data records and activities that process the data and
<b>...</b><br><cite>en.wikipedia.org/wiki/<b>Information</b>_<b>systems</b>
- 53k - 12 hours ago - </cite><span class=gl><a href="http://
209.85.129.132/search?q=cache:w7oAl30-yewJ:en.wikipedia.org/wiki/Info
rmation_systems+information+system&amp;cd=1&amp;hl=en&amp;ct=clnk&amp;ie=UTF
-8">Cached</a> - <a href="/search?hl=en&amp;ie=UTF-8&amp;q=
related:en.wikipedia.org/wiki/Information_systems">Similar pages</a>
</span></div><li class=g style="margin-left:3em">
```

Εικόνα 9: Απόσπασμα από το string που επιστρέφει το Google για ένα ερώτημα.

3. Άνοιγμα των αρχείων καταγραφής.
4. Εγγραφή στο πρώτο log αρχείο:

out.log

```
1 20090317003206 q:information system qid:1795 ip:127.0.0.1 s:NA ref:NA n:10
```

5.

- Εύρεση του URL του πρώτου αποτελέσματος `http://en.wikipedia.org/wiki/Information_systems`
- Εύρεση του τίτλου του πρώτου αποτελέσματος.

Information systems - Wikipedia, the free encyclopedia

- Εύρεση της περίληψης του πρώτου αποτελέσματος.

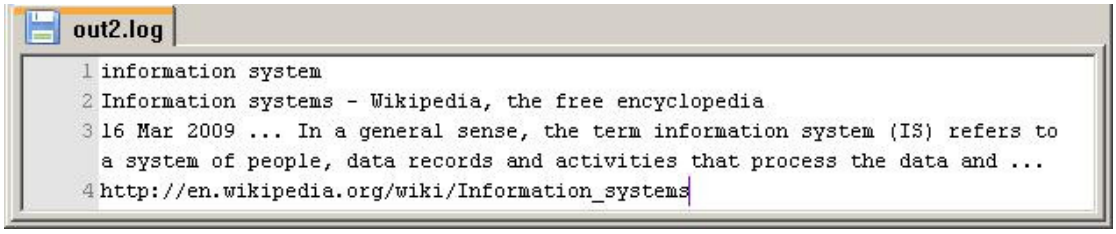
  16 Mar 2009 ... In a general sense, the term information system (IS) refers to a system of people, data records and activities that process the data and ...

- Αφαίρεση των HTML tags από τον τίτλο και την περίληψη, που γίνονται:

  Τίτλος: Information systems - Wikipedia, the free encyclopedia

  Περίληψη: 16 Mar 2009 ... In a general sense, the term information system (IS) refers to a system of people, data records and activities that process the data and ...

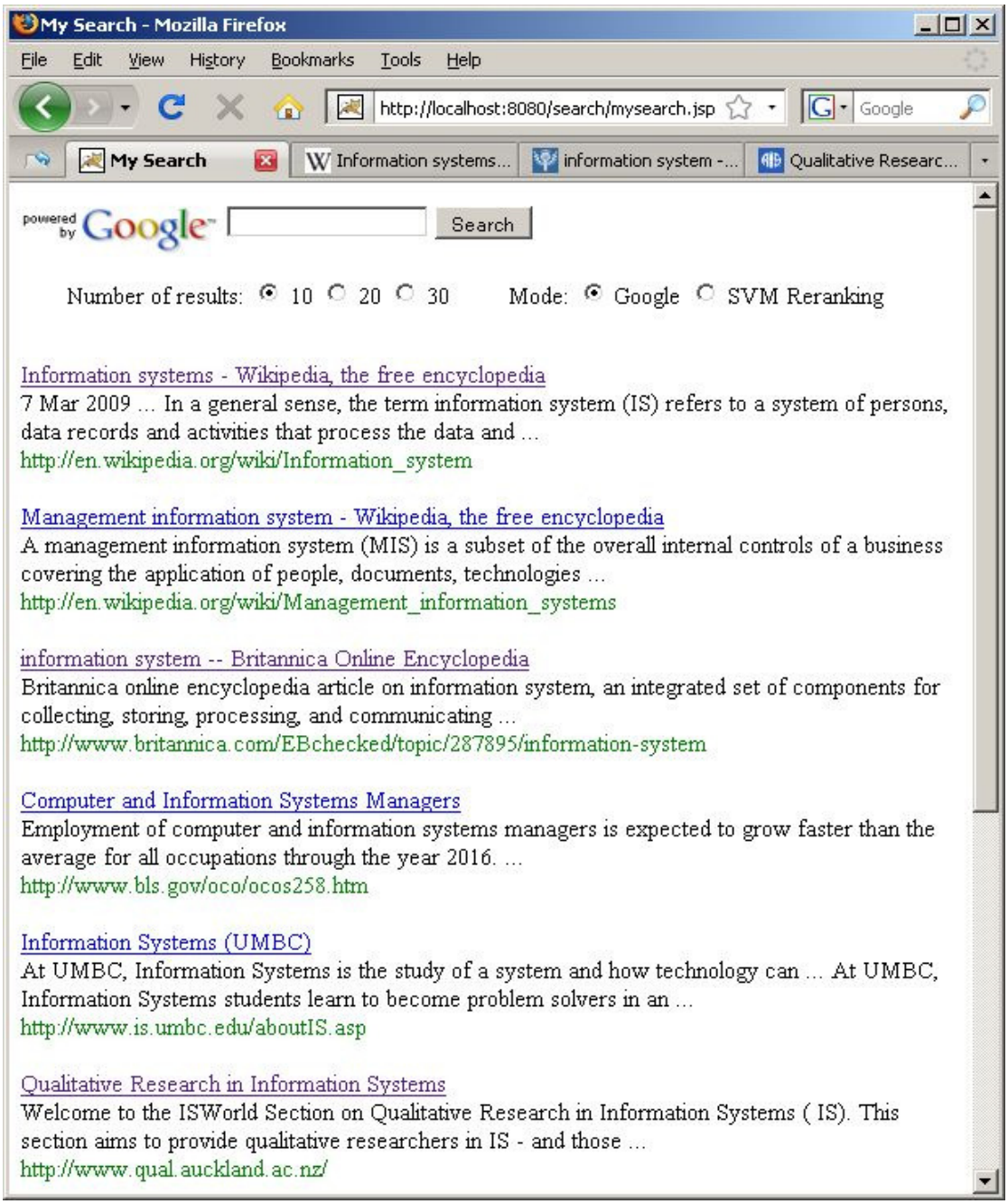
- Εγγραφή στο δεύτερο log αρχείο των παραπάνω, ως εξής:



- Αφαιρούμε από το συνολικό string το κομμάτι που επεξεργαστήκαμε και προχωράμε παρακάτω ομοίως

6. Οι επαναλήψεις τελειώνουν όταν κάνουμε τα παραπάνω βήματα για όλα τα αποτελέσματα
7. Γράφουμε στο πρώτο log αρχείο τα παρακάτω στοιχεία:

8. Εμφανίζουμε τα αποτελέσματα στο χρήστη, με τον τρόπο που φαίνεται στην παρακάτω εικόνα:



Εικόνα 10: Κατάταξη αποτελεσμάτων της εφαρμογής για το ερώτημα "information system"

Έχουμε καταγράψει και εμφανίσει λοιπόν τα αποτελέσματα της αναζήτησης του χρήστη. Τώρα πρέπει να καταγραφούν και τα αποτελέσματα στα οποία επιλέγει να κάνει κλικ. Για να επιτευχθεί αυτό, οι σύνδεσμοι των αποτελεσμάτων δεν δείχνουν απ' ευθείας στο αποτέλεσμα, αλλά καλούν ένα Java Servlet που έχουμε δημιουργήσει. Το Servlet αυτό (LogRedirect.class) καταγράφει στο πρώτο log αρχείο (out.log) τα παρακάτω στοιχεία:

- Ημερομηνία και ώρα που έγινε το κλικ
- URL διεύθυνση του αποτελέσματος

- Αύξων αριθμός του ερωτήματος (query id)
- IP διεύθυνση του χρήστη

Κατόπιν ανοίγει ένα νέο παράθυρο στον περιηγητή και γίνεται ανακατεύθυνση στη σελίδα του αποτελέσματος που είχε ζητήσει ο χρήστης. Η διαδικασία της καταγραφής του κλικ είναι αόρατη στο χρήστη σε κανονικές συνθήκες και δεν καθυστερεί ιδιαίτερα τη φόρτωση της ζητούμενης σελίδας.

3.2 Μορφή log αρχείων

Όπως αναφέραμε και στην παραπάνω παράγραφο, χρησιμοποιούμε δύο αρχεία για την καταγραφή της δραστηριότητας των χρηστών της μηχανής μας αναζήτησης.

3.2.1 Αρχείο για εξαγωγή προτιμήσεων (out.log)

Το πρώτο αρχείο (out.log) που περιλαμβάνει τις λέξεις – κλειδιά του κάθε ερωτήματος, τις URL διευθύνσεις των αποτελεσμάτων που παρουσιάστηκαν στο χρήστη και των κλικ που έκανε. Το αρχείο αυτό θα χρησιμοποιηθεί για να εξάγουμε τις σχετικές προτιμήσεις του χρήστη, σύμφωνα με τις στρατηγικές που αναλύσαμε στο κεφάλαιο 2.

Η κάθε γραμμή του αρχείου αυτού μπορεί να είναι είτε γραμμή που να αναπαριστά ένα ερώτημα του χρήστη, είτε γραμμή που να αναπαριστά ένα κλικ του χρήστη.

Οι γραμμές των ερωτημάτων αποθηκεύονται με την εξής μορφή:

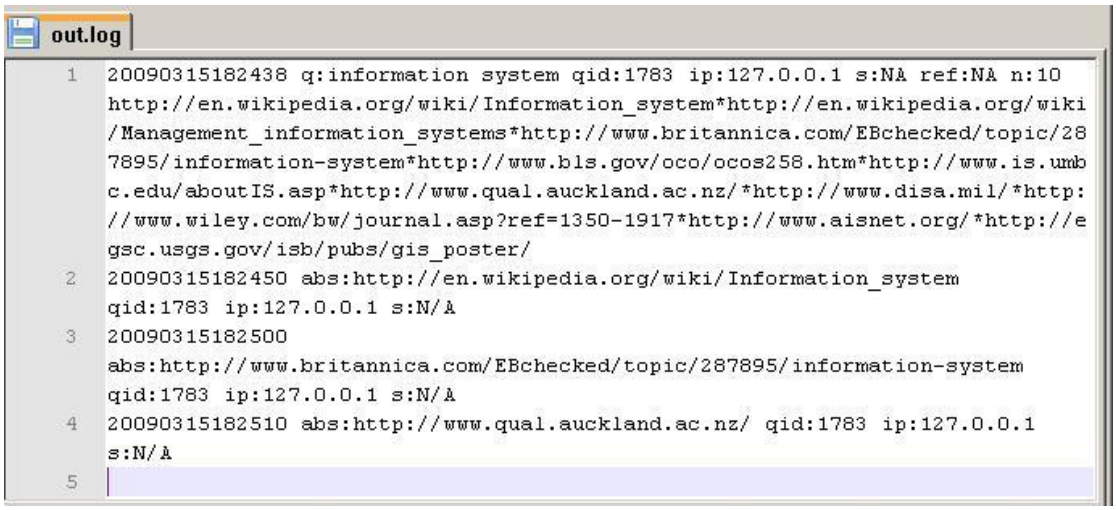
```
<Ημερομηνία & ώρα> q:<Ερώτημα> qid:<Query ID> ip:<IP διεύθυνση> s:NA
ref:NA n:<Αριθμός αποτελεσμάτων> <URL Result #1>*<URL Result #2>* …
*<URL Result #n>
```

Οι γραμμές των κλικ αποθηκεύονται με την εξής μορφή:

```
<Ημερομηνία & ώρα> abs:<URL Click> qid:<Query ID> ip:<IP διεύθυνση>
s:NA
```

Το αρχείο αυτό θα έχει δηλαδή M+N γραμμές οπού M ο αριθμός των ερωτημάτων που έχει θέσει ο χρήστης, και N ο αριθμός των κλικ σε συνδέσμους που έχει κάνει.

Στην επόμενη εικόνα φαίνεται το αρχείο για το παράδειγμά μας, όπου η γραμμή 1 δείχνει το ερώτημα και τα αντίστοιχα αποτελέσματα, ενώ οι γραμμές 2,3 και 4 δείχνουν τα αποτελέσματα στα οποία έκανε κλικ ο χρήστης .



Εικόνα 11: Μορφή αρχείου για εξαγωγή προτιμήσεων (out.log)

3.2.2 Αρχείο για δημιουργία ευρετηρίου (out2.log)

Στο δεύτερο αρχείο (out2.log), με του οποίου τα περιεχόμενα θα δημιουργηθεί ευρετήριο, περιλαμβάνονται τα ερωτήματα του χρήστη, οι τίτλοι, οι περιλήψεις και οι URL διευθύνσεις των αποτελεσμάτων που του παρουσιάστηκαν. Για κάθε αποτέλεσμα αποθηκεύονται τέσσερις γραμμές στο αρχείο ως εξής:

1. Ερώτημα
2. Τίτλος
3. Περίληψη
4. URL διεύθυνση

Για ένα ερώτημα δηλαδή που έχουν εμφανιστεί N αποτελέσματα, θα αποθηκευτούν 4·N γραμμές στο αρχείο αυτό.

out2.log

```
1  information system
2  Information systems - Wikipedia, the free encyclopedia
3  7 Mar 2009 ... In a general sense, the term information
   system (IS) refers to a system of persons, data records
   and activities that process the data and ...
4  http://en.wikipedia.org/wiki/Information_system
5  information system
6  Management information system - Wikipedia, the free
   encyclopedia
7  A management information system (MIS) is a subset of the
   overall internal controls of a business covering the
   application of people, documents, technologies ...
8  http://en.wikipedia.org/wiki/Management_information_systems
9  information system
10 information system -- Britannica Online Encyclopedia
11 Britannica online encyclopedia article on information
   system, an integrated set of components for collecting,
   storing, processing,  and communicating ...
12 http://www.britannica.com/EBchecked/topic/287895/informatio
   n-system
13 information system
14 Computer and Information Systems Managers
15 Employment of computer and information systems managers
   is expected to grow faster than the average for all
   occupations through the year 2016. ...
16 http://www.bls.gov/oco/ocos258.htm
```

Εικόνα 12: Μορφή αρχείου προς indexing (out2.log)

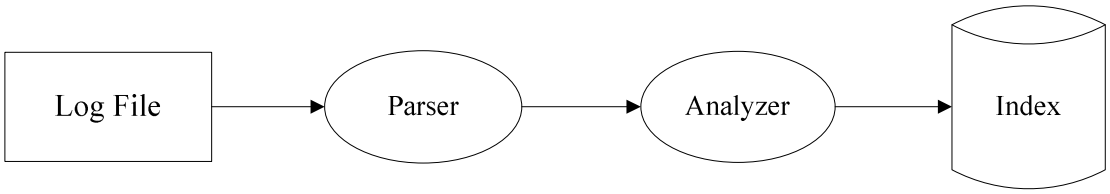
Στην παραπάνω εικόνα βλέπουμε την πληροφορία που αποθηκεύεται στο αρχείο για τα 4 πρώτα αποτελέσματα του ερωτήματος “information system”. Η πρώτη σειρά περιέχει τις λέξεις κλειδιά του ερωτήματος. Η δεύτερη σειρά έχει τον τίτλο του πρώτου αποτελέσματος, η τρίτη σειρά έχει την περίληψή του, ενώ η τέταρτη σειρά έχει τη διεύθυνσή του. Οι σειρές 5-8 έχουν την αντίστοιχη πληροφορία για το δεύτερο αποτέλεσμα, κ.ο.κ. Παρατηρούμε ότι η πληροφορία αυτή είναι ακριβώς αυτή που εμφανίζεται στο χρήστη (βλ. εικόνα 9)

3.3 Διαδικασία indexing του log αρχείου με το Lucene

Έχοντας μαζέψει ικανοποιητικό αριθμό δεδομένων από τη διαδικασία που περιγράψαμε στην προηγούμενη παράγραφο, μπορούμε να προχωρήσουμε στο επόμενο βήμα, το οποίο είναι η δημιουργία ευρετηρίου από τα στοιχεία του αρχείου out2.log. Το ευρετήριο αυτό θα δημιουργηθεί με τη βοήθεια του συστήματος Lucene.

Η δημιουργία ευρετηρίου χρησιμοποιώντας το Lucene χωρίζεται σε τρεις κύριες λειτουργίες

1. Προετοιμασία δεδομένων κειμένου για την προσθήκη στο ευρετήριο
2. Ανάλυση δεδομένων
3. Αποθήκευση δεδομένων στο ευρετήριο



Εικόνα 13: Διαδικασία indexing με το Lucene

3.3.1 Parsing δεδομένων και δημιουργία των Lucene Documents

Η πρώτη λειτουργία ξεκινάει διαβάζοντας το αρχείο καταγραφής out2.log και φτιάχνοντας ένα Lucene Document για κάθε τέσσερις γραμμές του αρχείου που διαβάζει. Στο Lucene, τα Documents (έγγραφα) είναι η μονάδα του ευρετηρίου και της αναζήτησης. Κάθε έγγραφο είναι ένα σύνολο από Fields (πεδία). Κάθε πεδίο έχει δύο μέρη, ένα όνομα και μία αξία. Κάθε έγγραφο πρέπει να περιέχει ένα ή περισσότερα πεδία, τα οποία να το κάνουν μοναδικά αναγνωρίσιμο. Στην περίπτωση μας, τα Lucene Documents που δημιουργούμε έχουν τέσσερα πεδία: Query (ερώτημα), Title (τίτλος), Abstract (περίληψη), URL (διεύθυνση).

Κατά τη δημιουργία των εγγράφων, δίνεται η δυνατότητα της επιλογής για κάθε πεδίο αν θα αποθηκευτεί εξ' ολοκλήρου με το έγγραφο ή/και αν θα προστεθεί στο ευρετήριο. Η επιλογή της αποθήκευσης (Store) χρησιμεύει στο να μπορεί να επιστραφεί το πεδίο όταν το έγγραφο είναι αποτέλεσμα μιας αναζήτησης. Η αξία αποθηκεύεται στην κανονική της μορφή, χωρίς δηλαδή να χρησιμοποιηθεί πρώτα κάποιος αναλυτής κειμένου (Analyzer). Η επιλογή του ευρετηρίου (Index) προσθέτει το πεδίο στο ευρετήριο, ώστε κάποιος να μπορεί να ψάξει για αυτό. Πριν αποθηκευτεί στο ευρετήριο, το πεδίο δέχεται επεξεργασία από έναν αναλυτή που χωρίζει το κείμενο σε κομμάτια – tokens. Εμείς θα κάνουμε Index όλα τα πεδία ώστε να μπορούμε να κάνουμε αναζήτηση σε αυτά και θα αποθηκεύσουμε τα πεδία Query και URL, τα οποία χρειάζεται να τα έχουμε και στην αρχική τους ολόκληρη μορφή (χωρίς δηλαδή να έχουν επεξεργαστεί από Analyzer).

3.3.2 Ανάλυση των δεδομένων

Αφού έχουμε δημιουργήσει τα Lucene Documents, προχωράμε στο επόμενο βήμα, δηλαδή την ανάλυση των δεδομένων ώστε να γίνουν καταλληλότερα για indexing. Μπορούμε να σκεφτούμε αυτή τη διαδικασία σαν ένα φίλτρο. Για να γίνει αυτό, το Lucene χωρίζει τα δεδομένα κειμένου σε κομμάτια – tokens και πραγματοποιεί μια σειρά από προαιρετικές λειτουργίες σε αυτά. Ένα παράδειγμα είναι η μετατροπή όλων των κεφαλαίων γραμμάτων σε πεζά. Συνηθίζεται επίσης η αφαίρεση των συχνών αλλά ανούσιων λέξεων, όπως είναι τα άρθρα. Άλλη συνηθισμένη τακτική είναι η ανάλυση των λέξεων στις ρίζες τους (stemming). Ο αναλυτής που χρησιμοποιούμε εμείς είναι ο WhitespaceAnalyzer, ο οποίος χωρίζει το κείμενο στα κενά που βρίσκει. Συνεχόμενες ακολουθίες μη-κενών χαρακτήρων συνιστούν ένα token.

3.3.3 Κατασκευή ανεστραμμένου ευρετηρίου

Μετά και την ανάλυση των δεδομένων, είμαστε πλέον έτοιμοι για την προσθήκη τους στο ευρετήριο. Τα δεδομένα αποθηκεύονται σε μια δομή γνωστή ως ανεστραμμένο ευρετήριο (inverted index). Αυτή η δομή δεδομένων κάνει αποτελεσματική χρήστη του χώρου του δίσκου, ενώ επιτρέπει την γρήγορη εύρεση λέξεων – κλειδιών. Αυτό που κάνει αυτή τη δομή ανεστραμμένη είναι ότι χρησιμοποιεί τις λέξεις που έχουν εξαχθεί από τα έγγραφα σαν κλειδιά αναζήτησης, αντί να θεωρεί τα έγγραφα σαν κεντρικές οντότητες. Με απλά λόγια, αντί να προσπαθεί να απαντήσει στην ερώτηση "τι λέξεις περιέχονται σε αυτό το έγγραφο?", αυτή η δομή είναι βελτιστοποιημένη για να παρέχει γρήγορες απαντήσεις στην ερώτηση "ποια έγγραφα περιέχουν τη λέξη X?". Όλες οι σύγχρονες μηχανές αναζήτησης χρησιμοποιούν ανεστραμμένα ευρετήρια. Αυτό που τις κάνει να διαφέρουν μεταξύ τους, είναι η προσθήκη επιπλέον παραμέτρων, όπως έχει κάνει το Google με το PageRank.

Ακολουθεί ένα απλό παράδειγμα ανεστραμμένου ευρετηρίου. Δεδομένων των εγγράφων

T_1 = "it is what it is", T_2 = "what is it", T_3 = "it is a ball"

δημιουργείται το ανεστραμμένο ευρετήριο που φαίνεται στον παρακάτω πίνακα (όπου οι αριθμοί στις αγκύλες δείχνουν σε ποια κείμενα υπάρχει η αντίστοιχη λέξη).

Λέξη	Έγγραφα
a	{3}
ball	{3}
is	{1, 2, 3}
it	{1, 2, 3}
what	{1, 2}

Πίνακας 2: Παράδειγμα ανεστραμμένου ευρετηρίου

Ένα ερώτημα με τις λέξεις – κλειδιά “what”, “is”, και “it” θα έδινε αποτέλεσμα το σύνολο $\{1,2\} \bigcap \{1,2,3\} \bigcap \{1,2,3\} = \{1,2\}$, δηλαδή τα έγγραφα T_1 και T_2.

4

Εκπαίδευση SVM μοντέλου

4.1 Εξαγωγή σχετικών προτιμήσεων χρήστη

Για να εκπαιδεύσουμε το μοντέλο που θα χρησιμοποιήσουμε για την ανακατάταξη των αποτελεσμάτων του Google, θα πρέπει να τροφοδοτήσουμε τον αλγόριθμο Support Vector Machine με τις σχετικές προτιμήσεις του χρήστη. Τις προτιμήσεις αυτές θα τις εξάγουμε έμμεσα από το αρχείο καταγραφής της δραστηριότητας του χρήστη (out.log).

Στο αρχείο αυτό καταγράφονται όλα τα αποτελέσματα που έχουν παρουσιαστεί στο χρήστη για τα ερωτήματα που έκανε, καθώς και τα κλικ που επέλεξε να κάνει. Αναλύοντας τα δεδομένα αυτά χρησιμοποιώντας τις στρατηγικές που περιγράψαμε στο κεφάλαιο 2, θα εξάγουμε τις σχετικές προτιμήσεις του χρήστη.

Για τη διαδικασία αυτήν της ανάλυσης του αρχείου καταγραφής και της εξαγωγής των προτιμήσεων του χρήστη από αυτό, θα χρησιμοποιήσουμε το Osmot. Το Osmot έχει έτοιμες συναρτήσεις για την ανάγνωση και ανάλυση log αρχείων καθώς και την εξαγωγή προτιμήσεων από αυτά. Χρησιμοποιώντας αυτές σαν βάση, κάναμε τις απαραίτητες αλλαγές και προσθήκες για τις ανάγκες τις εργασίας μας,

Η διαδικασία περιλαμβάνει τα εξής βήματα:

1. Ανάλυση του αρχείου καταγραφής και καταγραφή των ερωτημάτων και των κλικ.
2. Εξαγωγή προτιμήσεων για κάθε ερώτημα

Το πρώτο βήμα είναι να διαβάσουμε το log αρχείο και να μετατρέψουμε τα δεδομένα του σε μορφή που να είναι εύκολα επεξεργάσιμη. Χρησιμοποιώντας την συνάρτηση parseLog του Osmot, αναλύουμε το log σε εγγραφές από ερωτήματα (QueryEntry) και σε εγγραφές από κλικ (ClickEntry).

Αν έχει γίνει κλικ σε κάποιο από τα αποτελέσματα ενός ερωτήματος, καλείται η συνάρτηση που εξάγει τις σχετικές προτιμήσεις του χρήστη. Στην περίπτωση που δεν έχει γίνει κλικ σε κάποιο αποτέλεσμα, δεν εξάγεται κάποια προτίμηση με βάση τις απλές στρατηγικές. Μπορούμε όμως να εξάγουμε προτιμήσεις από αυτό, αν το ερώτημα αποτελεί μέρος μιας αλυσίδας ερωτημάτων.

Στην απλή περίπτωση του καθενός ερωτήματος ξεχωριστά, σε κάθε αποτέλεσμα που παρουσιάστηκε στο χρήστη εκχωρούμε μια τιμή που δηλώνει την σχετική προτίμηση του χρήστη για το αποτέλεσμα. Οι τιμές αυτές προκύπτουν ακολουθώντας τις τεχνικές που αναλύσαμε στο κεφάλαιο 2, δηλαδή την στρατηγική "Click > Skip above" καθώς και τη στρατηγική για προσθήκη υπάρχουσας γνώσης. Ακολουθεί ο αλγόριθμος που χρησιμοποιούμε για να βρούμε την κατάλληλη αξία για κάθε αποτέλεσμα:

- Κάθε αποτέλεσμα στο οποίο δεν έγινε κλικ από το χρήστη παίρνει την τιμή 1.
- Για τα αποτελέσματα στα οποία έγινε κλικ:
    - Το αποτέλεσμα που βρίσκεται χαμηλότερα στην κατάταξη του Google παίρνει τιμή 2.
    - Το επόμενο αποτέλεσμα που βρίσκεται πιο πάνω στην κατάταξη παίρνει την τιμή 3.
    - Ομοίως συνεχίζουμε για τα υπόλοιπα αποτελέσματα στα οποία έγινε κλικ.

Για το παράδειγμα όπου ο χρήστης έκανε κλικ στους συνδέσμους $\boldsymbol{l_1}$, $\boldsymbol{l_3}$, και $\boldsymbol{l_6}$.

$$\boldsymbol{l_1} \quad l_2 \quad \boldsymbol{l_3} \quad l_4 \quad l_5 \quad \boldsymbol{l_6} \quad l_7$$

Ακολουθώντας τα βήματα του παραπάνω αλγορίθμου έχουμε:

- Τα αποτελέσματα l_2, l_4, l_5, και l_7 παίρνουν όλα την τιμή προτίμησης 1, καθώς δεν έγινε κλικ σε αυτά.
- Το αποτέλεσμα $\boldsymbol{l_6}$ παίρνει τιμή προτίμησης 2, καθώς είναι το χαμηλότερο στην κατάταξη αποτέλεσμα στο οποίο έγινε κλικ.
- Το αποτέλεσμα $\boldsymbol{l_3}$ παίρνει τιμή προτίμησης 3, αφού είναι το επόμενο αποτέλεσμα στην κατάταξη στο οποίο έγινε κλικ.
- Το αποτέλεσμα $\boldsymbol{l_1}$ παίρνει τιμή προτίμησης 4, καθώς είναι το πρώτο αποτέλεσμα στην κατάταξη αποτέλεσμα στο οποίο έγινε κλικ.

Παρατηρούμε ότι τηρούνται όλες οι απαιτήσεις της στρατηγικής "Click > Skip above" :

value($\boldsymbol{l_3}$) > value(l_2), value($\boldsymbol{l_6}$) > value(l_2) , value($\boldsymbol{l_6}$) > value(l_4) , value($\boldsymbol{l_6}$) > value(l_5)

4.2 Επιλογή χαρακτηριστικών (features)

Για την εκπαίδευση της συνάρτησης ανάκτησης χρησιμοποιώντας τον αλγόριθμο Support Vector Machine, είναι απαραίτητο να σχεδιαστεί μια αντιστοίχηση χαρακτηριστικών μεταξύ ενός ερωτήματος q και ενός εγγράφου d. Έχοντας λοιπόν τα χαρακτηριστικά κάθε αποτελέσματος και τις προτιμήσεις του χρήστη για αυτά, το SVM μπορεί να εξάγει ένα μοντέλο. Ξέροντας ουσιαστικά τι χαρακτηριστικά έχουν τα αποτελέσματα που προτιμά ο χρήστης, μπορούμε να αλλάξουμε την κατάταξη των αποτελεσμάτων χρησιμοποιώντας το μοντέλο αυτό.

Παρακάτω ακολουθούν τα 11 χαρακτηριστικά που υλοποιήσαμε στην εργασία. Ο τρόπος υπολογισμού τους εξηγείται στην επόμενη ενότητα (4.3).

1. *Ομοιότητα ερωτήματος – τίτλου εγγράφου*, υπολογισμένη με τη συνάρτηση ομοιότητας κειμένου του Lucene.
2. *Ομοιότητα ερωτήματος – περίληψης εγγράφου*, υπολογισμένη με τη συνάρτηση ομοιότητας κειμένου του Lucene.
3. *Ομοιότητα ερωτήματος – URL διεύθυνσης εγγράφου*, υπολογισμένη με τη συνάρτηση ομοιότητας κειμένου του Lucene.
4. *Ομοιότητα ερωτήματος – τίτλου εγγράφου*, υπολογισμένη με τη συνάρτηση ομοιότητας κειμένου tf · idf.
5. *Ομοιότητα ερωτήματος – περίληψης εγγράφου*, υπολογισμένη με τη συνάρτηση ομοιότητας κειμένου tf · idf.
6. *Ομοιότητα ερωτήματος – URL διεύθυνσης εγγράφου*, υπολογισμένη με τη συνάρτηση ομοιότητας κειμένου tf · idf.
7. *Ομοιότητα ερωτήματος – τίτλου εγγράφου*, υπολογισμένη με τη συνάρτηση ομοιότητας κειμένου BM25.
8. *Ομοιότητα ερωτήματος – περίληψης εγγράφου*, υπολογισμένη με τη συνάρτηση ομοιότητας κειμένου BM25.
9. *Ομοιότητα ερωτήματος – URL διεύθυνσης εγγράφου*, υπολογισμένη με τη συνάρτηση ομοιότητας κειμένου BM25.
10. *Βαθμολογία με βάση τη θέση κατάταξης του εγγράφου στο Google*.
11. *Domain της URL διεύθυνσης του εγγράφου*.

Η λίστα των χαρακτηριστικών που χρησιμοποιήσαμε σίγουρα έχει πολλά περιθώρια για βελτίωση, καθώς υλοποιήσαμε μόνο μερικά βασικά χαρακτηριστικά που είναι σημαντικά για την κατάταξη και σχετικά απλά να υλοποιηθούν. Το σύνολο αυτό των χαρακτηριστικών μπορεί εύκολα να επεκταθεί και το σημαντικό είναι ότι κάθε προσθήκη ενός χαρακτηριστικού αυξάνει ακόμα περισσότερο την ακρίβεια του εξαγόμενου μοντέλου.

4.3 Υπολογισμός των feature vectors

4.3.1 Συνάρτηση ομοιότητας κειμένου του Lucene

Η συνάρτηση *tf · idf* αναλύθηκε στο κεφάλαιο 2, και πάνω σε αυτήν βασίζεται και η συνάρτηση του Lucene. Το Lucene χρησιμοποιεί ένα συνδυασμό του Μοντέλου Διανυσματικού Χώρου (Vector Space Model – VSM) και του Boolean μοντέλου για να καθορίσει πόσο σχετικό είναι ένα έγγραφο με το ερώτημα του χρήστη. Πρώτα χρησιμοποιεί το Boolean μοντέλο για να περιορίσει τα έγγραφα που πρέπει να βαθμολογηθούν. Η ιδέα πίσω από το VSM είναι ότι όσο περισσότερες φορές εμφανίζεται ένας όρος ερωτήματος σε ένα έγγραφο σε σχέση με τον αριθμό των φορών που εμφανίζεται σε όλα τα έγγραφα στη συλλογή, τόσο πιο σχετικό είναι το έγγραφο αυτό με το ερώτημα. Η βαθμολογία ενός ερωτήματος *q* για ένα έγγραφο *d* συσχετίζεται με το εσωτερικό γινόμενο μεταξύ των διανυσμάτων του εγγράφου και του ερωτήματος σε ένα Μοντέλο Διανυσματικού Χώρου. Ένα έγγραφο του οποίο το διάνυσμα βρίσκεται πιο κοντά στο διάνυσμα του ερωτήματος στο μοντέλο αυτό, βαθμολογείται υψηλότερα. Η βαθμολογίας ομοιότητας του Lucene υπολογίζεται λοιπόν ως εξής:

$$score(q,d) = coord(q,d) \cdot queryNorm(q) \cdot \sum_{t\ in\ q} (tf(t\ in\ d) \cdot idf(t)^2 \cdot t.getBoost() \cdot norm(t,d)$$

Όπου

1. $tf(t\ in\ d)$ συσχετίζεται με τη συχνότητα του όρου, ορισμένη από τον αριθμό των φορών που ο όρος *t* εμφανίζεται στο έγγραφο *d*. Τα έγγραφα που έχουν περισσότερες εμφανίσεις ενός δεδομένου όρου λαμβάνουν υψηλότερη βαθμολογία. Ο προκαθορισμένος υπολογισμός για το $tf(t\ in\ d)$ είναι:

$$tf(t\ in\ d) = frequency^{1/2}$$

2. $idf(t)$ είναι η αντίστροφη συχνότητα εγγράφου. Η τιμή αυτή συσχετίζεται με το αντίστροφο του *docFreq* (δηλαδή του αριθμού των εγγράφων στα οποία εμφανίζεται

ο όρος *t*). Αυτό σημαίνει ότι πιο σπάνιοι όροι δίνουν μια υψηλότερη συνεισφορά στη συνολική βαθμολογία. Ο προκαθορισμένος υπολογισμός για το $idf(t)$ είναι:

$$idf(t) = 1 + \log(\frac{numDocs}{docFreq + 1})$$

3. $coord(q,d)$ είναι ένας παράγοντας βασισμένος στο πόσοι από τους όρους του ερωτήματος βρίσκονται στο συγκεκριμένο έγγραφο. Τυπικά ένα έγγραφο που περιλαμβάνει περισσότερους από τους όρους του ερωτήματος λαμβάνει μια υψηλότερη βαθμολογία από ένα άλλο με λιγότερους όρους.
4. $queryNorm(q)$ είναι ένας παράγοντας κανονικοποίησης που χρησιμοποιείται για να κάνει βαθμολογίες μεταξύ ερωτημάτων συγκρίσιμες. Αυτός ο παράγοντας δεν επηρεάζει την κατάταξη των εγγράφων (αφού όλα τα έγγραφα πολλαπλασιάζονται με τον ίδιο παράγοντα), αλλά απλά προσπαθεί να κάνει τις βαθμολογίες μεταξύ διαφορετικών ερωτημάτων συγκρίσιμες.
5. $t.getBoost()$ και $norm(t,d)$ είναι παράγοντες που μπορούν προαιρετικά να προάγουν ένα συγκεκριμένο όρο *t* για το ερώτημα *q*. Εμείς θα χρησιμοποιήσουμε τις προκαθορισμένες τιμές του Lucene, που είναι 1.

Μπορούμε να συνοψίσουμε τα παραπάνω ως εξής:

- Έγγραφα που περιλαμβάνουν όλους τους όρους του ερωτήματος είναι καλά
- Αντιστοιχίες σπάνιων λέξεων μεταξύ εγγράφου και ερωτήματος είναι καλύτερες από συνηθισμένες λέξεις.
- Τα μεγάλα σε έκταση έγγραφα δεν είναι τόσο καλά όσο τα μικρότερα.
- Έγγραφα που επαναλαμβάνουν πολλές φορές τους όρους του ερωτήματος είναι καλά.

4.3.2 Συνάρτηση ομοιότητας κειμένου BM25

Η συνάρτηση ομοιότητας BM25 υπολογίζει τη βαθμολογία ενός εγγράφου *d* σε σχέση με ένα ερώτημα *q* ως εξής:

$$R(q,d) = \sum_{t\ in\ q} \frac{tf(t\ in\ d)}{k_1\left((1-b) + b \cdot \frac{l_d}{avl_d}\right) + tf(t\ in\ d)} \cdot idf(t)$$

Όπου τα *tf* και *idf* είναι όπως και πριν, l_d είναι το μήκος του εγγράφου *d*, avl_d είναι το μέσο μήκος των εγγράφων στη συλλογή, k_1 είναι μια ελεύθερη παράμετρος που συνήθως είναι 2 και *b* μια άλλη ελεύθερη παράμετρος που συνήθως είναι 0.75. Στην εργασία μας

χρησιμοποιήσαμε την υλοποίηση της συνάρτησης BM25 του Joaquín Pérez-Iglesias. Για να γίνει αυτό, καταγραφούμε το μήκος κάθε πεδίου του εγγράφου τη στιγμή που το προσθέτουμε στο ευρετήριο με το Lucene, έτσι ώστε στο τέλος να έχουμε και το μέσο μήκος, που χρειάζεται για τον υπολογισμό της ομοιότητας BM25.

4.3.3 Βαθμολογία θέσης κατάταξης στο Google

Για τη βαθμολογία με βάση τη θέση κατάταξης του αποτελέσματος στο Google, χρησιμοποιούμε τον ακόλουθο υπολογισμό:

$$G(q,d) = \begin{cases} 1 - r(q,d)/10, & r(q,d) < 10 \\ 0, & r(q,d) > 10 \end{cases}$$

Όπου *r(q,d)* είναι η θέση κατάταξη του αποτελέσματος *d* για το ερώτημα *q*, που δίνει το Google.

4.3.4 Domain του αποτελέσματος

Τέλος, το χαρακτηριστικό του domain είναι ουσιαστικά 73 χαρακτηριστικά που περιλαμβάνουν μερικά από τα υπάρχοντα Top-level domains του διαδικτύου. Το θεωρούμε όμως σαν ένα χαρακτηριστικό, καθώς το πολύ ένα από αυτά τα χαρακτηριστικά θα έχει την τιμή 1 (αν το έγγραφο έχει διεύθυνση σε αυτό το domain), και τα υπόλοιπα θα έχουν την τιμή 0.

Με το χαρακτηριστικό αυτό, μπορούμε να εξάγουμε εύκολα κάποια πιθανή προτίμηση του χρήστη για ιστοσελίδες κάποιου συγκεκριμένου domain. Για παράδειγμα, κάποιος που κάνει κλικ συχνά σε αποτελέσματα ιστοσελίδων από το domain *.edu* σημαίνει ότι τις προτιμάει και το μοντέλο μας θα δώσει μεγαλύτερο βάρος στο χαρακτηριστικό αυτό.

4.4 Μορφή αρχείου εισόδου SVM

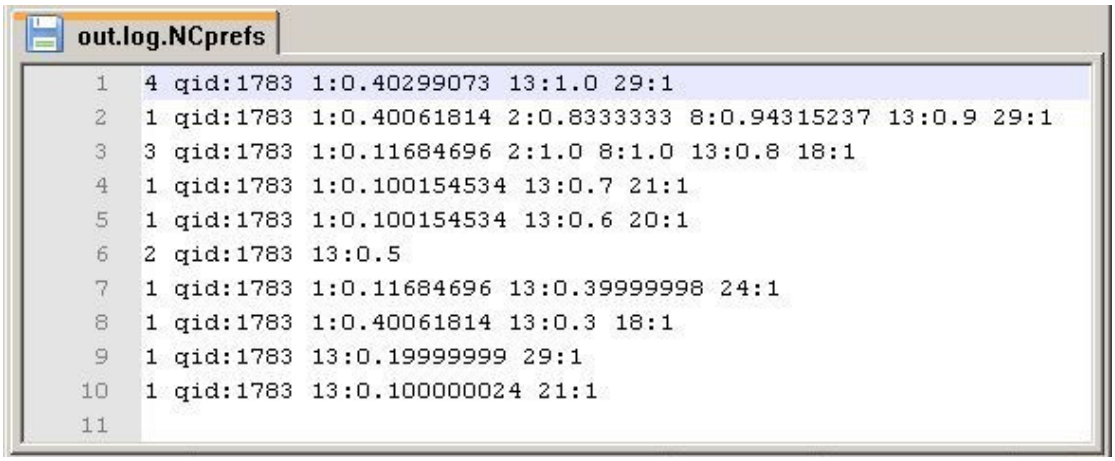
Το αρχείο που θα δώσουμε σαν είσοδο στον SVM αλγόριθμο για να εκπαιδεύσει το ζητούμενο μοντέλο πρέπει να περιέχει τις προτιμήσεις του χρήστη για τα αποτελέσματα που του παρουσιάστηκαν και τα χαρακτηριστικά αυτών. Στις προηγούμενες παραγράφους του κεφαλαίου αναλύθηκε με ποια στρατηγική υπολογίζεται η τιμή προτίμησης για κάθε

έγγραφο, καθώς και ο τρόπος υπολογισμού των χαρακτηριστικών των εγγράφων. Κάθε γραμμή του αρχείου αναπαριστά ένα έγγραφο και έχει την εξής μορφή:

```
<target> qid:<queryID> <feature>:<value> ... <feature>:<value>
```

όπου <target> είναι η τιμή προτίμησης του αποτελέσματος, <queryID> είναι ο αύξων αριθμός του ερωτήματος, <feature> είναι ο αύξων αριθμός του χαρακτηριστικού, και <value> είναι η τιμή για το αντίστοιχο χαρακτηριστικό. Τα ζευγάρια <feature>:<value> πρέπει να είναι καταταγμένα σε αύξουσα σειρά του αριθμού του χαρακτηριστικού. Χαρακτηριστικά τα οποία έχουν τιμή 0 μπορούν να παραλειφτούν.

Έτσι για το παράδειγμα μας, το αρχείο εισόδου του SVM θα πάρει την ακόλουθη μορφή:



Εικόνα 14: Μορφή αρχείου εισόδου SVM

4.5 Εκπαίδευση SVM Μοντέλου

Έχοντας ετοιμάσει πλέον το αρχείο παραδειγμάτων που θα δώσουμε σαν είσοδο στον SVM αλγόριθμο, μπορούμε να τρέξουμε την κατάλληλη εφαρμογή του *SVM-light* που εκπαιδεύει το μοντέλο. Το *SVM-light* περιλαμβάνει μια εφαρμογή εκμάθησης (svm_learn) και μια εφαρμογή ταξινόμησης (svm_classify). Η εφαρμογή εκμάθησης έχει τρεις τρόπους λειτουργίας, για ταξινόμηση, για παλινδρόμηση και για κατάταξη, που είναι και αυτή που θα χρησιμοποιήσουμε εμείς. Η εφαρμογή ταξινόμησης μπορεί να χρησιμοποιηθεί για να εφαρμόσει το εκπαιδευμένο μοντέλο σε νέα έγγραφα.

Όταν χρησιμοποιείται η εφαρμογή εκμάθησης για ταξινόμηση, εξάγεται μια σχετική προτίμηση για κάθε ζευγάρι αποτελεσμάτων στο αρχείο εισόδου που έχουν διαφορετική τιμή

προτίμησης (target value). Το ειδικό χαρακτηριστικό “qid” χρησιμοποιείται για να περιορίσει τις προτιμήσεις που θα δημιουργηθούν μόνο για αποτελέσματα που έχουν την ίδια τιμή “qid”.

Για παράδειγμα, αν το αρχείο έχει περιεχόμενο

```
3 qid:1 1:0.53 2:0.12
2 qid:1 1:0.13 2:0.1
7 qid:2 1:0.87 2:0.12
```

εξάγεται προτίμηση μόνο για τα πρώτα δύο παραδείγματα (ότι δηλαδή το πρώτο θα πρέπει να έχει θέση κατάταξης υψηλότερη από το δεύτερο), αλλά όχι για το τρίτο παράδειγμα, καθώς έχει διαφορετικό “qid”.

Τρέχοντας λοιπόν την εφαρμογή svm_learn με είσοδο το αρχείο που δημιουργήθηκε στην προηγούμενη ενότητα, θα εκπαιδευτεί το μοντέλο που φαίνεται στην παρακάτω εικόνα.

model.dat

```
1  SVM-light Version V6.02
2  0 # kernel type
3  3 # kernel parameter -d
4  1 # kernel parameter -g
5  1 # kernel parameter -s
6  1 # kernel parameter -r
7  empty# kernel parameter -u
8  29 # highest feature index
9  10 # number of training documents
10 10 # number of support vectors plus 1
11 0 # threshold b, each following line is a SV (starting with alpha*y)
12 2.2033490930117763 1:0.40299073 13:1 29:1 #
13 -1.3188353926446823 1:0.40061814 2:0.83333331 8:0.94315237 13:0.89999998 29:1 #
14 1.2800999763617817 1:0.11684696 2:1 8:1 13:0.80000001 18:1 #
15 -0.6898519079517389 1:0.10015453 13:0.69999999 21:1 #
16 -0.55501695711061338 1:0.10015453 13:0.60000002 20:1 #
17 1.7584471901929097 13:0.5 #
18 -0.43961179754822743 1:0.11684696 13:0.39999998 24:1 #
19 -1.3188353926446823 1:0.40061814 13:0.30000001 18:1 #
20 -0.91974481166652322 13:0.19999999 29:1 #
21
```

Εικόνα 15: Μορφή αρχείου εκπαιδευμένου μοντέλου SVM

Οι πρώτες γραμμές του αρχείου του μοντέλου περιέχουν τις παραμέτρους της εκπαίδευσης. Οτιδήποτε είναι μετά τον χαρακτήρα “#” είναι επεξηγηματικό σχόλιο. Οι επόμενες γραμμές περιέχουν η κάθε μία από ένα διάνυσμα υποστήριξης (support vector), σε τυχαία σειρά.

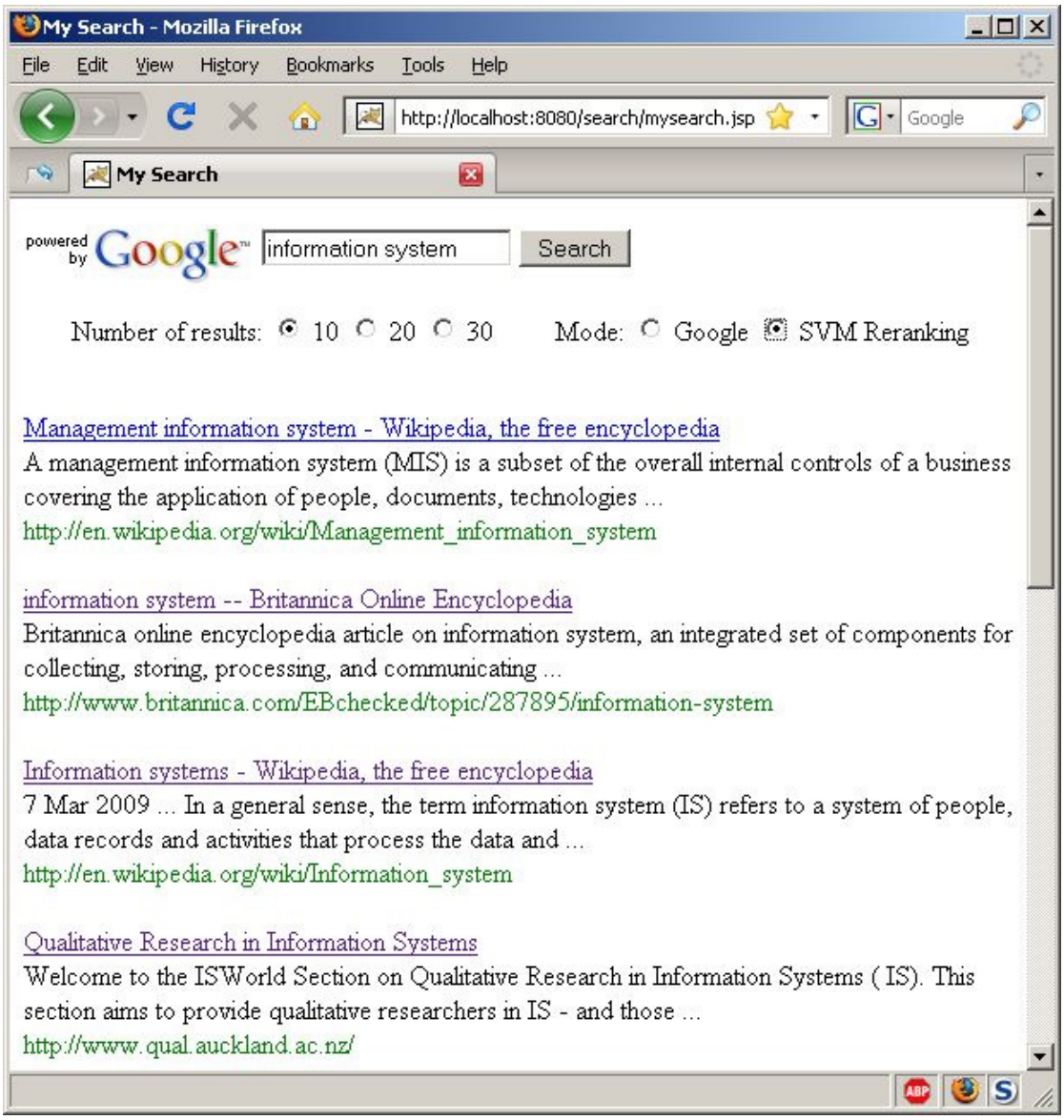
5

Αναζήτηση με ανακατάταξη αποτελεσμάτων

5.1 Διαδικασία της αναζήτησης

Στα προηγούμενα δείξαμε πως γίνεται η καταγραφή του ιστορικού του χρήστη και ο υπολογισμός των χαρακτηριστικών των ιστοσελίδων που προτιμά ο χρήστης. Κατόπιν χρησιμοποιήσαμε τον SVM αλγόριθμο εκπαίδευσης (εφαρμογή svm_learn) για τη δημιουργία ενός μοντέλου με αυτές τις προτιμήσεις του χρήστη. Έχοντας εκπαιδεύσει το μοντέλο αυτό, ο χρήστης μπορεί να χρησιμοποιεί την επιλογή "SVM Reranking" στο Web Interface της εφαρμογής μας όταν κάνει το ερώτημά του. Χρησιμοποιώντας την επιλογή αυτή, ενεργοποιείται η αναζήτηση με ανακατάταξη των αποτελεσμάτων. Δηλαδή, ανακτώνται κανονικά τα αποτελέσματα που θα παρουσίαζε το Google για το ερώτημα αυτό, αλλά αλλάζει η σειρά κατάταξής τους ανάλογα με τις προτιμήσεις του χρήστη. Η ανακατάταξη των αποτελεσμάτων πραγματοποιείται χρησιμοποιώντας τον SVM αλγόριθμο ταξινόμησης (εφαρμογή svm_classify), ο οποίος κατατάσσει τα αποτελέσματα αναλόγως με τις προτιμήσεις του χρήστη με τις οποίες έχει εκπαιδευτεί. Ο αλγόριθμος συγκρίνει τα χαρακτηριστικά των αποτελεσμάτων του ερωτήματος με τα χαρακτηριστικά τα οποία έχει μάθει ότι προτιμά ο χρήστης και πραγματοποιεί την αλλαγή της κατάταξης.

Στην επόμενη εικόνα φαίνεται το Web Interface της εφαρμογής με το χρήστη να επιλέγει την μέθοδο αναζήτησης με ανακατάταξη αποτελεσμάτων μέσω του SVM μοντέλου.



Εικόνα 16: Επιλογή "SVM Reranking" στο Web Interface της εφαρμογής

Πολλές από τις ενέργειες που εκτελούνται όταν ο χρήστης κάνει ένα ερώτημα ζητώντας αναταξινόμηση, είναι κοινές με αυτές του απλού τρόπου αναζήτησης, που περιγράψαμε στην ενότητα 3.1. Η διαφορά βρίσκεται στο γεγονός ότι τώρα κρατάμε τέσσερα αρχεία καταγραφής της δραστηριότητας του χρήστη.

Τα δύο αρχεία (out.log και out2.log) παραμένουν τα ίδια με τον απλό τρόπο αναζήτησης και καταγράφουν τα δεδομένα για όλα τα ερωτήματα και τα κλικ που γίνονται. Συνεχίζουμε να κρατάμε αυτά τα αρχεία, καθώς κάποια στιγμή θα τα ξαναχρειαστούμε για να κάνουμε επανεκπαίδευση του μοντέλου. Τα δύο καινούργια αρχεία (out3.log και out4.log) κρατάνε κάθε φορά τα δεδομένα για το τελευταίο ερώτημα που έγινε και τα αποτελέσματά του, και θα χρησιμοποιηθούν σε συνδυασμό με το εκπαιδευμένο μοντέλο, για την αναταξινόμηση των αποτελεσμάτων της αναζήτησης.

Αναλυτικά εκτελούνται οι εξής ενέργειες μετά από κάθε ερώτημα:

1. Αναζήτηση στο Google για τις λέξεις – κλειδιά που εισήγαγε ο χρήστης και με ζητούμενο αριθμό αποτελεσμάτων τον επιλεγμένο.
2. Το Google επιστρέφει ένα string το οποίο περιλαμβάνει ουσιαστικά όλο τον HTML κώδικα που θα εμφάνιζε αν κάποιος είχε κάνει το ίδιο ερώτημα από την κανονική ιστοσελίδα της Google. Μέσα από το string αυτό πρέπει να βρούμε και να απομονώσουμε τους τίτλους, τις περιλήψεις και τις URL διευθύνσεις των αποτελεσμάτων.
3. Άνοιγμα των τεσσάρων log αρχείων που θα χρησιμοποιήσουμε.
4. Καθαρισμός των περιεχομένων των log αρχείων out3.log και out4.log, καθώς πρέπει να έχουν τα δεδομένα μόνο του τελευταίου ερωτήματος κάθε φορά.
5. Εγγραφή στα out.log και out3.log των παρακάτω στοιχείων:
   - Ημερομηνία και ώρα
   - Λέξεις – κλειδιά του ερωτήματος
   - Αναγνωριστικός αύξων αριθμός του ερωτήματος (query id)
   - Διεύθυνση IP του χρήστη που πραγματοποίησε το ερώτημα
6. Εκτέλεση του παρακάτω επαναληπτικού βρόχου:
   - Εύρεση και αποθήκευση σε έναν πίνακα της πρώτης URL διεύθυνσης που βρίσκεται μέσα στο string που επέστρεψε το Google.
   - Αν δεν είναι κανονικό αποτέλεσμα κειμένου, αλλά αποτέλεσμα που παραπέμπει σε άλλες σελίδες αναζήτησης του Google, όπως είναι οι σελίδες του Google για εικόνες, βίντεο, βιβλία ή ειδήσεις, παραβλέπουμε αυτό το αποτέλεσμα και συνεχίζουμε στην επόμενη επανάληψη του βήματος 6.
   - Εύρεση στο string και αποθήκευση σε έναν πίνακα του τίτλου του αποτελέσματος.
   - Εύρεση στο string και αποθήκευση σε έναν πίνακα της περίληψης (abstract) του αποτελέσματος.
   - Αφαίρεση των HTML tags από τη διεύθυνση, τον τίτλο και την περίληψη, ώστε να μείνει μόνο καθαρό κείμενο, για να μπορεί να καταχωρηθεί σωστά στο ευρετήριο.
   - Εγγραφή στο τρίτο log αρχείο (out3.log) της URL διεύθυνσης του αποτελέσματος και μετά ενός αστερίσκου *.

- Εγγραφή στο δεύτερο και στο τέταρτο log αρχείο (out2.log & out4.log) των παρακάτω στοιχείων:
  - Ερώτημα χρήστη
  - Τίτλος αποτελέσματος
  - Περίληψη αποτελέσματος
  - URL διεύθυνση αποτελέσματος
- Αφαίρεση από το αρχικό string των στοιχείων του αποτελέσματος που βρήκαμε, ώστε η επόμενη επανάληψη να συνεχίσει στα επόμενα αποτελέσματα.

7. Οι επαναλήψεις τελειώνουν όταν έχουν απομονωθεί τα στοιχεία όλων των αποτελεσμάτων που επέστρεψε το Google.
8. Δημιουργία ευρετηρίου για τα αποτελέσματα του τελευταίου ερωτήματος, τα στοιχεία των οποίων βρίσκονται στο αρχείο out4.log.
9. Υπολογισμός των χαρακτηριστικών των αποτελεσμάτων του τελευταίου ερωτήματος.
10. Δημιουργία του αρχείου εισόδου του αλγορίθμου ταξινόμησης SVM. Το αρχείο αυτό περιλαμβάνει τα χαρακτηριστικά των αποτελεσμάτων που υπολογίστηκαν στο προηγούμενο βήμα.
11. Ανακατάταξη των αποτελεσμάτων του ερωτήματος με βάση τα χαρακτηριστικά τους και το μοντέλο προτιμήσεων του χρήστη χρησιμοποιώντας την εφαρμογή svm_classify του SVM^{light}.
12. Εγγραφή στο αρχείο out.log των URL διευθύνσεων όλων των αποτελεσμάτων με τη νέα κατάταξη, χωρισμένων μεταξύ τους με έναν αστερίσκο *.
13. Εμφάνιση των αποτελεσμάτων στο χρήστη με τη νέα κατάταξη.

5.2 Ανακατάταξη αποτελεσμάτων

Στην ενότητα αυτή θα αναλύσουμε τα τελευταία βήματα του παραπάνω αλγορίθμου που περιγράφουν τη διαδικασία αλλαγής της κατάταξης των αποτελεσμάτων του Google, σύμφωνα με τις προτιμήσεις του χρήστη που είναι καταγεγραμμένες στο μοντέλο που δημιουργήθηκε με τον SVM αλγόριθμο εκπαίδευσης.

5.2.1 *Indexing αποτελεσμάτων*

Για να γίνει η ανακατάταξη των αποτελεσμάτων, πρέπει πρώτα να υπολογίσουμε τα χαρακτηριστικά τους. Η διαδικασία όμως αυτή προϋποθέτει ότι τα αποτελέσματα είναι καταγεγραμμένα σε ένα ευρετήριο. Συνεπώς το πρώτο μας βήμα είναι να προσθέσουμε τα στοιχεία των αποτελεσμάτων του τελευταίου ερωτήματος σε ένα ευρετήριο.

Η δημιουργία του ευρετηρίου αυτού θα γίνει με παρόμοιο τρόπο με αυτόν που περιγράψαμε στη φάση της εκπαίδευσης, χρησιμοποιώντας και πάλι το Lucene. Αρχίζουμε διαβάζοντας το αρχείο out4.log και φτιάχνοντας ένα Lucene Document για τις τέσσερις γραμμές του αρχείου, με πεδία:

- Query (ερώτημα),
- Title (τίτλος),
- Abstract (περίληψη),
- URL (διεύθυνση).

Επιλέγουμε να κάνουμε Index όλα τα πεδία, ώστε να μπορούμε να κάνουμε αναζήτηση σε αυτά, και να αποθηκεύσουμε (Store) τα πεδία Query και URL. Κατόπιν χρησιμοποιούμε τον WhitespaceAnalyzer για να χωρίσουμε το κείμενο σε λέξεις (tokens), προκειμένου να γίνει καταλληλότερο για indexing. Αφού έχουν τελειώσει οι παραπάνω διαδικασίες, το κείμενο είναι έτοιμο για προσθήκη στο ευρετήριο.

5.2.2 *Υπολογισμός των χαρακτηριστικών των αποτελεσμάτων*

Αφού δημιουργηθεί το ευρετήριο, πρέπει να γίνει ο υπολογισμός των τιμών των χαρακτηριστικών των αποτελεσμάτων του ερωτήματος. Τα χαρακτηριστικά είναι τα ίδια που χρησιμοποιήσαμε και στη φάση της εκπαίδευσης και οι τιμές τους υπολογίζονται με τον τρόπο που αναλύθηκε στην ενότητα 4.3. Αναλυτικά έχουμε:

1. *Ομοιότητα ερωτήματος – τίτλου εγγράφου*, υπολογισμένη με τη συνάρτηση ομοιότητας κειμένου του Lucene.
2. *Ομοιότητα ερωτήματος – περίληψης εγγράφου*, υπολογισμένη με τη συνάρτηση ομοιότητας κειμένου του Lucene.
3. *Ομοιότητα ερωτήματος – URL διεύθυνσης εγγράφου*, υπολογισμένη με τη συνάρτηση ομοιότητας κειμένου του Lucene.
4. *Ομοιότητα ερωτήματος – τίτλου εγγράφου*, υπολογισμένη με τη συνάρτηση ομοιότητας κειμένου tf · idf.

5. *Ομοιότητα ερωτήματος – περίληψης εγγράφου*, υπολογισμένη με τη συνάρτηση ομοιότητας κειμένου tf · idf.
6. *Ομοιότητα ερωτήματος – URL διεύθυνσης εγγράφου*, υπολογισμένη με τη συνάρτηση ομοιότητας κειμένου tf · idf.
7. *Ομοιότητα ερωτήματος – τίτλου εγγράφου*, υπολογισμένη με τη συνάρτηση ομοιότητας κειμένου BM25.
8. *Ομοιότητα ερωτήματος – περίληψης εγγράφου*, υπολογισμένη με τη συνάρτηση ομοιότητας κειμένου BM25.
9. *Ομοιότητα ερωτήματος – URL διεύθυνσης εγγράφου*, υπολογισμένη με τη συνάρτηση ομοιότητας κειμένου BM25.
10. *Βαθμολογία με βάση τη θέση κατάταξης του εγγράφου στο Google*.
11. *Domain της URL διεύθυνσης του εγγράφου*.

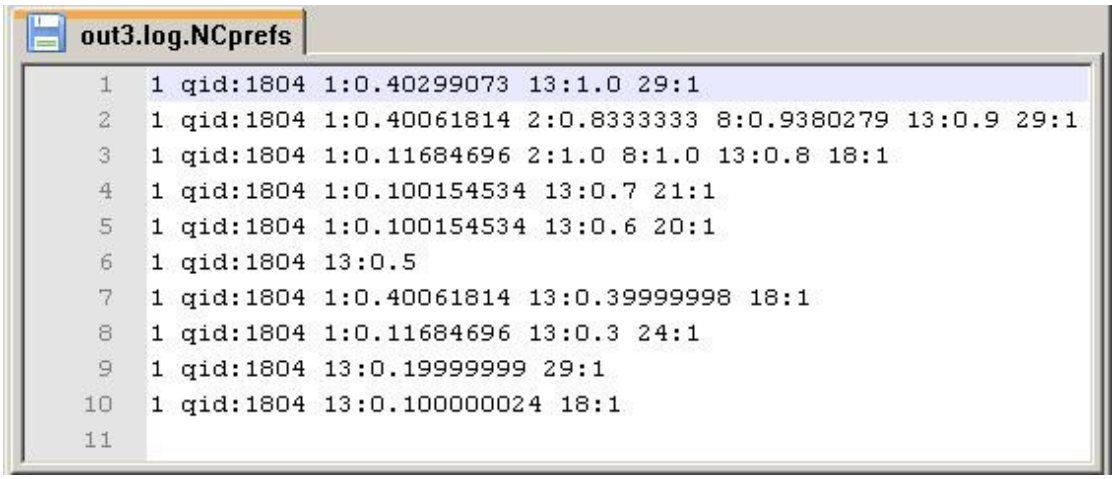
5.2.3 Αρχείο εισόδου SVM αλγορίθμου ταξινόμησης

Αφού υπολογιστούν οι παραπάνω τιμές των χαρακτηριστικών των αποτελεσμάτων, γράφονται στο αρχείο εισόδου του SVM αλγορίθμου ταξινόμησης με παρόμοιο τρόπο όπως έγινε και κατά την εκπαίδευση. Υπενθυμίζουμε ότι κάθε γραμμή του αρχείου αναπαριστά ένα έγγραφο και έχει την εξής μορφή:

```
<target> qid:<queryID> <feature>:<value> ... <feature>:<value>
```

όπου <queryID> είναι ο αύξων αριθμός του ερωτήματος, <feature> είναι ο αύξων αριθμός του χαρακτηριστικού, και <value> είναι η τιμή για το αντίστοιχο χαρακτηριστικό. Η διαφορά σε σχέση με το αρχείο της εκπαίδευσης είναι στην τιμή <target>. Στη φάση της εκπαίδευσης το <target> είναι η τιμή προτίμησης του αποτελέσματος που έχει υπολογιστεί σύμφωνα με τις έμμεσες προτιμήσεις του χρήστη. Τώρα όμως, στη φάση της ανακατάταξης, η τιμή αυτή δεν έχει τέτοιο νόημα καθώς ο χρήστης δεν έχει δει τα αποτελέσματα ακόμα. Συνεπώς δεν παίζει ρόλο στον αλγόριθμο ταξινόμησης τι τιμή θα έχει το <target>, οπότε επιλέγουμε αυθαίρετα να έχει πάντα την τιμή 1.

Στην επόμενη εικόνα φαίνεται ένα παράδειγμα αρχείου εισόδου του SVM αλγορίθμου ταξινόμησης.

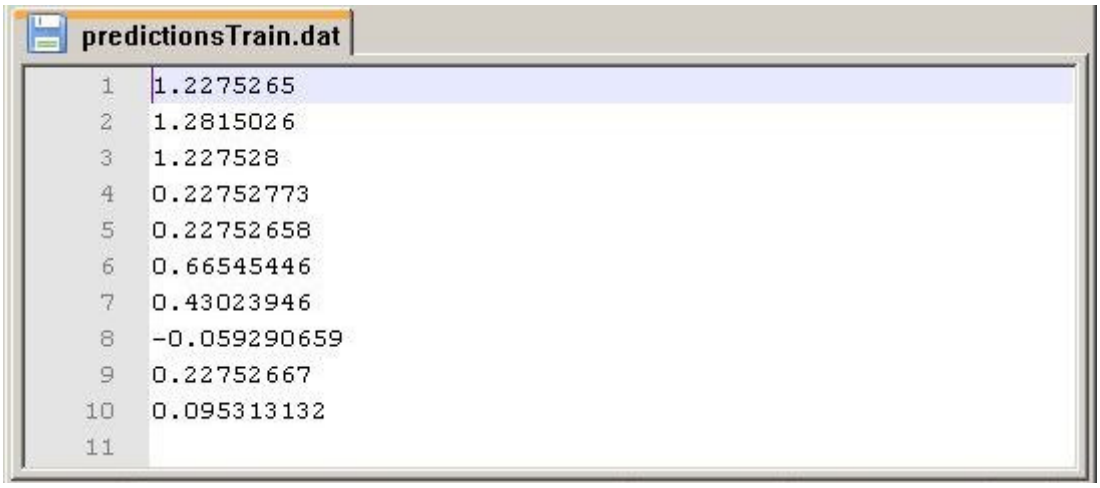


Εικόνα 17: Παράδειγμα αρχείου εισόδου εφαρμογής αναταξινόμησης svm_classify

5.2.4 Αναταξινόμηση αποτελεσμάτων με την εφαρμογή SVM_classify

Αφού ετοιμαστεί το παραπάνω αρχείο, μπορούμε να το δώσουμε σαν είσοδο στον αλγόριθμο SVM ταξινόμησης, μαζί με το μοντέλο που εκπαιδεύτηκε προηγουμένως. Ο αλγόριθμος αυτός αποφασίζει τι κατάταξη θα έπρεπε να έχουν τα αποτελέσματα, σύμφωνα με τα χαρακτηριστικά τους και τις προτιμήσεις του χρήστη για τα χαρακτηριστικά αυτά, όπως έχουν υπολογιστεί από το εκπαιδευμένο SVM μοντέλο. Η υλοποίηση του αλγορίθμου που χρησιμοποιούμε είναι αυτή της εφαρμογής svm_classify που προσφέρει το SVM-light.

Η έξοδος του αλγορίθμου είναι ένα αρχείο με τόσες γραμμές όσα τα αποτελέσματα που θα παρουσιαστούν στο χρήστη. Κάθε γραμμή αντιπροσωπεύει το αντίστοιχο αποτέλεσμα της αρχικής κατάταξης που επέστρεψε το Google. Σε κάθε γραμμή υπάρχει ένας πραγματικός αριθμός. Η γραμμή που περιέχει τον μεγαλύτερο από τους αριθμούς αυτούς αντιπροσωπεύει το αποτέλεσμα που θα πρέπει να είναι πρώτο στη νέα κατάταξη. Η γραμμή με τον αμέσως επόμενο μεγαλύτερο αριθμό αντιπροσωπεύει το δεύτερο αποτέλεσμα της νέας κατάταξης, και ομοίως συνεχίζεται η ίδια διαδικασία για να βρούμε τη θέση που θα έχουν στη νέα κατάταξη και τα υπόλοιπα αποτελέσματα.



Εικόνα 18: : Αρχείο εξόδου εφαρμογής αναταξινόμησης svm_classify

Στο παράδειγμα της παραπάνω εικόνας βλέπουμε ότι το αποτέλεσμα #2 έχει την μεγαλύτερη τιμή και άρα θα είναι στην πρώτη θέση της νέας κατάταξης. Ακολουθεί το αποτέλεσμα #3 που θα πάει στη δεύτερη θέση της νέας κατάταξης. Συνολικά, η νέα κατάταξη των παραπάνω αποτελεσμάτων θα είναι ως εξής: 2, 3, 1, 4, 5, 6, 7, 9, 10, 8.

5.2.5 Παρουσίαση στο χρήστη των αναταξινομημένων αποτελεσμάτων

Για να παρουσιάσουμε λοιπόν στο χρήστη τη νέα κατάταξη, ανακατατάσσουμε τα αποτελέσματα αναλόγως με την αντίστοιχη τιμή τους από τον αλγόριθμο ταξινόμησης SVM. Αφού γίνει η ανακατάταξη, επιστρέφονται τα αποτελέσματα σε HTML μορφή στο χρήστη. Όπως και στη φάση της εκπαίδευσης, όταν ο χρήστης κάνει κλικ σε κάποιο αποτέλεσμα, πρώτα θα καταγραφεί το κλικ του από τον proxy server και μετά θα ανακατευθυνθεί στη σελίδα που ζήτησε. Συνεπώς η δραστηριότητα του χρήστη συνεχίζεται να καταγράφεται ώστε να μπορούμε να κάνουμε όποτε θελήσουμε επανεκπαίδευση του μοντέλου, προκειμένου να έχουμε ακόμα πιο ενημερωμένες πληροφορίες για τις προτιμήσεις του χρήστη.

6

Έλεγχος συστήματος

6.1 Εκπαίδευση μοντέλου με την εφαρμογή svm_learn

Στην ενότητα αυτή, θα ελέγξουμε αν το σύστημα λειτουργεί όπως αναμένουμε, δείχνοντας παραδείγματα αναταξινομημένων αποτελεσμάτων για διάφορα εκπαιδευμένα μοντέλα.

Τα παραδείγματα που θα δείξουμε εκμεταλλεύονται κυρίως το χαρακτηριστικό του domain της διεύθυνσης του αποτελέσματος, καθώς είναι ένας από τους πιο εύκολους τρόπος για να δείξουμε τη διαφορά στην κατάταξη των αποτελεσμάτων μετά την εκπαίδευση του μοντέλου.

Για την αρχική εκπαίδευση του μοντέλου χρησιμοποιούμε την επιλογή της απλής αναζήτησης μέσω Google, ζητώντας 30 αποτελέσματα για κάθε ερώτημα. Φροντίζουμε να μην κάνουμε κλικ σε αποτελέσματα που έχουν διεύθυνση τύπου “edu”. Αντιθέτως, προτιμάμε να πατάμε σε αποτελέσματα που έχουν domain “com”, “org”, ή “net”. Συγκεκριμένα, πραγματοποιούμε την εκπαίδευση του μοντέλου (svm_learn) όταν έχουμε πατήσει σε 30 αποτελέσματα τύπου “com”, 20 αποτελέσματα τύπου “org”, και 10 αποτελέσματα τύπου “net”. Αναλυτικά τα αποτελέσματα της εκπαίδευσης φαίνονται στην επόμενη εικόνα.

svm_learn

```
Osmot loading config file from osmot.conf
Current working directory is C:\Windseeker\Lucene-src
Indexing to directory 'C:\Windseeker\index'...
Reading log 'C:/Windseeker/logs/out2.log'...
328 total milliseconds
20090221193919 all 8 queries found.
Searcher - search(Query)query:clickstream
Searcher - search(Query)query:computer
Searcher - search(Query)query:database
Searcher - search(Query)query:implicit query:feedback
Searcher - search(Query)query:magic
Searcher - search(Query)query:ntua
Searcher - search(Query)query:support query:vector
Searcher - search(Query)query:search query:engine
Command is: svm_learn -z p out.log.NCprefs model.dat
Scanning examples...done
Reading examples into memory...100..200..OK. (237 examples read)
Constructing 1477 rank constraints...done.
Setting default regularization parameter C=0.1897
Optimizing......done. (1683 iterations)
Optimization finished (334 misclassified, maxdiff=0.00077).
Runtime in cpu-seconds: 0.42
Number of SV: 799 (including 784 at upper bound)
L1 loss: loss=702.61570
Norm of weight vector: |w|=4.13852
Norm of longest example vector: |x|=7.81999
Estimated VCdim of classifier: VCdim<=1048.37337
Computing XiAlpha-estimates...done
Runtime for XiAlpha-estimates in cpu-seconds: 0.00
XiAlpha-estimate of the error: error<=54.03% (rho=1.00,depth=0)
XiAlpha-estimate of the recall: recall=>47.64% (rho=1.00,depth=0)
XiAlpha-estimate of the precision: precision=>67.38% (rho=1.00,depth=0)
Number of kernel evaluations: 109470
Writing model file...done
```

Εικόνα 19: Εκπαίδευση του μοντέλου με την εφαρμογή svm_learn

Από τα στοιχεία που παρέχει ο SVM αλγόριθμος κατά την εκπαίδευση του μοντέλου και φαίνονται στην παραπάνω εικόνα, έχουμε ότι:

- Πραγματοποιήθηκαν 8 ερωτήματα.
- Παρουσιάστηκαν συνολικά 237 αποτελέσματα για τα ερωτήματα αυτά.
- Προέκυψαν 1477 σχετικές προτιμήσεις από τα 60 κλικ που κάναμε.
- Δημιουργήθηκαν 799 support vectors.

Κατόπιν γίνεται μία αυτό-αξιολόγηση του μοντέλου για τα δεδομένα της εκπαίδευσης με τις εξής εκτιμήσεις:

- Σφάλμα $\leq 54{,}03\%$
- Ανάκληση $\geq 47{,}64\%$
- Ακρίβεια $\geq 67{,}38\%$

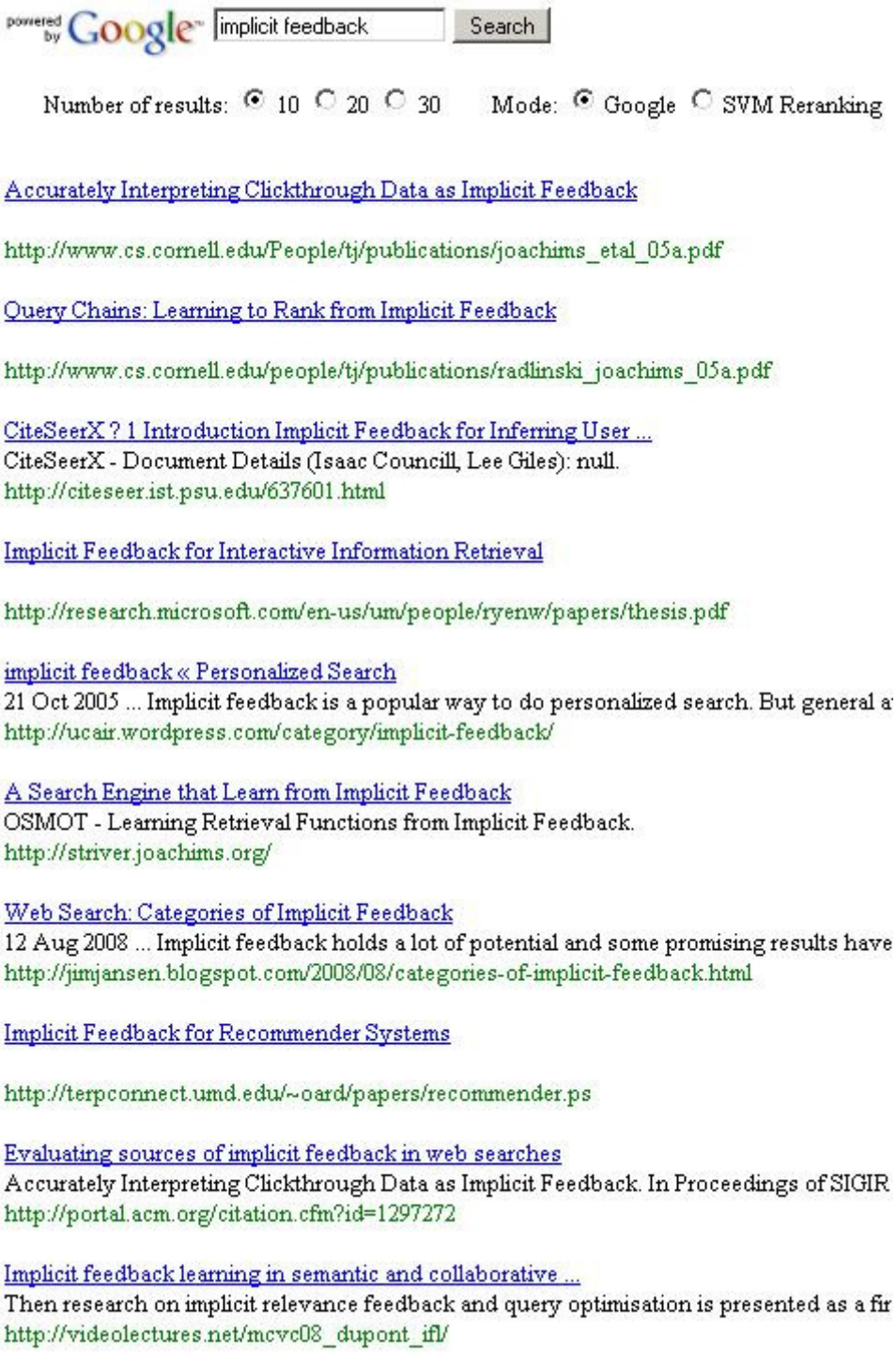
Από την αξιολόγηση αυτή φαίνεται ότι το μοντέλο δεν είναι ακόμα πολύ καλό στην ταξινόμηση των αποτελεσμάτων με τον τρόπο που θα ήθελε ο χρήστης. Αυτό οφείλεται στα λίγα ερωτήματα και κλικ που κάναμε. Παρ' όλα αυτά, για τους σκοπούς του παραδείγματός μας, που μελετάμε κυρίως την επίδραση του χαρακτηριστικού του domain, θεωρείται ικανοποιητικό.

6.2 Παράδειγμα χρήσης

6.2.1 Βήμα 1

Έχοντας εκπαιδεύσει το μοντέλο, κάνουμε μερικές αναζητήσεις χρησιμοποιώντας την επιλογή της ανακατάταξης των αποτελεσμάτων. Από τα αποτελέσματα που μας παρουσιάζονται είναι φανερό ότι οτιδήποτε αποτέλεσμα ανήκει στα domain που πατάγαμε κατά την εκπαίδευση (".com", ".org", ".net") είναι υψηλά στην κατάταξη, ενώ αντιθέτως τα υπόλοιπα είναι πάντα στα τελευταία αποτελέσματα της κατάταξης.

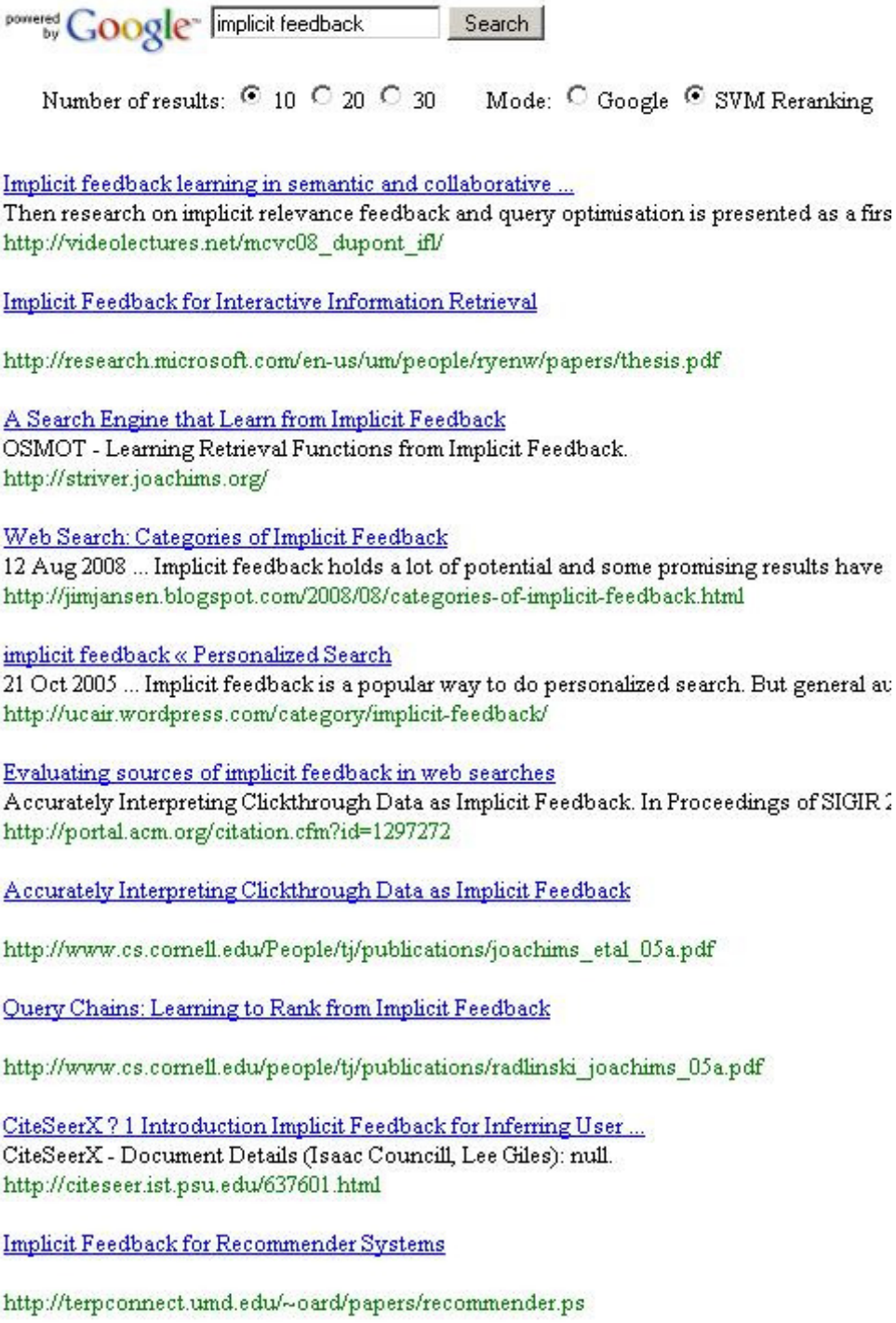
Για παράδειγμα, για το ερώτημα "implicit feedback", σε αναζήτηση χωρίς ανακατάταξη, αρκετά από τα αποτελέσματα στην κορυφή της κατάταξης είναι του domain ".edu", όπως φαίνεται στην παρακάτω εικόνα.



Εικόνα 20: Αναζήτηση χωρίς ανακατάταξη αποτελεσμάτων

6.2.2 Βήμα 2

Αντίθετα, όταν γίνει το ίδιο ερώτημα με ανακατάταξη με βάση το εκπαιδευμένο μοντέλο, η κατάταξη που παίρνουμε έχει πάει στο τέλος όλα τα αποτελέσματα του domain ".edu", καθώς κατά την εκπαίδευση δεν τα είχαμε προτιμήσει. Συμπεραίνουμε ότι το μοντέλο "κατάλαβε" στην εκπαίδευση ότι προτιμάμε τα έγγραφα τύπου ".com", ".org" και ".net" και τα ανεβάζει πλέον πιο ψηλά στην κατάταξη.



Εικόνα 21: Ανακατάταξη μετά από εκπαίδευση με κλικ σε έγγραφα: 30 com, 20 org, 10 net

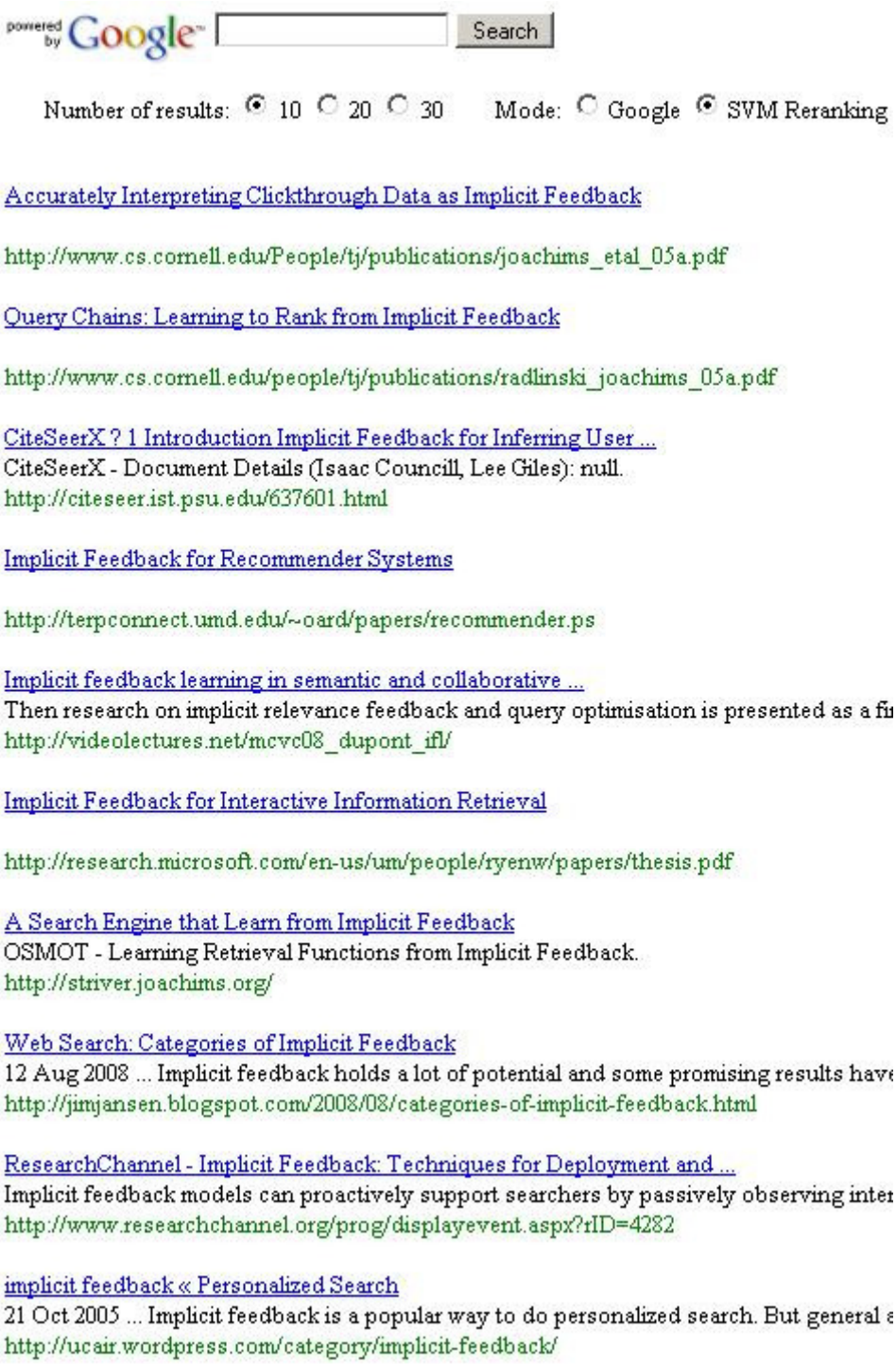
6.2.3 Βήμα 3

Για το επόμενο στάδιο του παραδείγματος, κάνουμε μερικά ερωτήματα και κάνουμε κλικ σε έξι αποτελέσματα με domain ".edu" για να δούμε τι επίδραση θα έχει στην κατάταξη. Εκπαιδεύουμε πάλι το μοντέλο, στο οποίο προστίθενται οι νέες προτιμήσεις αλλά διατηρούνται και οι προηγούμενες. Όπως φαίνεται και στην επόμενη εικόνα, τα αποτελέσματα του domain ".edu" άρχισαν να κερδίζουν πιο υψηλές θέσεις στην κατάταξη.

Εικόνα 22: Ανακατάταξη μετά από εκπαίδευση με κλικ σε έγγραφα: 30 com, 20 org, 10 net, 6 edu

6.2.4 Βήμα 4

Για το τελικό βήμα του παραδείγματος, κάνουμε μερικά ερωτήματα ακόμα και πατάμε σε τριάντα αποτελέσματα με domain ".edu" μόνο. Πλέον όλα τα αποτελέσματα στην κορυφή της κατάταξης είναι του domain ".edu", όπως φαίνεται παρακάτω.



Εικόνα 23: Ανακατάταξη μετά από εκπαίδευση με κλικ σε έγγραφα: 30com, 20org, 10net, 34edu

7

Επίλογος

7.1 Σύνοψη και συμπεράσματα

Η εξατομίκευση των αποτελεσμάτων των μηχανών αναζήτησης είναι ολοένα και πιο απαραίτητη στις μέρες μας, όπου η χρήση του Διαδικτύου και των μηχανών αναζήτησης αυξάνει με μεγάλους ρυθμούς. Είναι φανερό ότι η ενιαία κατάταξη των αποτελεσμάτων για όλους τους χρήστες δεν είναι η καλύτερη προσέγγιση και η εργασία μας απευθύνεται ακριβώς στο θέμα αυτό. Η λύση που προτείνουμε είναι ένα μοντέλο που εκπαιδεύεται από το ιστορικό των κλικ του χρήστη της μηχανής αναζήτησης, και καταγράφει τι χαρακτηριστικά έχουν τα έγγραφα που προτιμάει ο χρήστης. Χρησιμοποιώντας το μοντέλο αυτό, μπορούμε να παρουσιάσουμε τα αποτελέσματα της μηχανής αναζήτησης με κατάταξη βασισμένη στις προτιμήσεις του χρήστη.

Η καταγραφή των ερωτημάτων που θέτει ο χρήστης, των αποτελεσμάτων που του παρουσιάζονται, καθώς και των κλικ που κάνει, γίνονται στο παρασκήνιο χωρίς να επιβαρύνουν το χρήστη. Με τον τρόπο αυτό, μπορούμε να μαζέψουμε μεγάλο όγκο πληροφοριών με μικρό κόστος. Καταγράφουμε τα δεδομένα αυτά σε ένα ευρετήριο για να μπορέσουμε να τα επεξεργαστούμε πιο εύκολα και να υπολογίσουμε τα χαρακτηριστικά τους. Επίσης, εκφέρουμε κάποιες σχετικές προτιμήσεις από τις επιλογές που κάνει ο χρήστης όταν αγνοεί εσκεμμένα κάποια αποτελέσματα και προτιμά να πατήσει σε κάποια άλλα.

Όλα τα παραπάνω εισάγονται στον αλγόριθμο εκπαίδευσης Support Vector Machine και έτσι δημιουργείται το μοντέλο που θα χρησιμοποιήσουμε για την ανακατάταξη. Όταν ο χρήστης κάνει το ερώτημα του στη μηχανή αναζήτησης, έχει πλέον τη δυνατότητα να ζητήσει αναταξινόμηση των αποτελεσμάτων με βάση το εκπαιδευμένο μοντέλο. Κατόπιν αναλύονται τα χαρακτηριστικά των αποτελεσμάτων της μηχανής αναζήτησης και υπολογίζεται από τον αλγόριθμο κατάταξης Support Vector Machine τι θέση πρέπει να έχουν τα αποτελέσματα αυτά, με βάση τις προτιμήσεις που έχουν καταχωρηθεί στο μοντέλο. Έτσι τα αποτελέσματα παρουσιάζονται στο χρήστη με κατάταξη εξατομικευμένη στις προτιμήσεις του.
Η εφαρμογή χρειάζεται να εκπαιδευτεί με πολλά δεδομένα για να μπορέσουμε να κρίνουμε την αποτελεσματικότητα της. Έχει ήδη τεθεί σε λειτουργία στο Ινστιτούτο Πληροφοριακών Συστημάτων και Προσομοίωσης, καθώς και στο Εργαστήριο Συστημάτων Βάσων Γνώσεων και Δεδομένων και συλλέγονται δεδομένα.

7.2 Μελλοντικές επεκτάσεις

Η εφαρμογή που σχεδιάσαμε έχει το πλεονέκτημα ότι μπορεί εύκολα να επεκταθεί και να βελτιώσει την αποδοτικότητα της λειτουργίας της.

Ένας τρόπος επέκτασης είναι με την προσθήκη περισσότερων χαρακτηριστικών (features) για τα αποτελέσματα που παρουσιάζονται στο χρήστη, έτσι ώστε να μπορούμε να εξάγουμε με περισσότερη ακρίβεια τις προτιμήσεις του. Χαρακτηριστικά που θα μπορούσαμε να υλοποιήσουμε είναι:

- Επιπλέον συναρτήσεις ομοιότητας κειμένου.
- Θέση κατάταξης του αποτελέσματος σε άλλες μηχανής αναζήτησης.
- Συγκεκριμένες λέξεις – κλειδιά που παρουσιάζονται στον τίτλο ή την περίληψη του αποτελέσματος (δυαδικό χαρακτηριστικό: 1 αν εμφανίζεται η λέξη, 0 αν δεν εμφανίζεται – π.χ. forum ή wiki).

Άλλος τρόπος επέκτασης είναι να εξάγουμε επιπλέον προτιμήσεις χρησιμοποιώντας τις στρατηγικές που αναλύσαμε για τις αλυσίδες ερωτημάτων. Στον πηγαίο κώδικα της εργασίας έχει γίνει ήδη μια προεργασία για την επέκταση αυτή, οπότε μπορεί εύκολα να υλοποιηθεί.

Τέλος, η εφαρμογή μπορεί να μεταφερθεί και να υλοποιηθεί σαν extension του περιηγητή Mozilla Firefox. Αυτό θα μας δώσει πολλές επιπλέον πληροφορίες καταγράφοντας ενέργειες όπως ο χρόνος ανάγνωσης ή κύλισης σε κάθε ιστοσελίδα, η προσθήκη στα αγαπημένα, η αποθήκευση σε αρχείο ή η εκτύπωση. Από τις πληροφορίες αυτές μπορούμε να εξάγουμε ακόμα περισσότερες προτιμήσεις του χρήστη οι οποίες θα βοηθήσουν στο να είναι πιο ακριβές το μοντέλο.

8

Βιβλιογραφία

[CLW+01] M. Claypool, P. Le, M. Waseda, D. Brown, *Implicit Interest Indicators*, Proceedings of ACM Intelligent User Interfaces Conference, 2001.

[GH05] O. Gospodnetic, E. Hatcher, *Lucene in Action*, Manning Publications Co, 2005.

[JGP+05] T. Joachims, L. Granka, B. Pan, H. Hembrooke, G. Gay, *Accurately Interpreting Clickthrough Data as Implicit Feedback*, Proceedings of the Conference on Research and Development in Information Retrieval (SIGIR), 2005.

[JGP+07] T. Joachims, L. Granka, B. Pan, H. Hembrooke, F. Radlinski, G. Gay, *Evaluating the Accuracy of Implicit Feedback from Clicks and Query Reformulations in Web Search*, ACM Transactions on Information Systems (TOIS), Vol. 25, No. 2 (April), 2007.

[Joa02] T. Joachims, *Optimizing Search Engines Using Clickthrough Data*, Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD), ACM, 2002.

[Joa99] T. Joachims, *SVM^{light} Support Vector Machine*, http://svmlight.joachims.org/ , 1999.

[JR07] T. Joachims, F. Radlinski, *Search Engines that Learn from Implicit Feedback*, IEEE Computer, Vol. 40, No. 8, August, 2007.

[KT03] D. Kelly, J. Teevan, *Implicit Feedback for Inferring User Preference: A Bibliography*, ACM SIGIR Forum, Vol. 37, No. 2, 2003.

[LXQ+07] T. Liu, J. Xu, T. Qin, W. Xiong, H. Li, *LETOR: Benchmark dataset for research on learning to rank for information retrieval,* LR4IR 2007, in conjunction with SIGIR 2007

[Nic97] D. Nichols, *Implicit Rating and Filtering*, Proceedings of the 5th DELOS Workshop on Filtering and Collaborative Filtering, 1997

[Per08] J. Perez-Iglesias, *Integrating BM25 & BM25F into Lucene*, http://nlp.uned.es/~jperezi/Lucene-BM25 , 2008

[RJ05] F. Radlinski and T. Joachims, *Query Chains: Learning to Rank from Implicit Feedback*, Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD), ACM, 2005.

[She04] P. Sherrod, *Introduction to Support Vector Machine (SVM) Models*, http://www.dtreg.com/svm.htm , 2004