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

Ομαδοποίηση αποτελεσμάτων μηχανών αναζήτησης και επέκταση ερωτήματος με χρήση αυτο-οργανούμενων χαρτών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Βαγγέλης Β. Κώστας

Επιβλέπων : Ανδρέας-Γεώργιος Σταφυλοπάτης
Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2014

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

Ομαδοποίηση αποτελεσμάτων μηχανών αναζήτησης και επέκταση ερωτήματος με χρήση αυτο-οργανούμενων χαρτών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Βαγγέλης Β. Κώστας

Επιβλέπων : Ανδρέας-Γεώργιος Σταφυλοπάτης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 30η Οκτωβριου 2014.

.....	.....	.....
Ανδρέας Σταφυλοπάτης	Γιώργος Στάμου	Στέφανος Κόλλιας
Καθηγητής Ε.Μ.Π	Επίκουρος Καθηγητής Ε.Μ.Π	Καθηγητής Ε.Μ.Π

Αθήνα, Οκτώβριος 2014

.....
Βαγγέλης Β. Κώστας

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

Contents

1 Εισαγωγή

Μια μηχανή αναζήτησης είναι σχεδιασμένη να ψάχνει πληροφορία στο διαδίκτυο και να αναπαριστά τα αποτελέσματα σε μια λίστα. Η λίστα ταξινομείται με την χρήση ενός κριτηρίου που εισήγαγε πρώτη η google το 2000 και λέγετε PageRank. Αυτός είναι και ένας από τους λόγους που έφεραν την google στην σημερινή της θέση στην αγορά. Ο αριθμός PageRank ενός ιστότοπου υπολογίζετε με την χρήση των PageRanks των ιστοτοπων που συνδέουν προς αυτόν και των συνδέσεων που έχει ο ίδιος προς όλους ιστότοπους. Αλλοι αλγόριθμοι που έχουν προταθεί απο την microsoft για τον υπολογισμό του PageRank λαμβάνουν υπόψιν και τον μέσο χρόνο που περνάει ένας επισκέπης στον συγκεκριμένο ιστότοπο. Ένας εναλλακτικώς τρόπος παρουσίασης των αποτελεσμάτων είναι αυτός που τα παρουσιάζει μέσα σε κατηγορίες με τίτλους και ιεραρχία, αυτό προσπαθούν να πετύχουν οι μεταμηχανες αναζήτησης. Για να συμβεί η ιεραρχική παρουσίαση πρέπει να υπολογιστεί ενα ιεραρχικό clustering των αποτελεσμάτων, Με την παραπανω προσέγγιση συμφωνούν οι περισσοτερες σημερινές μεταμηχανες αναζήτησης αν και χρησιμοποιούν διαφορετικούς αλγορίθμους. Ο διευθυντής έρευνας της google Peter Norvig είπε για αυτό "clustering technology is the PageRank of the future". Η αποφυγή του PageRank ως το μόνο μέτρο ποιότητας ιστότοπου θα αποφύγει σχεδόν εξ ολοκλήρου την προκατάληψη μηχανής αναζήτησης(Search Engine Bias[9])η του λεγομένου "google bombing". Στην παρούσα εργασία θα προσπαθήσουμε να δώσουμε περισσότερη πληροφορία στον χρήστη για τα αποτελέσματα δίνοντας όχι μόνο την ιεραρχία αλλά και το μέτρο ομοιότητας μεταξύ διαφορετικών κατηγοριών. Για να κάνουμε την παραπάνω πληροφορία ευκολότερα προσβάσιμη στον μέσο χρήστη θα πρέπει να παραστήσουμε τα αποτελέσματα σε δισδιάστατο χώρο με την απόσταση να εκπροσωπεί την ομοιότητα. Για να το πετύχουμε αυτό θα πρέπει να επιλέξουμε έναν αλγόριθμο για clustering που να διαφυλάσσει την τοπολογία των αποτελεσμάτων. Επιλέξαμε μια αυξανόμενη έκδοση του αλγορίθμου αυτοοργανουμενων χαρτών SOM την Growing Grid Som γιατί εκπληρώνει τους παραπάνω όρους σε λογικούς χρόνους. Σε πρώτο στάδιο χρειάστηκε να αποφασίσουμε πως θα λαμβάνουμε τα αποτελέσματα κάποιας ερώτησης. Αποφασίσαμε να χρησιμοποιήσουμε τα αποτελέσματα που μας δίνει η google σαν το αρχικό σώμα αποτελεσμάτων πάνω στο οποίο θα εφαρμοστεί ο αλγόριθμος μας. Εδώ θα μπορούσαμε να υπολογίσουμε τα χαρακτηριστικά(features) που μας ενδιαφέρουν για το clustering χρησιμοποιώντας ολόκληρο το κείμενο του ιστότοπου αυτό όμως αποδείχθηκε να είναι αρκετά χρονοβόρο γιατί θα πρέπει να επισκεπτόμαστε όλους τους ιστότοπους των αποτελεσμάτων αφού δεν έχουμε μια δικιά μας βάση cached δεδομένων. Γιαυτό λοιπόν αποφασίσαμε να χρησιμοποιήσουμε μόνο το μικρο κείμενο που βρίσκετε στα snippets και titles των αποτελεσμάτων που γυρίζει η google. Ένα θετικό χαρακτηριστικό του μεγέθους των snippets είναι ότι έχουνε χώρο μόνο για ένα θέμα. Στα παραπάνω συμπεράσματα φαίνεται να κατέληξαν και οι σχεδιαστές διαφόρων μεταμηχανων αναζήτησης με χρήση snippet όπως VIVISIMO, MOOTER, COPENIC, iBoogie, KARTOO, Groxis, DOGPILE, Clusty.

Για να μεταφέρουμε τα snippets σε διανυσματικό χώρο ώστε να μπορεί μετά το corpus να χρησιμοποιηθεί για την εκπαίδευση του νευρωνικού δικτύου αποφασίσαμε να χρησιμοποιήσουμε την τεχνική Latent Semantic Analysis(LSA) η οποία αν και υπολογιστικά εντατική αποφέρει καλά αποτελέσματα. Αυτή η τεχνική βελτιώνει το κλασικό μοντέλο διανυσματικού χώρου με την χρήση της Singular Value Decomposition(SVD) η οποία βοηθάει στο να αφαιρεθεί ο θόρυβος που εισαγάγει

η χρήση συνώνυμων. Στην συνέχεια το corpus με τα διανυσματικά μοντέλα των snippets χρησιμοποιείτε για την εκπαίδευση ενός νευρωνικού. Τα αποτελέσματα αυτής της διαδικασίας τα χρησιμοποιούμε για να συμπεράνουμε την ιεραρχία των αποτελεσμάτων αναζήτησης και να υπολογίσουμε τα ονόματα των κατηγοριών στις οποίες φαίνεται να ανήκουν.

Η σωστή ομαδοποίηση των αποτελεσμάτων η οποία αντανακλά με κάποιο νοητό τρόπο τα διαφορετικά και δυνητικά απεριόριστα "θέματα" σε έναν τόσο δυναμικό χώρο σαν το διαδίκτυο αποτελεί μια απαιτητική εργασία. Σε αυτόν τον χώρο οποιαδήποτε ονομασία κατηγορίας από μια σταθερή λίστα από ονομασίες δεν είναι εφικτή η δεν θα είναι αρκετά ευέλικτη ώστε να συλλάβει το αντικείμενο του κάθε ιστότοπου γιαυτό αναγκαζόμαστε να υπολογίσουμε τις ονομασίες "on the fly". Στη συνέχεια θα αναλυθούν σε βάθος τα θέματα που προέκυψαν και οι τρόποι επίλυσης τους.

Στο κεφάλαιο 3 θα δούμε μια επισκόπηση όλου του συστήματος εξηγώντας περιληπτικά τις θεωρητικές έννοιες και την πρακτική υλοποίηση τους. Στα κεφάλαια 4-6 θα εμβαθύνουμε περαιτέρω στο θεωρητικό υπόβαθρο του συστήματος και στο 7 θα δούμε μερικούς μαθηματικούς υπολογισμούς και αριθμητικά παραδείγματα. Τέλος στο κεφάλαιο 8 θα εξηγήσουμε της επιλογές μας και στο 9 θα αναφέρουμε πιθανές βελτιώσεις.

2 Μεταμηχανές αναζήτησης

Υπάρχουν διάφορα συστήματα ομαδοποίησης των αποτελεσμάτων αναζήτησης στο διαδίκτυο που οπως εξηγήσαμε παραπάνω λέγονται μεταμηχανές αναζήτησης(search metamachines). Μια λεπτομερή έρευνα των διαφόρων αλγορίθμων που υπάρχουν στην αγορά σήμερα έγινε από Claudio[14]. Οι αλγόριθμοι χωρίζονται σε 3 κατηγορίες data centric, description-aware και description-centric.

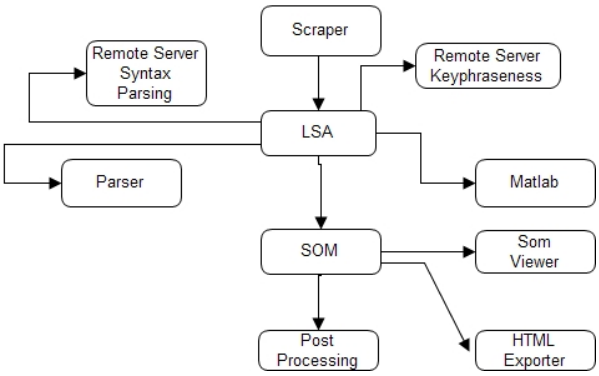
Data-centric αλγόριθμοι χειρίζονται περισσότερο αριθμητικά δεδομένα. Scatter/Gather, Lassi, WebCat, AIsearch είναι μερικοί από τους data-centric αλγόριθμους. Το label(ετικέτα) του κάθε cluster στους παραπάνω αλγόριθμους προέρχεται συνήθως καθαρά από τα βάρη του centroid. Η ονομασία αυτή είναι ανεπαρκής από τη σκοπιά των χρηστών, καθιστώντας τους αλγόριθμους αυτούς ανίκανους να ονομάσουν και να περιγράψουν το cluster με κάτι λογικό στα μάτια του χρήστη.

Description-aware αλγόριθμοι επικεντρώνονται στην κατασκευή περιγραφών που είναι ανθρώπινα ερμηνεύσιμες. Γνωστές μεταμηχανές που χρησιμοποιούν τον παραπάνω αλγόριθμο είναι Tree Clustering (STC) και SnakeT. Η αποτυχία αυτού του τύπου αλγόριθμων είναι ότι η ομαδοποίηση προηγείται της ονομασίας.

Description-centric αλγόριθμοι είναι σχεδιασμένοι να λάβουν υπόψιν τόσο την ποιότητα της ομαδοποίησης όσο και την ονομασία των cluster. Μερικές γνωστές μηχανές αυτού του τύπου είναι Lingo, SRC και DisCover, Το μειονέκτημα αυτών των αλγορίθμων είναι ότι τα clusters που δημιουργούνται βασίζονται στα αποτελέσματα της αναζήτησης τα όποια μπορεί να μην παρουσιάζουν αρκετή ποικιλία ώστε ο χρήστης να βρει το θέμα που τον ενδιαφέρει, η αντίθετος να παρουσιάζουν πολλά θεματικά ανεξάρτητα αποτελέσματα.

Η δικιά μας μεταμηχανη θα τοποθετηθεί κάπου ανάμεσα στους description-aware και description-centric αλγορίθμους. Οπως θα δούμε παρακάτω δίνουμε μεγάλο βάρος στην ομαδοποίηση, χρησιμοποιούμε ομως τα αποτελέσματα της για να συμπεράνουμε HyperClusters και αναλύουμε συντακτικά το κείμενο των snippet σε κάθε HC για να υπολογίσουμε την keyphraseness λέξεων η φράσεων που φαίνεται να έχουν σημασία σε ανθρώπινους όρους και να παρουσιάσουμε τελικά μερικές λέξεις και φράσεις που φαίνεται να χαρακτηρίζουν το cluster.

3 Επισκόπηση συστήματος



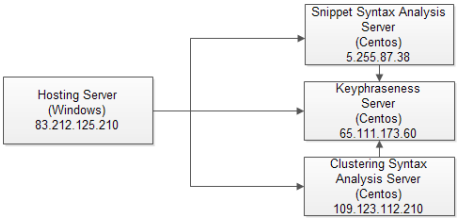
Παραπάνω βλέπουμε ένα dataflow διάγραμμα του συστήματος. Το αρχικό κομμάτι είναι ο Scraper που συλλέγει τα αποτελέσματα της αναζήτησης από την goolge. Για να αποφύγουμε τους περιορισμούς του API αποφασίσαμε αυτό το κομμάτι να υλοποιηθεί σαν Scraper, που μιμείται της αιτήσεις ενός κανονικού Chrome Browser, και όχι σαν απλός Parser. Σε αυτό το σημείο για να αποφύγουμε τυχόν προβλήματα πρόσβασης στο μέλλον έχουμε την δυνατότητα να επιλέξουμε να στείλουμε τις αιτήσεις μέσο διαφόρων proxy.

Τα αποτελέσματα μετά περνάνε στο επόμενο κομμάτι του συστήματος που ξεκινάει την διαδικασία του υπολογισμού της LSA και ένα αντίγραφο των snippet θα σταλεί σε έναν syntax analysis server(1) που ξεκινάει μια ανάλυση του κείμενου για την εξόρυξη των Noun Phrases και Named Entities. Σε αυτό το κομμάτι ανάλογα με τις επιλογές που έχει κάνει ο χρήστης ο αλγόριθμος μπορεί να τρέξει τοπικά η να χρειαστεί εισόδους από έναν syntax analysis server(2). Η σύνδεση με τον Remote Keyphraseness Server γίνεται για την ανάκτηση της keyphraseness[4] που υπολογίζετε με χρήση της Wikipedia.

Keyphraseness είναι η πιθανότητα μιας λέξης W να είναι λέξη κλειδί σε ένα έγγραφο και υπολογίζεται από την διαίρεση του αριθμού των εγγράφων στα οποία η λέξη ήταν κλειδί με τον ολικό αριθμό εγγράφων $P\left(keyword|W\right) \approx \frac{count(D_{key})}{count(D_w)}$. Η επιλογή κατάτμησης των υπολογισμών σε διαφόρους server θα εξηγηθεί παρακάτω.

Στο τέλος της LSA έχουμε αριθμητικά δεδομένα που τώρα πια μπορούμε να χρησιμοποιήσουμε στον αλγόριθμο SOM που έχουμε επιλέξει για να πάρουμε τα ομαδοποιημένα αποτελέσματα. Τέλος τα ομαδοποιημένα αποτελέσματα περνάνε από ένα στάδιο postprocessing, κατά την διάρκεια του όποιου υπολογίζονται μετρικές ομαδοποίησης και γίνετε μια περαιτέρω ομαδοποίηση των cluster σε hyperclusters. Η επιλογή ονομασίας των τελικών HC γίνεται στον HTML Exporter, μπορεί να παραλάβει τώρα τα αποτελέσματα της συντακτικής ανάλυσης που ξεκίνησε στον syntax analysis server(1) στην αρχή της LSA και να υπολογίσει την keyphraseness των NP για να επιλέξει ποιες εκφράζουν καλύτερα το αντικείμενο του cluster.

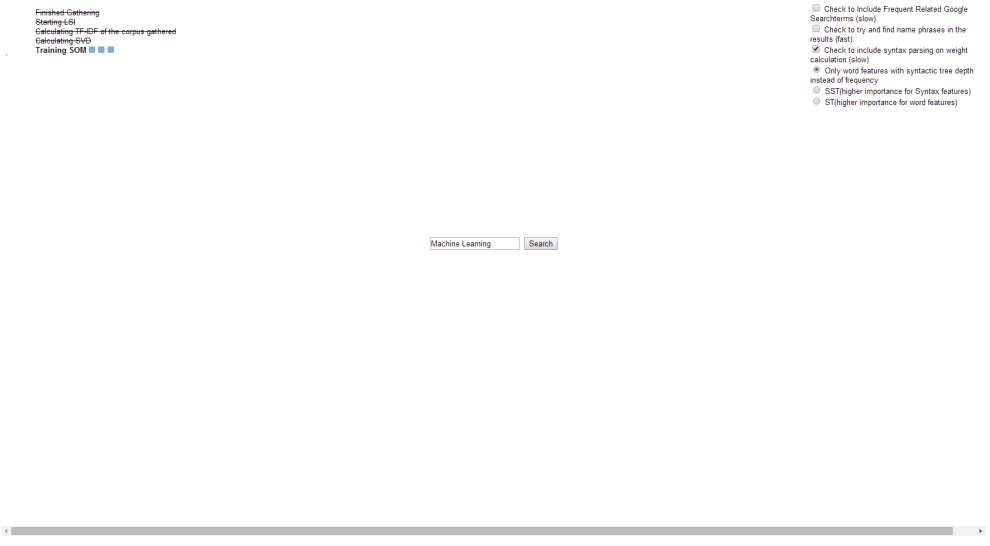
Το τελικό σύστημα μας χρησιμοποιεί 3 Server για να κρατήσουμε την σταθερότητα του κυρίου εξυπηρετητή. Τυχόν προβλήματα στους δευτερεύοντες Server θα προκαλέσουν βλάβη μόνο στην λειτουργικότητα που φιλοξενείται εκεί. Ένα script επόπτης θα ελέγχει σε τακτά χρονικά διαστήματα την κατάσταση του κάθε εξυπηρετητή και θα αλλάζει ανάλογα με τυχόν προβλήματα τις διατιθέμενες επιλογές έως ότου λυθεί το πρόβλημα.



Στην παραπάνω εικόνα μπορούμε να δούμε τους Server που αποτελούν το σύστημα μας. Ο Server που τρέχει Windows είναι ο hosting server του συστήματος, εδώ στέλνονται τα αιτήματα προς Google για ανάκτηση του σώματος αποτελεσμάτων καθώς και γίνεται ο υπολογισμός του Singular Value Decomposition με χρήση ενός Matlab standalone και τελικά η ομαδοποίηση με Growing Grid SOM. Ο παραπάνω Server είναι ο μόνος που έχει domain[1].

Οι υπόλοιποι server τρέχουν (Centos) και καλύπτουν κομμάτια του αλγόριθμου που απαιτούν είτε μεγάλο εύρος ζώνης είτε μεγάλη χρήση μνήμης.

Ο Keyphraseness[4]server στέλνει πολλές και συχνές αιτήσεις στην wikipedia και γιαυτό τον λόγο υποθέτουμε ότι οπό πραγματικές συνθήκες χρήσης (πολλοί χρήστες) θα χρειάζεται μεγάλο εύρος ζώνης καθώς κάθε αίτηση γίνεται παράλληλα. Οι υπόλοιποι δυο Centos servers χρησιμοποιούν έτοιμες βιβλιοθήκες σε java και groovy, καθώς και τον stanford parser, αυτές οι βιβλιοθήκες χρειάζονται πολλή μνήμη για να τρέξουν σωστά. Η υλοπόιηση του παραπάνω κομματιού έγινε στην μορφή ενός service που τρέχει στον κάθε ένα από τους δυο server και ακούει σε μια συγκεκριμένη θύρα. Για κάθε αίτημα που έρχεται από τον hosting server ξεκινάει ένα καινούργιο νήμα με τις βιβλιοθήκες προφορτωμένες, αποφεύγοντας έτσι την ανάγκη να φορτωθούν για κάθε αίτημα.



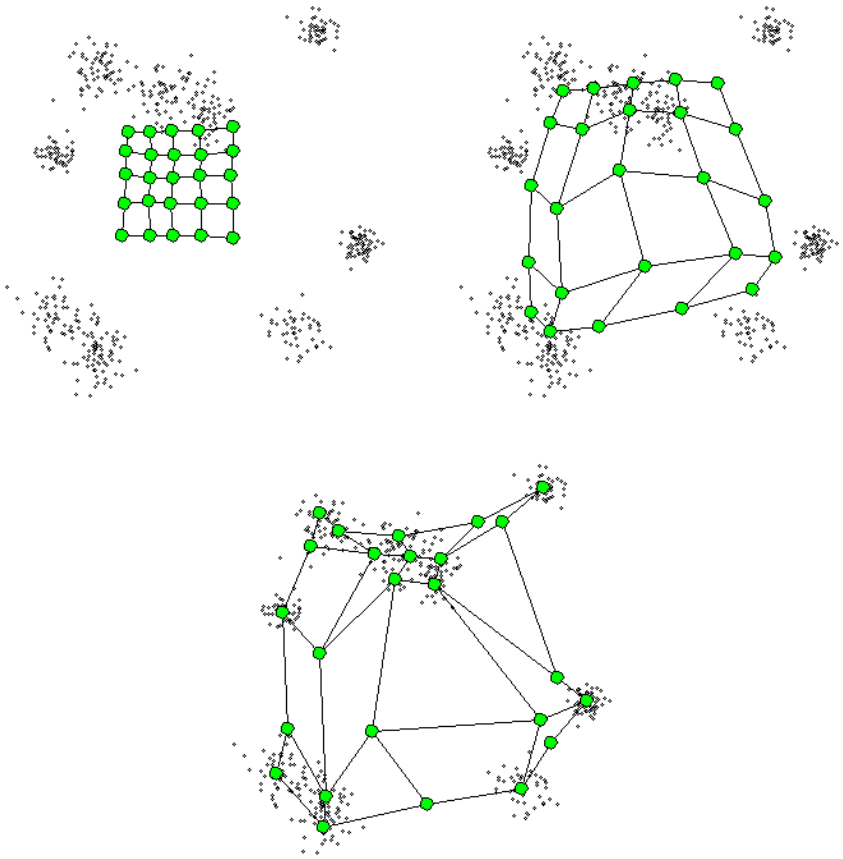
[1] http://latentsnippet.serveftp.com

4 Ομαδοποίηση

Ζούμε σε έναν κόσμο γεμάτο δεδομένα. Κάθε μέρα, οι άνθρωποι ασχολούνται με διαφορετικούς τύπους δεδομένων που προέρχονται από όλους τους τύπους των μετρήσεων και παρατηρήσεων. Μια από τις πιο σημαντικές δραστηριότητες ανάλυσης είναι η ταξινόμηση των δεδομένων σε ένα σύνολο κατηγοριών ή ομάδων. Τα δεδομένα που είναι στην ίδια ομάδα θα πρέπει να εμφανίζουν παρόμοιες ιδιότητες με βάση κάποιων κριτηρίων και χαρακτηριστικών.

4.1 Δίκτυα SOM

Για την ομαδοποίηση έχουν αναπτυχθεί μερικοί αλγόριθμοι εκ των οποίων αποφασίσαμε να χρησιμοποιήσουμε των αλγόριθμο αυτό-οργανουμενων χαρτών SOM για να κρατήσουμε την τοπολογία των ομάδων.



Οι αυτό-οργανουμενοι χάρτες (SOM), κοινώς γνωστοι και ως δίκτυα Kohonen (Kohonen 1982, Kohonen 2001) είναι μια υπολογιστική μέθοδος για την ποσοτικοποίηση και την ανάλυση πολυδιάστατων δεδομένων. Οι αυτό-οργανουμενοι χάρτες ορίζουν μια διατεταγμένη χαρτογράφηση, ένα είδος προβολής από ένα σύνολο δεδομένων σε ένα συνήθως δισδιάστατο πλέγμα οπου ενα centroid m_i σχετίζεται με κάθε κόμβο του δικτύου.

Τα centroid αυτά υπολογίζονται από τον SOM αλγόριθμο. Ένα στοιχείο δεδομένων θα πρέπει να αντιστοιχείται με τον κόμβο του οποίου το centroid είναι το πιο παρόμοιο , για παράδειγμα, έχει τη μικρότερη απόσταση από το στοιχείο δεδομένων σε ορισμένες μετρικές.

Το centroid είναι συνήθως ένας σταθμισμένος τοπικός μέσος όρος των συγκεκριμένων δεδομένων στο χώρο όλων των δεδομένων. Επί πλέον, όταν τα centroids υπολογίζονται από το SOM αλγόριθμο οι κόμβοι μοιάζουν περισσότερο στους κοντινότερους κόμβους από ότι στους κόμβους που βρίσκονται πιο μακριά. Με τον τρόπο αυτό το σύνολο των μοντέλων μπορεί να θεωρηθεί ότι αποτελεί μια γραφική παράσταση ομοιότητας των δεδομένων.

Ο αλγόριθμος SOM αναπτύχθηκε αρχικά για την ποσοτικοποίηση των κατανομών διανυσμάτων, όπως διατεταγμένα σύνολα μετρήσεων ή στατιστικά χαρακτηριστικά, αλλά μπορεί να αποδειχθεί ότι ένα SOM μπορεί να οριστεί για οποιαδήποτε στοιχεία δεδομένων τα οποία μπορούν να παραστούν σαν διανύσματα σε κάποιο πολυδιάστατο χώρο. Στην παρούσα εργασία χρειάστηκε να παραστούν συμβολοσειρες σαν διανύσματα σε πολυδιάστατο χώρο.

Ο SOM αλγόριθμος αναπτύχθηκε από μοντέλα νευρωνικών δικτύων, συγκεκριμένα τα μοντέλα της συσχετικής μνήμης και προσαρμοστικής μάθησης (βλ. Kohonen 1984). Ένα κίνητρο ήταν να εξηγήσει την χωρική οργάνωση των λειτουργιών του εγκεφάλου, όπως παρατηρείται ιδιαίτερα στον εγκεφαλικό φλοιό. Παρ όλα αυτά το SOM δεν ήταν το πρώτο βήμα προς αυτή την κατεύθυνση (βλ. von der Malsburg 1973 και Amari 1980). Ωστόσο, η αυτό-οργάνωτικη δύναμη αυτών των πρώτων μοντέλων ήταν μάλλον ανεπαρκής. Η κρίσιμη εφεύρεση του Kohonen ήταν να εισαγάγει ένα πρότυπο σύστημα που αποτελείται από τουλάχιστον δύο αλληλεπιδρώντα υποσυστήματα διαφορετικής φύσης. Ένα από τα υποσυστήματα αυτά είναι ένα ανταγωνιστικό νευρωνικό δίκτυο που υλοποιεί το winner-take-all, το άλλο υποσύστημα ελέγχεται από το νευρωνικό δίκτυο και τροποποιεί την τοπική συνοπτική πλαστικότητα των νευρώνων στη μάθηση. Η μάθηση περιορίζεται χωρικά στην τοπική γειτονιά από τους πιο δραστήριους νευρώνες. Η πλαστικότητα του υποσυστήματος ελέγχου θα μπορούσε να βασίζεται σε μη ειδικές αλληλεπιδράσεις νευρώνων , αλλά το πιο πιθανόν είναι ναι είναι αποτέλεσμα χημικού ελέγχου.

Παρ όλα αυτά, η αρχή του SOM μπορεί επίσης να εκφραστεί μαθηματικά σε μια καθαρή αφηρημένη μορφή, χωρίς αναφορά σε οποιαδήποτε νευρονική εξάρτηση. Ο πρώτος τομέας εφαρμογής των SOM ήταν αυτός της αναγνώρισης ομιλίας (βλ. σχήμα. 2). Στην αφηρημένη του μορφή, το SOM έχει τεθεί σε ευρεία χρήση στην ανάλυση δεδομένων (Kaski et al. 1998, Oja et al. 2003, Pöllä et al. 2007).

4.2 Αλγόριθμος εκμάθησης

Για την εκμάθηση θα χρησιμοποιηθούν συντελεστές στάθμισης των νευρώνων που θα αναφέρονται ως βάρη. Τα βάρη των νευρώνων αρχικοποιούνται είτε σε μικρές τυχαίες τιμές ή, για γρηγορότερη σύγκληση του αλγορίθμου σε τιμές ομοιόμορφα κατανεμημένες στον υπόχωρο που χαρακτηρίζεται από τα δύο μεγαλύτερα κύρια ιδιοδιανύσματα του σώματος δεδομένων. Με την τελευταία αυτή εναλλακτική λύση, η μάθηση είναι ταχύτερη, επειδή τα αρχικά βάρη έχουν ήδη δώσει καλή προσέγγιση των βαρών SOM.

Το δίκτυο θα πρέπει να λάβει μεγάλο αριθμό διανυσμάτων που εκπροσωπούν, όσο καλύτερα γίνεται, τα είδη των διανυσμάτων που αναμένεται να συναντηθούν κατά τη διάρκεια της χρήσης. Τα παραδείγματα χορηγούνται συνήθως αρκετές φορές.

Η εκπαίδευση χρησιμοποιεί ανταγωνιστική μάθηση. Όταν ένα παράδειγμα εκπαίδευσης τροφοδοτείται στο δίκτυο, υπολογίζεται η Ευκλείδεια απόσταση του με όλα τα διανύσματα βαρών. Ο νευρώνας με διάνυσμα βάρους πιο παρόμοιο με την είσοδο είναι ο νικητής (BMU). Τα βάρη των BMU και των νευρώνων που είναι κοντά τους στο πλέγμα SOM προσαρμόζονται προς το διάνυσμα εισόδου. Το μέγεθος της μεταβολής μειώνεται με το χρόνο και με απόσταση από το BMU. Ο τύπος ανανέωσης για ένα νευρώνα με διάνυσμα βαρών Wv (t) είναι

$Wv(t + 1) = Wv(t) + \Theta(v, t)\, \alpha(t)(D(t) - Wv(t))$,

όπου α (t) είναι μια μονότονα φθίνουσα συνάρτηση που δίνει τον συντελεστή μάθησης και D (t) είναι το διάνυσμα εισόδου. Η συνάρτηση γειτονιάς Θ(v, t) εξαρτάται από την πλεγματική απόσταση μεταξύ των BMU και του νευρώνα v. Στην απλούστερη μορφή είναι 1 για όλους τους νευρώνες αρκετά κοντά στον BMU και 0 για τους άλλους, αλλά μια Γκαουσιανη συνάρτηση είναι κοινή επιλογή υλοποίησης. Ανεξάρτητα από την μορφή της γειτονιάς, αυτη συρρικνώνεται με το χρόνο. (Simon Haykin 1999) Στην αρχή, όταν η περιοχή είναι ευρεία, η αυτό-οργάνωση λαμβάνει χώρα σε όλο το δίκτυο. Όταν η γειτονιά έχει συρρικνωθεί σε μόλις δύο νευρώνες τα βάρη συγκλίνουν στις τοπικές εκτιμήσεις.

Αυτή η διαδικασία επαναλαμβάνεται για κάθε διάνυσμα εισόδου για ένα (συνήθως μεγάλο) αριθμό κύκλων λ. Το δίκτυο σαν αποτέλεσμα συνδέει τους κόμβους εξόδου με ομάδες ή μοτίβα στα δεδομένα εισόδου.

Κατά τη διάρκεια της χαρτογράφησης, θα υπάρχει ένας και μόνος νευρώνας νικητής: ο νευρώνας του οποίου το διάνυσμα βαρών βρίσκεται πιο κοντά στο διάνυσμα εισόδου. Αυτό μπορεί να καθορίζεται απλώς από τον υπολογισμό της ευκλείδειας απόστασης μεταξύ διανύσματος εισόδου και διανύσματος βαρών του νευρώνα.

Ενώ η αναπαράσταση δεδομένων εισόδου ως διανύσματα έχει τονιστεί, θα πρέπει να σημειωθεί ότι κάθε είδους αντικείμενου που μπορεί να αναπαρασταθεί ψηφιακά με κατάλληλο μέτρο απόστασης, μπορεί να χρησιμοποιηθεί για να κατασκευαστέι ένας αυτό-οργάνουμενος χάρτης. Αυτα περιλαμβάνουν πίνακες, συνεχείς λειτουργίες ή ακόμα και άλλα SOM.

Παρακάτω θα εξηγήσουμε έναν βελτιωμένο αλγόριθμο SOM ονομαζόμενο Growing grid. Η επιλογή του συγκεκριμένου μοντέλου αντί του GHSOM(growing hierarchical SOM) έγινε για την δυνατότητα περισσοτέρων προτιμήσεων επεξεργασίας του τελικού SOM όπως και για την εφαρμογή μέτρων οργάνωσης και ποιότητας που χρησιμοποιούνται σε ανάλογες εργασίες.

4.3 Growing grid

Η μεταφορά των δεδομένων από πολυδιάστατο χώρο σε δις ή τρισδιάστατο πλέγμα κάνει τις διασυνδέσεις ανάμεσα στα σημεία δεδομένων αισθητή και παρέχει μια καλύτερη εικόνα της δομή των δεδομένων και της τάσης ομαδοποίησης. Η δυνατότητα αυτή έκανε το SOM ένα σημαντικό εργαλείο σε ένα ευρύ φάσμα εφαρμογών, όπως η εξόρυξη δεδομένων, ή και γενικότερα η αναγνώριση μοτίβων. Ωστόσο, ορισμένες δυσκολίες στην αξιοποίηση SOM παρέμειναν σε μεγάλο βαθμό ανέπαφες, ακόμη και αν υπάρχει ένας μεγάλος αριθμός ερευνητικών εργασιών επί των χρήσεων των SOM. Πρώτον, το SOM χρησιμοποιεί ένα δίκτυο σταθερής αρχιτεκτονικής όσον αφορά τον αριθμό και τη διάταξη των νευρώνων, η οποία πρέπει να έχει καθοριστεί πριν από την εκπαίδευση. Στις περισσότερες περιπτώσεις, ο χρήστης δεν έχει πολλές γνώσεις των εγγενών δομών δεδομένων. Έτσι είναι δύσκολο να προκαθοριστεί η κατάλληλη διάσταση του δικτύου. Θα ήτανε παράλογο να περιμένουμε

τον χρήστη να μπορεί να επιλέξει μόνος του το σωστό μέγεθος δικτύου για κάθε query προς την google, αφού εξαρτάτε από το μέγεθος και αλλά χαρακτηριστικά του σώματος δεδομένων που δίνεται ως απάντηση. Αυτό συχνά οδηγεί σε σημαντικό περιορισμό πιθανών εφαρμογών [1] . Είναι πολύ πιθανό ότι ένα προκαθορισμένο μέγεθος του δικτύου είναι είτε πολύ μικρό ή πολύ μεγάλο. Σε κάθε περίπτωση, το αποτέλεσμα θα είναι κακής ποιότητας. Έτσι σίγουρα αξίζει να εξεταστούν μοντέλα νευρωνικού δικτύου που καθορίζουν τον αριθμό και τη διάταξη των μονάδων κατά τη διάρκεια της εκπαίδευσης. Ανατρέξτε στο [8] για μοντέλα που προτάθηκαν και επιτρέπουν την προσαρμογή της αρχιτεκτονικής του δικτύου κατά τη διάρκεια της εκπαίδευσης. Ένα άλλο μειονέκτημα του σταθερού δικτύου είναι ότι αν και τα διανύσματα δεδομένων αντιστοιχούν στους καλύτερα ταιριάζοντες νευρώνες, είναι συνήθως δύσκολο να δοθούν πολλές πληροφορίες σχετικά με την ολική κατανομή των στοιχείων με μόνο την παρατήρηση των δισδιάστατων αποτελεσμάτων. Για τους παραπάνω λόγους αποφασίσαμε να χρησιμοποιήσουμε ένα SOM με αυξανόμενο δίκτυο βασισμένο στην κατανομή πιθανοτήτων όπως προτείνεται από Bernd Fritzke[8].

4.4 Αρχιτεκτονική δικτύου

Το δίκτυο θεωρούμε ότι αποτελείται από ένα ορθογώνιο k x m πλέγμα και A μονάδες(clusters):

A = $[\alpha_{ij}]$, l ≤i ≤k, l ≤j ≤m (1)

Υποθέτουμε πιθανότητα κατανομής P (ξ) των n-διάστατων διανυσμάτων δεδομένων ξ. Μια μεταβλητή t_c σχετίζεται με κάθε μονάδα και ορίζεται αρχικά σε 0. Οι μεταβλητές αυτές χρησιμοποιούνται για τη συλλογή στατιστικών πληροφοριών για να αποφασιστεί πού να γίνει η εισαγωγή νέας γραμμής ή στήλης (βλ. παρακάτω).

4.4.1 Προσαρμογή κέντρου

Οι νευρώνες νικητές προσαρμόζονται για κάθε διάνυσμα εισόδου όπως στον κανονικό αλγόριθμο Kohonen.

Για κάθε διάνυσμα εισόδου ξ προσδιορίζεται ο νευρώνας s με το περισσότερο παρόμοιο διάνυσμα βαρών :

$\|w_s - \xi\| < \|w_c - \xi\|$ $(\forall c \epsilon A)$(2)

Στη συνέχεια, το διάνυσμα S και οι γειτονικοί νευρώνες προσαρμόζονται προς το διάνυσμα εισόδου ξ. Το μέγεθος προσαρμογής της μονάδας(νευρώνας) εξαρτάται από την απόσταση στο πλέγμα από τον νευρώνα νικητή s όπως εξηγείται παρακάτω.

Για δύο μονάδες C1,C2 ϵ A με Cl = α_{i1j1} και C2 = α_{i2j2} ορίζουμε το ακόλουθο μέτρο απόστασης:

$d(c_1, c_2) = \|i_1 - i_2\| + \|j_1 - j_2\|$ (3)

το οποίο είναι επίσης γνωστό ως μέτρο City block η L1 νόρμα. Αν για ένα σήμα εισόδου ξ, ο νευρώνας s είναι ο νικητής τότε τα διανύσματα βαρών των γειτονικών νευρώνων του δικτύου προσαρμόζεται ανάλογα με

$\triangle w_c = \varepsilon_0 \exp\left(-\frac{d^2(c,s)}{2\sigma^2}\right)(\xi - w_c)(\forall c \epsilon A)$ (4)

Με αυτόν τον τρόπο έχουμε ένα σταθερό ρυθμό μάθησης ε_0 και ο εκθετικός όρος αντιπροσωπεύει Gaussian γειτονιά με σταθερή παράμετρο πλάτους σ. Σε κάθε βήμα της προσαρμογής η μεταβλητή που είχαμε ορίσει 0 ανανεώνεται

$\tau_s = \tau_s + 1$ (5)

και έτσι οι τιμές αυτές δείχνουν πόσο συχνά ο νευρώνας είναι νευρώνας νικητής.

4.4.2 Εισαγωγή των νέων γραμμών και στηλών

Για ένα δίκτυο μεγέθους k x m κάνουμε k x m x lg βήματα προσαρμογής. Έτσι, η παράμετρος lg δείχνει πόσα βήματα προσαρμογής γίνονται κατά μέσο όρο ανά νευρώνα . Αφού k x m x lg βήματα προσαρμογής έχουν διενεργηθεί, καθορίζουμε τον νευρώνα q με τη μέγιστη τιμή:

$\tau_q \geq \tau_c (\forall c \epsilon A) (6)$

Ο νευρώνας αυτός υπήρξε νευρώνας νικητής πιο συχνά και, προκειμένου να διανέμει τα σήματα πιο ομοιόμορφα πάνω από όλες τις γειτονικές νευρώνες, είναι λογικό να εισαγάγουμε μια νέα γραμμή ή στήλη δίπλα του. Δεδομένου ότι υπάρχουν αρκετές δυνατότητες για το πώς να το κάνουμε αυτό πρέπει να επιλέξουμε μονο μία. Αρχικά πρέπει να αναγνωριστεί ο γείτονας f του q με το πιο διαφορετικό διάνυσμα βαρών και να εισαχθεί μια νέα γραμμή (ή στήλη) μεταξύ q και f. Το σκεπτικό πίσω από αυτή την επιλογή είναι ότι κατά πάσα πιθανότητα ο f υποδεικνύει μια κατεύθυνση με μεγάλη διακύμανση στα δεδομένα. Το παραπάνω περιγράφεται ως εξής:

Συμβολίζουμε με N_q το σύνολο των (μέχρι 4) άμεσων γειτόνων του q:

$N_q = \{c \epsilon A | d(q, c) = 1\}$. (7)

Ο γειτονικός νευρώνας f με το πιο διαφορετικό διάνυσμα βαρών υπολογίζεται από:

$\|w_f - w_q\| < \|w_c - w_q\| \ (\forall c \epsilon N_q)$. (8)

Χωρίς απώλεια της γενικότητας υποθέτουμε ότι q και f είναι γειτονικές μονάδες, με $q = a_{ij}$ και $f = a_{ij+1}$. Η περίπτωση το q και f να μοιράζονται μια στήλη μπορεί να αντιμετωπιστεί με πλήρη αναλογία. Εισαγάγουμε μια νέα στήλη j' (με k μονάδες) μεταξύ των στηλών j και j + 1(βλ. σχήμα. 3). Τα νέα διανύσματα βαρών βρίσκονται με παρεμβολή από τους γείτονές τους, πράγμα που αυξάνει (όπως προβλέπεται) την πυκνότητα των μονάδων στην περιοχή γύρω από το w_q:

$w_{rj} = 0.5(w_{rj} + w_{rj+1}) \ (1 \leq r \leq k)$. (9)

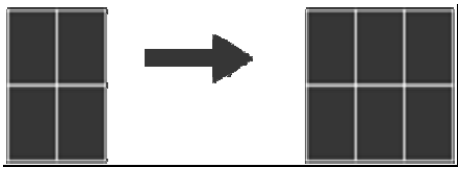
Από το παραπάνω ο αριθμός m των στηλών μεγαλώνει:

m = m + l. (10)

και όλες οι μεταβλητές τ επαναφέρονται στο 0

$\tau_c = 0 (\forall c \epsilon A) (11)$

και επαναλαμβάνουμε στον επόμενο γύρο των προσαρμογών εκτός εάν η το κριτήριο διακοπής πληρείται (βλ. παρακάτω).



4.4.3 Κριτήρια διακοπής

Για να αποφύγουμε την περίπτωση το δικτύου να αυξάνεται επ 'αόριστον πρέπει να καθοριστεί ένα κριτήριο διακοπής. Στην απλούστερη περίπτωση μπορούμε να καθορίσουμε το μέγιστο αριθμό των επιτρεπόμενων μονάδων για να σταματήσει η διαδικασία, όταν αυτός ο αριθμός επιτευχθεί ή ξεπεραστεί δεν θα επιτρέπονται καινούργιες στήλες. Είναι, ωστόσο, επίσης δυνατό να καθορίσουμε ένα έμμεσο

κριτήριο. Λόγω των ισχυρών περιορισμών της τοπολογίας δεν μπορούμε να αναμένουμε ότι κάθε μονάδα θα λαμβάνει ένα ίσο αριθμο διανύσματων εισόδου (δηλ. I / (m x k)). Ως εκ τούτου, θα μπορούσε κανείς να συνεχίσει την διαδικασία έως ότου το μερίδιο των διανυσμάτων εισόδου πέσει κάτω από ένα κατώφλι για κάθε μονάδα του δικτύου. Άλλα έμμεσα κριτήρια είναι δυνατά. Κάποιος πρέπει, ωστόσο, να προσέξει ότι το επιλεγέν κριτήριο θα εκπληρωθεί τελικά (ή θα μπορούσε να συνδυαστεί με ένα μέγιστο επιτρεπόμενο αριθμό μονάδων).

4.4.4 Ακριβής ρύθμιση της θέσης διανύσματος

Ο σταθερός ρυθμός προσαρμογής εμποδίζει την σύγκλιση των διανυσμάτων στις βέλτιστες θέσεις, αυτό θα μπορούσε (θεωρητικά) να επιτευχθεί με την στοχαστική προσέγγιση. Κατά τη διάρκεια της φάσης εκπαίδευσης του δικτύου, ο ρυθμός αυτός προσαρμογής είναι ζωτικής σημασίας για να αναπροσαρμόσουμε τα βάρη μετά από τις προσθήκες. Όταν η εκπαίδευση τελειώσει είναι δυνατόν να γίνει μια πιο ακριβής ρύθμιση με την χρήση ενός φθίνοντα ρυθμού μάθησης: Θα χρειαστούμε

$\mathrm{t'}_{max} = \mathrm{x\ x\ m\ x}\lambda_{\mathrm{f}}$ (12)

βήματα προσαρμογής σύμφωνα με την εξίσωση (4) χρησιμοποιώντας τον παρακάτω χρονοεξαρτώμενο ρυθμό μάθησης

$\mathrm{e\ (t')} = \varepsilon_0\,(\varepsilon_1/\varepsilon_0)^{\mathrm{t'/t'max}}$ (t'=1$\ldots$, t'_{max}). (13)

Με αυτόν τον τρόπο t' δηλώνει το χρόνο στη φάση της ακριβής ρύθμισης, η οποία αρχίζει αφού η φάση εκπαίδευσης έχει τελειώσει. Η παράμετρος λ_{f} καθορίζει πόσα βήματα προσαρμογής χρειάζονται (κατά μέσο όρο) σε αυτή την τελική φάση.

5 Μοντέλο διανυσματικού χώρου

Οπως είναι προφανές από τα παραπάνω ότι για να επεξεργαστούμε τα snippets που είναι μικρά κομμάτια κειμένου με οποιαδήποτε μαθηματική τεχνική από τις παραπάνω πρέπει να τα μεταφέρουμε σε κάποιον πολυδιάστατο διανυσματικό χώρο. Ο τρόπος με τον οποίο το καταφέρνουμε αυτό είναι με την χρήση του vector space model και πριν αρχίσουμε να υπολογίζουμε τις συχνότητες των λέξεων για το VSM θα πρέπει να βρούμε κάποιον τρόπο να αναγνωρίσουμε την σχέση μεταξύ διαφορετικών μορφών της ιδίας λέξης(εκπαιδευτικαι, εκπαιδεύουμε, εκπαίδευση).

Η τεχνική που το επιτυγχάνει αυτό λέγετε Stemming και θα περιγραφεί παρακάτω.

5.1 Stemming

Στη γλωσσική μορφολογία και ανάκτηση πληροφοριών , stemming είναι η διαδικασία για τη μείωση λέξεων στο βασικό τους στέλεχος , ή ριζική μορφή. Το βασικό στέλεχος δεν χρειάζεται να είναι ταυτόσημο με την μορφολογική ρίζα της λέξης. Είναι συνήθως αρκετό σχετικές λέξεις να έχουν το ίδιο βασικό στέλεχος, ακόμη και αν αυτό το βασικό στέλεχος δεν είναι από μόνο του μια έγκυρη ρίζα. Πολλές μηχανές αναζήτησης μεταχειρίζονται λέξεις με το ίδιο βασικό στέλεχος ως συνώνυμα , ως ένα είδος διεύρυνσης ερωτήματος , μια διαδικασία που ονομάζεται "conflation". Υπάρχουν πολλοί αλγόριθμοι για τον υπολογισμό του stem, και αποκαλούνται stemmers.

Ένας αλγόριθμος για την αγγλική γλώσσα , για παράδειγμα, θα πρέπει να προσδιορίσει ότι η συμβολοσειρα "cats" (και, ενδεχομένως, "catlike", "catty" κ.λπ.), έχει βάση τη ρίζα "cat". Ένας αλγόριθμος stemmer μειώνει τις λέξεις "fishing", "fished", "fish" και "fisher" στην ρίζα, "fish". Στην παρούσα εργασέια για λόγους απλότητας, θα γίνει χρήση του Porter stemmer[3].

5.2 Vector space model

Το VSM είναι ένα αλγεβρικό μοντέλο για την αναπαράσταση έγγραφων κειμένου (και κάθε αντικειμένου) ως διανύσματα. Έχει χρησιμοποιηθεί στο φιλτράρισμα των πληροφοριών, ανάκτηση πληροφοριών, ταξινόμηση και κατάταξη σύμφωνα με την σχετικότητα. Η πρώτη χρήση του ήταν στο SMART (σύστημα αναζήτησης πληροφοριών).

Τα έγγραφα και τα ερωτήματα εκπροσωπούνται από διανύσματα.

dj = (w1,j,w2,j,...,wt,j)

q = (w1,q,w2,q,...,wt,q)

Κάθε διάσταση αντιστοιχεί σε ένα ξεχωριστό token. Εάν ένα token εμφανίζεται στο έγγραφο, η αξία του στο διάνυσμα είναι μη μηδενική. Υπάρχουν αρκετοί διαφορετικοί τρόποι για των υπολογισμό αυτών των αξιών, γνωστών και ως term weights. Ένα από τα πιο γνωστά συστήματα είναι το tf-idf.

Ο ορισμός του token εξαρτάται από την εφαρμογή. Συνήθως οι όροι είναι μεμονωμένες λέξεις, λέξεις-κλειδιά, φράσεις ή και περισσότερα. Στην δικιά μας εφαρμογή η λέξη token χρησιμοποιείται για να χαρακτηρίσει την ριζα των λέξεων κλειδιών που λαμβάνουμε αφού τελειώσει το στάδιο του stemming. Η διάσταση του κάθε διανύσματος εγγράφου είναι ο αριθμός των λέξεων κλειδιών που παρουσιάζονται όχι μόνο σε αυτό το έγγραφο αλλά σε ολόκληρο το σώμα δεδομένων.

Το παραπάνω μοντέλο μπορεί να χρησιμοποιηθεί για να συγκρίνουν τα έγγραφα με τα ερωτήματα, αντιμετοπίζωντας τα ερωτήματα σαν ένα οποιοδήποτε

άλλο έγγραφο και υπολογίζοντας το VSM τους. Στην συνέχεια θα μπορούσαμε να υπολογίσουμε ένα κριτήριο συνάφειας γνωστό ως cosine similarity measure.

5.3 Cosine Measure

Το cosine measure είναι ένα μέτρο ομοιότητας μεταξύ δύο διανυσμάτων με τη μέτρηση του συνημίτονου της γωνίας μεταξύ τους. Το συνημίτονο 0^o είναι 1, και λιγότερο από 1 για οποιαδήποτε άλλη γωνία. Το συνημίτονο της γωνίας μεταξύ δύο διανυσμάτων καθορίζει έτσι αν δύο διανύσματα δείχνουν, σε γενικές γραμμές, την ίδια κατεύθυνση.

Αυτό χρησιμοποιείται συχνά για να συγκριθούν τα έγγραφα στον τομέα της εξόρυξης δεδομένων. [10].

5.3.1 Υπολογισμός cosine

Το συνημίτονο των δύο διανυσμάτων προκύπτει εύκολα, χρησιμοποιώντας τον τύπο του εσωτερικού γινομένου:

$\alpha \cdot \beta = \|\alpha\| \, \|\beta\| \cos\vartheta$

Δεδομένων δύο διανυσμάτων A και B, με γωνία ϑ, η ομοιότητα θα υπολογίζεται από τον παρακάτω τύπο:

$\text{similarity} = \cos\vartheta = \frac{\text{A B}}{\|\text{A}\|\|\text{B}\|} = \frac{\Sigma \text{ AixBi}}{\sqrt{\Sigma \text{Ai}\hat{}2}\sqrt{\text{Bi}\hat{}2}}$

Το αποτέλεσμα κυμαίνεται από 0 -1 με αντίστροφη συσχέτιση, το 1 σημαίνει ακριβώς ίδια, και 0 συνήθως να υποδεικνύει την ανεξαρτησία.

Για το θέμα μας τα διανύσματα A και B είναι τα διανύσματα με τις συχνότητες λέξεων(tokens) που παρουσιάζουν τα έγγραφα. Η ομοιότητα μπορεί να θεωρηθεί ως μια μέθοδος για την κοινωνικοποίηση μήκους έγγραφου κατά τη διάρκεια της σύγκρισης.

Στην περίπτωση μας, η ομοιότητα των δύο εγγράφων θα κυμαίνεται από 0 έως 1, δεδομένου ότι ο όρος συχνότητας (TF-IDF παρακάτω) δεν μπορεί να είναι αρνητικός. Η γωνία μεταξύ δύο διανυσμάτων συχνότητας δεν μπορεί να είναι μεγαλύτερη από 90 °.

6 TF-IDF

Για βελτίωση των αποτελεσμάτων θα πρέπει να γίνει μια κανωνικοποίηση των βαρών των διαφόρων λέξεων λαμβάνοντας υπόψιν τα βάρη που αυτή η λέξη παρουσιάζει σε ολόκληρο το σώμα δεδομένων.

Το TF-IDF βάρος (term frequency–inverse document frequency) είναι ένα βάρος που χρησιμοποιείται συχνά στα information retrival και text mining. Το βάρος αυτό είναι ένα στατιστικό μέτρο που χρησιμοποιείται για να αξιολογήσει πόσο σημαντική είναι μια λέξη σε ένα έγγραφο σε μια συλλογή ή σώμα δεδομένων . Η σημασία αυξάνει αναλογικά με τον αριθμό των εμφανίσεων μιας λέξης στο έγγραφο, αλλά αντισταθμίζεται από τη συχνότητα της λέξης σε όλο το σώμα εγγράφων. Παραλλαγές του tf-idf χρησιμοποιούνται συχνά από τις μηχανές αναζήτησης ως ένα κεντρικό εργαλείο για την βαθμολόγηση και κατάταξη ενός εγγράφου δεδομένου ενός ερωτήματος από τον χρήστη. Tf-idf μπορεί να χρησιμοποιηθεί επιτυχώς για φιλτράρισμα λέξεων κλειδιών(stopwords) σε διάφορα θεματικά πεδία, όπως περιλήψεις [13] .

Μια από τις απλούστερες λειτουργίες κατάταξης υπολογίζεται από την άθροιση των tf-idf για κάθε token της ερώτησης, πολλές περισσότερο προηγμένες λειτουργίες κατάταξης είναι παραλλαγές αυτού το απλού μοντέλου.

6.1 Ιδέα

Ας υποθέσουμε ότι έχουμε σε ένα σύνολο αγγλικών εγγράφων κειμένου και επιθυμούμε να προσδιορίσουμε το έγγραφο που είναι πιο σχετικό με το ερώτημα "the brown cow". Ένας απλός τρόπος για να ξεκινήσουμε είναι με την εξάλειψη έγγραφων που δεν περιέχουν όλες τις τρείς λέξεις "the", "brown" και "cow", αλλά αυτό εξακολουθεί να αφήνει πολλά έγγραφα. Στο επόμενο στάδιο υπολογίζουμε για κάθε έγγραφο πόσες φορές εμφανίζετε η κάθε λέξη, όρος συχνότητάς της, και πόσες φορές εμφανίζετε σε ολόκληρο το σώμα. Ωστόσο, επειδή ο όρος "the" είναι τόσο κοινός, αυτός θα τείνει να τονίσει εσφαλμένα τα έγγραφα που τυχαίνει να χρησιμοποιούν τη λέξη "the" πιο συχνά, χωρίς να δώσει αρκετό βάρος στους ουσιαστικούς όρους “brown” και “cow”. Επίσης, ο όρος "the" δεν είναι μια καλή λέξη-κλειδί για να βοηθήσει τον διαχωρισμό σχετικών και μη σχετικών έγγραφων. Αντίθετα, οι λέξεις “brown” και “cow” που συμβαίνουν σπάνια είναι καλές λέξεις-κλειδιά για να διακρίνουν τα σχετικά έγγραφα από τα μη-σχετικά έγγραφα. Ως εκ τούτου, κανονικοποιωντας τις συχνότητες εμφάνισης της λέξης σε κάθε έγγραφο ως προς της συχνότητες εμφάνισης τους σε ολόκληρο το σώμα θα έχουμε μειωση των βαρών εκείνων των λέξεων που εμφανίζονται πιο συχνά σε όλο το corpus και αύξηση εκείνων που παρουσιάζονται μόνο σε μερικά έγγραφα.

6.2 Υπολογισμός

Η συχνότητα εμφάνισης λέξεων σε κάποιο έγγραφο(d) θα λέγεται συχνότητα εγγράφου αυτής της λέξης(t) ενώ η συχνότητα εμφάνισης της σε ολόκληρο το σώμα θα λέγετε συχνότητα σώματος. Αυτή η συχνότητα συνήθως κανωνικοποιείται ως προς το μέγεθος του εγγράφου ούτως ώστε να αποφευχθεί η προκατάληψη προς μεγαλύτερα έγγραφα (τα οποία μπορεί να έχουν υψηλότερη αρίθμηση κάποιας λέξης, ανεξάρτητα από την πραγματική σημασία της στο έγγραφο). Έτσι έχουμε την συχνότητα tf (t, d), που ορίζεται στην απλούστερη περίπτωση ως ο αριθμός

εμφανίσεων ενός όρου σε ένα έγγραφο. (Πολλές παραλλαγές έχουν προταθεί: Βλ. π.χ. Manning, Raghavan και Schütze, σ. 118.)

Η αντιστροφή συχνότητα εγγράφου είναι ένα μέτρο της γενικότερης σημασίας της λέξης (προκύπτει από τη διαίρεση του συνολικού αριθμού των εγγράφων από τον αριθμό των εγγράφων που περιέχουν τον όρο, και στη συνέχεια τον λογάριθμο του πηλίκου).

idf(t)=log $\frac{\|D\|}{\|\{d:t \epsilon d\}\|}$

με

- $\|\Delta\|$: αριθμό πληθικότητας του D, ή ο συνολικός αριθμός των εγγράφων που υπάρχει στο σώμα
- $\|\{d : t \epsilon d\}\|$: Αριθμός των εγγράφων, όπου εμφανίζεται η λέξη t (δηλαδή,tf(t,d)$\neq$ 0). Εάν η λέξη δεν είναι στο σώμα, αυτό θα οδηγήσει σε διαίρεση με 0. Συνεπώς, συνηθίζεται να ρυθμίζεται ο τύπος με $1 + \|\{d : t \epsilon d\}\|$.

Τότε έχουμε: tf-idf(t,d)= tf(t,d) x idf(t).

Το υψηλό βάρος tf-idf σημαίνει ότι η λέξη έχει υψηλή συχνότητα στο έγγραφο και παρουσιάζεται μόνο σε μερικά έγγραφα από όλο το σώμα εγγράφων έτσι τα βάρη, έχουν την τάση να φιλτράρουν τους κοινούς όρους. Η tf-idf τιμή για μια λέξη θα είναι μεγαλύτερη από το μηδέν μόνο αν ο λόγος μέσα στον λογάριθμο της IDF είναι μεγαλύτερος από 1. Ανάλογα με το αν 1 προστίθεται στον παρανομαστή, μερικά από όλα τα έγγραφα θα έχουν μηδενικό ή αρνητικό IDF, και εάν το 1 προστίθεται στον παρανομαστή για μια λέξη που εμφανίζεται σε όλα εκτός από ένα έγγραφο θα έχει idf ίση με μηδέν .

6.3 Παράδειγμα

Για να καταλάβουμε όλα τα παραπάνω θα επιλύσουμε ένα απλό παράδειγμα. Έστω ότι όλο το σώμα των δεδομένων μας αποτελείται από 3 έγγραφα τα οποία είναι μια πρόταση το καθένα και δεν εφαρμόζουμε stemming ούτε βελτιστοποιήσεις που λαμβάνουν υπόψιν και την θέση της κάθε λέξης στο έγγραφο καθώς και δεν αναφέρουμε τις πολύ συχνές λέξεις την αγγλικής(stopwords). Παράδειγμα από Professors David Grossman and Ophir Frieder, from the Illinois Institute of Technology .

Το corpus είναι το παρακάτω:

D1: "Shipment of gold damaged in a fire"

D2: "Delivery of silver arrived in a silver truck"

D3: "Shipment of gold arrived in a truck"

TERM VECTOR MODEL BASED ON $w_i = tf_i * IDF_i$											
Query, Q: "gold silver truck" D_1: "Shipment of gold damaged in a fire" D_2: "Delivery of silver arrived in a silver truck" D_3: "Shipment of gold arrived in a truck" D = 3; IDF = log(D/df_i)											
		Counts, tf_i						Weights, $w_i = tf_i * IDF_i$			
Terms	**Q**	**D_1**	**D_2**	**D_3**	**df_i**	**D/df_i**	**IDF_i**	**Q**	**D_1**	**D_2**	**D_3**
a	0	1	1	1	3	3/3 = 1	0	0	0	0	0
arrived	0	0	1	1	2	3/2 = 1.5	0.1761	0	0	0.1761	0.1761
damaged	0	1	0	0	1	3/1 = 3	0.4771	0	0.4771	0	0
delivery	0	0	1	0	1	3/1 = 3	0.4771	0	0	0.4771	0
fire	0	1	0	0	1	3/1 = 3	0.4771	0	0.4771	0	0
gold	1	1	0	1	2	3/2 = 1.5	0.1761	0.1761	0.1761	0	0.1761
in	0	1	1	1	3	3/3 = 1	0	0	0	0	0
of	0	1	1	1	3	3/3 = 1	0	0	0	0	0
silver	1	0	2	0	1	3/1 = 3	0.4771	0.4771	0	0.9542	0
shipment	0	1	0	1	2	3/2 = 1.5	0.1761	0	0.1761	0	0.1761
truck	1	0	1	1	2	3/2 = 1.5	0.1761	0.1761	0	0.1761	0.1761

1. Στήλες 1 - 5: Αρχικά κατασκευάσουμε ένα ευρετήριο λέξεων από τα έγγραφα υπολογίζοντας της συχνότητες τους (tfi) για το ερώτημα και κάθε dj έγγραφο.

2. Στήλες 6 - 8: Υπολογίζουμε το συχνότητες έγγραφου di για κάθε ένα. Από IDFi = log (D / dfi) όπου D = 3.

3. Στήλες 9 έως 12: Παίρνουμε το TF * IDF και συνυπολογίζουμε τα τελικά βάρη. Αυτές οι στήλες μπορεί να θεωρηθούν ως μια αραιή μήτρα στην οποία οι περισσότερες τιμές είναι μηδέν.

Το αποτέλεσμα φαίνεται στην εικόνα 1. Παρακάτω θα μπορούσαμε να χρησιμοποιήσουμε ένα μέτρο ομοιότητας όπως το cosine ανάμεσα στο διάνυσμα Q και κάθε έγγραφο για να αποφανθούμε ως προς την κατάταξη τους.

7 LSA

Latent semantic indexing η σημασιολογική ανάλυση (LSA) είναι μια τεχνική στην επεξεργασία φυσικής γλώσσας , για την ανάλυση των σχέσεων ανάμεσα σε μια σειρά εγγράφων καθώς και τις λέξεις που περιέχουν με την παραγωγή ενός σύνολου εννοιών που σχετίζονται με τα έγγραφα και τις λέξεις. Η LSA υποθέτει ότι οι λέξεις των οποίων η έννοια μιαζει θα εμφανίζονται κοντά η μια στην άλλη στο κείμενο. Δημιουργείται μια μήτρα που περιλαμβάνει τα αποτελέσματα της tf-idf με σειρά την λεξη και κολόνα το έγγραφο πάνω στην οποια εφαρμόζεται μια μαθηματική τεχνική που ονομάζεται ανάλυση σε ιδιάζουσες τιμές, singular value decomposition (SVD) και χρησιμοποιείται για να μειώσει τον αριθμό των στηλών, διατηρώντας τη δομική ομοιότητα μεταξύ των γραμμών. [6]

7.1 Πίνακας εμφανίσεων

Η LSA μπορεί να χρησιμοποιήσει μια μήτρα συχνοτήτων των λέξεων στα έγγραφα που είναι αποτέλεσμα της tf-idf και, όπως είδαμε στο παράδειγμα, είναι μια αραιή μήτρα της οποίας οι γραμμές αντιστοιχούν στους όρους και οι στήλες αντιστοιχούν στα έγγραφα. Την παραπάνω μήτρα θα την ονομάσουμε πίνακα εμφανίσεων (occurrence matrix).

7.1.1 Μείωση Βαθμίδας

Μετά την κατασκευή του πίνακα εμφανίσεων η LSA βρίσκει μια προσέγγιση του με χρήση του SVD. Θα μπορούσαν να υπάρχουν διάφοροι λόγοι για αυτές τις προσεγγίσεις:

- Ο αρχικός πίνακας θεωρείται πολύ μεγάλος για τους πόρους του συστήματος. Σε αυτή την περίπτωση, η προσέγγιση ερμηνεύεται ως (ένα « αναγκαίο κακό»).
- Ο αρχικός πίνακας τεκμαίρεται θορυβώδης, για παράδειγμα, τυχαίες λέξεις πρόκειται να καταργηθούν. Από αυτή την άποψη, η προσέγγιση μπορεί ερμηνευθεί ως καλύτερη του αρχικού πίνακα.
- Ο αρχικός πίνακας θεωρείται υπερβολικά αραιός. Δηλαδή, παραθέτει μόνο τις λέξεις που παρουσιάζονται στην πραγματικότητα σε κάθε έγγραφο, ενώ μας ενδιαφέρουν όλες οι λέξεις που σχετίζονται με κάθε έγγραφο, γενικά ένα πολύ μεγαλύτερο σύνολο, λόγω συνωνυμίας όπως εξηγήθηκε παραπάνω .

Η συνέπεια της προσέγγισης είναι η μείωση του βαθμού της μήτρας πράγμα που κάνει μερικές διαστάσεις να συνδυαστούν και κάθε τιμή να εξαρτάτε από πολλές άλλες:

{(αυτοκίνητο), (φορτηγό), (λουλούδι)} -> {(1,3452 * αυτοκίνητο + 0,2828 * φορτηγό), (λουλούδι)}

Αυτό μετριάζει το πρόβλημα του προσδιορισμού συνωνυμίας, καθώς αναμένεται να συγχωνευτούν διαστάσεις που σχετίζονται με λέξεις που έχουν παρόμοια σημασία(τείνουν να παρουσιάζονται στα ίδια έγγραφα η έχουν πάντα μαζί τους την ίδια τρίτη λέξη). Επίσης αντιμετωπίζει σε κάποιο βαθμό το πρόβλημα με την πολυσημία, αφού οι διαστάσεις τους που δείχνουν προς την "σωστή" κατεύθυνση

προστίθενται στις διαστάσεις των λέξεων που μοιράζονται την ίδια σημασία. Αντίθετα διαστάσεις που δείχνουν στην "λάθος" κατεύθυνση τείνουν είτε να απλά να αλληλοακυρώνονται, ή, στη χειρότερη περίπτωση, να είναι μικρότερες από διαστάσεις στις κατευθύνσεις που αντιστοιχούν στο ζητούμενο νόημα.

7.1.2 Singular value decomposition

Στην γραμμική άλγεβρα, (SVD) είναι μια παραμετροποίηση μιας μήτρας. Η ανάλυση σε ιδιάζουσες τιμές μιας m × n πραγματικής ή μιγαδικής μήτρας M είναι μια παραμετροποίηση της μορφής:

M=UΣV'

όπου U είναι μια m × m πραγματική ή μιγαδική μήτρα, Σ είναι ένας m × n διαγώνιος πίνακας με μη αρνητικούς πραγματικός αριθμός στη διαγώνιο, και V' (ανάστροφος του V) είναι μια n × n πραγματική ή μιγαδική μήτρα. Η διαγώνιες τιμές του S είναι οι ιδιοτιμες του αρχικού πίνακα M.

7.1.3 Δήλωση θεωρήματος

Ας υποθέσουμε ότι M είναι μια m×n μήτρα με δεδομένα από το πεδίο K, το οποίο είναι είτε το πεδίο των πραγματικών αριθμών είτε αυτό των μιγαδικών αριθμών . Τότε υπάρχει μια παραμετροποίηση της μορφής

M=UΣV'

όπου U είναι ένας m × m πίνακας, η μήτρα Σ είναι ένας m × n διαγώνιος πίνακας με μη αρνητικούς πραγματικούς αριθμούς στη διαγώνιο, και V' έιναι n×n ενιαία μήτρα και ανάστροφος του V. Μια τέτοια παραμετροποίηση ονομάζεται ανάλυση σε ιδιάζουσες τιμές του M.

Τα διαγώνια στοιχεία Σ_{ii} του Σ είναι γνωστά ως ιδιοτιμές του M. Μια κοινή σύμβαση είναι η παράταξη στη λίστα των ιδιοτιμών σε φθίνουσα σειρά. Στην περίπτωση αυτή, η διαγώνιος μήτρα Σ είναι μοναδικά καθορισμένη από M (αν και η πίνακες U και V δεν είναι).

7.1.4 Ιδιοτιμές, ιδιοδιανύσματα και svd

Ένας μη-αρνητικός πραγματικός αριθμός σ είναι μια ιδιοτιμη για M αν και μόνο αν υπάρχει μοναδιαίο διάνυσμα u σε K"' και v σε K" τέτοιο ώστε

M υ=σ u και M' u= σ υ

Τα διανύσματα u και v ονομάζονται αριστερό και δεξιό μοναδιαίο διάνυσμα αντίστοιχα.

Σε κάθε ανάλυση σε ιδιάζουσες τιμές

M=UΣV'

τα διαγώνια στοιχεία του Σ είναι ίσα με τις ιδιοτιμές του πίνακα M. Οι στήλες του U και V είναι, αντίστοιχα, το αριστερό και δεξιό μοναδιαίο διάνυσμα για την αντίστοιχη ιδιοτιμή. Το παραπάνω θεώρημα συνεπάγεται ότι:

- Ένα m × n πίνακας M έχει τουλάχιστον μια και το πολύ p = min (m, n) διακριτές ιδιοτιμες.
- Είναι πάντα δυνατό να βρεθεί μια ορθογώνια βάση U για K"' που αποτελείται από τα αριστερά ιδιοδιανύσματα του πίνακα M.
- Είναι πάντα δυνατό να βρεθεί μια ορθογώνια βάση V για K" που αποτελείται από τα δεξιά ιδιοδιανύσματα M.

Μια ιδιοτιμή για την οποία μπορούμε να βρούμε δύο αριστερά (ή δεξιά) ιδιοδιανύσματα που είναι γραμμικά εξαρτώμενα ονομάζεται εκφυλισμένη.

Μια μη εκφυλισμένη ιδιοτιμή έχει πάντα αριστερό και δεξί ιδιοδιανύσμα. Συνεπώς, εάν όλες οι ιδιοτιμές του πίνακα M είναι μη-εκφυλισμένες και μη μηδενικές, τότε η ανάλυση σε ιδιάζουσες τιμές του είναι μοναδική.

Οι εκφυλισμένες ιδιοτιμές, εξ ορισμού, έχουν μη μοναδικά ιδιοδιανύσματα. Επιπλέον, εάν u1 και u2 είναι δύο αριστερά ιδιοδιανύσματα που αντιστοιχούν στην ιδιοτιμη σ, τότε κάθε κανονικοποιημένος γραμμικός συνδυασμός των δύο διανυσμάτων είναι επίσης ένα αριστερό ιδιοδιανύσμα που αντιστοιχεί στην ιδιοτιμη σ. Το ίδιο ισχύει και για τα δεξιά ιδιοδιανύσματα. Συνεπώς, αν M έχει εκφυλισμένες ιδιοτιμές, τότε η ανάλυση σε ιδιάζουσες τιμές του δεν είναι μοναδική.

7.1.5 Μήτρα προσέγγισης

Μερικές πρακτικές εφαρμογές του SVD πρέπει να λύσουν το πρόβλημα της προσέγγισης του πίνακα M με έναν άλλο πίνακα $\tilde{\text{M}}$ ο οποίος έχει μια συγκεκριμένη τάξη r προσέγγισης. Σε αυτήν την περίπτωση αποδεικνύεται ότι η λύση δίνεται από την SVD του M, και συγκεκριμένα

$\tilde{\text{M}}=\text{U}\tilde{\Sigma}V'$

όπου $\tilde{\Sigma}$ είναι η ίδια μήτρα όπως Σ εκτός από το ότι περιέχει μόνο τις r μεγαλύτερες ιδιοτιμές (οι άλλες ιδιοτιμές αντικαθίστανται από το μηδέν). Αυτό είναι γνωστό ως ο Eckart-Young θεώρημα, όπως αποδείχθηκε από τους δύο αυτούς συγγραφείς το 1936 (αν και βρέθηκε αργότερα να είναι γνωστό απο προηγούμενους συγγραφείς Βλέπε Stewart 1993).

Απόδειξη: Προσπαθούμε να ελαχιστοποιήσουμε $\left\|\text{M-}\tilde{\text{M}}\right\|_{\text{F}}$ χρησιμοποιώντας το $\text{rank}(\tilde{M})=\text{r}$.

Ας υποθέσουμε ότι η SVD του M = U Σ V'. Δεδομένου ότι η νόρμα Frobenius είναι μοναδική, έχουμε μια ισοδύναμη ισότητα:

$\min\left\|\Sigma\text{-U'}\tilde{M}\text{V}\right\|_F$

Σημειώστε ότι αφού ο Σ είναι διαγώνιος τότε και U'$\tilde{M}$V θα πρέπει να είναι διαγώνια, ώστε να ελαχιστοποιείται η νόρμα Frobenius . Η νόρμα Frobenius είναι η τετραγωνική ρίζα του αθροίσματος των τετραγώνων όλων των απολύτων τιμών. Αυτό σημαίνει ότι η U και V είναι επίσης μοναδιαίες μήτρες του $\tilde{M}$. Έτσι μπορούμε να υποθέσουμε ότι ο $\tilde{\text{M}}$ για να ελαχιστοποιηθεί η παραπάνω δήλωση έχει τη μορφή:

$\tilde{\text{M}}=\text{USV'}$

όπου S είναι διαγώνιος και οι διαγώνιες τιμές s_{ii} δεν είναι απαραίτητα διατεταγμένες όπως στο SVD.

$\min_{\tilde{M}}\|\Sigma\text{-S}\|_F\equiv\min_{s_i}\sqrt{\sum(\sigma_i\text{-s}_\iota)^2}$

Από τον περιορισμό τάξης, δηλαδή η S έχει r μη μηδενικά διαγώνια στοιχεία, το ελάχιστο των παραπάνω επιτυγχάνεται ως εξής:

δδ $\min_{s_i}\sqrt{\sum(\sigma_i\text{-s}_\iota)^2+\sum\sigma_i^2}=\sqrt{\sum\sigma_i^2}$

Ως εκ τούτου, $\tilde{\text{M}}$ βαθμού r είναι η καλύτερη προσέγγιση της M κατά νόρμα Frobenius όταν $\sigma_i=\text{s}_i(\text{i}=1\ldots r)$ και τα αντίστοιχα ιδιοδιανύσματα είναι ίδια με αυτα του M.

Στην παρακάτω εικόνα μπορούμε να δούμε τις διαφορές που επιφέρει η εφαρμογή SVD στον πήνακα της εικόνας 4.

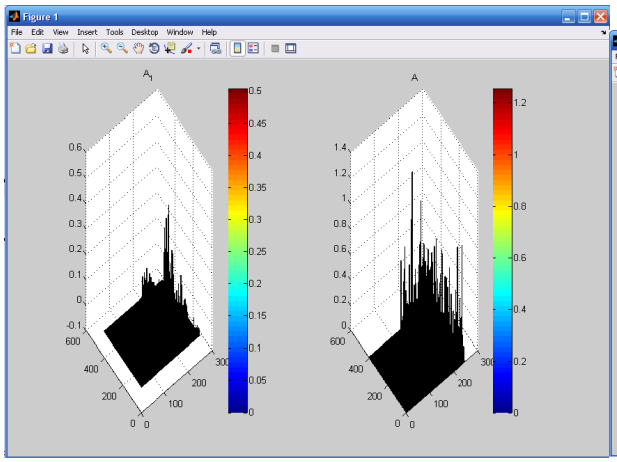
Redistribution of Term Weights in LSI

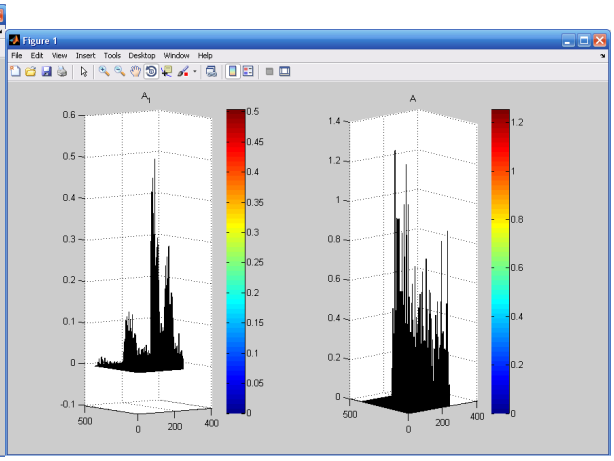
			After SVD (A_k, k = 2) d1	d2	d3	Totals		Before SVD A d1	d2	d3	Totals
a	=	k1	0.9662	0.985	1.0453	2.9965		1	1	1	3
arrived	=	k2	0.3003	1.1328	0.5974	2.0305		0	1	1	2
damaged	=	k3	0.6659	-0.1478	0.4478	0.9659		1	0	0	1
delivery	=	k4	-0.148	0.9347	0.1982	0.9853		0	1	0	1
fire	=	k5	0.6659	-0.1478	0.4478	0.9659		1	0	0	1
gold	=	k6	1.114	0.0506	0.8473	2.0119		1	0	1	2
in	=	k7	0.9662	0.985	1.0453	2.9965		1	1	1	3
of	=	k8	0.9662	0.985	1.0453	2.9965		1	1	1	3
shipment	=	k9	1.114	0.0506	0.8473	2.0119		1	0	1	2
silver	=	k10	-0.296	1.8692	0.396	1.9697		0	2	0	2
truck	=	k11	0.3003	1.1328	0.5974	2.0305		0	1	1	2
Totals			6.6159	7.8301	7.5151	21.9611	Totals	7	8	7	22
Expected Totals			7	8	7	22					

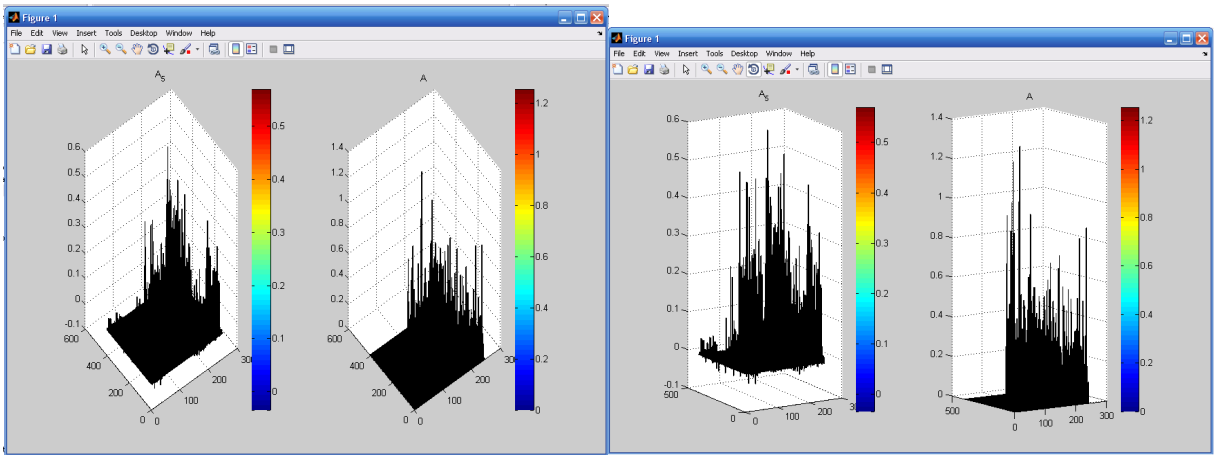
Net Noise Removed = 22 - 21.9611 = 0.0389

7.1.6 Παράδειγμα SVD

Στις παρακάτω εικόνες θα δούμε την διαφορά ανάμεσα στον κανονικό πήνακα και το SVD τάξης 1 μέχρι 5 του ιδίου πήνακα (πραγματικός πίνακας για σώμα αποτελεσμάτων απο Google).







7.2 Παράδειγμα LSI

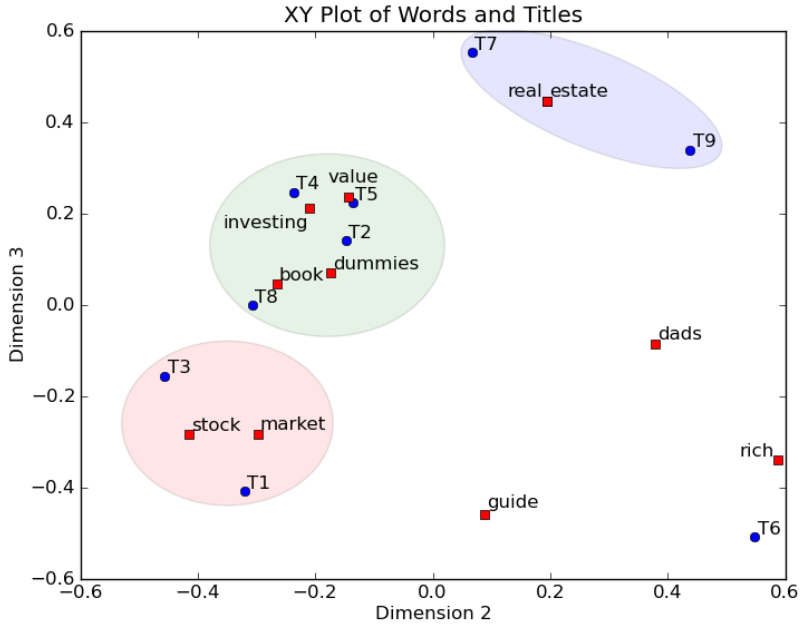
Σαν ένα μικρό παράδειγμα, έψαξα για τα βιβλία που χρησιμοποιούν τη λέξη "investing" στην Amazon.com και πήρα τους 9 πρώτους τίτλους βιβλίων που εμφανίστηκαν. Ενα token είναι οποιαδήποτε λέξη που:

- εμφανίζεται σε 2 ή περισσότερους τίτλους, και
- δεν είναι μια πολύ κοινή λέξη όπως "and", "the", και ούτω καθεξής (γνωστές ως stop words). Οι λέξεις αυτές δεν περιλαμβάνονται, επειδή δεν συμβάλλουν πολύ στο νόημα.

Σε αυτό το παράδειγμα, έχουμε αφαιρέσει τις ακόλουθες κοινές λέξεις stop words: "and", "edition", "for", "in", "little", "of", "the", "to".

1. The Neatest Little Guide to Stock Market Investing
2. Investing For Dummies, 4th Edition
3. The Little Book of Common Sense Investing: The Only Way to Guarantee Your Fair Share of Stock Market Returns
4. The Little Book of Value Investing
5. Value Investing: From Graham to Buffett and Beyond
6. Rich Dad's Guide to Investing: What the Rich Invest in, That the Poor and the Middle Class Do Not!
7. Investing in Real Estate, 5th Edition
8. Stock Investing For Dummies
9. Rich Dad's Advisors: The ABC's of Real Estate Investing: The Secrets of Finding Hidden Profits Most Investors Miss

Μόλις η Latent Semantic Analysis έχει εκτελεστεί σε αυτό το παράδειγμα, μπορούμε να σχεδιάσουμε τις λέξεις και τους τίτλους σε ένα γράφημα XY και να εντοπίσουμε clusters των τίτλων. Οι 9 τίτλοι απεικονίζονται με μπλε κύκλους και οι 11 tokens (διαστασεις) απεικονίζονται με κόκκινα τετράγωνα. Όχι μόνο μπορούμε να διακρίνουμε clusters τίτλων, αλλά αφού τα tokens μπορεί να απεικονίζονται μαζί με τους τίτλους, μπορούμε να ονομάσουμε τα clusters. Για παράδειγμα, το μπλε cluster που περιέχει τίτλους T7 και T9, αφορα την ακίνητη περιουσία. Το πράσινο cluster, με τους τίτλους T2, T4, T5 και T8, αφορα την επένδυση αξίας, και, τέλος, το κόκκινο cluster, με τίτλους T1 και T3, αφορά το χρηματιστήριο. Ο τίτλος T6 είναι μοναδικός και έτσι μένει από μόνος του.



Παρακάτω θα περάσουμε από όλα τα βήματα που χρειάζεται να τρέξουν για τον υπολογισμό της LSA σε αυτό το παράδειγμα.

7.2.1 Matrix Count

Το πρώτο βήμα σε αυτή την παρουσίαση της Latent Semantic Indexing είναι η δημιουργία της μήτρας συχνοτήτων των λέξεων. Κάθε κελί περιέχει τον αριθμό των επαναλήψεων που κάθε λέξη εμφανίζει σε αυτόν τον τίτλο. Για παράδειγμα, η λέξη "book" εμφανίζεται μία φορά στον τίτλο T3 και μία φορά στον τίτλο T4, ενώ η "investing" εμφανίζεται μία φορά σε κάθε τίτλο. Σε γενικές γραμμές, οι μήτρες που χτίζονται κατά τη διάρκεια αυτής της φάσης του LSA τείνουν να είναι πολύ μεγάλες, αλλά και πολύ αραιές (τα περισσότερα κελιά περιέχουν 0). Αυτό οφείλεται στο γεγονός ότι κάθε τίτλος ή έγγραφο περιέχει συνήθως μόνο ένα μικρό αριθμό από όλες τις πιθανές λέξεις του σώματος. Αυτή η σποραδικότητα μπορεί να αξιοποιηθεί τόσο στη χρήση μνήμης όσο και το χρόνο σε περισσότερο εξελιγμένες εφαρμογές LSA.

Λεξεις	T1	T2	T3	T4	T5	T6	T7	T8	T9
book			1	1					
dads						1			1
dummies		1						1	
estate							1		1
guide	1					1			
investing	1	1	1	1	1	1	1	1	1
market	1		1						
real							1		1
rich						2			1
stock	1		1					1	
value				1	1				

Εδώ θα σας παρουσιάσουμε Python κώδικα που υλοποιεί όλες τις απαραίτητες ενέργειες για να υπολογιστει η LSA. Ο Python κώδικας που χρησιμοποιείται παρακάτω θα κάνει χρήση των Python NumPy και SciPy βιβλιοθήκων.

7.2.2 Python - Εισαγωγή

Πρώτα πρέπει να εισάγουμε μερικές λειτουργίες από Python βιβλιοθήκες για να χειριστούν κάποιοι από τους υπολογισμούς που χρειάστει να κάνουμε. NumPy είναι η αριθμητική βιβλιοθήκη της Python, και εμείς θα εισαγάγουμε zeros, μια λειτουργία που δημιουργεί μια μήτρα μαι μηδενικά που θα χρησιμοποιήσουμε όταν χτίσουμε την μήτρα συχνοτήτων. Από το μέρος γραμμικής άλγεβρας του επιστημονικού πακέτου (scipy.linalg) εισάγουμε την SVD λειτουργία που υπολογισει στην πραγματικότητα την SVD, η οποία είναι η καρδιά του LSA.

```
from numpy import zeros
from scipy.linalg import svd
```

7.2.3 Python - Ορισμός Δεδομένων

Στη συνέχεια, ορίζουμε τα δεδομένα που θα χρησιμοποιήσουμε. Συνολικά 9 τίτλοι βιβλίων έχουν συγκεντρωθεί, και θα καταχωρηθούν στον πίνακα titles ενώ ο πινάκας stopwords κατέχει τις 8 κοινές λέξεις που πρόκειται να αγνοήσουμε(στην τελική εφαρμογή αυτός ο πινάκας θα είναι αρκετά διαφορετικός), όταν μετράμε τις λέξεις σε κάθε τίτλο, και ο πίνακας ignorechars έχει όλους τους χαρακτήρες στίξης που θα αφαιρεθούν. Αυτός ο πινάκας είναι μικρός γιατί πρόκειται να συναντήσουμε μόνο 4 σημεία στίξης : κόμμα (,), άνω και κάτω τελεία (:), απόστροφος (’), και το θαυμαστικό (!).

```
titles =
[
"The Neatest Little Guide to Stock Market Investing",
"Investing For Dummies, 4th Edition",
"The Little Book of Common Sense Investing: The Only Way to
Guarantee Your Fair Share of Stock Market Returns",
"The Little Book of Value Investing",
"Value Investing: From Graham to Buffett and Beyond",
"Rich Dad's Guide to Investing: What the Rich Invest in,
That the Poor and the Middle Class Do Not!",
```

```
"Investing in Real Estate, 5th Edition",
"Stock Investing For Dummies",
"Rich Dad's Advisors: The ABC's of Real Estate Investing:
The Secrets of Finding Hidden Profits Most Investors Miss"
]
stopwords = ['and','edition','for','in','little','of','the','to']
ignorechars = ''',:'!''
```

7.2.4 Python - Ορίσμος LSA κλάσης

Η τάξη LSA έχει μεθόδους για ανάλυση έγγραφων, και υπολογισμό των συχνοτήτων. Η πρώτη μέθοδος είναι η μέθοδος __init__, η οποία καλείται κάθε φορά που μια instance της κλάσης LSA δημιουργείται. Αποθηκεύει τα stopwords και ignorechars ώστε να μπορούν να χρησιμοποιηθούν αργότερα, και στη συνέχεια προετοιμάζει το λεξικό και υπολογίζει την μήτρα.

```
class LSA(object):
def __init__(self, stopwords, ignorechars):
self.stopwords = stopwords
self.ignorechars = ignorechars
self.wdict = {} self.dcount = 0
```

7.2.5 Python - Parse

Η μέθοδος parse παίρνει ένα έγγραφο, το χωρίζει σε λέξεις, καταργεί τους χαρακτήρες από τον πίνακα ignorechars και μετατρέπει τα πάντα σε πεζά, ώστε οι λέξεις να μπορεί να συγκριθούν με τα stopwords. Αν η λέξη είναι stopword, αγνοείται και προχωράμε στην επόμενη λέξη. Αν δεν είναι βάζουμε τη λέξη στο λεξικό, αλλά και επισυνάπτουμε τον τρέχον αριθμό για να παρακολουθείθουν τα έγγραφα στα οποία η λέξη εμφανίζεται.

Τα έγγραφα στα οποία κάθε λέξη εμφανίζεται φυλάσσονται σε έναν κατάλογο που σχετίζεται με αυτή τη λέξη στο λεξικό. Για παράδειγμα αφού η λέξη book εμφανίζεται στους τίτλους 3 και 4, θα είχαμε self.wdict ['book'] = [3, 4] αφού έχουν αναλυθεί όλοι οι τίτλοι.

Μετά την επεξεργασία όλων των λέξεων από το παρόν έγγραφο, θα αυξήσουμε τον αριθμό εγγράφων στο πλαίσιο της προετοιμασίας για το επόμενο έγγραφο που πρέπει να αναλυθεί.

```
def parse(self, doc):
words = doc.split(); for w in words:
w = w.lower().translate(None, self.ignorechars)
if w in self.stopwords:
continue
elif w in self.wdict:
self.wdict[w].append(self.dcount)
else:
self.wdict[w] = [self.dcount]
self.dcount += 1
```

7.2.6 Python - Υπολογισμός Πίνακα Εμφάνισεων

Αφού αναλυθούν όλα τα έγγραφα, όλες οι λέξεις κλειδιά που εμφανίζονται σε περισσότερα από 1 έγγραφο εξάγονται και να ταξινομούνται και χτίζεται η μήτρα εμφάνισης με αριθμό των γραμμών ίσο με τον αριθμό των λέξεων και αριθμό των στηλών ίσο με τον αριθμό έγγραφων όπως εξηγήθηκε παραπάνω.

```
def build(self):
self.keys = [k for k in self.wdict.keys() if len(self.wdict[k]) > 1]
self.keys.sort() self.A = zeros([len(self.keys), self.dcount])
for i, k in enumerate(self.keys):
for d in self.wdict[k]:
self.A[i,d] += 1
def printA(self):
print self.A
```

7.2.7 Python - Δοκιμή του LSA Τάξης

Μετά τον ορισμό της κλάσης LSA μπορούμε να δοκιμάσουμε στους 9 τίτλους βιβλίων που έχουμε. Πρώτα θα δημιουργήσουμε μια instance του LSA, που ονομάζεται mylsa, και θα αρχικοποιήθουν τα stopwords και ignorechars που ορίσαμε. Κατά τη διάρκεια της δημιουργίας, καλείται η μέθοδος __init__ η οποία αποθηκεύει τα stopwords και ignorechars και αρχικοποιεί το λεξικό και τον μετρητή έγγραφων.

Στη συνέχεια, καλούμε την μέθοδο parse για κάθε τίτλο. Η μέθοδος αυτή αποσπά λέξεις από κάθε τίτλο, διαγράφει χαρακτήρες στίξης, μετατρέπει κάθε λέξη σε πεζά, διαγράφει stopwords, και αποθηκεύει τις εναπομείναντες λέξεις σε ένα λεξικό μαζί με τον αριθμό τίτλου από τον οποίο προήλθαν.

Τέλος καλούμε την build() μέθοδο για τη δημιουργία του πίνακα εμφανίσεων. Αυτό αποσπά όλες τις λέξεις που έχουμε δει μέχρι τώρα, διαγράφει τις λέξεις που εμφανίζονται σε λιγότερο από 2 τίτλους, τις ταξινομεί, χτίζει μια μήτρα με μηδενικά σωστού μεγέθους και στη συνέχεια αυξάνει την τιμή στο σωστό κελί κάθε φορά που μια λέξη εμφανίζεται στον τίτλο.

```
mylsa = LSA(stopwords, ignorechars)
for t in titles: mylsa.parse(t)
mylsa.build()
mylsa.printA()
```

Αυτή είναι η μήτρα που εκτυπώνεται στην οθόνη από την printA(). Όπως μπορείτε να δείτε, είναι ο ίδια με την μήτρα που υπολογίσαμε νωρίτερα με το χέρι.

```
[[0. 0. 1. 1. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 1. 0. 0. 1.]
[0. 1. 0. 0. 0. 0. 0. 1. 0.]
[0. 0. 0. 0. 0. 0. 1. 0. 1.]
[1. 0. 0. 0. 0. 1. 0. 0. 0.]
[1. 1. 1. 1. 1. 1. 1. 1. 1.]
[1. 0. 1. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 1. 0. 1.]
[0. 0. 0. 0. 0. 2. 0. 0. 1.]
[1. 0. 1. 0. 0. 0. 0. 1. 0.]
```

```
[0. 0. 0. 1. 1. 0. 0. 0. 0.]]
```

7.2.8 TFIDF

Δεδομένου ότι έχουμε ένα τόσο μικρό παράδειγμα, και του οτι η TF-IDF υπολογίστικε παραπάνω σαν παράδειγμα θα παραλείψουμε αυτό το βήμα και να προχωρήσουμε στην καρδιά του LSA, υπολογίζοντας το SVD της μήτρας μας. Ωστόσο, αν θέλαμε να προσθέθει TFIDF στην LSA κλάση μας θα μπορούσαμε να προσθέσουμε τις ακόλουθες δύο γραμμές στην αρχή του python αρχείου μας, για την εισαγωγή του αρχείου καταγραφής, asarray, και τις λειτουργίες πρόσθεσης.

```
from math import log
from numpy import asarray, sum
```

Στη συνέχεια, θα προσθέσουμε την ακόλουθη μέθοδο TFIDF στην LSA τάξη μας. WordsPerDoc (N*,j) κατέχει το άθροισμα της κάθε στήλης, το οποίο είναι ο συνολικός αριθμός των λέξεων κλειδιών σε κάθε έγγραφο. DocsPerWord (Di) χρησιμοποιεί asarray να δημιουργήσει ένα διάνυσμα (array) από True και False αξίες, ανάλογα με το αν η τιμή του κελιού είναι μεγαλύτερη από 0 ή όχι. Στη συνέχεια, κάθε σειρά αθροίζεται και το αποτέλεσμα μας λέει σε πόσα έγγραφα εμφανίζεται κάθε λέξη. Τέλος εφαρμόζουμε τον τύπο TFIDF σε κάθε κελί. Πρέπει να μετατρέψουμε τον αριθμό στηλών σε float ώστε να αποφύγουμε την ακέραια διαίρεση.

```
def TFIDF(self):
WordsPerDoc = sum(self.A, axis=0)
DocsPerWord = sum(asarray(self.A > 0, 'i'), axis=1)
rows, cols = self.A.shape
for i in range(rows):
for j in range(cols):
self.A[i,j] = (self.A[i,j] / WordsPerDoc[j]) * log(float(cols) / DocsPerWord[i]
```

7.2.9 Εφαρμογή της Singular Value Decomposition

Έχοντας υπολογίσει την μήτρα εμφανίσεων, καλούμαστε να εφαρμόσουμε μια ισχυρή αλλά λίγο γνωστή τεχνική που ονομάζεται Singular Value Decomposition ή SVD για την ανάλυση του πίνακα μας.

Ο λόγος που η SVD είναι χρήσιμη, είναι γιατί υπολογίζει μια αναπαράσταση της μήτρας μας με μειομένη διάσταση που τονίζει τις ισχυρότερες σχέσεις και απομακρύνει το θόρυβο. Με άλλα λόγια, υπολογίζει την καλύτερη δυνατή προσέγγιση της μήτρας με τη λιγότερη δυνατή πληροφορία. Για να το κάνει αυτό, απομακρύνει τον θόρυβο, ο οποίος δεν βοηθά, και τονίζει ισχυρά πρότυπα και τάσεις, οι οποίες βοηθούν. Το τέχνασμα στη χρήση της SVD είναι το πόσες διαστάσεις ή "έννοιες" θα πρέπει να χρησιμοποιηθούν κατά την προσέγγιση της μήτρας. Εαν οι σημαντικές διαστάσεις είναι πολύ λίγες τότε σημαντικά patterns μπορεί να χαθούν, εάν οι διαστάσεις είναι πάρα πολλές τότε ο θόρυβος που προκαλείται από τυχαίες επιλογές λέξεων θα καθορίσει patterns που στην πραγματικότητα δεν υπάρχουν.

Ο αλγόριθμος SVD είναι λίγο περίπλοκος, αλλά ευτυχώς η Python έχει μια βιβλιοθήκη που τον καθιστά εύκολο στη χρήση. Με την προσθήκη αυτής της μεθόδου κάτω από την LSA κλάση μας, μπορούμε να παράγωγισουμε την μήτρα μας σε 3 άλλες μήτρες. Την U μήτρα που μας δίνει τις συντεταγμένες της κάθε

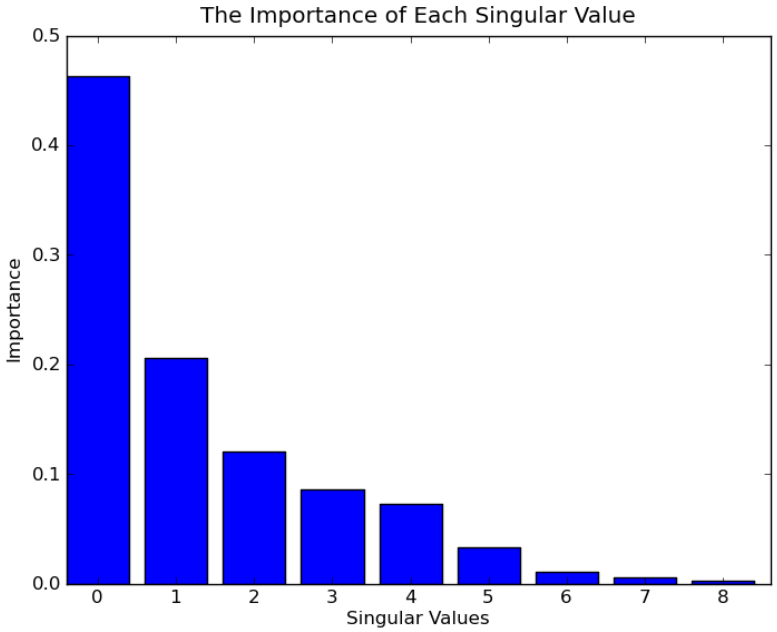


Figure 7.1:

λέξης στον χώρο "εννοιών", την Vt (V') μήτρα που μας δίνει τις συντεταγμένες του κάθε εγγράφου στον χώρο "εννοιών", και την μήτρα S των ιδιοτιμών που μας δίνει μια ιδέα για το πόσες διαστάσεις ή «έννοιες» θα πρέπει να περιληφθούν.

```
def calc(self):
self.U, self.S, self.Vt = svd(self.A)
```

Για να επιλέξουμε το σωστό αριθμό των διαστάσεων μπορούμε να κάνουμε ένα ιστόγραμμα του τετραγώνου των ιδιοτιμών. Σε αυτό το γράφημα βλέπουμε κατά πόσο κάθε ιδιοτιμή συμβάλλει στην προσέγγιση της μήτρας μας.

Για μεγάλες συλλογές εγγράφων, όπως θα είναι το σώμα των αποτελεσμάτων που θα μας δίνει η αναζήτηση στην google, ο αριθμός των διαστάσεων που χρησιμοποιείται είναι σε εύρος από 100 έως 500. Στο παράδειγμά μας θα χρησιμοποιήσουμε 3 διαστάσεις.

Παρακάτω θα παρουσιάσουμε τον υπολογισμό της SVD 3ων διαστάσεων της μήτρας μας. Κάθε λέξη έχει 3 αριθμούς που είναι η τιμή της σε κάθε διάσταση. Ο πρώτος αριθμός τείνει να αντιστοιχεί στον αριθμό των εμφανίσεων που κάθε λέξη έχει σε όλους τους τίτλους και δεν είναι τόσο κατατοπιστική, όσο η δεύτερη και η τρίτη διάσταση. Ομοίως, κάθε τίτλος έχει επίσης 3 αριθμούς που συνδέονται με αυτόν, ένα για κάθε διάσταση. Η πρώτη διάσταση δεν είναι πολύ ενδιαφέρουσα, επειδή τείνει να αντιστοιχεί στον αριθμό των λέξεων στον τίτλο.

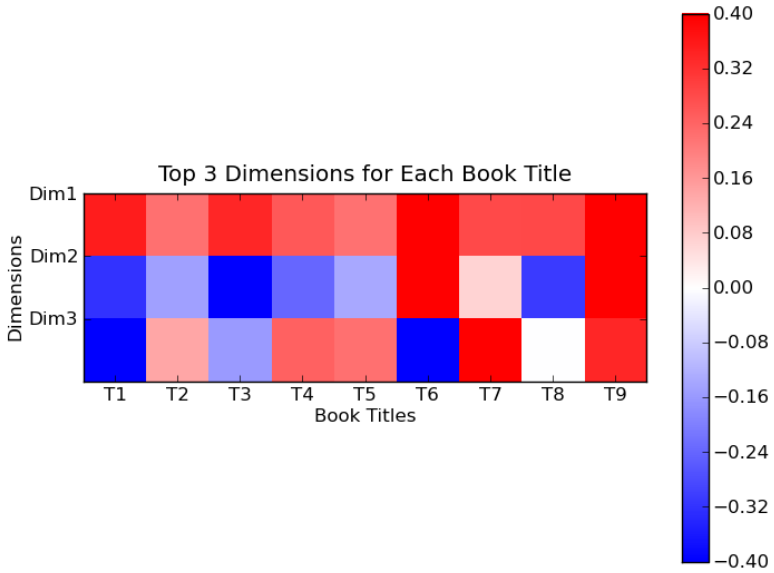


Figure 7.2:

book	0.15	-0.27	0.04			
dads	0.24	0.38	-0.09			
dummies	0.13	-0.17	0.07			
estate	0.18	0.19	0.45			
guide	0.22	0.09	-0.46	3.91	0	0
investing	0.74	-0.21	0.21	0	2.61	0
market	0.18	-0.30	-0.28	0	0	2.00
real	0.18	0.19	0.45			
rich	0.36	0.59	-0.34			
xtock	0.25	-0.42	-0.28			
value	0.12	-0.14	0.23			

T1	T2	T3	T4	T5	T6	T7	T8	T9
0.35	0.22	0.34	0.26	0.22	0.49	0.28	0.29	0.44
-0.32	-0.15	-0.46	-0.24	-0.14	0.55	0.07	-0.31	0.44
-0.41	0.14	-0.16	0.25	0.22	-0.51	0.55	0.00	0.34

7.2.10 Clustering με Χρώμα

Μπορούμε επίσης να μετατρέψουμε τους αριθμούς σε χρώματα. Για παράδειγμα, εδώ είναι μια έγχρωμη αναπαράσταση που αντιστοιχεί στις 3 πρώτες διαστάσεις του πίνακα των τίτλων που δείξαμε παραπάνω. Περιέχει ακριβώς τις ίδιες πληροφορίες, εκτός από το ότι το μπλε δείχνει αρνητικούς αριθμούς, το κόκκινο δείχνει θετικούς αριθμούς, και οι αριθμοί κοντά στο 0 είναι λευκοί. Για παράδειγμα, ο τίτλος 9, ο οποίος είναι πολύ θετικός και στις 3 διαστάσεις, είναι επίσης έντονα κόκκινος και στις 3 διαστάσεις.

Μπορούμε να χρησιμοποιήσουμε αυτά τα χρώματα για ομαδοποίηση(clustering) των τίτλων. Αγνοούμε την πρώτη διάσταση για την ομαδοποίηση, επειδή όλοι οι τίτλοι είναι κόκκινοι. Στη δεύτερη διάσταση, έχουμε το ακόλουθο αποτέλεσμα.

Dim2	Titles
red	6-7,9
blue	1-5,8

Χρησιμοποιώντας την τρίτη διάσταση, μπορούμε να χωρίσουμε κάθε μία από αυτές τις ομάδες ξανά με τον ίδιο τρόπο. Για παράδειγμα, εξετάζοντας την τρίτη διάσταση, τίτλος 6 είναι μπλε, αλλά ο τίτλος 7 και τίτλος 9 είναι ακόμη κόκκινοι. Με αυτόν τον τρόπο για τις δύο ομάδες, θα καταλήξουμε στην παρακάτω ομαδοποίηση.

Dim2	Dim3	Titles
red	red	7,9
red	blue	6
blue	red	2,4-5,8
blue	blue	1,3

7.2.11 Ομαδοποίηση κατά τιμή

Απορρίπτοντας την πρώτη διάσταση, όπως συζητήσαμε, μπορούμε να φτιάξουμε μια γραφική παράσταση δυο διαστάσεων χρησιμοποιώντας ένα διάγραμμα XY και τις διαστάσεις Dim2 και Dim3. Θα τεθεί η δεύτερη διάσταση στον άξονα X και η τρίτη διάσταση στον άξονα Υ. Είναι ενδιαφέρον να συγκρίνουμε το γράφημα XY με τον πίνακα που δημιουργήσαμε παραπάνω χρησιμοποιώντας μόνο τα χρώματα.

Στο γράφημα που ακολουθεί, οι λέξεις αντιπροσωπεύονται από κόκκινα τετράγωνα και οι τίτλοι αντιπροσωπεύονται από μπλε κύκλους. Για παράδειγμα η λέξη "book" έχει τιμές διαστάσεων (0,15, -0,27, 0,04). Αγνοούμε την τιμή πρώτης διάστασης 0,15 και παραστούμε την λέξη "book" στη θέση (x = -0,27, y = 0,04) όπως φαίνεται στο γράφημα. Το ίδιο κάνουμε και για τους τίτλους.

Ένα πλεονέκτημα αυτής της τεχνικής είναι ότι και λέξεις και οι τίτλοι μπορούν να διατίθενται στην ίδια γραφική παράσταση. Όχι μόνο μπορούμε να προσδιορίσουμε clusters των τίτλων, αλλά μπορούμε να ονομάσουμε τα clusters εξετάζοντας τι λέξεις εμφανίζονται. Για παράδειγμα, το κάτω αριστερό cluster έχει τίτλους 1 και 3, οι οποίοι είναι και οι δύο για την επένδυση στο χρηματιστήριο(stock, market). Οι λέξεις «stock» και «market» βρίσκονται στο ίδιο cluster και είναι εύκολο αποφανθούμε ως προς την ονομασία του. Ένα άλλο παράδειγμα είναι το μεσαίο cluster που έχει τους τίτλους 2, 4, 5, και, κατα εναν μικρότερο βαθμό τον τίτλο 8.Οι τίτλοι 2, 4, και 5 βρίσκονται δίπλα στις λέξεις «value» και «investing» που συνοψίζουν τους εν λόγω τίτλους αρκετά καλά.

Το παραπάνω αν και αρκετά επιτυχές για ενα μικρό σώμα δεδομένων δεν θα μπορούσε να εφαρμοστεί σε μεγαλήτερα σώματα οπου η επιλογή 3 ενοιών θα ειναι υπεραπλούστευση. Στην εργασία μας η μεταφορά στο δισδιάστατο επίπεδο θα γίνει με εφαρμογή του SOM.

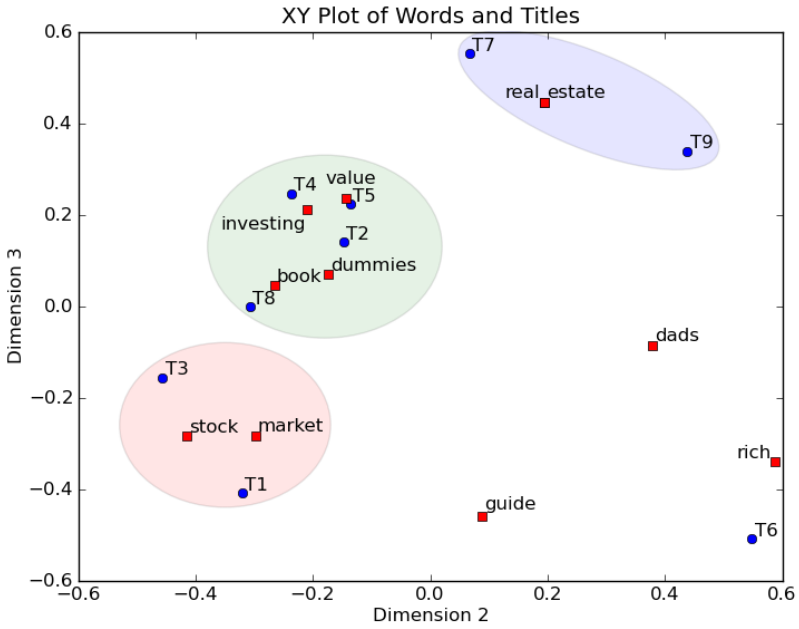
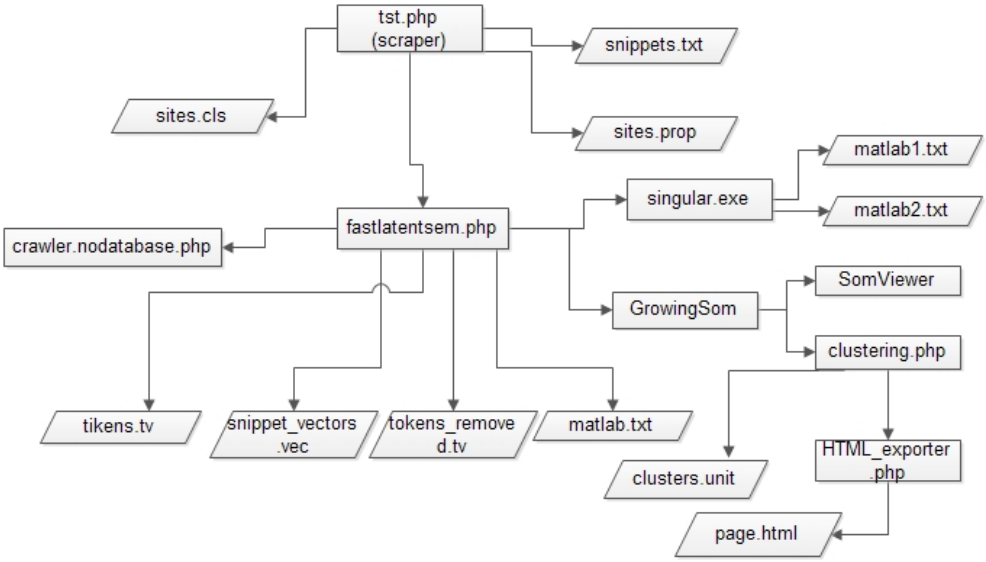

XY Plot of Words and Titles
T7
real estate
T9
value
T4
T5
investing
T2
dummies
book
T8
dads
T3
stock
market
rich
T1
guide
T6
Dimension 3
Dimension 2
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6

Figure 7.3:

8 Μοντέλο

Για την εφαρμογή ολλων των παραπάνω στην πράξη χρειάστηκε να αναπτύξουμε ένα σύστημα αναζήτησης στο διαδίκτυο που αναλύει και ομαδοποιεί τα αποτελέσματα της αναζήτησης της μηχανής της Google. Αποφασίστηκε να χρησιμοποιηθεί αυτή η μηχανή καθώς η χρήση προγράμματος bot για το χτίσιμο δικιάς μας βάσης με δεδομένα από της ιστοσελίδες που το bot έχει επισκεφθεί θα ήταν αρκετά χρονοβόρα και θα χρειαζόταν TB μνήμης για να μιμηθεί τα αποτελέσματα της google.



Στήν παραπάνω εικόνα φαίνεται ένα πιο αναλυτικό dataflow διάγραμμα του εν λόγου συστήματος. Η ανάπτυξη έγινε σε μερικές γλώσσες όπως η PHP,Python που είναι αρκετά αποτελεσματικές στην ανάπτυξη εφαρμογών διαδικτύου και οχι τοσο αποδοτικές σε πολλαπλασιασμούς μεγάλων πινάκων, που θα χρειαστούν για τον υπολογισμό της SVD. Για το υπολογιστικά εντατικό αυτό κομμάτι του αλγορίθμου επιλέξαμε την Matlab, η οποία είναι σχεδιασμένη με στόχο την αποτελεσματικότητα σε τέτοιους υπολογισμούς. Για το κομμάτι της ομαδοποίησης και την ποσοτικοποίηση των αποτελεσμάτων χρησιμοποιήθηκε μια έτοιμη σουίτα εργαλείων για νευρωνικά δίκτυα από το πανεπιστήμιο της Βιένης υλοποιημένη σε Java[2]. Παρακάτω θα εξηγήσουμε κάθε βήμα του αλγορίθμου καθώς και τους μικροσυντονισμούς και επιλογές που έχουμε κάνει στην δικιά μας υλοποίηση, για βέλτιστα αποτελέσματα για κείμενα με χαρακτηριστικά σαν αυτά των snippets.

8.1 Επιλογή μεγέθους σώματος αποτελεσμάτων

Αρχικά θα πρέπει να αποφασίσουμε πόσα αποτελέσματα θα πάρουμε από την Google. Για το παραπάνω το API της Google προσφέρει 3 επιλογές 10,50,100. Για να επιλέξουμε το μέγεθος θα πρέπει να υπολογίσουμε μετρικές για την ποιότητα του clustering και για αυτό θα χρειαστεί να ξέρουμε τις κλάσεις που εμφανίζονται

[2] http://www.ifs.tuwien.ac.at/dm/somtoolbox/

στο σώμα αποτελεσμάτων πριν την εκπαίδευση του SOM. Το παραπάνω θα επιτευχθεί με την συγχώνευση 5 διαφορετικών σετ αποτελεσμάτων, το κάθε ένα για διαφορετική query. Οι όροι που επιλέξαμε είναι 'artificial neural networks','genetic programming','association rule learning','decision tree learning','machine learning'. Ξέροντας την ομάδα στην οποία ανήκει το κάθε snippet μπορούμε να υπολογίσουμε της μετρικές και τα αποτελέσματα φαίνονται παρακάτω. Οι μετρικές που χρησιμοποιήθηκαν είναι cluster uncompactness, cluster separation, sigma, ocq, rand statistic, jaccard coefficient, folkens and mallows, purity και τελικά F1 που είναι συνάρτηση μερικών από τις παραπάνω μετρικές.

Cluster Uncompactness: είναι το αντίστροφο του cluster compactness που έχει να κάνει με το πόσο στενά σχετίζονται τα αντικείμενα σε ένα cluster. Το μέτρο εκτιμά την πυκνότητα διασποράς με βάση την απόσταση από το κέντρο του κάθε cluster(centroid).

Cluster Separation: μέτρα πόσο διαφέρει η πόσο καλά διαχωρισμένο είναι ένα cluster από τα γειτονικά του εκτιμώντας τις αποστάσεις των centroids.

OCQ: ένας συνδυασμός του cluster uncompactness και cluster separation:

$Ocq\,(\beta) = \beta/CU + (1-\beta)\cdot \mathrm{CS}$

όπου CU : cluster uncompactness, CS : cluster separation και β είναι ένα βάρος που επιλέξαμε να είναι β=0.5

Rand Statistic: Ο δείκτης Rand [12] ή μέτρο Rand (από William M. Rand) στην ομαδοποίηση δεδομένων, είναι ένα μέτρο της ομοιότητας μεταξύ δύο ομαδοποιήσεων που σχετίζεται με την ακρίβεια, αλλά ισχύει ακόμα και όταν οι ετικέτες κατηγορίας(labels) δεν χρησιμοποιούνται. Εστω ένα σύνολο n στοιχείων S= $\{o_1, ..., o_n\}$ και δύο διαμερισμοί του S, X= $\{\mathrm{X}_1, ..., \mathrm{X}_r\}$ και Y= $\{\mathrm{Y}_1, ..., \mathrm{Y}_s\}$:

a: ο αριθμός των ζευγών των στοιχείων σε S που βρίσκονται στο ίδιο υποσυνολο σε X και στο ίδιο υποσύνολο στο Υ

b: ο αριθμός των ζευγών των στοιχείων σε S που βρίσκονται σε διαφορετικά σύνολα σε X και Y

c: ο αριθμός των ζευγών των στοιχείων σε S που βρίσκονται στο ίδιο σετ σε X και σε διαφορετικά στο Y

d: ο αριθμός των ζευγών των στοιχείων σε S που βρίσκονται σε διαφορετικά σύνολα σε X και στο ίδιο σε Υ

Ο τύπος του δείκτη Rand θα είναι:

$R = \frac{a+b}{a+b+c+d}$

Jaccard Coefficient: μετρά την ομοιότητα μεταξύ σύνολων δεδομένων, και ορίζεται ως το μέγεθος της τομής διαιρούμενο με το μέγεθος της ένωση των συνόλων:

$J\,(A,B) = \frac{|A \cap B|}{|A \cup B|}$

Fowlkes and Mallows[7]: χρησιμοποιείται για τον προσδιορισμό της ομοιότητας δυο ομαδοποιείσεων.

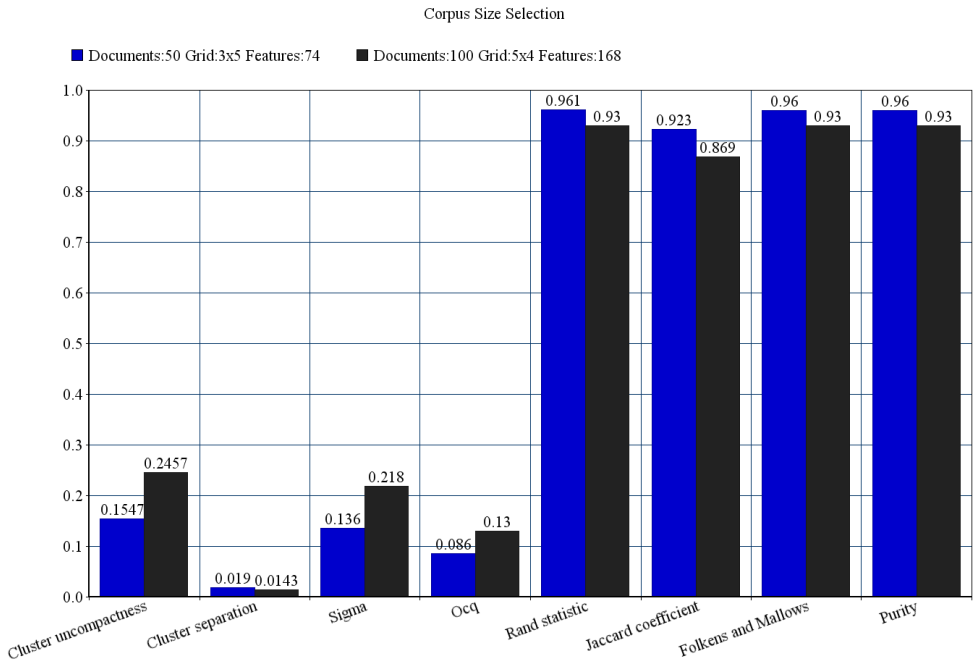
$FM = \sqrt{\frac{TP}{TP+FP} \cdot \frac{TP}{TP+FN}}$

όπουTP: true positives, FP: false positives, FN: false negatives

F1: είναι συνάρτηση της ακρίβειας και ανάκλησης μιας ομαδοποίησης:

$F_1 = 2\frac{precision \cdot recall}{precision + recall}$

όπου ακρίβεια σχετίζεται με πόσα δεδομένα έχουν ταξινομηθεί στο ίδιο cluster δεδομένου ότι έχουν την ίδια ετικέτα(label), και ανάκληση σχετίζεται με πόσα από τα δεδομένα κάθε ετικέτας έχουν ταξινομηθεί σε ίδιους cluster.



Όπως φαίνεται στην εικόνα τα αποτελέσματα του clustering είναι καλύτερα για 50 results. Στην περίπτωση των 100 αποτελεσμάτων φαίνεται να προστίθεται αρκετός θόρυβος που επιδρά αρνητικά, καθώς και ο χρόνος εκτέλεσης γίνεται 3x μεγαλύτερος.

8.2 Αφαίρεση Όρων

Στο τέλος του Scraper έχει δημιουργηθεί ένα σώμα με τα 50 πρώτα αποτελέσματα από όλους τους ορούς αναζήτησης. Τότε ο έλεγχος και τα αποτελέσματα περνάνε στο επόμενο script που είναι μια υλοποίηση της LSI σε PHP (LSI). Η LSI δέχεται αυτά τα 50 αποτελέσματα σαν 2 λίστες, μια η λίστα με τα urls και η άλλη αυτή που περιεχέι τα snippets και τίτλους. Από αυτό το σημείο και παρακάτω για λόγους σύγκρισης τα 50 παραπάνω αποτελέσματα θα είναι σταθερά και θα ανακτώνται από μια βάση mysql. Το URL θα χρησιμοποιηθεί σαν label ενώ το snippet αποστέλλεται στον Parser που υπολογίζει το Tf κομμάτι της Tf-Idf.

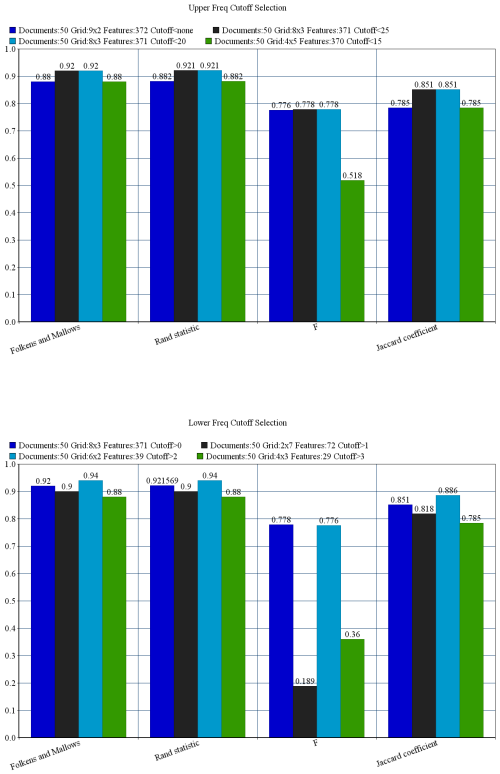
Στην κλάση αυτή το κείμενο πρώτα επιδέχεται μια επεξεργασία, αρχικά αφαιρούνται κοινές λέξεις που προσθέτουν θόρυβο με απλή σύγκριση με μια λίστα από συχνές λέξεις της αγγλικής, στην συνεχεία με χρήση μιας υλοποίησης του porter stemmer σε php(stemmer.php) οι λέξεις που έχουνε απομείνει ανακόπτονται.

Η προκύπτουσα συμβολοσειρά είναι έτυμη για το τελευταίο βήμα του parsing που είναι η μέτρηση των επαναλήψεων των λεκτικών μονάδων (token). Με λεκτικές μονάδες εννοούμε της ρίζες των λέξεων που έχουνε υπολογιστεί από τον αλγόριθμο porter. Σε αυτό το σημείο μετά από μερικές δοκιμές αποφασίσαμε να δώσουμε μεγαλύτερο βάρος στις λέξεις που προέρχονται από τον τίτλο σε σύγκριση με αυτές που προέρχονται από το υπόλοιπο κείμενο του snippet. Οι διαφοροποιήσεις

του βάρους λέξεων από τους τίτλους των snippet δεν επέφερε μεγάλη διαφορά στο μέτρο F1(0.612-0.667) άλλα κρίθηκε αναγκαίο. Μεγαλύτερες τιμές από 1,5x είχαν αρνητικές επιπτώσεις στο μέτρο F.

Η ανάλυση επιστρέφει σαν αποτέλεσμα τον αριθμό επαναλήψεων των λέξεων που μας ενδιαφέρουν από το snippet καθώς και το stem τους. Όταν επεξεργαστούν όλα τα snippets που είχαμε συλλέξει αρχικά στον Scraper η δισδιάστατη μήτρα στην μνήμη της LSI είναι ο πίνακας εμφανίσεων για το συγκεκριμένο σώμα δεδομένων και μπορούμε να την επεξεργαστούμε με τα συνήθη εργαλεία της LSI.

Πριν προχωρήσει οποιοδήποτε περαιτέρω βήμα της LSI πρέπει η μήτρα να υποβληθεί σε μια προεπεξεργασία, σκοπός της οποίας είναι η μείωση του θορύβου που προσθέτουν λέξεις με μικρή η μεγάλη συχνότητα εμφάνισης. Για την αξιολόγησή των αποτελεσμάτων θα χρειαστούμε ένα μέτρο. Το μέτρο αξιολόγησης που χρησιμοποιήθηκε είναι το F-measure (F1), αν και υπολογίζονται και άλλα μέτρα. Το F1 κρίθηκε να είναι ο κατάλληλος συνδυασμός ανάκλησης και ακρίβειας (Recall & Precission) για την συγκεκριμένη εφαρμογή καθώς χρησιμοποιείται σε άλλες ανάλογες εργασίες.



Οι λέξεις που εμφανίζονται σε περισσότερα από τα μισά(25) έγγραφα φαίνεται ότι προσθέτουν θόρυβο όπως βλέπουμε στην πρώτη εικόνα, όμως εάν συνεχίσουμε να κατεβάζουμε το cutoff η ποιότητα του clustering μειώνεται. Σαν πάνω όριο θα επιλεγεί το 25. Τα πράγματα δυσκολεύουν όταν βλέπουμε το κάτω όριο, όπως φαίνεται στην εικόνα υπάρχει μεγάλη διαφοροποίηση στις τιμές των μετρικών ιδιαίτερα στην F. Η διαφορές αυτές δεν ακολουθούν καθαρά τον αριθμό των features που απαλείφονται, πράγμα που σημαίνει ότι δεν μπορούμε να κόψουμε αυτές

τις λέξεις και να περιμένουμε αυτό να επιφέρει μια μεγάλη βελτίωση στο μέτρο ομαδοποίησης F1 για κάθε σωμα δεδομένων. Σαν χαμηλότερο όριο θα επιλεγεί το 1 γιατί αν και παραπάνω βλέπουμε μια μεγάλη πτώση στην F1, αυτο ενδέχεται να έχει να κάνει με το σώμα των δεδομένων και το γεγονός ότι δημιουργήθηκε από 5 διαφορετικές αναζητήσεις και τα αποτελέσματα είναι αρκετά διαφορετικά για μια ομοιόμορφη αναζήτηση.

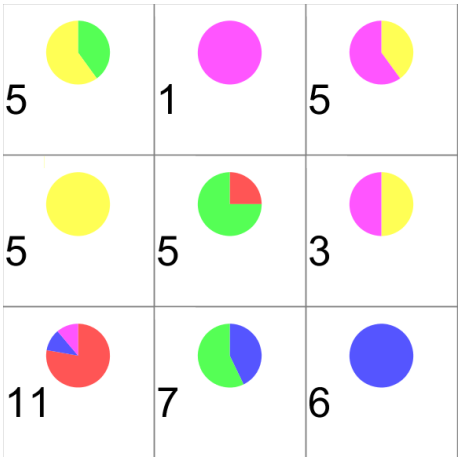
8.3 Απεικόνιση

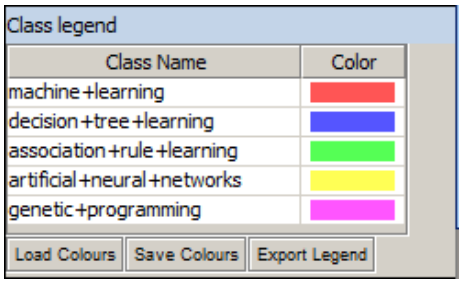
Έχοντας πια τελειώσει με το στάδιο της επεξεργασίας και αφαίρεσης θορύβου τα αποτελέσματα είναι έτυμα για το επόμενο βήμα της LSI, υπολογισμός του SVD. Σε αυτό το σημείο αντιμετωπίσαμε ένα πρόβλημα που είχε να κάνει με την ασυμβατότητα της PHP με πολύπλοκους μαθηματικούς υπολογισμούς που απαιτούν εντατική και πιο άμεση χρήση του επεξεργαστή. Η σχεδίαση της PHP έκανε το script σε αυτό το σημείο πολύ αργό και με τεράστιες απαιτήσεις μνήμης.

Αναγκαστήκαμε να περάσουμε σε matlab και να σχεδιάσουμε μια διεπαφή των δυο. Το πρόγραμμα που σχεδιάσαμε σε matlab υπολογίζει SVD τάξεις k ενός πίνακα που διαβάζει από ένα αρχείο με k ένα input του χρήστη. Ή PHP από την μεριά της αποθηκεύει τον πίνακα που μέχρι τώρα υπήρχε μόνο στην μνήμη της σε ένα αρχείο, το οποίο διαβάζεται από την matlab και έχοντας υπολογίσει το SVD, το αποθηκεύει σε ένα δεύτερο αρχείο το οποίο αυτή την φορά διαβάζει η php.

Με αυτό το τέχνασμα αποφεύγουμε τον υπολογισμό του SVD σε php και παρατηρούμε μεγάλη βελτίωση στους χρόνους ολοκλήρωσης και στην χρήση μνήμης.

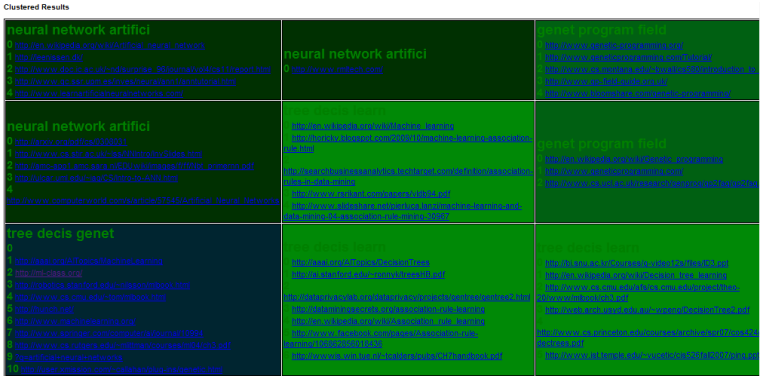
Σε αυτό το σημείο ο πίνακας είναι έτυμος για την χρήση στην εκπαίδευση ενός Growing Grid. Ο αλγόριθμος Growing Grid που χρησιμοποιούμε χρειάζεται μερικές παραμέτρους που έχουν να κάνουν με το αρχικό μέγεθος του πλέγματος και την ταχύτητα σύγκλησης, ο υπολογισμός αυτών γίνεται στον Scraper και εξαρτάτε από μερικά χαρακτηριστικά του σώματος που έχουμε συσσωρεύσει, όπως το μέγεθος του. Σαν αποτέλεσμα έχουμε δυναμικό υπολογισμό των παραμέτρων και διαφορετική εκπαίδευση για κάθε σώμα δεδομένων, πράγμα που βελτιώνει ορατά τα αποτελέσματα που θα είχαμε με στατικές παραμέτρους. Η εκπαίδευση του δικτύου γίνεται με την υλοποίηση του αλγορίθμου Growing Grid σε Java από το πανεπιστήμιο της Βιένης όπως αναφέραμε παραπάνω και τα αποτελέσματα μπορούμε να τα απεικονίσουμε με την χρήση ενός εργαλείου από το ίδιο πακέτο (SomViewer).





Παραπάνω βλέπουμε ότι αποτελέσματα από την ίδια αναζήτηση(ίδιο χρώμα, class legend) τείνουν να αντιστοιχίσουν στους ίδιους η γειτονικούς clusters. Παρατηρούμε επίσης ότι τα αποτελέσματα της αναζήτησης για Machine Learning (χρώμα κόκκινο) βρίσκονται κοντά στο γεωμετρικό κέντρο του πλέγματος όπως θα περιμέναμε αφού επικαλύπτουν όλες τις άλλες αναζητήσεις.

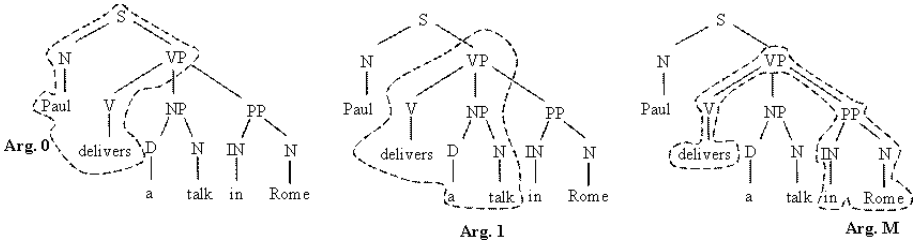
Τα αποτελέσματα από την ομαδοποιηση περνάνε σε ένα άλλο script (Post_Processing) που τα επεξεργάζεται και υπολογίζει διάφορα μέτρα αξιολόγησης τα οποία αποθηκεύονται σε ένα αρχείο (results.unit). Όπως βλέπουμε στις παραπάνω εικόνες οι διαφορετικές κλάσεις είναι αρκετά ορατές στα μάτια μας όμως θα πρέπει να βρούμε έναν μαθηματικά ρητό τρόπο για να αποφασίσουμε αν το κάθε cluster ανήκει σε διαφορετική κλάση(HyperCluster) η όχι. Γιαυτό τον λόγο μερικά από τα μεγέθη που έχουν υπολογιστεί στο Post_Processing περνάνε σε μια κλάση (HTML_Exporter.php) που αποφασίζει αν κάποιο cluster ανήκει στην ίδια κλάση με κάποιον άλλο συγκρίνοντας τα centroids των δυο cluster. Αν οι πρώτες 10 πιο σημαντικές λέξεις του ενός centroid, δλδ αυτές με μεγαλύτερα βάρη είναι κατά 70% ίδιες με αυτές του άλλου centroid τότε τα δυο clusters ανήκουν στην ίδια κλάση. Και σαν αποτέλεσμα των παραπάνω συγκρίσεων που εφαρμόζονται μόνο για γειτονικά clusters υπολογίζονται όλες η κλάσεις που φαίνεται να παρουσιάζονται στα αποτελέσματα. Σαν ονομασία της κλάσης σε αυτό το σημείο θα χρησιμοποιήσουμε τις 3 πιο σημαντικές λέξεις του centroid.



8.4 Συντακτικό βάρος

Ένα πρόβλημα με την δικιά μας εργασία είναι το μικρό μέγεθος των εγγράφων. Αυτό θα προκαλεί στην μέση περίπτωση να εμφανίζονται το πολύ 1-2 φορές σε ολόκληρο το έγγραφο λέξεις που είναι πολύ πιο σημαντικές από άλλες που εμφανίζονται 1 φορά. Ένας απλός υπολογισμός συχνότητας σε αυτόν τον τύπο εγγράφου θα χάνει την πληροφορία που προσφέρει η σύνταξη τις πρότασης.

Για τον παραπάνω λόγο αποφασίσαμε σε αυτό το σημείο να προσθέσουμε ένα επιπλέον κριτήριο για τον υπολογισμό του βάρους tf. Αποφασίσαμε να χρησιμοποιήσουμε τα συντακτικά χαρακτηριστικά των προτάσεων του snippet οπως προτείνεται από τον Alessandro Moschitti, [11]. Θα προσπαθήσουμε να χρησιμοποιήσουμε ολόκληρα υποδέντρα του συντακτικού δέντρου σαν χαρακτηριστικά οπως φαίνεται στην παρακάτω εικόνα.



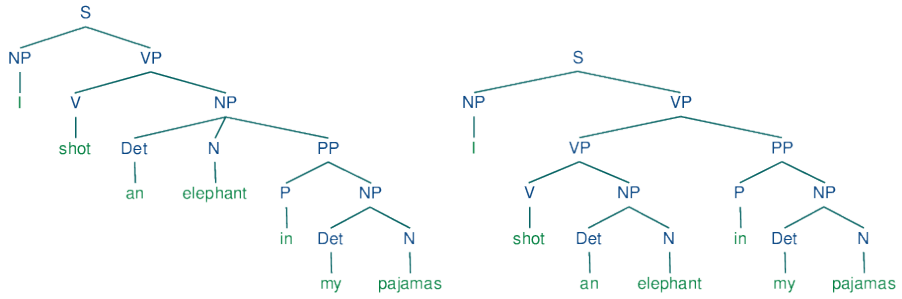
Σε αυτό το σημείο είναι προφανές ότι θα πρέπει κάπως να υπολογίσουμε το συντακτικό δέντρο για κάθε πρόταση σε κάθε snippet. Για αυτό θα χρησιμοποιήσουμε μια υλοποίηση ενός parser που προτάθηκε [2].

Σαν παράδειγμα θα υπολογίσουμε τα 2 πιθανά συντακτικά δέντρα για την παρακάτω πρόταση

I shot an elephant in my pajamas.

```
(S (NP I) (VP (V shot) (NP (Det an) (N elephant)
(PP (P in) (NP (Det my) (N pajamas))))))
```

```
(S (NP I) (VP (VP (V shot) (NP (Det an) (N elephant)))
(PP (P in) (NP (Det my) (N pajamas)))))
```



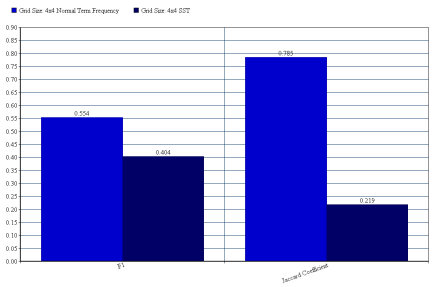
Τεχνικά προβλήματα παρουσιάστηκαν στο συγκεκριμένο στάδιο γιατί ο παραπάνω υπολογισμός χρειάζεται τουλάχιστον 3GB RAM για να ανεβάσει στην μνήμη μοντέλα που χρησιμοποιούνται για τους υπολογισμούς και περνούν αρκετό χρόνο. Για τον λόγο αυτόν αποφασίσαμε το συγκεκριμένο στάδιο να είναι προαιρετικό, και να τρέχει σε έναν απομακρυσμένο server. Το σώμα των δεδομένων επεξεργάζετε πολυνηματικά στον απομακρυσμένο server σε 8 νήματα, αυτό επέφερε μια αρκετά μεγάλη βελτίωση στον χρόνο εκτέλεσης.

Ο αλγόριθμος που υπολογίζει την συντακτική ανάλυση δεν χρησιμοποιεί λεξικό, γιαυτό δεν θα έχουμε σωστά αποτελέσματα πάντα. Παρακάτω βλέπουμε τα αποτελέσματα του αλγορίθμου για την αναγνώριση των μερών του λόγου στις συνήθεις μετρικές.

```
[root@sp4755e python]# /usr/local/bin/python2.7 test.py
     Type        Total    Precision       Recall      F1-Score
============================================================
     ADJP           15        0.538        0.467        0.500
     ADVP           22        0.667        0.727        0.696
     FRAG            0        0.000        0.000        0.000
      NAC            0        0.000        0.000        0.000
       NP          294        0.710        0.741        0.725
       NX            4        0.000        0.000        0.000
       PP           89        0.686        0.663        0.674
      PRN            0        0.000        0.000        0.000
      PRT            0        0.000        0.000        0.000
       QP            9        1.000        0.667        0.800
        S           80        0.444        0.550        0.492
     SBAR           17        0.140        0.353        0.200
    SBARQ           37        0.846        0.892        0.868
     SINV            3        0.333        0.333        0.333
       SQ           39        0.750        0.846        0.795
      UCP            0        0.000        0.000        0.000
       VP          102        0.500        0.529        0.514
   WHADJP            6        1.000        1.000        1.000
   WHADVP            2        0.500        1.000        0.667
     WHNP           42        0.929        0.929        0.929
     WHPP            2        1.000        1.000        1.000

    total          763        0.626        0.689        0.656
```

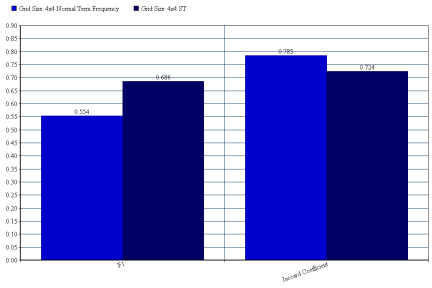
Αρχικά θα χρησιμοποιήσουμε ένα συντακτικό βάρος με s=1, δλδ τα υποδέντρα μπορεί να αποτελούνται μόνο από pre-pre-terminals χωρίς να έχουν καμία συγκεκριμένη λέξη από την πρόταση. Το σώμα δεδομένων που θα χρησιμοποιήσουμε είναι διαφορετικό από αυτό που είχαμε παραπάνω.



Από τα αποτελέσματα των μετρικών βλέπουμε ότι δεν βελτιώσαμε την ομαδοποίηση. Αυτό σημαίνει ότι για την ομαδοποίηση με χρήση snippet τα συντακτικά χαρακτηριστικά τους προσθέτουν πληροφορία που δεν έχει να κάνει με το θέμα του snippet.

Παρακάτω βλέπουμε τα αποτελέσματα για υποδεντρα που δεν έχουν μόνο pre-terminals ομως περιέχουν και λέξεις. Η βελτίωση που προσέχουμε έχει να κάνει με το ότι στην κανονική TF τα βάρη τείνουν να είναι 1 λόγω του μικρού μεγέθους των

snippet ενώ με την χρήση του ST τα βάρη διαφοροποιούνται ανάλογα με το βάθος του συνταχτικού δέντρου.



8.5 Επέκταση Ερωτήματος

Παραπάνω είπαμε ότι θα κρατήσουμε τις πρώτες 3 λέξεις του centroid για χρήση στην ονομασία τις κάθε κλάσης και την επέκταση του ερωτήματος. Σε ομοιογενείς αναζητήσεις τα αποτελέσματα που θα είχαμε αν χρησιμοποιούσαμε απλά αυτές τις 3 λέξεις για την επέκταση και ονομασία δεν θα ήταν τόσο καθαρά όσο αυτά που φαίνονται στην απεικόνιση των 5 αναζητήσεων σε ένα grid.

Οι πρώτες 3 λέξεις σε αυτές τις περιπτώσεις συμβαίνει συχνά να μην έχουνε μεγάλη σημασία στα μάτια του χρήστη.

Για να παρακάμψουμε αυτό το γεγονός αποφασίσαμε να χρησιμοποιήσουμε μια μετρική "σημαντικότητας" που θα επιλέγει μερικές λέξεις που έχουνε μεγάλη σημαντικότητα, από το σύνολο των 20 λέξεων που εμφανίζουν τα μεγαλύτερα βάρη στον centroid. Την μετρική αυτή θα την υπολογίσουμε με χρήση της έννοιας της "keyphraseness" [4].

Η υλοποίηση του αλγορίθμου για τον υπολογισμό εκτελείται σε αρκετά μικρό χρόνο για μια λέξη, όμως εμείς θα χρειαστούμε την keyphraseness όλων των λέξεων του σώματος που είναι ~100. Για να επιταχύνουμε τον παραπάνω αλγόριθμο θα υπολογίζουμε την keyphraseness κάθε 50 λέξεων παράλληλα και για να κρατήσουμε την σταθερότητα του συστήματος αυτό το στάδιο θα εκτελείτε στον απομακρυσμένο server.

Στο τέλος της παραπάνω διαδικασίας θα έχουμε μερικές λέξεις που θα δίνουνε μια ιδέα για την ομοιότητα ανάμεσα σε γειτονικά clusters. Αυτές τις λέξεις μπορούμε τώρα να τις να εμπλουτίσουμε περαιτέρω με χρήση των συχνών αναζητήσεων που μας δίνει η Google και να δώσουμε στον χρήστη την δυνατότητα να επεκτείνει περαιτέρω το ερώτημα του ανακτώντας αποτελέσματα που διατηρούν κατά κάποιον τρόπο την σημασιολογική έννοια του cluster.

```
physics generation
physics research
generation research
generation faculty
physics faculty
research faculty
Array
(
    [0] =>  generation undergraduate scholarships
    [1] =>  generation undergraduate
    [2] =>  generation undergraduate students
    [3] => undergraduate phd
    [4] => program
    [5] => student
    [6] => comics
    [7] => phd undergraduate gpa
    [8] => undergraduate masters phd
    [9] => undergraduate postgraduate phd
    [10] => undergraduate graduate phd
    [11] => undergraduate  phd
    [12] => undergraduate dating phd
    [13] => phd generation
    [14] =>  generation phd
    [15] =>  student
    [16] => 3rd generation phd
    [17] =>  fellowship
    [18] => distributed generation phd
    [19] => hot generation phd
    [20] => mesh generation phd
    [21] => procedural generation phd
```

Για τον όρο αναζήτησης "physics", ο παραπάνω αλγόριθμος συμπεραίνει ότι μια από τις κλάσεις που βρέθηκαν έχουν τις λέξεις υψηλής keyphraseness "physics,generation,research,faculty" και χαρακτηριστικά τοπικά stems "inform event research", το οποίο μπορεί να μας βοηθήσει να συμπεράνουμε ότι οι σύνδεσμοι που έχουν ταξινομηθεί σε αυτή την κλάση θα έχουν να κάνουν με έρευνα σε πανεπιστήμια. Αν ακολουθήσουμε τους συνδέσμους θα δούμε ότι είναι οι ιστοσελίδες μεταπτυχιακών και προπτυχιακών τμημάτων.

http://www.princeton.edu/physics/
http://www.physics.utoronto.ca/
http://www.pa.ucla.edu/
http://physics.uchicago.edu/
http://www.tcd.ie/Physics/

Με χρήση των λέξεων υψηλής keyphraseness η διαδικασία εμπλουτισμού με όρους συχνής αναζήτησης αποφάνθηκε ότι ίσως ο χρήστης να θέλει να χρησιμοποιήσει τον όρο "phd" εάν θέλει να πάρει αποτελέσματα που να αφορούν αυτή την κλάση.

Enrichment results: generation, undergraduate, letter, phd, jobs, research, faculty.

Περαιτέρω χρήση των λέξεων του centroid θα γίνει για την ταξινόμηση γειτονικών cluster σε hyperclusters. Όταν δυο clusters έχουν μια μεγάλη επικάλυψη των σημαντικότερων λέξεων του centroid τους τότε ταξινομούνται κάτω από τον ίδιο hypercluster

και τους δίνεται το ίδιο χρώμα στο φόντο.

8.6 Ονομασία κλάσης

Για την ονομασία κλάσης αποφασίσαμε ότι είναι πιο κατανοητό στον χρήστη η χρήση ονοματικών φράσεων(Noun Phrase, NP) και επώνυμων οντοτήτων(Named Entities).

Μια ονοματική φράση (συντομογραφία NP), είναι μια φράση που έχει ένα ουσιαστικό (ή αόριστη αντωνυμία) ως πρώτη λέξη, ή εκτελεί την ίδια γραμματική λει-

τουργία ως μια τέτοια φράση.

This ***sentence*** contains *two noun* ***phrases***.

Οι επώνυμες οντότητες είναι στοιχεία των προτάσεων που μπορούν να ταξινομηθούν σε προκαθορισμένες κατηγοριες όπως τα ονόματα των προσώπων, οργανισμών και περιοχών.

Jim bought 300 shares of **Acme Corp.** in 2006.

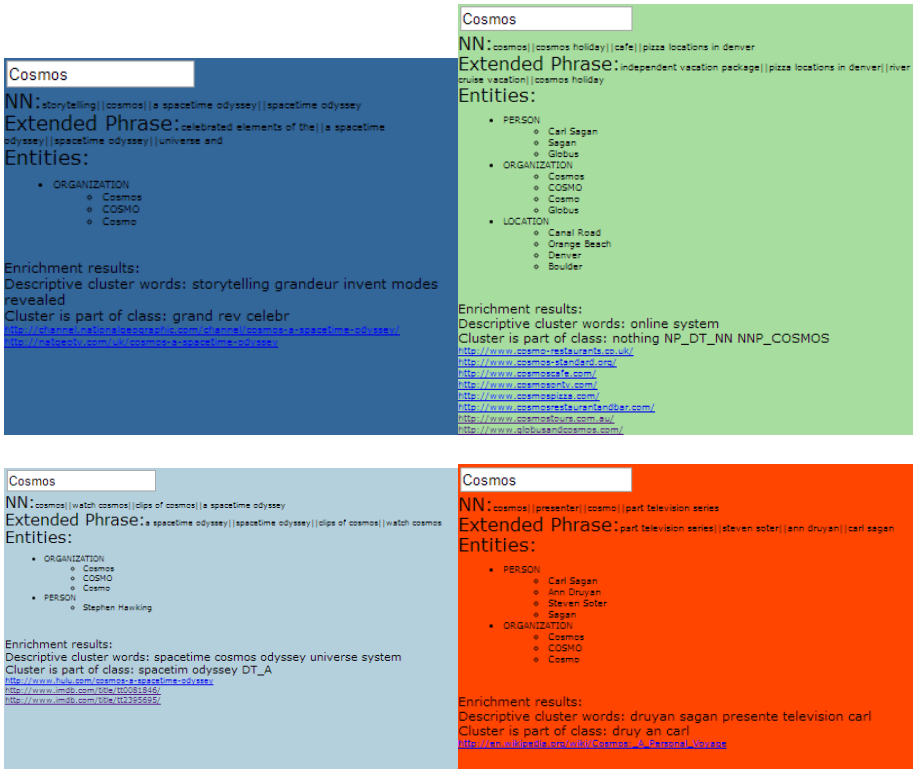
[Jim] Person

[Acme Corp.]Organization.

Η εξόρυξη των παραπάνω δεδομένον από το κείμενο των snippet είναι μια σχετικά χρονοβόρα διαδικασία ταξινομησης που υλοποιήθηκε με χρήση έτοιμων βιβλιοθηκών DKpro[3], και δεδομένου ότι πρέπει να απαντήσουμε στον χρήστη σε λογικούς χρόνους αυτό μας ανάγκασε να αλλάξουμε λίγο την ροή του προγράμματος.

Δεδομένου ότι οι NP και NE δεν θα χρησιμοποιηθούν σε κανένα σημείο από την αναζήτηση μέχρι την αναπαράσταση των αποτελεσμάτων καταλάβαμε ότι για να μην αναγκάζουμε τον χρήστη να περιμένει περισσότερο χρόνο για την εξόρυξη θα μπορούσαμε να τρέξουμε την επεξεργασία παράλληλα και να αναθέσουμε ανάλογα της ονοματικές φράσεις στους clusters στους όποιους τυχαίνει να έχει ομαδοποιήθει το κάθε snippet που τις περιλαμβάνει. Για να επιλέξουμε τις πιο σημαντικές φράσεις χρησιμοποιούμε την ίδια διαδικασία που παραπάνω υπολογίζει την keyphraseness για την επέκταση ερωτήματος.

Για την αναζήτηση του όρου "Cosmos" και ανάλυση των αποτελεσμάτων με συνταχτικά βάρη ST συμπεράνουμε τους παρακάτω clusters.



[3] https://code.google.com/p/dkpro-core-asl/wiki/DKProGroovyCookbook

Βλέπουμε ότι για τους παραπάνω clusters έχουμε 3 clusters που αφορούν τις σειρές ντοκιμαντέρ και ένα cluster που αφορά πακέτα διακοπών, restaurants και cafe. Σε αυτόν το cluster η φράσεις που έχουν επιλεχτεί είναι "independent vacation package, pizza locations in denver". Από τα υπόλοιπα clusters 2 έχουν να κάνουν με την καινούργια σειρά και είναι ταξινομημένοι σε περιεχόμενο που αφορά αποτελέσματα video σε imdb και hulu και σε επισκοπησεις της προβολής του show στο nationalgeographic. Το τρίτο cluster έχει να κάνει με ένα άρθρο στην wikipedia για την παλιά σειρά ντοκιμαντέρ οπως μπορούμε να δούμε από την φράση και τις επώνυμες οντότητες.

9 Συμπεράσματα και βελτιώσεις

Ο στόχος της εργασίας ήταν να μπορέσουμε να δώσουμε στον χρήστη περισσότερη πληροφορία για τα αποτελέσματα της αναζήτησης του και να του προσφέρουμε την δυνατότητα να επεκτείνει το ερώτημα του με ελαχίστη προαπαιτούμενη γνώση του αντικειμένου, ονομάζοντας τις ομάδες των αποτελεσμάτων και δίνοντας υποδείξεις επέκτασης.

Το πρώτο μέρος του στόχου μας θεωρείται ότι έχει επιτευχθεί με την ομαδοποίηση και αναπαράσταση των αποτελεσμάτων σε clusters όπου ο χρήστης μπορεί να καταλάβει την ομοιότητα των αποτελεσμάτων από την θέση τους. Η ολοκλήρωση του δευτέρου μέρους του στόχου μας αποδείχθηκε να είναι πιο δύσκολη χωρείς την χρήση εξωτερικών δομημένων Πληροφοριών. Παρόλαυτά με χρήση της Wikipedia, του Stanford Parser και των υποδείξεών της Google πιστεύουμε ότι ολοκληρώσαμε σε κάποιο βαθμό και την ονομασία cluster και επέκταση ερωτήματος.

Κατά την ολοκλήρωση της παρούσας εργασίας παρατηρήσαμε μερικά σημεία όπου μπορούμε να βελτιώσουμε η αλλάξουμε την προσέγγιση μας. Το το κύριο σημείο που θα επέφερε της μεγαλύτερες αλλαγές/βελτιώσεις είναι φυσικά το σώμα των δεδομένων. Σε μερικές περιπτώσεις τα αποτελέσματα της αναζήτησης δεν παρουσιάζουν αρκετή ανομοιογένεια για να παρουσιάζουν εύκολα διαπιστωτεες κλάσεις που να έχουν νόημα για τον ανθρώπινο χρήστη. Φυσικά δεν έχουμε την δυνατότητα να συγκεντρώσουμε μια δικιά μας βάση δεδομένων για αρκετά μεγάλο κομμάτι του διαδικτύου ώστε να μπορούμε να απαντήσουμε τυχαία ερωτήματα όπως η Google. Θα μπορούσαμε όμως να χρησιμοποιήσουμε το advanced search api της goolge ώστε να αλλάζουμε κάθε απλή αναζήτηση του χρήστη σε πιο περίπλοκες αναζητησεις περιορισμένες κατά domain η κατά την ύπαρξη η απουσία μερικών λέξεων. Τα αποτελέσματα αυτών των πολλαπλών αναζητήσεων μετά θα μπορούν να συγχωνευτούν και να χρησιμοποιηθούν σαν το καινούργιο corpus.

Η διαδικασία ομαδοποίησης εξαρτάται από τον αλγόριθμο SOM όμως μπορούμε να βελτιώσουμε περαιτέρω τα αποτελέσματα της με τον εμπλουτισμό των snippets με προτάσεις που κάνουν χρήση συνωνύμων. Τα συνώνυμα μπορούν να βρεθούν χρησιμοποιώντας WordNet[4].

Η ονομασία των κλάσεων μπορεί να βελτιωθεί με την πρόσθεση ontology lookup. Η υλοποίησης μπορεί να γίνει με χρήση κάποιας online υπηρεσίας η να χρησιμοποιήσουμε την προσέγγιση που βλέπουμε στην FREyA[5], ένα σύστημα μετάφρασης ερωτηματων σε SPARQL για μια φιλική διεπαφή με το Semantic Web και για την απάντηση περίπλοκων ερωτημάτων. Η freya μεταχειρίζεται ένα υποσύνολο οντολογιον για να αποφασίσει αν μια λεξη της ερωτησης αποτελεί ένα "ontology consept". Οι οντολογιες είναι αποθηκευμένες σε ένα ευρετήριο Lucene[5] που υποστηρίζει πολύ γρήγορες αναζητήσεις με εφαρμογές stemming και περίπλοκων ερωτημάτων σε αρχεία αρκετά μεγάλου μεγέθους. Θα μπορούσαμε λοιπόν να κάνουμε χρήση ενός τέτοιου ευρετηρίου για να επιλέξουμε πιο σωστές ονομασίες για τις κλάσεις(HC) βρίσκοντας τις κείνες SuperClasses ανάμεσα στα clusters.

[4] http://wordnet.princeton.edu/
[5] https://github.com/deepgraphs/pyFreya/

References

[1] Halgarmuge S.K. Srinivasan B. Alahakoon, D. Dynamic self-organizing maps with controlled growth for knowledge discovery. *IEEE Transactions on Neural Networks*, 2000.

[2] E. Charniak. A maximum-entropy-inspired parser. *Technical Report CS-99-12*, 1999.

[3] S.E. Robertson C.J. van Rijsbergen and M.F. Porter. New models in probabilistic information retrieval. *British Library Research and Development Report, no. 5587*, 1980.

[4] Rada Mihalcea Andras Csomai. Wikify! linking documents to encyclopedic knowledge. *CIKM '07*, 2007.

[5] Danica Damljanovic Milan Agatonovic Hamish Cunningham. Freya: an interactive way of querying linked data using natural language. *gate.ac.uk*, 2012.

[6] Susan T. Dumais. Latent semantic analysis. *Annual Review of Information Science and Technology*, 2005.

[7] C. L. Fowlkes, E. B.; Mallows. A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association*, 1983.

[8] B. Fritzke. Growing grid a self-organizing network with constant neighborhood range and adaption strength. *Neural Processing Letters*, 1995.

[9] ERIC GOLDMAN. Search engine bias and the demise of search engine utopianism. *YALE JOURNAL OF LAW AND TECHNOLOGY*, 2006.

[10] M. Steinbach & V. Kumar. *Introduction to Data Mining.* Addison-Wesley, 2005.

[11] Moschitti. Making tree kernels practical for natural language learning. *EACL*, 2006.

[12] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 1971.

[13] Karen Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 1972.

[14] CLAUDIO CARPINETO STANISlAW OSINSKI GIOVANNI ROMANO DAWID WEISS. A survey of web clustering engines. *ACM Computing Surveys*, 2009.