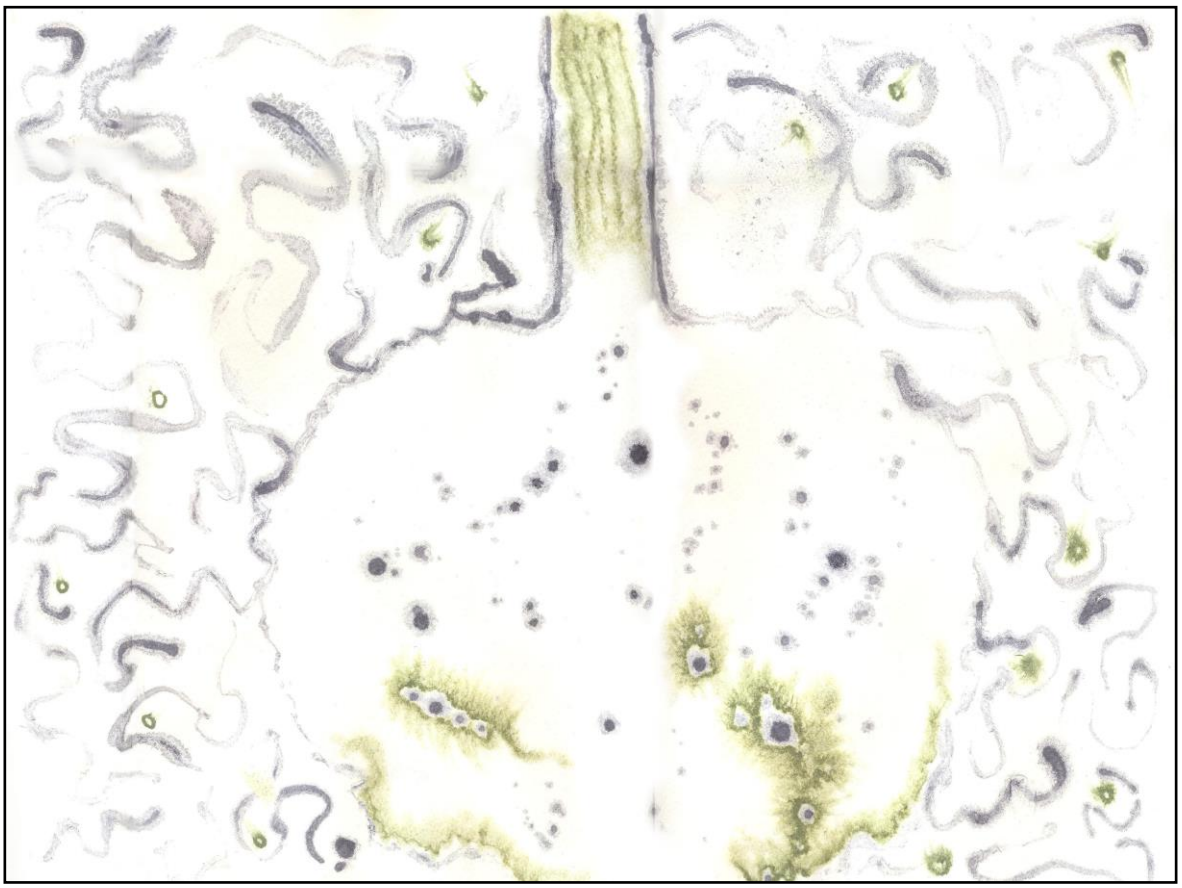
ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή εργασία

«Οπτικοποίηση ενός μεγάλου συνόλου δεδομένων με τη χρήση κατάλληλων εργαλείων»



Του φοιτητή
Λαζαρίδη Συμεών
Αρ.Μητρώου: 05/2784

Επιβλέπων καθηγητής
Βοζαλής Εμμανουήλ

ΠΡΟΛΟΓΟΣ

Η παρούσα πτυχιακή εργασία ασχολείται με το ζήτημα της οπτικοποίησης δεδομένων. Άπτεται επίσης θεμάτων όπως η ανάκτηση και εξόρυξη πληροφορίας, τα γραφικά, η ανάλυση κειμένου και ο προγραμματισμός.

Εκπονήθηκε στη Θεσσαλονίκη το 2011 ως προαπαιτούμενο της αποφοίτησής μου από το τμήμα πληροφορικής του αλεξάνδρειου τεχνολογικού εκπαιδευτικού ιδρύματος της Θεσσαλονίκης (ΑΤΕΙΘ).

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή αυτής της εργασίας Βοζαλή Εμμανουήλ και όλους τους ανθρώπους που με δίδαξαν, όσο φοίτησα στο τμήμα πληροφορικής του ΑΤΕΙΘ, με τη στάση τους ως καθηγητές αλλά και ως άνθρωποι. Θα ήθελα να ευχαριστήσω ιδιαίτερα τους κυρίους: Σταμάτη Δημοσθένη, Κότσιαλο Ευθύμιο, Μάργαρη Αθανάσιο, Γουλιάνα Κωνσταντίνο, Βαφειάδη Αντώνιο και Κατωπόδη Κωνσταντίνο, ως ελάχιστη ένδειξη ευγνωμοσύνης για ό,τι προσέφεραν σε εμένα και τους συμφοιτητές μου. Τέλος, θέλω να ευχαριστήσω τους φίλους και τις φίλες με τους οποίους και τις οποίες έζησα τον τελευταίο καιρό.

Αφιερώνω την εργασία στους γονείς μου Θεόδωρο Λαζαρίδη και Βασιλική Κελέκη, οι αξίες, ο κόπος και η αγάπη των οποίων, είχαν και εξακολουθούν να έχουν καταλυτικό ρόλο στη διαμόρφωσή μου ως άνθρωπο και κατ’ επέκταση σε ό,τι κάνω.

ΠΕΡΙΛΗΨΗ

Η δυνατότητα των υπολογιστών για αποθήκευση δεδομένων είναι απεριόριστη. Αυτό έχει ως αποτέλεσμα να έχουμε στη διάθεση μας όσα δεδομένα θελήσουμε, αλλά ταυτόχρονα προκύπτει το πρόβλημα της μετατροπής τους σε αξιοποιήσιμη πληροφορία και γνώση. Πολλές φορές, αυτή η πληροφορία είναι θεμιτό να γίνει αντιληπτή άμεσα και γρήγορα. Η οπτικοποίηση των δεδομένων είναι ο τρόπος να εκμεταλλευτούμε την ισχυρότερη ανθρώπινη αίσθηση για να πετύχουμε το σκοπό μας. Η οπτικοποίηση δεδομένων είναι η γραφική αναπαράσταση αυτών, με σκοπό να αποκαλύψει την περίπλοκη πληροφορία με μια ματιά .

Σε αυτήν την εργασία επιχειρείται η οπτικοποίηση του ερευνητικού πεδίου τεσσάρων καθηγητών του ΑΤΕΙ Θεσσαλονίκης. Παρουσιάζεται η γενική ιδέα της οπτικοποίησης όπως και κάποιες εφαρμογές πραγματικών παραδειγμάτων. Για την εξόρυξη της πληροφορίας χρησιμοποιείται η απόδοση tf-idf βαρών στις λέξεις ενός μεγάλου όγκου κειμένων για την εξαγωγή των σημαντικών εννοιών. Ο σχηματισμός ενός λεξικού με έννοιες ενδιαφέροντος είναι απαραίτητος. Η καταμέτρηση των εμφανίσεων τους, γίνεται με την συγγραφή ενός προγράμματος σε Java. Τέλος ο πίνακας διανυσμάτων που δημιουργείται εισάγεται στον VOS viewer που είναι ένα πρόγραμμα για δημιουργία χαρτών που οπτικοποιούν κάποιο δεδομένο γνωσιακό πεδίο.

ABSTRACT

The capability of computers to store data is unlimited. As a result, we have at our service as much data as we want, but at the same time emerges the problem of converting them in useful information and knowledge. There are many cases where this information is required to be perceived directly and quickly. The visualization of data is the way to take advantage of the strongest human sense, in order to achieve our goal. Data visualization is the graphical representation of data, aiming to reveal the complex information at a glance.

In this thesis, the visualization of the research field of four professors of ATEI of Thessaloniki is attempted. The general idea of visualization, as well as the

implementations of some real examples, is also presented. For the data mining, the tf-idf weighting scheme is used upon the words of a corpus, for the extraction of the important concepts. The creation of a thesaurus including the interesting concepts, is compulsory. The counting of their occurrences is done by writing a program in Java. Finally, the vector array which is created, together with the thesaurus, forms the input for the VOS viewer, which is a program for creating knowledge domain visualization maps.

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΡΟΛΟΓΟΣ 2
ΕΥΧΑΡΙΣΤΙΕΣ 2
ΠΕΡΙΛΗΨΗ 3
ABSTRACT 3
ΠΕΡΙΕΧΟΜΕΝΑ 5
ΕΥΡΕΤΗΡΙΟ ΕΙΚΟΝΩΝ 7
ΚΕΦΑΛΑΙΟ 1 – ΕΙΣΑΓΩΓΗ 9
1.1 Ορισμός του Προβλήματος 9
1.2 Στόχοι της Εργασίας 9
1.3 Δομή πτυχιακής εργασίας 10
ΚΕΦΑΛΑΙΟ 2 – ΘΕΩΡΗΤΙΚΟ ΚΑΙ ΠΡΑΚΤΙΚΟ ΥΠΟΒΑΘΡΟ 11
2.1 Εισαγωγή-Ορισμός 11
2.2 Βήματα οπτικοποίησης 12
2.3 Θετικά – γιατί οπτικοποίηση; 13
2.4 Κίνδυνοι – συμβατικές (τυποποιημένες) μορφές οπτικοποίησης 14
2.5 Ανάλυση κειμένου (δημιουργία λεξικού – απόδοση tf-idf βαρών στις λέξεις) 15
2.6 Εργαλεία 17
2.6.1 Εργαλεία Οπτικοποίησης 17
2.6.1.1 VOS viewer (VOS viewer, 2011) 17
2.6.1.2 Processing 19
2.6.1.3 Prefuse (Prefuse, 2011) 20
2.6.1.4 Graphviz 22
2.6.2 Εργαλεία Ανάλυσης Κειμένου 24
2.6.3 Άλλα εργαλεία 28
2.7 Παραδείγματα 29
2.7.1 Wordle 29
2.7.2 Flight patterns 30
2.7.3 Χάρτες Μετρό 31
2.7.4 Οπτικοποίηση πρόσβασης στις ιστοσελίδες nytimes.com και mobile.nytimes.com 33
2.7.5 Yellow pages και Netflix 36
2.8 Αντικείμενο Πτυχιακής: Οπτικοποίηση γνωσιακού πεδίου 40
2.9 Επίλογος 41
ΚΕΦΑΛΑΙΟ 3 - ΑΝΑΛΥΣΗ ΚΑΙ ΣΧΕΔΙΑΣΜΟΣ 43
3.1 ΕΙΣΑΓΩΓΗ 43
3.2 Η πορεία που ακολουθεί η δημοσίευση “Visualizing the WCCI 2006 Knowledge Domain” 43
3.2.1 Σύντομη αναφορά στα βήματα 44
3.2.2 Σημεία - κλειδιά αναλυτικά 46
3.2.2.1 Λεξικό (thesaurus) του πεδίου CI 46
3.2.2.2 Βάρος συσχετίσεως (association strength) 47
3.2.2.3 VOS 48
3.3 Ανάλυση των βημάτων της παρούσας εργασίας 52
3.3.1 Περιληπτικά 52
3.3.2 Αναλυτικά 53
3.3.2.1 Συλλογή δεδομένων 53
3.3.2.2.Αντικείμενο προς ανάλυση / στοιχεία-κλειδιά 53

3.3.2.3 Η δημιουργία του λεξικού 54
3.3.2.4 Ο αλγόριθμος που εκτελεί το πρόγραμμα για τη δημιουργία των διανυσμάτων 60
3.3.2.5 Μια πρώτη οπτικοποίηση 63
3.5 Επίλογος 64
ΚΕΦΑΛΑΙΟ 4 - ΥΛΟΠΟΙΗΣΗ- ΑΠΟΤΕΛΕΣΜΑΤΑ ΧΡΗΣΗΣ- ΑΠΟΤΙΜΗΣΗ 65
4.1 Εισαγωγή 65
4.2 Υλοποίηση στα κομβικά σημεία (από τα 106 abstracts στη βασική εικόνα) 65
4.2.1 Πειράματα – κατασκευη θησαυρών 65
4.2.1.1 Το βασικό πείραμα 66
4.2.1.2 Παραλλαγές πειραμάτων (λίστα μεγέθους 50,150,300,200,1000 λέξεων) 68
4.2.1.3 Βελτίωση θησαυρών (θησαυροί 100 και 150 λέξεων χωρίς άσχετες με την πληροφορική λέξεις) 71
4.2.2 Κώδικας σε Java 73
4.2.2.1 Η κλάση IntVector 73
4.2.2.2 Η μέθοδος fileLister 74
4.2.2.3 Η μέθοδος exists 75
4.2.2.4 Η μέθοδος occurenciesFill 77
4.2.2.5 Η μέθοδος arrayTransformer 77
4.3 Τί παρατηρούμε; 78
4.3.1 Συγγένειες 78
4.3.2 Ομαδοποίηση (Clustering) (ή αλλιώς συσταδοποίηση) 80
4.3.3 Κοινές λέξεις 85
4.4 Παράμετροι και όψεις στον VOS 87
4.4.1 Label view 87
4.4.2 Density view 89
4.4.3 Cluster Density view 92
4.4.4 Scatter View 95
4.5 Αξιολόγηση – Αποτίμηση αποτελεσμάτων 98
4.6 Επίλογος 99
ΚΕΦΑΛΑΙΟ 5 - ΣΥΜΠΕΡΑΣΜΑΤΑ – ΠΡΟΤΑΣΕΙΣ ΓΙΑ ΜΕΛΛΟΝΤΙΚΗ ΕΡΓΑΣΙΑ 100
5.1 Εισαγωγή 100
5.2 Συμπεράσματα 100
5.2.1 Το αρχικό πρόβλημα 100
5.2.2 Σύντομη αναφορά στην αντιμετώπιση 101
5.2.3 Η θέαση του αποτελέσματος 101
5.3 Προτάσεις για Μελλοντική Εργασία 102
5.3.1 Νέος VOS viewer 102
5.3.2 Οπτικοποίηση με διαφορετικό εργαλείο - Εισαγωγή διάστασης χρόνου 111
5.3.3 Ανάπτυξη διεπαφής για τη δημιουργία των διανυσμάτων 111
5.3.4 Αλλαγή των ποσοστών συμμετοχής των καθηγητών – παραπάνω καθηγητές-πλήρεις δημοσιεύσεις. 112
5.4 Επίλογος 113
ΒΙΒΛΙΟΓΡΑΦΙΑ 114
ΠΑΡΑΡΤΗΜΑ Α’ 116
ΠΑΡΑΡΤΗΜΑ Β’ 128
ΠΑΡΑΡΤΗΜΑ Γ’ 138

ΕΥΡΕΤΗΡΙΟ ΕΙΚΟΝΩΝ

Εικονα 1 παραδειγμα του «wordle» με stop words στα αγγλικα 16
Εικονα 2 Στιγμιοτυπο λειτουργιας του VOS viewer 18
Εικονα 3 Έξοδος του VOS viewer για το ερευνητικο προφιλ του πανεπιστημιου του Leiden 19
Εικονα 4 Προγραμματιστικο περιβαλλον στο processing 20
Εικονα 5 Παραδειγμα εξοδου του Prefuse 22
Εικονα 6 Παραδειγμα οπτικοποιησης με το GraphViz 23
Εικονα 7 Γραφος φτιαγμενος με το GraphViz 24
Εικονα 8 Στιγμιοτυπο εκτελεσης του montyLingua απο τη γραμμη εντολων 26
Εικονα 9 Εξαγωγη λεξεων με το rapid miner 27
Εικονα 10 Παραδειγματα του wordle 30
Εικονα 11 Οπτικοποιηση flight patterns με τις πτησεις των ΗΠΑ και του Καναδα 30
Εικονα 12 Χαρτης του υπογειου δικτυου του Μετρο της Νεας Υορκης 32
Εικονα 13 Παλιος χαρτης του Μετρο του Λονδινου 33
Εικονα 14 Χαρτης του μετρο της Βαρκελωνης 33
Εικονα 15 Χαρτης με τις πολεις του κοσμου 34
Εικονα 16 Χαρτης με τις πολεις των οποιων οι χρηστες επισκεφθηκαν το nytimes.com και το mobile.nytimes.com 35
Εικονα 17 Στιγμιοτυπο απο το βιντεο με την εξελιξη των επισκεψεων παγκοσμιως στο nytimes.com και το mobile.nytimes.com 36
Εικονα 18 Οπτικοποιηση του yellowpages.com 39
Εικονα 19 Οπτικοποιηση του Netflix 40
Εικονα 20 Πινακας με τα βηματα της εργασιας (Van Eck et al., 2006) 44
Εικονα 21 Το εσωτερικο της διεργασιας επεξεργασιας κειμενου του rapid miner 58
Εικονα 22 Εισαγωγη του αρχειου “items” και του αρχειου “occurrences” στον VOS viewer. 63
Εικονα 23 Οπτικοποιηση με 100 λεξεις 64
Εικονα 24 Συνοπτικη αναπαρασταση της δημιουργιας του θησαυρου 67
Εικονα 25 Οπτικοποιηση βασισμενη σε θησαυρο 50 λεξεων 68
Εικονα 26 Οπτικοποιηση βασισμενη σε θησαυρο 150 λεξεων 69
Εικονα 27 Οπτικοποιηση βασισμενη σε λιστα 200 λεξεων 69
Εικονα 28 Οπτικοποιηση βασισμενη σε λιστα 300 λεξεων 70
Εικονα 29 Οπτικοποιηση βασισμενη σε λιστα 1000 λεξεων 71
Εικονα 30 Οπτικοποιηση βασισμενη σε λιστα 100 διαλεγμενων λεξεων. 72
Εικονα 31 Οπτικοποιηση βασισμενη σε λιστα 150 διαλεγμενων λεξεων. 72
Εικονα 32 Συνοπτικη αναπαρασταση του αλγοριθμου συμπληρωσης του πινακα διανυσματων 75
Εικονα 33 Οπτικοποιηση με 1000 λεξεις 79
Εικονα 34 Ομαδοποιηση στην οπτικοποιηση των 100 λεξεων 80
Εικονα 35 Οπτικοποιηση 100 λεξεων με πλαισια που περικλειουν το συνολο των λεξεων καθε ομαδας. 81
Εικονα 36 Οπτικοποιηση με 100 λεξεις και περιγραφη της βασικης κατανομης. 82
Εικονα 37 Οπτικοποιηση 300 λεξεων με πλαισια που περικλειουν το συνολο των λεξεων καθε ομαδας. 83
Εικονα 38 Οπτικοποιηση με 300 λεξεις και περιγραφη της βασικης κατανομης 83
Εικονα 39 Οπτικοποιηση 300 λεξεων με προσαρμοσμενα πλαισια που περιλαμβανουν ολους τους ορους καθε ομαδας 84
Εικονα 40 Οπτικοποιηση 100 διαλεγμενων λεξεων με πλαισια που περικλειουν το συνολο των λεξεων καθε ομαδας. 85
Εικονα 41 Συμπτωσεις των λεξεων performance, model, value κ.α στην οπτικοποιηση των 300 λεξεων 86
Εικονα 42 Συμπτωσεις των λεξεων pca, system, independent κ.α στην οπτικοποιηση των 300 λεξεων 87
Εικονα 43 Οπτικοποιηση με 100 διαλεγμενες λεξεις, με ετικετες και πλαισια αντι για κυκλους. 88
Εικονα 44 Οπτικοποιηση με 100 διαλεγμενες λεξεις σε πλαισια, με επιλογη που επιτρεπει επικαλυψεις 89
Εικονα 45 Οπτικοποιηση 150 διαλεγμενων λεξεων σε density view. 90
Εικονα 46 Οπτικοποιηση 1000 λεξεων σε density view. 91
Εικονα 47 Οπτικοποιηση 1000 λεξεων σε density view με μειωμενο ευρος πυρηνων (kernel

WIDTH). 92
ΕΙΚΟΝΑ 48 ΟΠΤΙΚΟΠΟΙΗΣΗ 200 ΛΕΞΕΩΝ ΣΕ CLUSTER DENSITY VIEW. 93
ΕΙΚΟΝΑ 49 ΟΠΤΙΚΟΠΟΙΗΣΗ 200 ΛΕΞΕΩΝ ΜΕ ΜΕΙΩΜΕΝΟ KERNEL WIDTH. 94
ΕΙΚΟΝΑ 50 ΟΠΤΙΚΟΠΟΙΗΣΗ 50 ΛΕΞΕΩΝ ΣΕ CLUSTER DENSITY VIEW. ΔΕΝ ΥΠΑΡΧΕΙ ΑΝΑΓΚΗ ΝΑ ΜΕΙΩΘΕΙ ΤΟ KERNEL WIDTH. ΟΙ ΟΡΟΙ ΕΧΟΥΝ ΧΩΡΟ ΝΑ ΕΚΦΡΑΣΤΟΥΝ ΧΡΩΜΑΤΙΚΑ ΧΩΡΙΣ ΕΠΙΚΑΛΥΨΕΙΣ. 95
ΕΙΚΟΝΑ 51 ΟΠΤΙΚΟΠΟΙΗΣΗ 1000 ΛΕΞΕΩΝ ΣΕ SCATTER VIEW ΚΑΙ SIMILARITY MEASURE 1. 96
ΕΙΚΟΝΑ 52 ΟΠΤΙΚΟΠΟΙΗΣΗ 1000 ΛΕΞΕΩΝ ΣΕ SCATTER VIEW ΚΑΙ SIMILARITY MEASURE 2 97
ΕΙΚΟΝΑ 53 ΟΠΤΙΚΟΠΟΙΗΣΗ 1000 ΛΕΞΕΩΝ ΣΕ SCATTER VIEW ΚΑΙ SIMILARITY MEASURE 3. 98
ΕΙΚΟΝΑ 54 1Ο ΒΗΜΑ ΟΠΤΙΚΟΠΟΗΣΗΣ ΜΕ ΤΟΝ VOS VIEWER 1.4.0 104
ΕΙΚΟΝΑ 55 ΣΤΙΓΜΙΟΤΥΠΟ ΤΗΣ ΟΠΤΙΚΟΠΟΙΗΣΗΣ ΜΕ ΤΟΝ VOS VIEWER 1.4.0 105
ΕΙΚΟΝΑ 56 2Ο ΒΗΜΑ ΟΠΤΙΚΟΠΟΙΗΣΗΣ ΜΕ ΤΟΝ VOS VIEWER 1.4.0 106
ΕΙΚΟΝΑ 57 3Ο ΒΗΜΑ ΟΠΤΙΚΟΠΟΙΗΣΗΣ ΜΕ ΤΟΝ VOS VIEWER 1.4.0 107
ΕΙΚΟΝΑ 58 4Ο ΒΗΜΑ ΟΠΤΙΚΟΠΟΙΗΣΗΣ ΜΕ ΤΟΝ VOS VIEWER 1.4.0 108
ΕΙΚΟΝΑ 59 5Ο ΒΗΜΑ ΟΠΤΙΚΟΠΟΙΗΣΗΣ ΜΕ ΤΟΝ VOS VIEWER 109
ΕΙΚΟΝΑ 60 ΟΠΤΙΚΟΠΟΙΗΣΗ ΣΤΟΝ ΝΕΟ VOS VIEWER ΜΕ ΚΑΤΩΦΛΙ 2 109
ΕΙΚΟΝΑ 61 ΟΠΤΙΚΟΠΟΙΗΣΗ ΣΤΟΝ ΝΕΟ VOS VIEWER ΜΕ ΚΑΤΩΦΛΙ 4 110
ΕΙΚΟΝΑ 62 ΟΠΤΙΚΟΠΟΙΗΣΗ ΣΤΟΝ ΝΕΟ VOS VIEWER ΜΕ ΚΑΤΩΦΛΙ 8 110
ΕΙΚΟΝΑ 63 ΟΠΤΙΚΟΠΟΙΗΣΗ ΣΤΟΝ ΝΕΟ VOS VIEWER ΜΕ ΚΑΤΩΦΛΙ 10 110
ΕΙΚΟΝΑ 64 LABEL VIEW 138
ΕΙΚΟΝΑ 65 DENSITY VIEW 139
ΕΙΚΟΝΑ 66 CLUSTER DENSITY VIEW 140
ΕΙΚΟΝΑ 67 SCATTER VIEW 141

ΚΕΦΑΛΑΙΟ 1 – ΕΙΣΑΓΩΓΗ

1.1 Ορισμός του Προβλήματος

Με τον όρο Οπτικοποίηση (Visualization) αναφερόμαστε στην προσπάθεια αναπαράστασης των δεδομένων χρησιμοποιώντας γραφικά, με στόχο την ανακάλυψη σύνθετων πληροφοριών, συσχετίσεων (associations) ή μοτίβων (patterns), τα οποία πιθανόν να μην είναι δυνατόν να γίνουν αντιληπτά με ευκολία, με διαφορετικό τρόπο.

Η συγκεκριμένη εργασία αποτελεί μια προσπάθεια οπτικοποίησης του ερευνητικού προφίλ του Α. Τεχνολογικού Εκπαιδευτικού Ιδρύματος Θεσσαλονίκης.

Το ερευνητικό προφίλ του ιδρύματος κωδικοποιείται μέσω των δημοσιεύσεων (σε συνέδρια ή περιοδικά) τεσσάρων καθηγητών που διδάσκουν σε αυτό. Στη συνέχεια, οι δημοσιεύσεις αναλύονται λεκτικά ώστε να επιλεγούν αντιπροσωπευτικές λέξεις - κλειδιά, και να σχηματιστεί τελικά ένα “λεξικό” (thesaurus) – μια λίστα από λέξεις δηλαδή που αναζητούνται στις δημοσιεύσεις των καθηγητών, και αργότερα νοηματοδοτούν το οπτικό σχήμα. Από τα επεξεργασμένα δεδομένα προκύπτει ένας όγκος αριθμητικών συμβολισμών, τον οποίο δέχονται ως είσοδο τα διάφορα εργαλεία για την τελική οπτική αποτύπωση. Στο τέλος, αξιολογούνται οι εικόνες που παράγονται και τα εργαλεία που χρησιμοποιήθηκαν κατά τη διαδικασία.

1.2 Στόχοι της Εργασίας

Στόχος της εργασίας είναι η αξιολόγηση της μεθόδου οπτικοποίησης που υλοποιείται, των εργαλείων που χρησιμοποιούνται και των συμπερασμάτων που μπορούν να προκύψουν παρατηρώντας τις εικόνες που κατασκευάζονται. Στόχος δηλαδή δεν είναι μόνο να μεταποιηθούν τα δεδομένα σε ένα οπτικό σχήμα, αλλά να αναλυθεί και όλη η διαδικασία που οδηγεί στην παραγωγή αυτού, η οποία είναι απαιτητική σε χρόνο υλοποίησης, σχεδιασμό και προσπάθεια, καθώς και η χρησιμότητα της παραχθείσας οπτικοποίησης.

1.3 Δομή πτυχιακής εργασίας

Η δομή που ακολουθεί το κείμενο της πτυχιακής εργασίας είναι η ακόλουθη:

Στο πρώτο κεφάλαιο ορίζεται το πρόβλημα και οι στόχοι της εργασίας.

Στο δεύτερο κεφάλαιο γίνεται ανάλυση του όρου «οπτικοποίηση» καθώς και πιο σύνθετων εννοιών, όπως «οπτικοποίση δεδομένων», «ανάλυση κειμένου», «οπτικοποίηση γνωσιακού πεδίου» και των υπόλοιπων θεωρητικών εργαλείων που χρειάζονται για την εκπόνηση της παρούσας εργασίας και την κατανόησή της. Αναφέρονται επίσης διάφορα παραδείγματα εφαρμογής της οπτικοποίησης και παρουσιάζονται εργαλεία που προσφέρονται για τους σκοπούς της.

Στο τρίτο κεφάλαιο παρουσιάζεται η δημοσίευση με τίτλο “Visualizing the WCCI 2006 Knowledge Domain” των Nees Jan van Eck, Ludo Waltman, Jan van den Berg, και Uzay Kaymak (Van Eck et al. 2006). Κατά την ίδια κατεύθυνση παρουσιάζονται τα βήματα που ακολουθούνται για τη δημιουργία της πρώτης εικόνας.

Στο τέταρτο κεφάλαιο αναλύονται κάποιες τεχνικές λεπτομέρειες που αφορούν τα διάφορα στάδια από την εξόρυξη των δεδομένων μέχρι την οπτικοποίησή τους. Παρουσιάζονται κάποιες αλλαγές που γίνονται πειραματικά και έχουν επίπτωση στην οπτικοποίηση και σχολιάζονται τα διάφορα αποτελέσματα.

Στο πέμπτο κεφάλαιο καταγράφονται τα συμπεράσματα σε σχέση με την αντιμετώπιση του προβλήματος και το αποτέσμα που επιτεύχθηκε, όπως επίσης προτάσεις και σκέψεις για μελλοντική βελτίωση της εργασίας.

ΚΕΦΑΛΑΙΟ 2 – ΘΕΩΡΗΤΙΚΟ ΚΑΙ ΠΡΑΚΤΙΚΟ ΥΠΟΒΑΘΡΟ

2.1 Εισαγωγή-Ορισμός

Πώς φαίνονται τα μονοπάτια που ακολουθούν εκατομμύρια χρηστών μέσα από ένα site; Πώς σχετίζονται τα 3.1 δισ γραμμάτων A, C, G, και T του ανθρώπινου γονιδιώματος με αυτό του χιμπατζή ή του ποντικιού; Στις εκατοντάδες χιλιάδες αρχεία του υπολογιστή μας, ποια καταλαμβάνουν περισσότερο χώρο και πόσο συχνά τα χρησιμοποιούμε; Εφαρμόζοντας μεθόδους από την επιστήμη των υπολογιστών, την στατιστική, τον εξόρυξη δεδομένων, το γραφικό σχεδιασμό και την οπτικοποίηση μπορούμε να αρχίσουμε να απαντάμε αυτές τις ερωτήσεις με ένα τέτοιο τρόπο, που να κάνει τις απαντήσεις προσβάσιμες στους άλλους.

Όλες οι προηγούμενες ερωτήσεις απαιτούν τεράστιο όγκο δεδομένων, πράγμα που κάνει δύσκολο το να αποκτηθεί η «μεγάλη εικόνα» του νοήματος. Το πρόβλημα περιπλέκεται περισσότερο, από τη συνεχώς εναλλασσόμενη φύση των δεδομένων, που προκαλείται από την προσθήκη καινούριας πληροφορίας ή την συνεχή εκλέπτυνση της παλαιότερης. Αυτός ο κατακλυσμός δεδομένων επιβάλει καινούργια εργαλεία βασισμένα σε λογισμικό, και η πολυπλοκότητά τους απαιτεί επιπλέον μελέτη. Όποτε αναλύουμε δεδομένα, ο στόχος μας είναι να τονίσουμε τα χαρακτηριστικά τους με σειρά ως προς την σημαντικότητά τους, να αποκαλύψουμε μοτίβα (patterns), και ταυτόχρονα να αναδείξουμε γνωρίσματα που υπάρχουν μέσω πολλαπλών διαστάσεων.

Πώς μπορούμε να απαντήσουμε αυτές τις ερωτήσεις γρήγορα, αν όχι αμέσως; Γινόμαστε τόσο καλοί στο να μετράμε και να καταγράφουμε στοιχεία, γιατί δεν συνεχίζουμε με μεθόδους κατανόησης και επικοινωνίας αυτής της πληροφορίας;

Σε αυτό το σημείο ακριβώς είναι που βοηθάει η οπτικοποίηση των δεδομένων. (Ben Fry, 2008)

·Οπτικοποίηση είναι η γραφική παρουσίαση των δεδομένων – αναπαραστάσεις που σκοπεύουν να αποκαλύψουν την περίπλοκη πληροφορία με μια ματιά .

·Οπτικοποίηση πληροφορίας, είναι η χρήση διαδραστικών οπτικών

αναπαραστάσεων αφηρημένων δεδομένων για ενίσχυση (διεύρυνση) της γνώσης.

·Η οπτικοποίηση πληροφορίας στην πραγματικότητα είναι εξωτερική γνώση, που σημαίνει, πώς κάποιες πηγές γνώσης που βρίσκονται εκτός του μυαλού μπορούν να χρησιμοποιηθούν για να ενισχύσουν τη γνωστική ικανότητά του ίδιου του μυαλού.(Colin Ware, 2004)

Αλλά η καλή οπτικοποίηση πληροφορίας ποτέ δε ξεκινά από τη θέση (standpoint) του συνόλου των δεδομένων (data set). Ξεκινά με ερωτήσεις. Γιατί συλλέχθηκαν τα δεδομένα; Τί ενδιαφέρον έχουν; Τί μπορούν να μας διηγηθούν;

Εν συντομία, μια σωστή οπτικοποίηση είναι ένα είδος αφηγήματος, που παρέχει ξεκάθαρη απάντηση σε ένα ερώτημα χωρίς περιττές λεπτομέρειες. Συγκεντρώνοντας την προσοχή στην αρχική πρόθεση της ερώτησης, είναι δυνατό να ελαχιστοποιήσουμε αυτές τις λεπτομέρειες γιατί η ερώτηση παρέχει ένα μέτρο σύγκρισης (benchmark) για το τί είναι απαραίτητο και τί όχι. (Ben Fry, 2008)

2.2 Βήματα οπτικοποίησης

Η διαδικασία κατανόησης των δεδομένων ξεκινά με ένα σύνολο αριθμών και μια ερώτηση. Τα ακόλουθα βήματα σχηματίζουν ένα δρόμο προς την απάντηση:

Απόκτηση (*Acquire*):

Πάρε τα δεδομένα, είτε από ένα αρχείο στο δίσκο είτε από μια πηγή στο δίκτυο.

Ανάλυση(*Parse*):

Όρισε κάποια δομή για το νόημα των δεδομένων, και ταξινόμησέ τα σε κατηγορίες

Φιλτράρισμα (*Filter*):

Αφαίρεσε ό,τι δεν αποτελεί δεδομένο ενδιαφέροντος.

Εξόρυξη(*Mine*):

Εφάρμοσε μεθόδους στατιστικής ή εξόρυξης πληροφορίας (data mining) για να διακρίνεις μοτίβα ή θέσε τα δεδομένα με μαθηματική έκφραση.

Αναπαράσταση (*Represent*):

Διάλεξε ένα βασικό οπτικό μοντέλο, όπως ένας γράφος από στήλες μια λίστα ή δέντρο.

Εκλέπτυνση- βελτίωση (*Refine*):

Βελτίωσε τη βασική αναπαράσταση για να την κάνεις πιο σαφή και οπτικά πιο ευχάριστη.

Διαδραση(*Interact*):
Πρόσθεσε μεθόδους για διαχείριση των δεδομένων ή για έλεγχο των γνωρισμάτων που είναι ορατά.

Η εξόρυξη δεδομένων και η οπτικοποίηση δεδομένων πάνε μαζί. Βρίσκοντας περίπλοκα μοτίβα στα δεδομένα και κάνοντάς τα ορατά για περαιτέρω ερμηνεία χρησιμοποιεί τη δύναμη των υπολογιστών, μαζί με τη δύναμη του ανθρώπινου μυαλού. Όταν χρησιμοποιηθεί σωστά, είναι ένας σπουδαίος συνδυασμός, που επιτρέπει πιο αποδοτικό διαχωρισμό των δεδομένων και αναγνώριση μοτίβων (pattern recognition).

Η μεγαλύτερη αξία μιας εικόνας φανερώνεται όταν μας αναγκάζει να προσέξουμε αυτό που ποτέ δεν περιμέναμε να δούμε.

-John Tukey- (Ben Fry, 2008)

2.3 Θετικά – γιατί οπτικοποίηση;

Σχεδόν κάθε ενδιαφέρουσα διαδικασία είναι πολύ δύσκολο να πραγματοποιηθεί πνευματικά και μόνο. Η οπτικοποίηση πληροφορίας:

- Προσφέρει στις πνευματικές λειτουργίες πρόσβαση σε μεγάλες ποσότητες δεδομένων που βρίσκονται εκτός του μυαλού.
- Μειώνει τις απαιτήσεις της λειτουργικής μνήμης του χρήστη.
- Επιτρέπει στη μηχανή να συμμετάσχει σε μια συνδετική διαδικασία, αλλάζοντας την οπτικοποίηση δυναμικά καθώς η δουλειά προχωράει.(Colin Ware, 2004)

Μια συμπληρωματική προσέγγιση στην κατανόηση της συμπεριφοράς διαφόρων συστημάτων, είναι η χρήση οπτικοποίησης πληροφορίας. Με την οπτικοποίηση, μπορούμε μερικές φορές να κερδίσουμε βαθύτερη γνώση που δεν είναι διαθέσιμη στις μετρικές από μόνες τους.(Julie Steele, Noah Iliinsky, 2010)

2.4 Κίνδυνοι – συμβατικές (τυποποιημένες) μορφές οπτικοποίησης

Υπερβολικά πολλή πληροφορία: Όταν ακούει κανείς τον όρο «υπερφόρτωση πληροφορίας» (information overload), πιθανότατα καταλαβαίνει αμέσως περί τίνος πρόκειται, μιας και είναι κάτι που αντιμετωπίζει καθημερινά. Ενδεικτικό είναι ότι κατά μέσο όρο σε μια κυριακάτικη New York Times περιέχεται όση πληροφορία είχε συνολικά πρόσβαση στη ζωή του ένας άνθρωπος της αναγέννησης. (Richard Saul Wurman's book Information Anxiety)

Συχνά, λιγότερη λεπτομέρεια, στην πραγματικότητα μεταφέρει περισσότερη πληροφορία γιατί η υπερβολικά συγκεκριμένη - ειδική λεπτομέρεια κάνει τον θεατή να χάσει αυτό που είναι σημαντικότερο ή να παραβλέψει εντελώς την εικόνα γιατί είναι πολύ περίπλοκη. Ενδείκνυται η χρήση όσον το δυνατόν λιγότερων δεδομένων, παρότι μπορεί τα επιπλέον δεδομένα να φαίνονται πολύτιμα.

Μάθε το κοινό σου (σε ποιούς απευθύνεσαι): Ποιό είναι το κοινό σου; Ποιοί είναι οι στόχοι τους όταν προσεγγίζουν μια οπτικοποίηση; Τι θέλουν να μάθουν; Αν δεν είναι προσβάσιμο το έργο για το κοινό σου, γιατί το κάνεις; Με ποιό τρόπο θα χρησιμοποιήσει το έργο το κοινό σου; Το να κάνεις τα πράγματα απλά και ξεκάθαρα δε σημαίνει να θεωρείς ότι οι χρήστες έχουν χαμηλό επίπεδο αντίληψης και να απλοποιείς εντελώς την διεπιφάνεια για αυτούς.

Η συντριπτική πλειοψηφία των οπτικοποιήσεων γίνεται με εντελώς παγιωμένες μορφές. Βασικοί τρόποι παρουσίασης όπως γράφοι με μπάρες, γραμμές, διασκορπισμένων σημείων (scatter graphs), και «πίτες», οργανωτικά διαγράμματα και διαγράμματα ροής, και κάποιοι άλλοι τρόποι, είναι εύκολο να δημιουργηθούν με κάθε είδους λογισμικό. Οποιοδήποτε bar chart ή scatter plot φτιαγμένο με το excel θα δείχνει σαν κάτι φτιαγμένο με το excel. Αυτές οι μορφές είναι ευρέως διαδεδομένες και παρέχουν συμβατικά και εύκολα σημεία εκκίνησης. Η θεωρία και η χρήση τους είναι, δικαιολογημένα, αρκετά κατανοητή από τους δημιουργούς αλλά και από τους καταναλωτές. Γι αυτό το λόγο είναι καλές, δυνατές λύσεις για κοινά προβλήματα οπτικοποίησης. Παρόλα αυτά, η βέλτιστη χρήση τους περιορίζεται σε μερικούς πολύ συγκεκριμένους τύπους δεδομένων και η τυποποίηση και η οικειότητα τους σημαίνουν ότι σπάνια θα πετύχουν καινοτομία. Οι πακεταρισμένες λύσεις παρέχουν τυποποιημένες απαντήσεις. Κάθε πρόβλημα

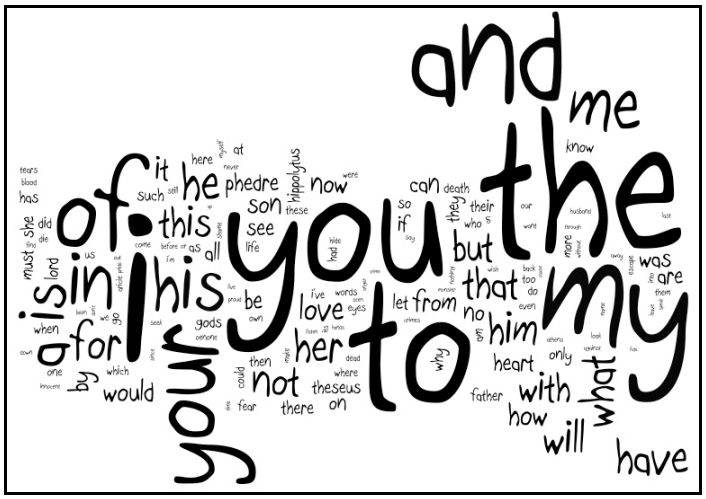
είναι μοναδικό, άρα χρειάζεται έμφαση σε εκείνη τη μοναδικότητα του για να λυθεί. (Julie Steele, Noah Iliinsky, 2010)-(Ben Fry, 2008)

2.5 Ανάλυση κειμένου (δημιουργία λεξικού – απόδοση tf-idf βαρών στις λέξεις)

Για να εξαχθεί μια εικόνα που οπτικοποιεί όρους προερχόμενους από έναν όγκο κειμένων χρειάζεται να δημιουργηθεί μία λίστα από λέξεις ενδιαφέροντος (thesaurus). Για να προκύψει αυτή η λίστα πρέπει τα διαθέσιμα κείμενα να αναλυθούν, και έπειτα να εξαχθούν από αυτά οι σημαντικοί όροι. Η ανάλυση, δηλαδή, στην προκειμένη περίπτωση περιλαμβάνει κάποιο φιλτράρισμα και κάποιους υπολογισμούς.

Αρχικά αγνοούνται συγκεκριμένοι χαρακτήρες όπως τα σημεία στίξης. Η διαδικασία της τμηματοποίησης ενός ορισμένου κειμένου σε λέξεις, ονομάζεται συμβολοποίηση (tokenization). Ως token (σύμβολο ή τεκμήριο) ορίζεται μια σειρά χαρακτήρων μέσα σε κάποιο συγκεκριμένο έγγραφο, που είναι ομαδοποιημένοι ως μια χρήσιμη σημασιολογικά μονάδα προς επεξεργασία. Πρέπει να τονισθεί ότι μια λέξη θεωρείται token αλλά όχι απαραίτητα και όρος (term).

Μερικές φορές, κάποιες υπερβολικά κοινές λέξεις, οι οποίες εκτιμάται ότι είναι μηδαμινής σημασίας ώστε να χρησιμοποιηθούν για το σκοπό της έρευνας, αποκλείονται από το λεξικό (thesaurus) εξαρχής. Αυτές οι λέξεις οι οποίες αφαιρούνται αφού γίνει η συμβολοποίηση του εγγράφου, λέγονται stop words.



Εικόνα 1 παράδειγμα του «wordle» με stop words στα αγγλικά

Για την αξιολόγηση του κατά πόσο είναι σημαντικές οι υπόλοιπες λέξεις, χρησιμοποιείται η απόδοση του tf-idf βάρους σε αυτές. Αναλυτικά:
Τον αριθμό των εμφανίσεων ενός όρου t σε ένα έγγραφο d τον συμβολίζουμε με tf_{td} . *(term frequency του όρου t στο έγγραφο d)*

Τον συνολικό αριθμό εμφανίσεων ενός όρου t σε μια συλλογή εγγράφων c τον συμβολίζουμε με cf_{tc}. *(collection frequency του όρου t στη συλλογή c)*

Τον αριθμό των εγγράφων d μιας συλλογής c που περιέχουν έναν όρο t τον συμβολίζουμε με df_t. *(document frequency του όρου t στα έγγραφα D της συλλογής c)*

Ο συνολικός αριθμός των εγγράφων d μιας συλλογής c συμβολίζεται με D.

Χρησιμοποιώντας τα παραπάνω ορίζουμε την inverse document frequency ενός όρου t ως $idf_t = \log(D/df_t)$.

Προκειμένου να δημιουργήσουμε ένα σύνθετο βάρος για κάθε όρο σε κάθε έγγραφο, χρησιμοποιούμε τα παραπάνω για να ορίσουμε τη σχέση:

$tf\text{-}idf_{td} = tf_{td} * idf_t$.

Αυτό το σχήμα απόδοσης βάρους στον όρο t, του αποδίδει ένα βάρος που είναι:

- Μέγιστο όταν ο t εμφανίζεται πολλές φορές σε μικρό αριθμό από έγγραφα.
- Μικρότερο όταν ο t εμφανίζεται λίγες φορές σε ένα έγγραφο, ή όταν

εμφανίζεται σε πολλά έγγραφα.

- Ελάχιστο όταν ο όρος t εμφανίζεται πρακτικά σε όλα τα έγγραφα.

(Christopher D. Manning et al, 2008)

2.6 Εργαλεία

2.6.1 Εργαλεία Οπτικοποίησης

Για λόγους σφαιρικής κάλυψης ίσως είναι σκόπιμο να αναφέρουμε μερικά εργαλεία που έχουν αναπτυχθεί ώστε να διευκολύνουν το πεδίο έρευνας της οπτικοποίησης.

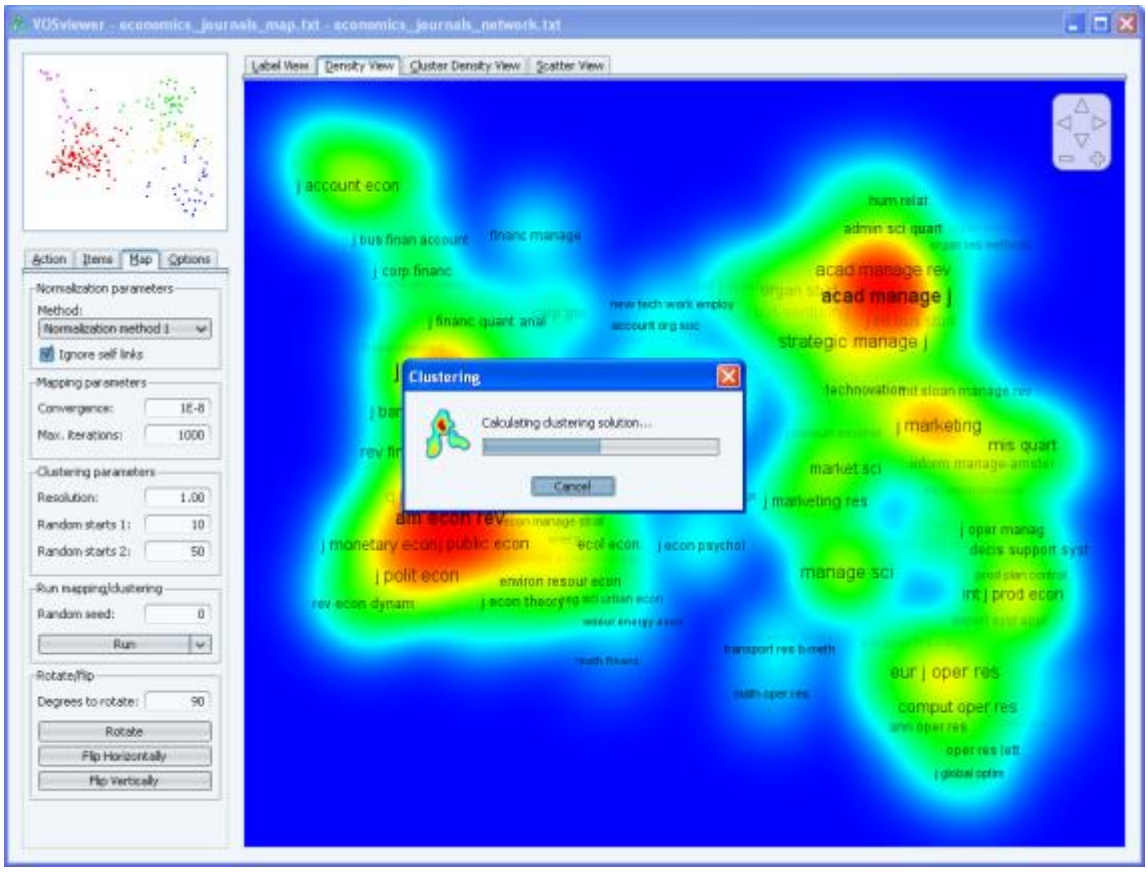
2.6.1.1 VOS viewer (VOS viewer, 2011)

Στόχος του VOS είναι να παρέχει μια λίγων διαστάσεων (low-dimensional) οπτικοποίηση, στην οποία τα αντικείμενα είναι τοποθετημένα με τέτοιο τρόπο ώστε η απόσταση μεταξύ οποιουδήποτε ζευγαριού αντικειμένων να αντανακλά την ομοιότητά τους όσον το δυνατόν ακριβέστερα.

Ο VOS viewer είναι ένα πρόγραμμα που μπορεί να χρησιμοποιηθεί για τους κάτωθι σκοπούς:

- Για κατασκευή χάρτη βασισμένη σε έναν πίνακα «συμπτώσεων» (co-occurence matrix) των στοιχείων. Οι χάρτες κατασκευάζονται με την τεχνική χαρτογράφησης και ομαδοποίησης VOS (Van Eck and Waltman, 2009a, 2009b).
- Για επισκόπηση ενός χάρτη. Μπορεί να εμφανίζει ένα χάρτη με ποικίλους διαφορετικούς τρόπους, που κάθε ένας δίνει έμφαση σε μια διαφορετική άποψη αυτού. Περιλαμβάνει λειτουργίες όπως zoom, scrolling και αναζήτηση, που διευκολύνουν την λεπτομερή εξέταση της εικόνας. Οι δυνατότητες επισκόπησης του προγράμματος είναι ιδιαιτέρως χρήσιμες για την επεξεργασία χαρτών που έχουν ένα σχετικά μεγάλο πλήθος στοιχείων

(δηλαδή το λιγότερο 100 ή 200).



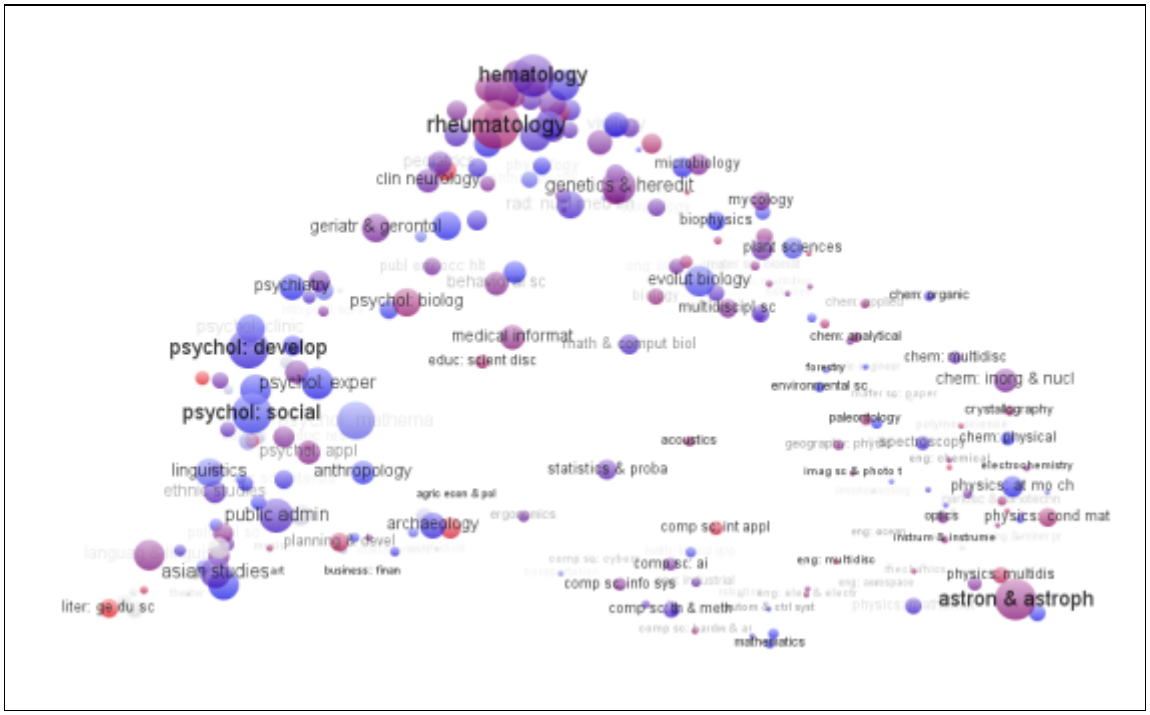
Εικόνα 2 Στιγμιότυπο λειτουργίας του VOS viewer

Για να εμφανίσει έναν χάρτη ο VOS viewer, δεν είναι απαραίτητο ο χάρτης να έχει κατασκευαστεί από το ίδιο πρόγραμμα, αλλά μπορεί να έχει κατασκευαστεί με τη χρήση άλλου προγράμματος. Για παράδειγμα ο VOS viewer επιτρέπει την επισκόπηση χαρτών φτιαγμένων με ένα πρόγραμμα πολυδιάστατης κλιμάκωσης (multidimensional scaling program) όπως το PROXCAL του SPSS.

Ο VOS viewer προορίζεται κυρίως για χρήση βιβλιομετρικών αναλύσεων (bibliometric analyses). Μπορεί για παράδειγμα να χρησιμοποιηθεί για την κατασκευή χαρτών συγγραφέων ή περιοδικων βασισμένων σε δεδομένα αναφορών ή κατασκευή χαρτών από λέξεις - κλειδιά που βασίζονται σε δεδομένα δημιουργημένα από τις συμπτώσεις αυτών των λέξεων στο ίδιο κείμενο (όπως στην πρακτική εφαρμογή αυτής της εργασίας).

Ο VOSviewer βρίσκεται ανεβασμένος στην ιστοσελίδα www.vosviewer.com. Το πρόγραμμα μπορεί να χρησιμοποιηθεί ελεύθερα, και για εμπορικούς σκοπούς.

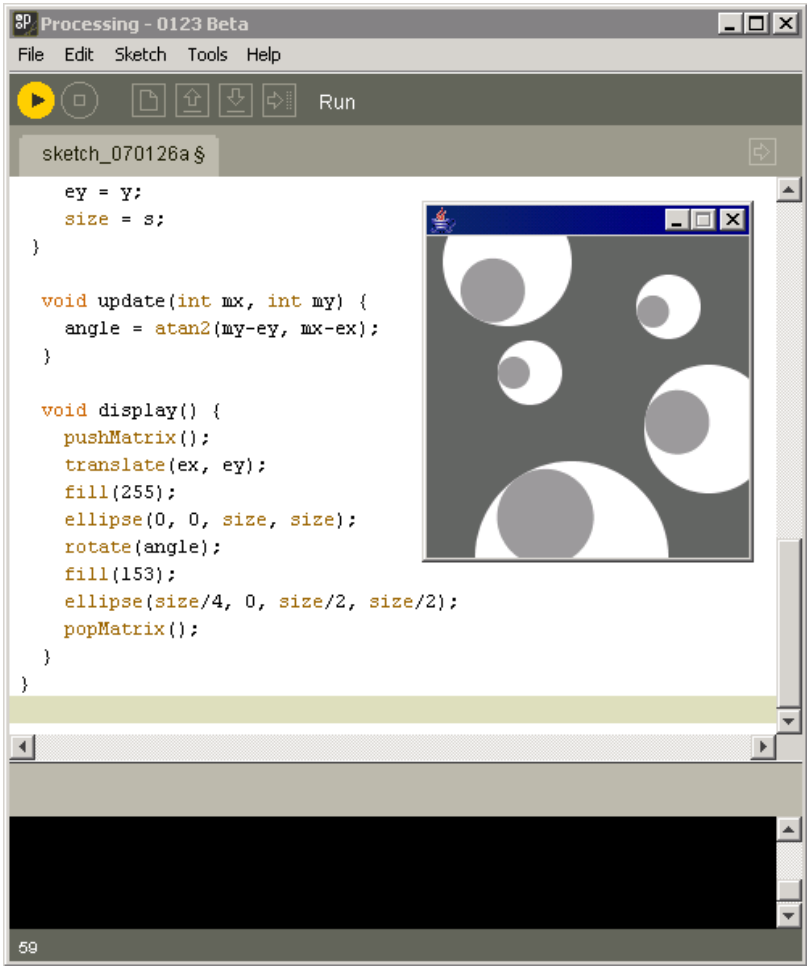
Ένα παράδειγμα χρήσης του VOSviewer, είναι η περιληπτική απεικόνιση του ερευνητικού προφίλ του πανεπιστημίου του Leiden (Εικόνα 3).



Εικόνα 3 Έξοδος του VOS viewer για το ερευνητικό προφίλ του πανεπιστημίου του Leiden

2.6.1.2 Processing (Ben Fry, 2008)

Το Processing ξεκίνησε ως project το 2001. Αρχικά φτιάχτηκε σαν μια domain-specific επέκταση για Java, όμως αργότερα κατέληξε να είναι ένα εργαλείο σχεδίασης και πρωτοτυποποίησης για κινούμενα γραφικά και σύνθετες οπτικοποιήσεις δεδομένων. Το Processing είναι ένα απλό προγραμματιστικό περιβάλλον που δημιουργήθηκε για να κάνει ευκολότερη την ανάπτυξη οπτικών εφαρμογών με έμφαση στην κίνηση και για να παρέχει στους χρήστες feedback άμεσα, μέσω διάδρασης.



Εικόνα 4 Προγραμματιστικό περιβάλλον στο processing

Είναι βασισμένο στην Java, αλλά τα προγραμματιστικά στοιχεία είναι αρκετά απλά και μπορεί κανείς να το μάθει χωρίς να ξέρει καθόλου Java. Η τελευταία έκδοση του Processing βρίσκεται εδώ: http://processing.org/download
Ένας σημαντικός στόχος του project ήταν να καταστήσει αυτό το είδος προγραμματισμού προσβάσιμο σε ένα ευρύτερο κοινό. Για αυτόν το λόγο είναι ελεύθερο προς κατέβασμα, ελεύθερο προς χρήση και open source.

Το Processing έχει χρησιμοποιηθεί στην περίπτωση του flight patterns και του nytimes.com website access visualization. Οι εικόνες που παράγονται στα εν λόγω projects φαίνονται στην ακόλουθη ενότητα με τα παραδείγματα και συγκεκριμένα στις παραγράφους 2.7.2 και 2.7.4.

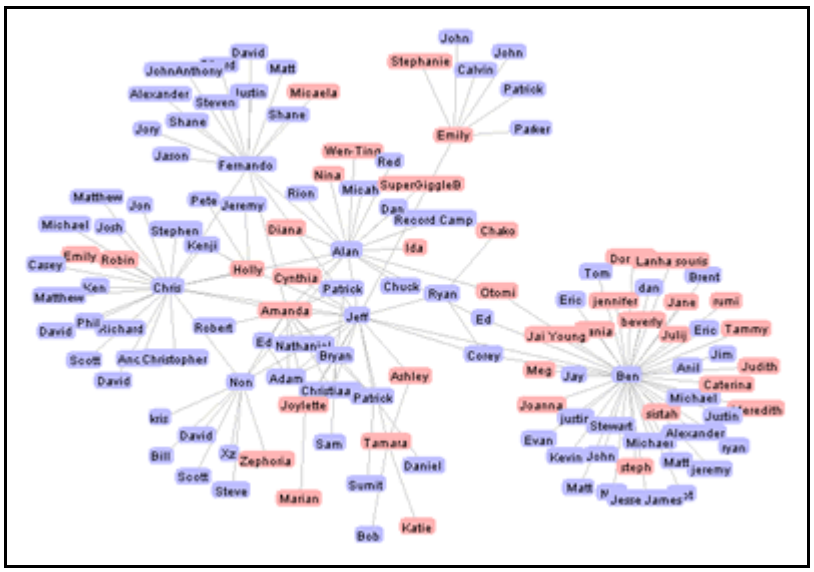
2.6.1.3 Prefuse (Prefuse, 2011)

Το Prefuse είναι ένα επεκτάσιμο software framework που βοηθά τους προγραμματιστές να δημιουργήσουν διαδραστικές εφαρμογές οπτικοποίησης

δεδομένων, χρησιμοποιώντας την Java ως γλώσσα προγραμματισμού. Μπορεί να χρησιμοποιηθεί για κατασκευή standalone εφαρμογών, οπτικών components ενσωματωμένων σε μεγαλύτερες εφαρμογές, και web applets. Το Prefuse προσπαθεί να απλοποιήσει τις διεργασίες: της αναπαράστασης και του αποδοτικού χειρισμού των δεδομένων, της χαρτογράφησης των δεδομένων μέσω οπτικών αναπαραστάσεων (δηλαδή μέσω χωρικής τοποθέτησης, μεγέθους σχήματος, χρώματος κτλ) και της διάδρασης με τα δεδομένα. Μερικά features του Prefuse είναι:

- Πίνακες, γράφοι και δομές δεδομένων δέντρου που δέχονται διάφορα χαρακτηριστικά (data attributes), τοποθέτηση δεικτών στα δεδομένα (data indexing), και ερωτήματα (queries).
- Components για layout, χρώμα, μέγεθος, και κωδικοποιήσεις σχημάτων, τεχνικές παραμόρφωσης, animation κ.α.
- Μια βιβλιοθήκη με ελέγχους διάδρασης για κοινές διαδραστικές, άμεσου χειρισμού, λειτουργίες.
- Υποστήριξη για animation μέσω ενός μηχανισμού προγραμματισμού γενικών δραστηριοτήτων.
- Μετατροπή θέασης που υποστηρίζει panning και zoom, συμπεριλαμβανομένων και του γεωμετρικού και του σημασιολογικού zoom.
- Δυναμικά ερωτήματα για διαδραστικό φιλτράρισμα των δεδομένων.
- Ενσωματωμένη αναζήτηση κειμένου χρησιμοποιώντας μερικές από τις διαθέσιμες μηχανές αναζήτησης.
- Μια μηχανή προσομοίωσης φυσικών δυνάμεων για δυναμικό layout και animation.
- Ευελιξία για πολλαπλούς τρόπους θέασης.
- Ενσωματωμένη γλώσσα της μορφής της SQL για τη συγγραφή ερωτημάτων προς τις δομές δεδομένων του Prefuse και τη δημιουργία παραγόμενων πεδίων δεδομένων.
- Υποστήριξη για αποστολή ερωτημάτων σε SQL βάσεις δεδομένων και χαρτογράφηση των αποτελεσμάτων των ερωτημάτων μέσα στις δομές δεδομένων του Prefuse.
- Απλά APIs για τη δημιουργία συστατικών κατά περίπτωση, για επεξεργασία, διάδραση και rendering.

Το Prefuse χρησιμοποιείται σε σχολικά projects, στην ακαδημαϊκή και βιομηχανική έρευνα και για ανάπτυξη εμπορικού λογισμικού.

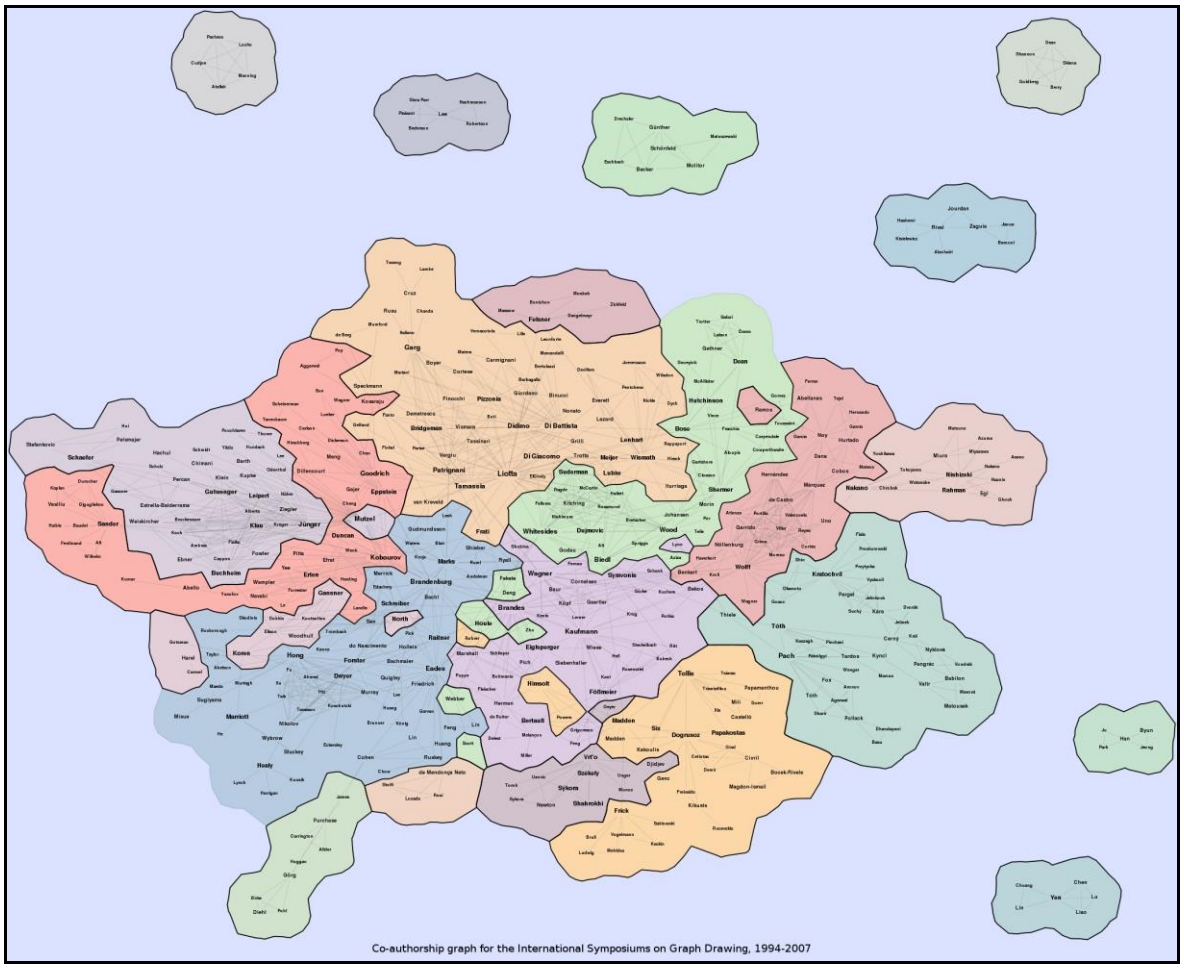


Εικόνα 5 Παράδειγμα εξόδου του Prefuse

2.6.1.4 Graphviz (GraphViz, 2011)

Η οπτικοποίηση γράφων (Graph Visualization) είναι ένας τρόπος αναπαράστασης δομικής πληροφορίας (structural information), όπως διαγράμματα αφαιρετικών γράφων και δικτύων. Διαθέτει σημαντικές εφαρμογές στη δικτύωση, στο web design, στη βιοπληροφορική, τη μηχανική λογισμικού, στο σχεδιασμό βάσεων δεδομένων και σε οπτικές διεπαφές για άλλα τεχνικά πεδία.

Το Graphviz είναι ένα open source λογισμικό οπτικοποίησης. Διαθέτει διάφορα προγράμματα για main layout. Διαθέτει επίσης web και διαδραστικές γραφικές διεπαφές, και βοηθητικά εργαλεία, βιβλιοθήκες και γλωσσικούς συνδέσμους. Δεν είναι δυνατόν να χωρέσουν πολλές λειτουργίες σε ένα GUI editor αλλά υπάρχουν πολλά εξωτερικά projects και εμπορικά εργαλεία που ενσωματώνουν το Graphviz.



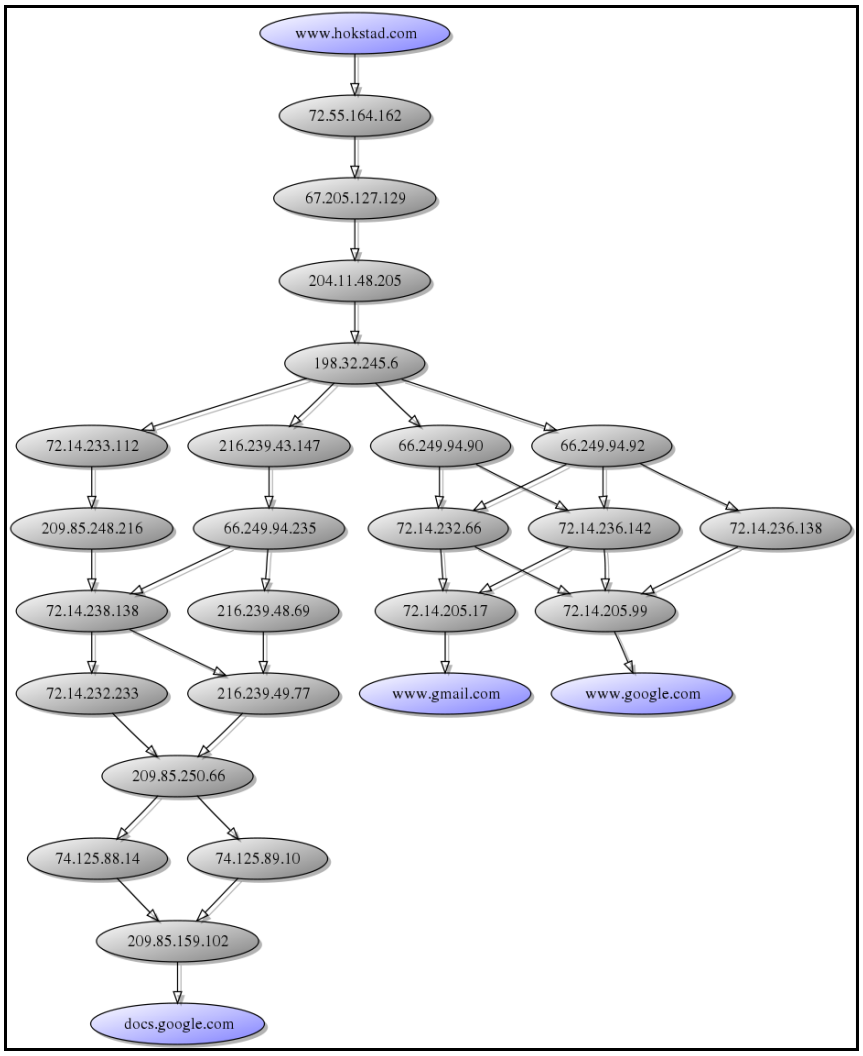
Εικόνα 6 Παράδειγμα οπτικοποίησης με το GraphViz

Τα προγράμματα για layout του Graphviz δέχονται περιγραφές γράφων σε απλή γλώσσα κειμένου, και δημιουργούν διαγράμματα σε χρήσιμες μορφές, όπως μορφές εικόνας και SVG για ιστοσελίδες, pdf και postscript για να συμπεριληφθούν σε άλλα κείμενα ή να εμφανιστούν σε κάποιον διαδραστικό graph browser. Υποστηρίζει επίσης GXL που είναι «διάλεκτος» της XML.

Το Graphviz έχει πολλές επιλογές για συμπαγή διαγράμματα, όπως επιλογές για χρώμα, γραμματοσειρές, layouts με καρτέλες, τύπους γραμμών, υπερσυνδέσμους και τροποποιήσιμα σχήματα.

Στην πράξη οι γράφοι δημιουργούνται από εξωτερικές πηγές δεδομένων, αλλά επίσης μπορούν να δημιουργηθούν και να τροποποιηθούν «χειροκίνητα», είτε ως απλό κείμενο ή μέσα σε ένα πρόγραμμα σχεδίασης γραφικών.

Το TraceViz οπτικοποιεί την έξοδο μιας εντολής trace route και χρησιμοποιεί το GraphViz για τον σχηματισμό του γράφου εξόδου:



Εικόνα 7 Γράφος φτιαγμένος με το GraphViz

2.6.2 Εργαλεία Ανάλυσης Κειμένου

2.6.2.1 MontyLingua (MontyLingua, 2011)

Το MontyLingua είναι έλα ελεύθερο end-to-end πρόγραμμα κατανόησης της αγγλικής γλώσσας. Δέχεται ως είσοδο ένα κείμενο στα αγγλικά και το αποτέλεσμα είναι μια σημασιολογική ανάλυση αυτού του κειμένου. Χρησιμεύει για ανάκτηση και εξαγωγή πληροφορίας, επεξεργασία αιτημάτων και απάντηση ερωτήσεων. Από αγγλικές προτάσεις, εξάγει το υποκείμενο, το ρήμα, το αντικείμενο, εξάγει επίθετα, φράσεις ουσιαστικών και ρημάτων, και ονόματα ανθρώπων, τοποθεσιών, γεγονότων, ημερομηνίες και ώρες και άλλες σημασιολογικές πληροφορίες.

Η έκδοση 2.0 έχει δοκιμαστεί σε Windows, διάφορες εκδόσεις του UNIX, και σε mac OS X, καθώς και σε διάφορες εκδόσεις της Java. Χρησιμοποιείται από αρκετά πανεπιστημιακά ερευνητικά projects και από εμπορικά περιβάλλοντα.

Το MontyLingua διαφέρει από άλλα εργαλεία επεξεργασίας φυσικής γλώσσας γιατί:

Είναι εντελώς *end-to-end*. Παίρνει για είσοδο ένα κείμενο και παράγει ως έξοδο τη σημασιολογική ερμηνεία.

Δεν είναι πολλά εργαλεία και εφαρμογές ραμμένα μεταξύ τους, είναι μία συμπαγής εφαρμογή.

Δε χρειάζεται εκπαίδευση και λειτουργεί κατευθείαν.

Διαθέτει γνώση “κοινής λογικής” σχετικά με τον καθημερινό κόσμο, που του επιτρέπει την αποφυγή λαθών όπως:

- "(NX the/DT mosquito/NN bit/NN NX) (NX the/DT boy/NN NX)"***** ==corrected==>
- "(NX the/DT mosquito/NN NX) (VX bit/VBD VX) (NX the/DT boy/NN NX)"*****

*(*Στην πρώτη περίπτωση θεωρείται ότι η λέξη mosquito προσδιορίζει την λέξη bit που κατατάσσεται ως ουσιαστικό, ενώ στη δεύτερη περίπτωση, αναγνωρίζεται η λέξη mosquito ως ουσιαστικό και υποκείμενο της πρότασης, και η λέξη bit ως ρήμα, που έτσι βγαίνει και νόημα.)*

Είναι ελαφρύ και φορητό ανάμεσα στις πλατφόρμες, γραμμένο σε Python και επίσης διαθέσιμο ως μεταγλωτισμένη βιβλιοθήκη σε Java.

Είναι εύκολο να τροποποιηθεί επιτρέποντας λεξικό χρήστη.

Το MontyLingua εκτελεί τις εξής εργασίες πάνω σε ένα κείμενο:

1.*MontyTokenizer* – Συμβολοποιεί το ακατέργαστο αγγλικό κείμενο (ευαίσθητο σε συντομογραφίες), και αναλύει γραμματικές εκθλίψεις, πχ "you're" ==> "you are"

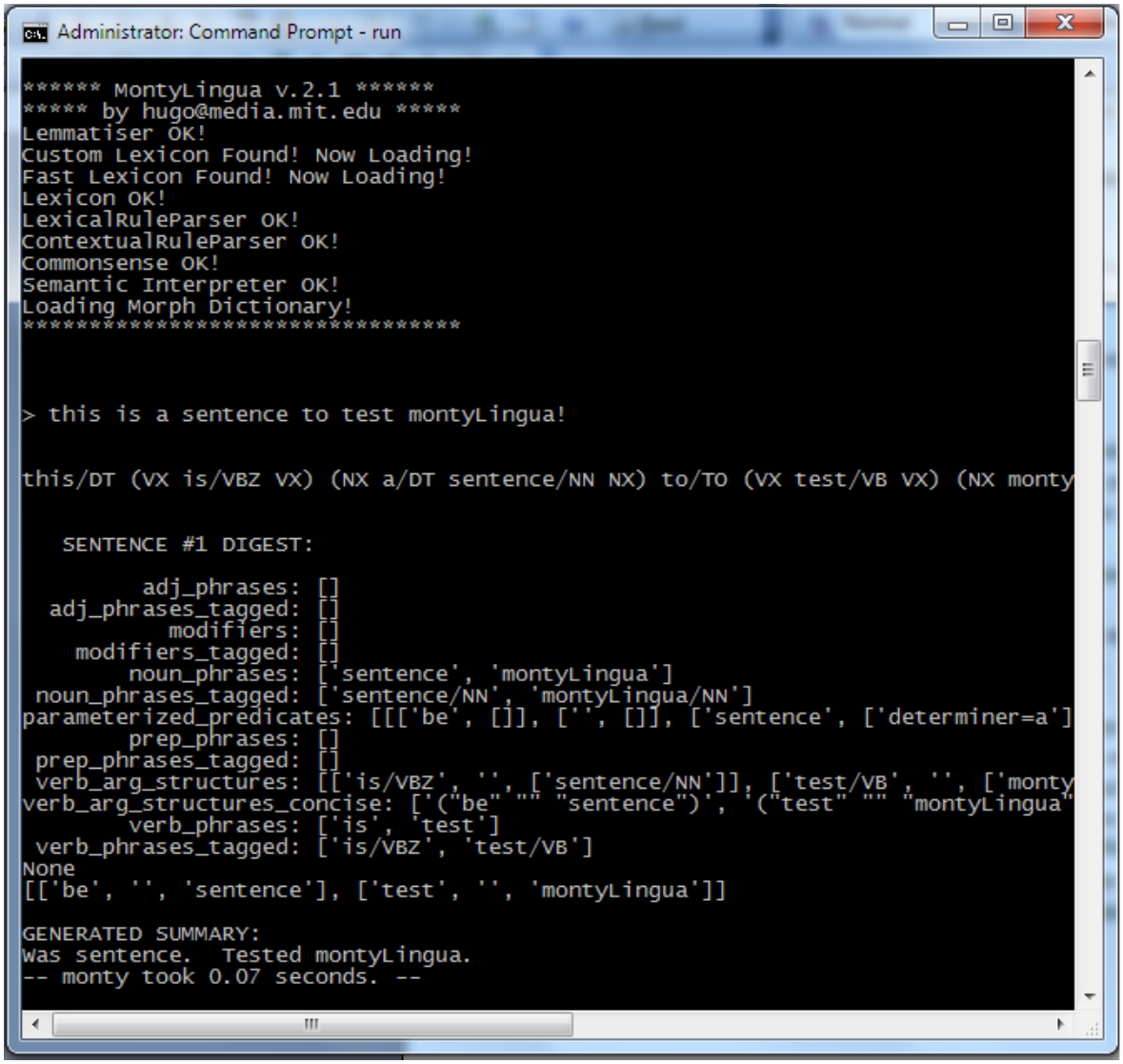
2.*MontyTagger* -Τοποθετεί ετικέτες στα μέρη του λόγου.

3.*MontyChunker* - Εντοπίζει κανονικές εκφράσεις.

4.*MontyExtractor* - Εξάγει φράσεις και τριπλέτες υποκειμένου/ ρήματος/ αντικειμένου από τις προτάσεις.

5.*MontyLemmatiser* - Απογυμνώνει την κλιτή μορφολογία, δηλαδή αλλάζει τα ρήματα στη μορφή απαρεμφάτου και τα ουσιαστικά στον ενικό αριθμό.

6.*MontyNLGenerator* – Χρησιμοποιεί την σύντομη και περιεκτική αναπαράσταση των κατηγορημάτων του MontyLingua, για να παράξει νατουραλιστικές αγγλικές προτάσεις και συνόψεις των κειμένων.



Εικόνα 8 Στιγμιότυπο εκτέλεσης του montyLingua από τη γραμμή εντολών

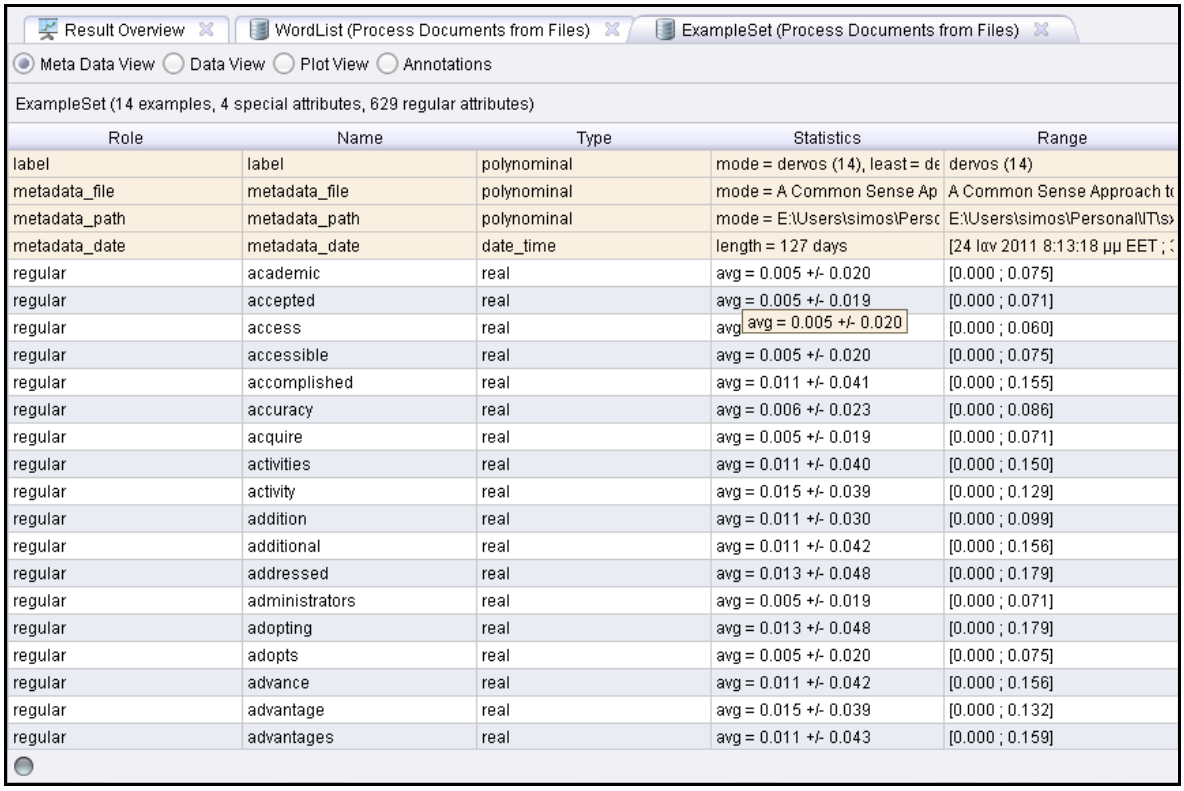
2.6.2.2 Rapid Miner (Rapid Miner, 2011)

Το Rapid Miner είναι ένα σύστημα open - source για εξόρυξη πληροφορίας (data mining) και ανακάλυψη γνώσης (knowledge discovery). Είναι διαθέσιμο ως μια stand-alone εφαρμογή για ανάλυση δεδομένων και ως μια μηχανή εξόρυξης πληροφορίας που μπορεί να ενσωματωθεί σε άλλα προϊόντα. Μέχρι τώρα, υπάρχουν χιλιάδες εφαρμογές του Rapid Miner σε περισσότερες από 30 χώρες. Διαθέτει:

- Αποθήκες για το χειρισμό διεργασιών, δεδομένων και μεταδεδομένων.
- Λειτουργίες φόρτωσης δεδομένων, μετασχηματισμό δεδομένων (data transformation), μοντελοποίηση δεδομένων (data modeling) και μεθόδους

οπτικοποίησης δεδομένων.

- Ελεύθερα διαθέσιμο open-source σύστημα εξόρυξης και ανάλυσης δεδομένων
- Συμβατότητα με όλες τις κύριες πλατφόρμες και λειτουργικά συστήματα.
- Πιο διαισθητικό σχεδιασμό διαδικασίας.
- Οπτική δεδομένων πολλών επιπέδων .
- GUI mode, server mode (command line), πρόσβαση μέσω Java API.
- Απλό μηχανισμό επέκτασης.
- Πολυδιάστατη απεικόνιση.
- Περισσότερους από 500 τελεστές για ενσωμάτωση και μετασχηματισμό δεδομένων, εξόρυξη δεδομένων, αξιολόγηση και οπτικοποίηση.
- Ορισμό επαναχρησιμοποιήσιμων δομικών κομματιών.
- Γραφικό σχεδιασμό διεργασιών για σταθερά ζητήματα και γλώσσα για scripting για ιδιόμορφες λειτουργίες.
- Πρόσβαση σε πηγές δεδομένων όπως Excel, Access, Oracle, IBM DB2, Microsoft SQL, Sybase, Ingres, MySQL, Postgres, SPSS, dBase, Text files και άλλες.



Role	Name	Type	Statistics	Range
label	label	polynominal	mode = dervos (14), least = de	dervos (14)
metadata_file	metadata_file	polynominal	mode = A Common Sense Ap	A Common Sense Approach t
metadata_path	metadata_path	polynominal	mode = E:\Users\simos\Persc	E:\Users\simos\Personal\IT\s
metadata_date	metadata_date	date_time	length = 127 days	[24 Ιαν 2011 8:13:18 μμ EET ;
regular	academic	real	avg = 0.005 +/- 0.020	[0.000 ; 0.075]
regular	accepted	real	avg = 0.005 +/- 0.019	[0.000 ; 0.071]
regular	access	real	avg	[0.000 ; 0.060]
regular	accessible	real	avg = 0.005 +/- 0.020	[0.000 ; 0.075]
regular	accomplished	real	avg = 0.011 +/- 0.041	[0.000 ; 0.155]
regular	accuracy	real	avg = 0.006 +/- 0.023	[0.000 ; 0.086]
regular	acquire	real	avg = 0.005 +/- 0.019	[0.000 ; 0.071]
regular	activities	real	avg = 0.011 +/- 0.040	[0.000 ; 0.150]
regular	activity	real	avg = 0.015 +/- 0.039	[0.000 ; 0.129]
regular	addition	real	avg = 0.011 +/- 0.030	[0.000 ; 0.099]
regular	additional	real	avg = 0.011 +/- 0.042	[0.000 ; 0.156]
regular	addressed	real	avg = 0.013 +/- 0.048	[0.000 ; 0.179]
regular	administrators	real	avg = 0.005 +/- 0.019	[0.000 ; 0.071]
regular	adopting	real	avg = 0.013 +/- 0.048	[0.000 ; 0.179]
regular	adopts	real	avg = 0.005 +/- 0.020	[0.000 ; 0.075]
regular	advance	real	avg = 0.011 +/- 0.042	[0.000 ; 0.156]
regular	advantage	real	avg = 0.015 +/- 0.039	[0.000 ; 0.132]
regular	advantages	real	avg = 0.011 +/- 0.043	[0.000 ; 0.159]

Εικόνα 9 Εξαγωγή λέξεων με το rapid miner

2.6.3 Άλλα εργαλεία

2.6.3.1 Netbeans (Netbeans, 2011)

Το Netbeans είναι ταυτόχρονα μια πλατφόρμα για εφαρμογές Java, και ένα IDE (Integrated Development Environment) για ανάπτυξη σε Java, Javascript, PHP, Python, Groovy, C, C++, Scala, Clojure και άλλα.

Είναι γραμμένο σε Java και μπορεί να τρέξει οπουδήποτε υπάρχει μια συμβατή JVM (Java Virtual Machine) εγκατεστημένη, συμπεριλαμβανομένων των Windows, Mac OS, Linux και Solaris. Για λειτουργίες ανάπτυξης σε Java ένα JDK (java development Kit) είναι απαραίτητο να υπάρχει, αλλά όχι για προγραμματισμό σε άλλες γλώσσες.

Το Netbeans επιτρέπει την ανάπτυξη εφαρμογών από ένα σύνολο υπομονάδων/ συστατικών λογισμικού που λέγονται Modules. Υπάρχει η δυνατότητα οι εφαρμογές που βασίζονται στην πλατφόρμα του Netbeans να επεκταθούν από τρίτους. Σκοπό έχει να απλοποιήσει την ανάπτυξη JavaSwing εφαρμογών. Το πακέτο Netbeans IDE για τη 2η έκδοση της Java (Java SE) περιέχει ότι χρειάζεται για την ανάπτυξη Netbeans plugins και εφαρμογών βασισμένων στην πλατφόρμα του, χωρίς να χρειάζεται επιπλέον SDK.

Οι εφαρμογές μπορούν να εγκαταστήσουν τα modules δυναμικά. Οποιαδήποτε εφαρμογή μπορεί να περιλάβει το module του κέντρου ενημερώσεων (update center module) για να επιτρέψει στους χρήστες της εφαρμογής να κατεβάσουν ψηφιακά υπογεγραμμένες αναβαθμίσεις και πρόσθετα, απευθείας μέσα στην τρέχουσα εφαρμογή. Με αυτόν τον τρόπο γίνεται αναβάθμιση χωρίς να χρειάζεται οι χρήστες να κατεβάσουν ξανά ολόκληρη την εφαρμογή.

Μερικά από τα χαρακτηριστικά της πλατφόρμας είναι:

- Διαχείριση μέσω διεπαφής χρήστη (δηλαδή μενού και εργαλειογραμμές)
- Διαχείριση επιλογών χρήστη
- Διαχείριση αποθήκευσης (αποθήκευση και φόρτωση οποιουδήποτε είδους πληροφορίας)

- Διαχείριση παραθύρων
- Λειτουργία wizard (υποστηρίζει βήμα προς βήμα διαλόγους)
- Οπτική βιβλιοθήκη Netbeans
- Ενσωματωμένα εργαλεία ανάπτυξης

2.7 Παραδείγματα (Julie Steele, Noah Iliinsky, 2010)

Στην ενότητα αυτή, παρατίθενται μερικά παραδείγματα οπτικοποίησης πληροφορίας. Κάποια από αυτά έχουν περισσότερο αισθητική αξία, και άλλα είναι προσανατολισμένα στο να παρουσιάσουν μια περιεκτική εικόνα κάποιων δεδομένων, μια άποψη αυτών, που δεν ήταν ορατή πριν αυτά να οπτικοποιηθούν.

2.7.1 Wordle

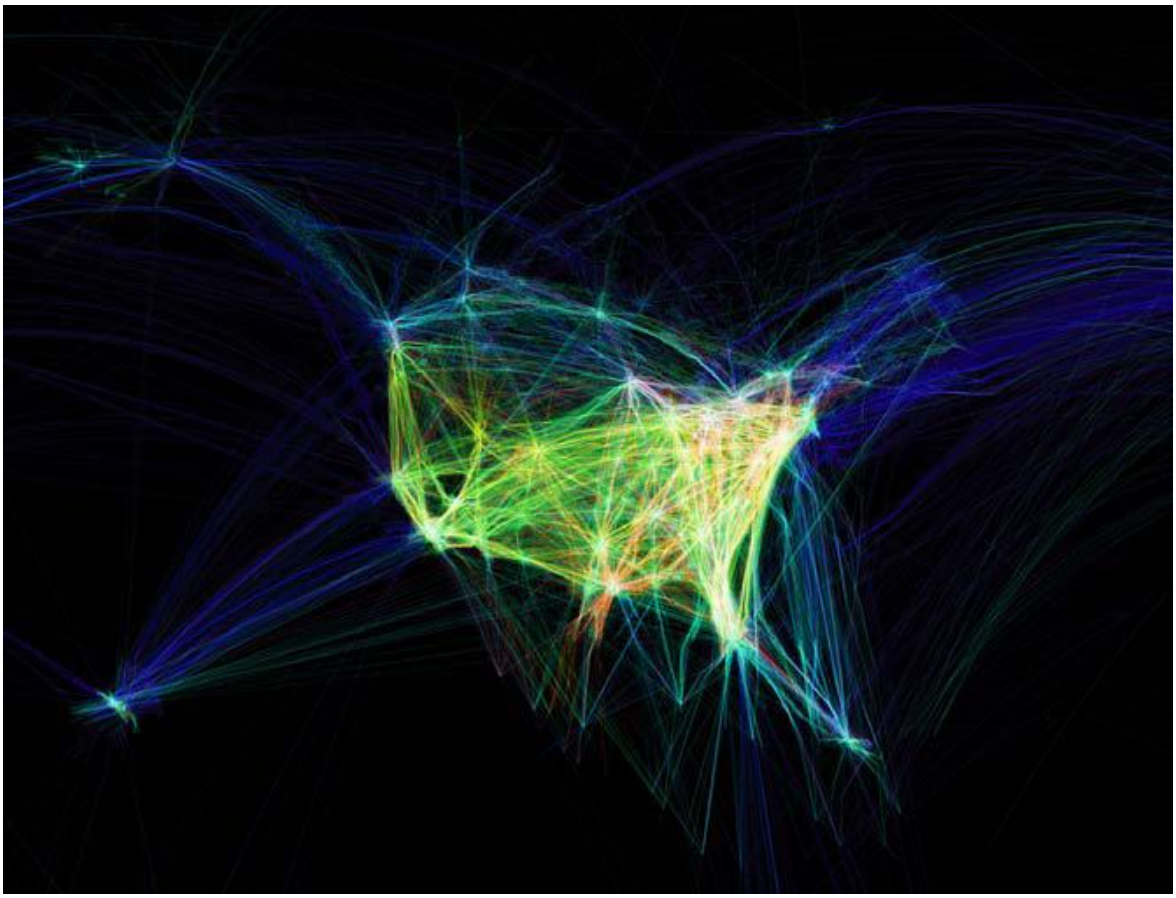
Το Wordle είναι ίσως μια γνώριμη μορφή οπτικοποίησης πληροφορίας, ένα πολύχρωμο κολάζ φτιαγμένο από λέξεις ή αλλιώς ένα “ναρκωτικό” που ανοίγει το δρόμο για την ανάλυση κειμένου. Ως “ναρκωτικό” είναι σχεδιασμένο για να προσφέρει απόλαυση, παρόλα αυτά όμως οι ρίζες του βρίσκονται στα χρηστικά σύννεφα ετικετών (tag clouds) τα οποία έγιναν γνωστά από ιστότοπους όπως το del.ico.us και το Flickr.

Το πρόγραμμα φιλτράρει τις λέξεις και αφαιρεί αυτές που θεωρούνται stop words δημιουργεί μια λίστα με αυτές και τους αποδίδει ένα βάρος ανάλογα με το πόσες φορές εμφανίζονται σε ένα δεδομένο κείμενο. Αναγνωρίζει τί γλώσσα είναι (ανάμεσα από αυτές που υποστηρίζει) και σχηματίζει ένα σύννεφο λέξεων, στο οποίο η κάθε μία έχει μέγεθος ανάλογο με το βάρος της.

Εικόνα 10 Παραδείγματα του wordle

2.7.2 Flight patterns

Υπάρχουν δρόμοι στον ουρανό. Δε μπορούμε να τους δούμε αλλά βρίσκονται εκεί. Αυστηρά καθορισμένες «λεωφόροι» που διασχίζονται από χιλιάδες αεροπλάνα κάθε μέρα. Αποδεικνύεται από την γραφική αποτύπωση (Εικόνα 11):



Εικόνα 11 Οπτικοποίηση flight patterns με τις πτήσεις των ΗΠΑ και του Καναδά

Το Flight Patterns είναι ένα project που ξεκίνησε το 2005, το οποίο οπτικοποιεί την εναέρια επιβατική κυκλοφορία στις Ηνωμένες Πολιτείες και τον

Καναδά. Υπάρχει σε δύο μορφές: ως ακίνητη εικόνα που εντοπίζει τα αεροσκάφη που φτάνουν και φεύγουν από τα αεροδρόμια του Καναδά και των ΗΠΑ μέσα σε 24 ώρες, και η απεικόνιση μέσω βίντεο, που δείχνει το ίδιο πράγμα σε κίνηση.

Τεχνικές και δεδομένα.
Το αρχικό dataset προέρχεται από το ASDI και έχει για κάθε data point τα εξής χαρακτηριστικά:
• Γεωγραφικό μήκος
• Γεωγραφικό πλάτος
• Υψόμετρο
• Κατασκευαστή Αεροσκάφους
• Μοντέλο αεροσκάφους
• Σφραγίδα ημερομηνίας - ώρας (Timestamp)
• Αριθμό πτήσης

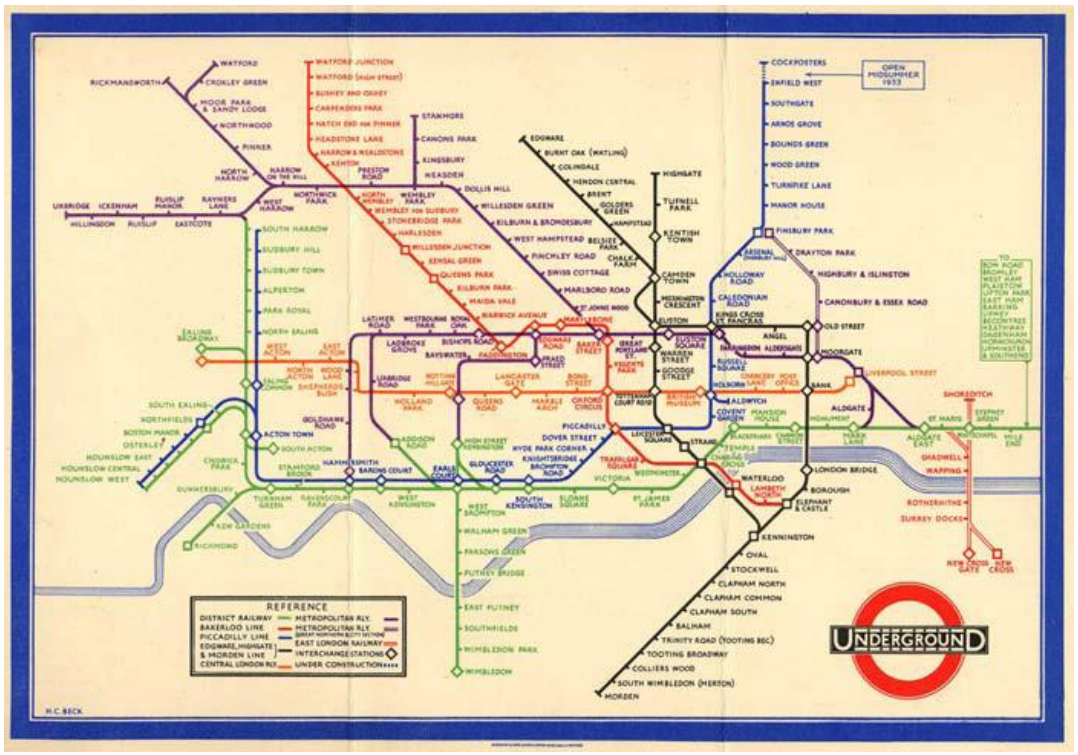
Οι οπτικοποιήσεις δημιουργήθηκαν με τη χρήση του Processing. Με τα δεδομένα και τη χρήση του προγράμματος τα γεωγραφικά μήκη και πλάτη μεταφράστηκαν σε έναν δισδιάστατο χάρτη. Προστέθηκε και χρώμα στο κάθε σημείο για χαρακτηριστικά όπως το υψόμετρο και το μοντέλο του αεροσκάφους.
Στο βίντεο ο συμβολισμός των αεροσκαφών ως κινούμενα σημεία, απέτυχε στο να αποκαλύψει την εξέλιξη της κάθε πτήσης. Για να φαίνεται η κίνηση τους, εμφανίζονται γραμμές που ενώνουν κάθε data point. Μετά από ένα χρονικά διάστημα (τριών ή πέντε λεπτών ανάλογα με το dataset) προστίθεται ένα 4% μαύρης αδιαφάνειας (opacity) σε ολόκληρο τον χάρτη με αποτέλεσμα οι παλιές πτήσεις να σβήνουν στο φόντο σταδιακά και να αναδεικνύεται η πρόοδος των αεροπλάνων.

2.7.3 Χάρτες Μετρό

Οι χάρτες είναι από τις πιο βασικές οπτικοποιήσεις δεδομένων που χρησιμοποιούμε. Φτιάχνονται εδώ και αιώνες. Ακόμη όμως δεν έχουν τελειοποιηθεί ως εργαλείο για την κατανόηση πολύπλοκων συστημάτων. Ένα παράδειγμα είναι το υπόγειο δίκτυο της Νέας Υόρκης με 26 γραμμές και 468 σταθμούς που διασχίζουν πέντε δημοτικά διαμερίσματα (Εικόνα 12).

Εικόνα 12 Χάρτης του υπογείου δικτύου του Μετρό της Νέας Υόρκης

Σκοπός αυτών των οπτικοποιήσεων είναι να βοηθήσουν τον χρήστη να αντιληφθεί σαφώς τη σύνδεση μεταξύ των γραμμών για ευκολότερη πλοήγηση, και η παροχή μιας ξεκάθαρης αναπαράστασης του πού βρίσκεται, όταν βγαίνει έξω από έναν σταθμό.



Εικόνα 13 Παλιός χάρτης του Μετρό του Λονδίνου

Εικόνα 14 Χάρτης του μετρό της Βαρκελώνης

2.7.4 Οπτικοποίηση πρόσβασης στις ιστοσελίδες nytimes.com και mobile.nytimes.com

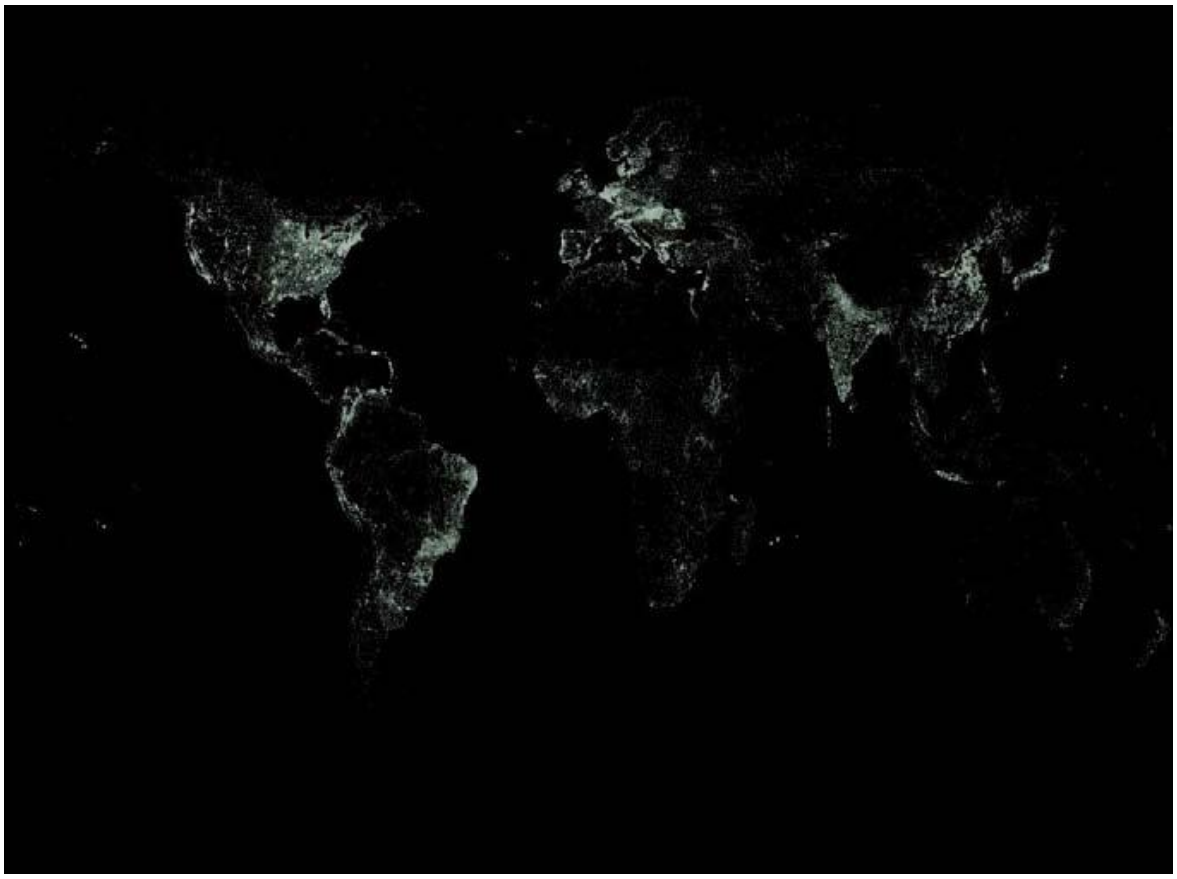
Ποιοί είναι οι αναγνώστες του New York Times website; Τι ώρα της ημέρας τείνουν να το επισκέπτονται; Τι συσκευές χρησιμοποιούν για το σκοπό αυτό; Από

ποιό μέρος του κόσμου είναι; Απάντηση στις ερωτήσεις αυτές προσπαθεί να δώσει η ανάπτυξη μιας οπτικοποίησης των New York Times Research and Development Labs.

Σταδιακά δημιουργήθηκε μια οπτικοποίηση που δείχνει την ημερήσια κίνηση στο nytimes.com και στο mobile.nytimes.com πάνω σε ένα παγκόσμιο χάρτη και έναν των ΗΠΑ.

Τα χαρακτηριστικά που χρησιμοποιήθηκαν για δεδομένα είναι η σφραγίδα ημερομηνίας - ώρας από κάθε επίσκεψη στο web και στο mobile site, για μια περίοδο 24 ωρών, και το γεωγραφικό μήκος και πλάτος του κάθε χρήστη.

Οι χάρτες που χρησιμοποιήθηκαν για φόντο στην οπτικοποίηση φτιάχτηκαν από ακριβείς αναπαραστάσεις των ηνωμένων πολιτειών και του παγκόσμιου χάρτη και από ένα dataset του UCLA's CENS group που τυπώνει τις συντεταγμένες κάθε πόλης του κόσμου (Εικόνα 15).

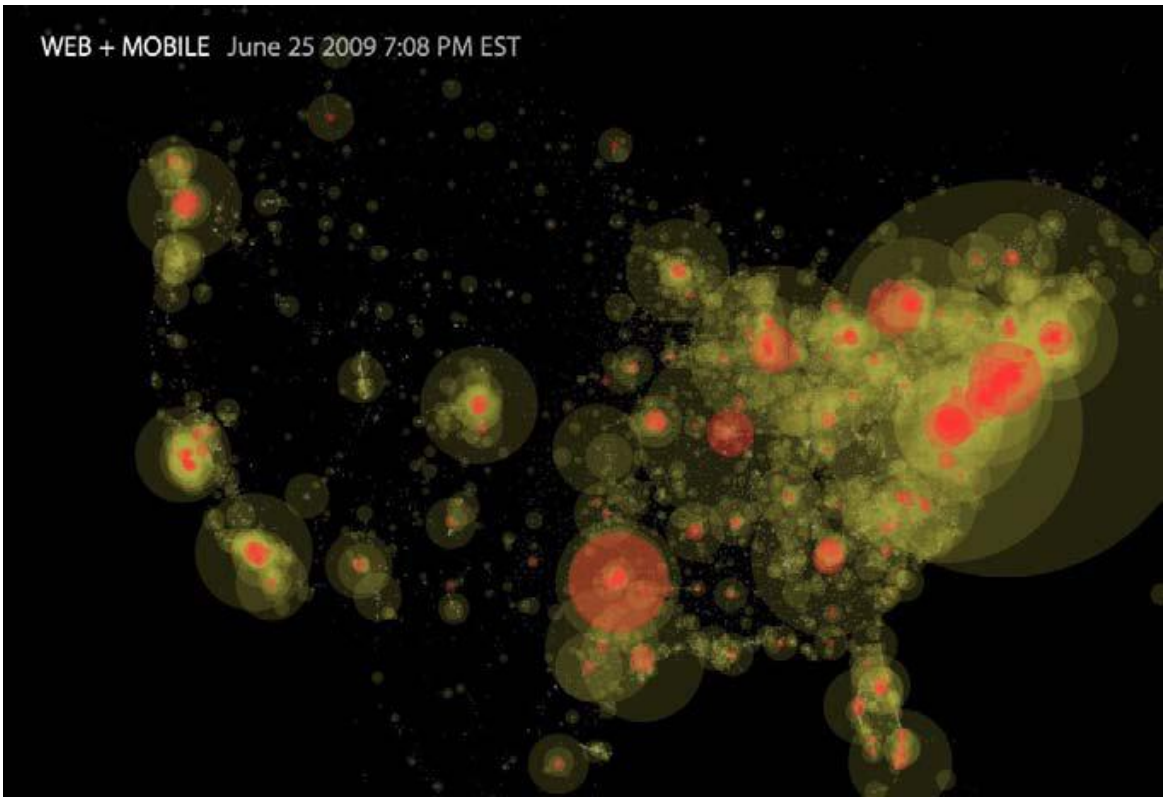


Εικόνα 15 Χάρτης με τις πόλεις του κόσμου

Σε επόμενο στάδιο προστέθηκαν στην οπτικοποίηση κίτρινες και μπλε κουκκίδες που αναπαριστούν επισκέψεις στο nytimes.com και στο mobile.nytimes.com αντίστοιχα (Εικόνα 16).

Εικόνα 16 Χάρτης με τις πόλεις των οποίων οι χρήστες επισκέφθηκαν το nytimes.com και το mobile.nytimes.com

Στην τελευταία φάση η οπτικοποίηση μέσω βίντεο δείχνει την προοδευτικά αυξανόμενη επίσκεψη από τα διάφορα σημεία του πλανήτη. Αυτό επιτυγχάνεται με την εμφάνιση φυσαλίδων που σταδιακά μεγαλώνουν στα σημεία απ όπου γίνονται επισκέψεις ανάλογα με τον αριθμό αυτών. Η επίσκεψη στο nytimes.com φαίνεται με κίτρινο και στο Mobile.nytimes.com με κόκκινο (Εικόνα 17).



Εικόνα 17 Στιγμιότυπο από το βίντεο με την εξέλιξη τωνεπισκέψεων παγκοσμίως στο nytimes.com και το mobile.nytimes.com

2.7.5 Yellow pages και Netflix

Η “Αναζήτηση και Ανακάλυψη” (Search and discovery) είναι δύο τύποι ανάκτησης πληροφορίας. Η αναζήτηση είναι μια γνώριμη φόρμα, πολύ καλό παράδειγμα της οποίας είναι το google και άλλες δικτυακές μηχανές αναζήτησης. Παρότι υπάρχει μια προοπτική ανακάλυψης στις μηχανές αναζήτησης, υπάρχουν πιο ακριβή παραδείγματα συστημάτων ανακάλυψης (discovery systems), όπως οι προτάσεις προϊόντων του Amazon και οι υποδείξεις ταινιών στο Netflix.

Αυτοί οι δύο τύποι συστημάτων ανάκτησης έχουν ως κοινό ότι μπορεί να είναι απίστευτα περίπλοκα στο εσωτερικό τους. Τα αποτελέσματα που παρέχουν εξαρτώνται όχι μόνο από το περιεχόμενο της ερώτησης (query) και των στοιχείων που ανακτώνται, αλλά επίσης και από τη συνολική συμπεριφορά των χρηστών του συστήματος. Για παράδειγμα, το πώς και ποιές ταινίες αξιολογείς στο Netflix θα επηρεάσουν τις ταινίες που προτείνονται στους άλλους χρήστες, και στο Amazon, το να διαβάσεις την περίληψη ενός βιβλίου, το να το αγοράσεις ή ακόμη το να το βάλεις στο καλάθι σου και μετά να το βγάλεις μπορεί να επηρεάσει τις υποδείξεις

που θα δοθούν στους υπόλοιπους χρήστες. Ομοίως στο Google, όταν επιλέγεις ένα αποτέλεσμα, ή όταν δεν το επιλέγεις, αυτή η συμπεριφορά έχει επιπτώσεις σε μελλοντικά αποτελέσματα αναζήτησης. Συνέπεια αυτής της πολυπλοκότητας είναι η δυσκολία να εξηγηθεί η συμπεριφορά τους συστήματος. Πρωτίστως βασιζόμαστε σε μετρικές απόδοσης για να ποσοτικοποιήσουμε την επιτυχία ή αποτυχία των αποτελεσμάτων της ανάκτησης, ή για να δούμε ποιές παραλλαγές ενός συστήματος λειτουργούν καλύτερα από άλλες. Αυτές οι μετρικές επιτρέπουν τη συνεχή βελτίωση του συστήματος.

Μια συμπληρωματική προσέγγιση στην κατανόηση της συμπεριφοράς αυτών των συστημάτων είναι η οπτικοποίηση πληροφορίας. Με την οπτικοποίηση, μπορούμε μερικές φορές να κερδίσουμε βαθύτερη γνώση που δεν αποκτάται από τις μετρικές μονάχα.

Σχετικά με το σύστημα του yellowpages.com στόχος είναι να πάρουμε μια σφαιρική οπτική από τη δραστηριότητα των χρηστών (τα ερωτήματα που υποβάλουν στην ιστοσελίδα), δραστηριότητα η οποία μπορεί να χρησιμοποιηθεί για τη βελτίωση του σχεδίου του συστήματος.

Το δεύτερο σύστημα είναι φτιαγμένο από το dataset του Netflix Prize (ένας διαγωνισμός προγνωστικού σχεδιασμού (predictive modeling) με έπαθλο ένα εκατομμύριο δολάρια που έληξε σχετικά πρόσφατα) και προτείνει ταινίες. Η οπτικοποίηση μπορεί να μας βοηθήσει να καταλάβουμε εγγενή ζητήματα σε ένα μοντέλο ανακάλυψης (discovery model) βασισμένο στις προτιμήσεις του χρήστη.

Η τεχνική οπτικοποίησης:

Η οπτικοποίηση γίνεται για να συγκριθούν τα στοιχεία του ίδιου τύπου – ερωτήματα (queries) στο πρώτο παράδειγμα, και ταινίες στο δεύτερο. Ο συλλογισμός είναι απλός: τα στοιχεία θα τοποθετηθούν στη σελίδα έτσι ώστε τα παρόμοια να βρίσκονται κοντά το ένα με το άλλο και τα ανόμοια αρκετά μακριά. Αυτή η πρόταση βασίζεται στην αρχή Gestalt της εγγύτητας, που ισχυρίζεται ότι όταν δύο αντικείμενα είναι τοποθετημένα κοντά το ένα με το άλλο, οι άνθρωποι τείνουν να τα αντιλαμβάνονται σαν να ανήκουν αυτά στην ίδια ομάδα.

Το πρώτο βήμα σε αυτές τις οπτικοποιήσεις είναι επομένως ο ορισμός των χαρακτηριστικών που καθορίζουν ποια στοιχεία είναι όμοια και ποια ανόμοια. Αυτά μπορεί να είναι οποιαδήποτε. Στο παράδειγμα του Netflix Prize, η εγγύτητα των ταινιών ορίστηκε με βάση τις αξιολογήσεις των χρηστών, αλλά εναλλακτικά θα

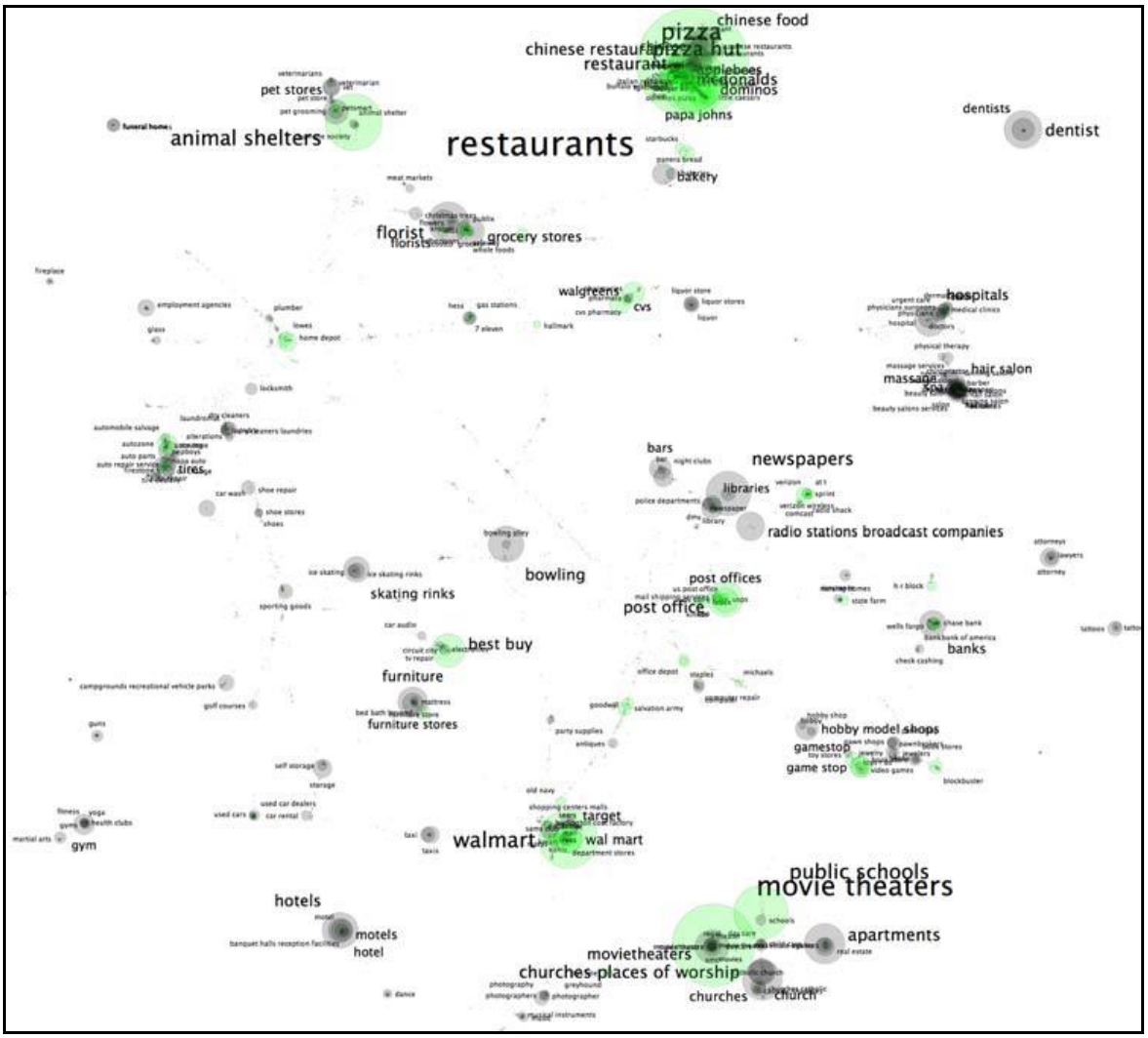
μπορούσαν να χρησιμοποιηθούν χαρακτηριστικά των ταινιών όπως το είδος ή οι ηθοποιοί.

Αφού οριστεί η ομοιότητα, απαιτείται μια διαδικασία για να μετατραπεί σε τιμές συντεταγμένων με 2 ή 3 διαστάσεις. Υπάρχουν κυρίως δύο τρόποι για να γίνει αυτή η κανονικοποίηση. Ο πρώτος είναι με τη χρήση ενός τύπου που μετατρέπει ένα χώρο ανώτερων διαστάσεων σε έναν χαμηλότερων, δύο ή τριών, διαστάσεων. Η εναλλακτική προσέγγιση είναι να εμφανιστούν τα στοιχεία σαν να είναι κόμβοι ενός γράφου, με τα όμοια να ενώνονται με ακμές.

Στη συνέχεια, με την κανονικοποίηση γίνεται μια προσπάθεια να τοποθετηθούν οι συνδεδεμένοι κόμβοι κοντά ο ένας στον άλλο, και οι αποσυνδεδεμένοι μακριά.

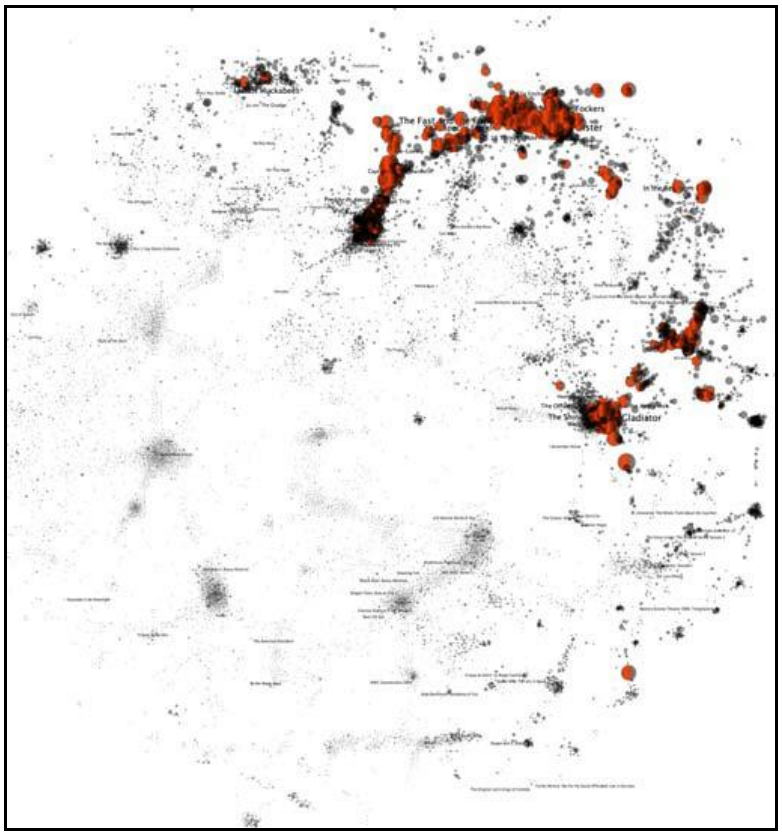
Μετά την κανονικοποίηση – που είναι η ανάθεση συντεταγμένων στα στοιχεία- αναπαραστάσεις των στοιχείων αυτών (απλοί κύκλοι και στις δύο περιπτώσεις) τοποθετούνται σε αυτές τις συντεταγμένες. Τα τελικά βήματα για να δημιουργηθούν οι οπτικοποιήσεις, απαιτούσαν να περιληφθούν και οι τίτλοι των στοιχείων, επικαλύπτοντας οποιαδήποτε επιπλέον ανάλυση.

Στην περίπτωση του yellowpages δύο ερωτήματα θεωρούνται παρόμοια αν στην περίπτωση που κάποιος χρήστης εισάγει είτε το πρώτο, είτε το δεύτερο, είναι πιθανόν και τις δύο φορές να επιλέξει τελικά την ίδια κατηγορία επιχειρήσεων. Η διάταξη γίνεται με αυτό τον ορισμό ομοιότητας δεδομένο.



Εικόνα 18 Οπτικοποίηση του yellowpages.com

Στην περίπτωση του Netflix, χρησιμοποιήθηκε ένα γνωστό μέτρο ομοιότητας σε πολλά συστήματα συστάσεων (recommendation systems), που είναι η ομοιότητα συνημιτόνου (cosine similarity). Στην περίπτωση των ταινιών, διαισθητικά, το μέτρο δείχνει ότι δύο ταινίες είναι παρόμοιες αν χρήστες που αξιολόγησαν με υψηλή βαθμολογία τη μία, έκαναν το ίδιο και στην άλλη, ή αντιθέτως αν βαθμολόγησαν με χαμηλό βαθμό τη μία έκαναν το αντίστοιχο και στην άλλη.



Εικόνα 19 Οπτικοποίηση του Netflix

2.8 Αντικείμενο Πτυχιακής: Οπτικοποίηση γνωσιακού πεδίου

Η παρούσα πτυχιακή ασχολείται με την οπτικοποίηση γνωσιακού πεδίου (knowledge domain visualization – KDV). Ο όρος knowledge domain visualization (ο οποίος φαίνεται πιο δόκιμος και θα επιλέγεται όταν χρειάζεται) αναφέρεται στην κατασκευή «χαρτών εννοιών» (concept maps) που οπτικοποιούν τη δομή και την εξέλιξη ενός επιστημονικού πεδίου. (Θα προτιμάται ο αγγλικός όρος “concept map”, αντί για «χάρτης εννοιών» καθώς και ο όρος “concept” αντί της λέξης «έννοια»).

Η οπτικοποίηση πεδίου (domain visualization) στοχεύει να αποκαλύψει το εύρος της επιστημονικής επικοινωνίας όπως καθρεφτίζεται μέσα από την σύνθεση της επιστημονικής βιβλιογραφίας και των μονοπατιών των αναφορών που υπάρχουν μέσα σε δημοσιεύσεις επιστημόνων. (Katy Börner et al., Annual Review of Information Science & Technology, Volume 37)

Πιο συγκεκριμένα, μέσα από την πορεία της παρούσας εργασίας, βασική επιδίωξη ήταν να παραχθεί ένα χάρτης που να απεικονίζει χαρακτηριστικούς όρους πληροφορικής που βρίσκονται μέσα στο δημοσιοποιημένο έργο τεσσάρων

καθηγητών του τμήματος πληροφορικής του ΑΤΕΙΘ, και τη σχέση μεταξύ αυτών. Μέσα από αυτήν την οπτικοποίηση μπορεί να φανεί το ερευνητικό πεδίο με το οποίο έχει ασχοληθεί ο κάθε καθηγητής, καθώς και το ποιά είναι η σχέση του με αυτά των υπολοίπων.

Η ιδέα για αυτήν την οπτικοποίηση είναι βασισμένη στην εργασία των Van Eck, N.J., Waltman, L., Van den Berg, J., & Kaymak, U. (2006). Στην εν λόγω εργασία εφαρμόζεται οπτικοποίηση (KDV) στο πεδίο της υπολογιστικής νοημοσύνης (Computational intelligence – CI). Συγκεκριμένα δίνεται έμφαση σε αυτό που λέγεται concept map. Αυτοί οι χάρτες απεικονίζουν συσχετίσεις μεταξύ των concepts σε ένα επιστημονικό πεδίο. Κατασκευάζεται ένα concept map βασισμένο στα abstracts των δημοσιεύσεων που έγιναν δεκτά για παρουσίαση στο 2006 IEEE World Congress on Computational Intelligence (WCCI 2006), το οποίο είναι ένα συνδετικό συνέδριο μεταξύ των 2006 International Joint Conference on Neural Networks (IJCNN 2006), του 2006 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2006), και του 2006 IEEE Congress on Evolutionary Computation (CEC 2006). Τα τρία υποπεδία του CI είναι τα νευρωνικά δίκτυα, τα fuzzy systems, και το evolutionary computation, καθένα από τα οποία έχει το δικό του υποσυνέδριο μέσα στο WCCI 2006. Γίνεται η υπόθεση ότι αφού τα τρία πιο σημαντικά υποπεδία του CI έχουν το καθένα το δικό του υποσυνέδριο μέσα στο WCCI 2006, είναι επόμενο ότι ένα concept map βασισμένο στις περιλήψεις (abstracts) των δημοσιεύσεων του WCCI 2006, θα παρέχει μια περιεκτική σύνοψη του πεδίου CI. Αυτός ο χάρτης υποδεικνύει τα σημαντικά θέματα του πεδίου CI και τη μεταξύ τους σχέση. Προκειμένου να επιτραπεί μια ανάλυση της πρόσφατης ανάπτυξης στον τομέα του CI, παρουσιάζεται και ένας χάρτης βασισμένος στα abstracts των δημοσιεύσεων που παρουσιάστηκαν στο WCCI 2002. Η σύγκριση των δύο χαρτών αποκαλύπτει την εισαγωγή νέων concepts στο πεδίο του CI, το ενδιαφέρον σε συγκεκριμένα θέματα, και τις διαφορές στις σχέσεις μεταξύ των θεμάτων.

2.9 Επίλογος

Στο κεφάλαιο 2 καλύψαμε κάποια βασικά θεωρητικά στοιχεία που είναι χρήσιμα για να κατανοήσει κανείς τον σκοπό της οπτικοποίησης και το πρόβλημα

με το οποίο ασχολείται η παρούσα εργασία.

Αναφερθήκαμε στην οπτικοποίηση ως πεδίο, τα χαρακτηριστικά και τους σκοπούς της, σε εργαλεία που διευκολύνουν την υλοποίηση της και σε παραδείγματα που προσφέρουν στο να καλυφθούν όσα κενά αφήνει μια αφαιρετική θεώρηση.

Βλέποντας τα παραπάνω γίνεται φανερό ότι η οπτικοποίηση δεδομένων εμπλέκει πολλά επιστημονικά πεδία της πληροφορικής και όχι μόνο, όπως η ανάλυση κειμένου, οι βάσεις δεδομένων, η εξόρυξη πληροφορίας, η στατιστική, τα γραφικά και ο προγραμματισμός.

ΚΕΦΑΛΑΙΟ 3 - ΑΝΑΛΥΣΗ ΚΑΙ ΣΧΕΔΙΑΣΜΟΣ

3.1 ΕΙΣΑΓΩΓΗ

Το ζητούμενο από την αρχή ήταν η δημιουργία ενός concept map που να απεικονίζει τα πεδία έρευνας των τεσσάρων καθηγητών του ΑΤΕΙΘ και τη μεταξύ αυτών συσχέτιση. Τα concepts αναγκαστικά αντιπροσωπεύονται από σημαντικούς όρους για τον τομέα της πληροφορικής, που έχουν εξαχθεί από τον όγκο των abstracts των δημοσιεύσεων των καθηγητών. Αυτοί οι όροι οπτικοποιημένοι, φαίνονται ως χρωματιστές φυσαλίδες. Κάθε μία φυσαλίδα είναι τόσο μεγάλη ανάλογα με το πόσες περιλήψεις περιέχουν έστω και μία φορά τον όρο τον οποίο αντιπροσωπεύει. Το χρώμα υπάρχει για να γίνεται διακριτό το ποιός καθηγητής ασχολείται με το εκάστοτε αντικείμενο.

Η ανάλυση της διαδικασίας που οδήγησε στην οπτικοποίηση αυτή βασίζεται κυρίως στην εργασία των van Eck και Ludo Waltman (Van Eck et al., 2006), στα βήματα που όρισαν και στη γενικότερη προσέγγιση τους, και λιγότερο στα βήματα που αναφέρει ο Ben Fry (Ben Fry 2008) και παρατίθενται στο 2ο κεφάλαιο, αν και σε πολλά σημεία συμπίπτουν.

Σκόπιμο επομένως είναι να γίνει μια σύντομη παρουσίαση των βημάτων της εργασίας τους, ώστε να γίνει φανερό αργότερα, πώς προσαρμόστηκαν αυτά για να δομηθεί η παρούσα πτυχιακή. Θα παρουσιαστούν επίσης λεπτομερώς οι ενέργειες που έγιναν σε κάθε ένα από αυτά τα βήματα καθώς και οι ειδικές θεωρήσεις που χρειάστηκαν για να συμβολιστούν τα αντικείμενα του προβλήματος και τα επιμέρους στοιχεία του, που με τη σειρά τους εμφανίζονται στις εισόδους, στους αλγόριθμους και στις εξόδους τους.

3.2 Η πορεία που ακολουθεί η δημοσίευση “Visualizing the WCCI 2006 Knowledge Domain”

3.2.1 Σύντομη αναφορά στα βήματα

Όπως αναφέρθηκε και στο υποκεφάλαιο 2.8 που είναι σχετικό με το knowledge domain visualization, η δημοσίευση των Van Eck, N.J., Waltman, L., Van den Berg, J., & Kaymak, U. (2006) ασχολείται με την οπτικοποίηση του πεδίου ονόματι computational intelligence, χρησιμοποιώντας τις περιλήψεις των δημοσιεύσεων που έγιναν δεκτές προς παρουσίαση στο 2006 IEEE World Congress on Computational Intelligence (WCCI 2006). Τα βήματα που ακολούθησαν οι συγγραφείς για να προσεγγίσουν το ζήτημα φαίνονται συνοπτικά στον παρακάτω πίνακα (Εικόνα 20):

TABLE I
SUMMARY OF THE WAY IN WHICH THE KDV PROCESS IS IMPLEMENTED IN THIS PAPER

Step of the KDV process	Implementation
(1) Collection of data	Corpus of WCCI 2006 abstracts
(2) Selection of type of item	Concepts
(3) Extraction of information	Co-occurrence frequencies (Paragraph II-A)
(4) Calculation of similarities	Association strengths (Paragraph II-B)
(5) Positioning of items	VOS (Paragraph II-C)
(6) Visualization	Concept map viewer (Paragraph II-D)

Εικόνα 20 Πίνακας με τα βήματα της εργασίας (Van Eck et al., 2006)

Σύμφωνα πάντα και με τον πίνακα, το πρώτο βήμα είναι ξεκάθαρο. Ως σύνολο δεδομένων ορίστηκαν οι περιλήψεις των δημοσιεύσεων, όπως αναφέρθηκε και παραπάνω.

Δεύτερο βήμα είναι ο εντοπισμός των στοιχείων που έχουν κάποιο ενδιαφέρον, μέσα από το σύνολο των ακατέργαστων δεδομένων. Περιοδικά, δημοσιεύσεις, συγγραφείς, περιγραφικοί όροι και λέξεις επιλέγονται συνήθως ως ο τύπος του αντικειμένου προς ανάλυση. Κάθε είδος αντικειμένου παρέχει μια διαφορετική οπτικοποίηση ενός επιστημονικού πεδίου και καταλήγει σε διαφορετική ανάλυση. Στην εν λόγω εργασία το προς ανάλυση αντικείμενο είναι το “concept” (στα ελληνικά «έννοια», αλλά όπως αναφέρθηκε και στο κεφάλαιο 2 θα

χρησιμοποιείται ο αγγλικός όρος).

Τρίτο βήμα είναι η εξαγωγή της σχετικής πληροφορίας από τα ακατέργαστα δεδομένα που συλλέχθηκαν στο πρώτο βήμα. Στην προκειμένη περίπτωση τη σχετική πληροφορία αποτελεί η συχνότητα των “συμπτώσεων” (co-occurrence frequency) των concepts εντός της ίδιας περίληψης. Η συχνότητα συμπτώσεως ανάμεσα σε δύο concepts υπολογίζεται από το σύνολο των περιλήψεων του WCCI 2006 στις οποίες εμφανίζονται και τα δύο. Για να ταυτοποιηθούν τα concepts αυτά μέσα σε μία περίληψη, χρειάζεται ένα λεξικό όρων (thesaurus) του γνωσιακού πεδίου που ενδιαφέρει. Επειδή ένα λεξικό για το CI δεν ήταν εξ αρχής διαθέσιμο, κατασκευάστηκε ένα από την αρχή. Η προσέγγιση για την κατασκευή του θα αναφερθεί παρακάτω.

Το τέταρτο βήμα του KDV (αναλυτικά τι σημαίνει στο 2.8) είναι ο υπολογισμός της ομοιότητας μεταξύ των αντικειμένων, που βασίζεται στην πληροφορία που εξάγεται από το τρίτο βήμα. Ο υπολογισμός δηλαδή, της ομοιότητας μεταξύ των concepts βασίζεται στις συχνότητες συμπτώσεων. Στη βιβλιογραφία, δύο προσεγγίσεις υπάρχουν για υπολογισμό ομοιότητας μεταξύ αντικειμένων βασισμένες στις συχνότητες συμπτώσεων. Η μία είναι η κανονικοποίηση των συχνοτήτων αυτών, χρησιμοποιώντας για παράδειγμα, τους συντελεστές Dice, Jaccard ή την ομοιότητα συνημιτόνου (cosine similarity) **(**H. P. F. Peters and A. F. J. van Raan, 1993). Η άλλη προσέγγιση είναι να χρησιμοποιηθεί συσχέτιση Pearson (Pearson correlation) μεταξύ των διανυσμάτων των συχνοτήτων συμπτώσεως δύο αντικειμένων ως μέτρο για την μεταξύ τους ομοιότητα. Στην εν λόγω εργασία ακολουθείται η πρώτη προσέγγιση. Παρόλα αυτά δε χρησιμοποιείται κάποιο από τα υπάρχοντα μέτρα για την κανονικοποίηση των συχνοτήτων συμπτώσεως. Αντιθέτως προτείνεται ένα νέο κριτήριο που λέγεται “βάρος συσχέτισης” (association strength).

Το πέμπτο βήμα είναι η τοποθέτηση των στοιχείων σε ένα χώρο λίγων διαστάσεων, βασισμένη στις ομοιότητες που υπολογίστηκαν στο προηγούμενο βήμα. Στην εν λόγω εργασία, τα αντικείμενα ονομάζονται concepts και ο χώρος λίγων διαστάσεων λέγεται concept map. Επιπλέον, η τοποθέτηση των concepts μέσα σε έναν concept map βασίζεται στα βάρη συσχετίσεων, και μόνο οι χάρτες δύο διαστάσεων μελετώνται. Σε πολλές μελέτες το πέμπτο βήμα της οπτικοποίησης γνωσιακού πεδίου εκτελείται με τη χρήση πολυδιάστατου scaling. Παρόλα αυτά, η εμπειρία δείχνει ότι αυτό δεν παρέχει πάντα ικανοποιητικά

αποτελέσματα όταν χρησιμοποιείται για το KDV. Για αυτό το λόγο θα χρησιμοποιηθεί μια εναλλακτική λύση αντι του πολυδιάστατου scaling που ονομάζεται VOS (αναλυτικά στο 3.2.2.3).

Το έκτο βήμα της διαδικασίας είναι η οπτικοποίηση του χώρου λίγων διαστάσεων που προκύπτει από το πέμπτο βήμα. Για την υλοποίηση αυτού του βήματος χρησιμοποιείται ένα πρόγραμμα που το αποκαλούμε concept map viewer. Επιπλέον έχει αναπτυχθεί και ένα άλλο πρόγραμμα που εμφανίζει έναν χάρτη που λέγεται concept density map (χάρτης πυκνότητας). Πρόκειται για έναν concept map στον οποίο χρώματα χρησιμοποιούνται για να υποδηλώσουν την πυκνότητα των concepts. Ο τρόπος με τον οποίο υπολογίζεται η πυκνότητα περιγράφεται σε άλλη δημοσίευση (N. J. van Eck, F. Frasincar, and J. van den Berg, “Visualizing concept associations using concept density maps,” in Proc. 10th International Conference on Information Visualization, 2006, accepted for publication.). Στο κεφάλαιο της υλοποίησης παρατίθενται και παραδείγματα από density maps που παράγονται από τον VOS viewer.

3.2.2 Σημεία - κλειδιά αναλυτικά

3.2.2.1 Λεξικό (thesaurus) του πεδίου CI

Για να κατασκευαστεί ένα λεξικό του πεδίου CI χρησιμοποιήθηκε ένα εργαλείο που αναπτύχθηκε από τους συγγραφείς της δημοσίευσης αυτής. Το εν λόγω εργαλείο δέχεται ως είσοδο έναν όγκο από περιλήψεις δημοσιεύσεων σχετικών με το CI πεδίο. Χρησιμοποιώντας το MontyLingua, αναθέτει σε κάθε λέξη του συνόλου μια ετικέτα ανάλογα με το μέρος του λόγου που είναι η λέξη αυτή. Μετά, βασιζόμενο στην συντακτική δομή των όρων, το εργαλείο διαλέγει λέξεις ή ακολουθίες λέξεων που πιθανώς αποτελούν όρους. Αυτό επιτυγχάνεται με τη χρήση μιας κανονικής έκφρασης. Η έξοδος του εργαλείου είναι μια λίστα υποψήφιων όρων, ταξινομημένη σύμφωνα με τη συχνότητα εμφάνισης του όρου στο σύνολο των κειμένων (corpus). Η λίστα επικυρώνεται χειροκίνητα. Αποφασίζεται για τον κάθε όρο αν είναι σχετικός με το CI. Επιπλέον, όταν θεωρείται σχετικός κάποιος όρος, εντοπίζονται και οι συνώνυμοι του, επίσης χειροκίνητα. Με αυτή τη διαδικασία, δημιουργείται ένα απλό λεξικό του πεδίου CI

που αποτελείται από τους πιο σημαντικούς όρους του πεδίου και τις συνωνυμικές σχέσεις μεταξύ αυτών.

3.2.2.2 Βάρος συσχετίσεως (association strength)

Προτείνεται μια νέα προσέγγιση για τον υπολογισμό των ομοιοτήτων μεταξύ αντικειμένων που βασίζεται στη συχνότητες συμπτώσεων (co-occurence frequencies). Το προτεινόμενο κριτήριο ονομάζεται “βάρος συσχετίσεως”.

Πρώτα παρέχεται μια σημειογραφία. Υποθέτουμε ότι είναι n αντικείμενα, που συμβολίζονται το καθένα από 1,... μέχρι n. Υποθέτουμε επίσης ότι είναι ένας n × n πίνακας C = (c_{ij}) των συχνοτήτων συμπτώσεων. Για i ≠ j, το στοιχείο c_{ij} του C συμβολίζει τη συχνότητα συμπτώσεων των αντικειμένων i και j. Το στοιχείο c_{ii} της κύριας διαγωνίου του πίνακα C συμβολίζει τον αριθμό των εμφανίσεων του αντικειμένου i. Ο μέγιστος αριθμός των φορών που ένα αντικείμενο μπορεί να εμφανιστεί, συμβολίζεται με m. Στην εργασία που αναφερόμαστε, τα αντικείμενα είναι concepts και η συχνότητα συμπτώσεων των αντικειμένων i και j αντιστοιχεί στον αριθμό των περιλήψεων μέσα στις οποίες και τα δύο αυτά concepts εμφανίζονται. Επιπροσθέτως, ο αριθμός των εμφανίσεων του αντικειμένου i αντιστοιχεί στον αριθμό των περιλήψεων στις οποίες το concept i εμφανίζεται, και το m αντιστοιχεί στον συνολικό αριθμό των περιλήψεων.

Χρησιμοποιώντας την παραπάνω σημειογραφία, εισάγεται τώρα το κριτήριο του βάρους συσχέτισης. Το A = (a_{ij}) θεωρούμε ότι συμβολίζει έναν πίνακα n × n των βαρών συσχετίσεων. Το στοιχείο a_{ij} του A συμβολίζει το βάρος συσχέτισης των αντικειμένων i και j και ορίζεται ως $a_{ij} = m\ c_{ij} / c_{ii}\ c_{jj}$ για i ≠ j. (1)

Τα στοιχεία στην κύρια διαγώνιο του A ισούνται με 0. Ανεπίσημα, το βάρος συσχέτισης δύο αντικειμένων μπορεί να ερμηνευθεί ως η πραγματική πιθανότητα συμπτώσεως των αντικειμένων, αναφορικά με την πιθανότητα συμπτώσεως που κάποιος θα περίμενε, αν δεν υπάρχει ιδιαίτερη σχέση μεταξύ των αντικειμένων. Αυτό μπορεί να ιδωθεί ως εξής. Θεωρούμε ότι το $p_i = c_{ii} / m$ συμβολίζει τον υπολογισμό της πιθανότητας εμφάνισης του αντικειμένου i, και το $p_{ij} = c_{ij} / m$ έστω ότι συμβολίζει τον υπολογισμό της πιθανότητας συμπτώσεως των αντικειμένων i και j, όπου i ≠ j. Λέμε ότι δεν υπάρχει ιδιαίτερη σχέση μεταξύ των αντικειμένων i και j αν οι εμφανίσεις του αντικειμένου i είναι στατιστικά ανεξάρτητες από τις εμφανίσεις του αντικειμένου j. Υποθέτοντας ότι οι εμφανίσεις του αντικειμένου i είναι ανεξάρτητες των εμφανίσεων του j, η πιθανότητα συμπτώσεως αυτών των

δύο υπολογίζεται από τη σχέση $p^*_{ij} = p_i\ p_j$. Το βάρος συσχετίσεως των αντικειμένων i και j ορίζεται ως η υπολογισμένη πιθανότητα συμπτώσεως αυτών όταν δε γίνεται η υπόθεση ανεξαρτησίας, σε σχέση με την υπολογισμένη πιθανότητα σύμπτωσης τους όταν γίνεται η υπόθεση αυτή. Το συγκεκριμένο βάρος αποτυπώνεται από τη σχέση $a_{ij} = p_{ij}/p^*_{ij}$, που είναι ισοδύναμη με την (1).

Με άλλα λόγια, το βάρος συσχέτισης δύο αντικειμένων είναι ίσο με την υπολογισμένη πιθανότητα συμπτώσεως, κανονικοποιημένη για την υπολογισμένη πιθανότητα συμπτώσεως που προκύπτει υπό την υπόθεση ότι δεν υπάρχει ιδιαίτερη σχέση μεταξύ των αντικειμένων.

3.2.2.3 VOS

Σε αυτήν την παράγραφο παρουσιάζεται η καινούρια αυτή μέθοδος στην οποία αναφερθήκαμε προηγουμένως, για την τοποθέτηση των αντικειμένων σε έναν χαμηλών διαστάσεων χώρο με βάση τις ομοιότητές τους. Η μέθοδος ονομάζεται VOS, που είναι μια συντομογραφία για το “visualization of similarities”. Μια πιο λεπτομερής συζήτηση του VOS, συμπεριλαμβανομένης και μιας ανάλυσης της σχέσης μεταξύ του VOS και του πολυδιάσταστου scaling, βρίσκεται στη δημοσίευση “N. J. van Eck and L. Waltman, “VOS: a new method for visualizing similarities between objects,” Erasmus University Rotterdam, Erasmus Research Institute of Management, Tech. Rep. ERS-2006-020-LIS, 2006.”

Θεωρούμε ότι υπάρχουν n αντικείμενα, που συμβολίζονται το καθένα από 1,... μέχρι n. Ας θεωρήσουμε επίσης έναν n×n πίνακα ομοιοτήτων $S = (s_{ij})$ που ικανοποιεί ότι $s_{ij} \geq 0$, $s_{ii} = 0$ και $s_{ij} = s_{ji}$ για κάθε $i,j \in \{1,\ldots,n\}$. Το στοιχείο s_{ij} του S συμβολίζει την ομοιότητα μεταξύ του αντικειμένου i και j. Υποτίθεται ότι οι ομοιότητες στον S μπορούν να θεωρηθούν ως μετρήσεις σε μια κλίμακα αναλογίας. Στην εν λόγω εργασία, οι ομοιότητες μεταξύ των αντικειμένων ανταποκρίνονται στα βάρη συσχετίσεων των concepts όπως ορίστηκαν στην προηγούμενη ενότητα, και γι’ αυτό ο πίνακας ομοιοτήτων S ισούται με τον πίνακα A των βαρών συσχέτισης. Το VOS στοχεύει στην παροχή ενός χώρου λίγων διαστάσεων στον οποίο τα αντικείμενα 1, . . . , n είναι τοποθετημένα με τέτοιο τρόπο ώστε η απόσταση μεταξύ οποιουδήποτε ζεύγους αντικειμένων i και j να αντανακλά την μεταξύ τους ομοιότητα s_{ij} όσο γίνεται ακριβέστερα. Τα αντικείμενα που έχουν μεγάλη ομοιότητα πρέπει να τοποθετούνται κοντά το ένα στο άλλο, ενώ

τα αντικείμενα που έχουν χαμηλή ομοιότητα πρέπει να τοποθετούνται μακρία το ένα από το άλλο. Ο n × p πίνακας X, όπου το p συμβολίζει τον αριθμό των διαστάσεων του χώρου που χρησιμοποιείται, περιλαμβάνει τις συντεταγμένες των αντικειμένων 1, . . . , n. Το διάνυσμα $x_i = (x_{i1}, . . . , x_{ip}) \in R^p$ συμβολίζει την i-οστή σειρά του X και περιέχει τις συντεταγμένες του αντικειμένου i. Η ιδέα του VOS είναι να ελαχιστοποιήσει ένα σταθμισμένο άθροισμα του τετραγώνου των ευκλείδιων αποστάσεων μεταξύ όλων των ζευγών των αντικειμένων. Όσο πιο μεγάλη είναι η ομοιότητα μεταξύ δύο αντικειμένων, τόσο πιο μεγάλο είναι το βάρος του τετραγώνου της απόστασής τους στην άθροιση. Για να αποφευχθούν λύσεις στις οποίες όλα τα αντικείμενα θα τοποθετούνται στις ίδιες συντεταγμένες, επιβάλεται ο περιορισμός ότι το άθροισμα όλων των αποστάσεων πρέπει να είνα ίσο με κάποια θετική σταθερά. Με μαθηματικό συμβολισμό, η «αντικειμενική» συνάρτηση προς ελαχιστοποίηση στο VOS δίνεται από τη σχέση:

$$E(\mathbf{X};\mathbf{S}) = \sum_{i<j} s_{ij}\|\mathbf{x}_i - \mathbf{x}_j\|^2 \qquad (2)$$

όπου το || . || συμβολίζει τον ευκλείδειο κανόνα. Η ελαχιστοποίηση της αντικειμενικής συνάρτησης εκτελείται υποκείμενη στον ακόλουθο περιορισμό

$$\sum_{i<j} \|\mathbf{x}_i - \mathbf{x}_j\| = 1. \qquad (3)$$

Να σημειωθεί ότι οι αποστάσεις $|| x_i – x_j ||$ στον περιορισμό δεν είναι εις το τετράγωνο.

Σημειώνουμε επίσης ότι όταν οπτικοποιούμε ομοιότητες φαίνεται φυσικό να περιμένουμε ότι κάθε αντικείμενο i τοποθετείται κοντά σε αυτό που λέμε “ιδανικές του συντεταγμένες”, οι οποίες δίνονται από τη σχέση:

$$\mathbf{x}_i^*(\mathbf{X},\mathbf{S}) = \frac{\sum_j s_{ij}\mathbf{x}_j}{\sum_j s_{ij}}. \qquad (4)$$

Με άλλα λόγια, κάθε αντικείμενο i ενδέχεται να τοποθετηθεί κοντά σε ένα σταθμισμένο μέσο όρο των συντεταγμένων όλων των άλλων αντικειμένων, όπου οι συντεταγμένες των αντικειμένων που είναι πιο όμοια με το αντικείμενο i, έχουν μεγαλύτερο βάρος στον υπολογισμό αυτού. Να τοποθετηθεί το κάθε αντικείμενο ακριβώς στις ιδανικές του συντεταγμένες $x^*_i(X,S)$ είναι εφικτό μόνο με την τοποθέτηση όλων των αντικειμένων στις ίδιες συντεταγμένες, που ξεκάθαρα δεν οδηγεί σε μια χρήσιμη λύση. Αντί να τοποθετείται κάθε αντικείμενο ακριβώς στις ιδανικές του συντεταγμένες, η αντικειμενική συνάρτηση στο (2) τοποθετεί τα αντικείμενα κοντα στις ιδανικές τους συντεταγμένες. Αυτό μπορεί να φανεί ως εξής. Ας υποθέσουμε ότι οι συντεταγμένες όλων των αντικειμένων εκτός από κάποιο αντικείμενο i είναι σταθερές. Η ελαχιστοποίηση της αντικειμενικής συνάρτησης στο (2) απλοποιείται στην ελαχιστοποίηση της σχέσης:

$$E_i(\mathbf{x}_i; \mathbf{X}, \mathbf{S}) = \sum_j s_{ij} \|\mathbf{x}_i - \mathbf{x}_j\|^2. \quad (5)$$

Η ελαχιστοποίηση του (5) μπορεί να εκτελεσθεί αναλυτικά και καταλήγει στη λύση $x_i = x^*_i(X,S)$. Με άλλα λόγια, αν οι συντεταγμένες όλων των αντικειμένων εκτός από κάποιο αντικείμενο i, είναι σταθερές, τότε η αντικειμενική συνάρτηση του VOS θα τοποθετήσει το αντικείμενο i ακριβώς στις ιδανικές του συντεταγμένες. Φυσικά, τα αντικείμενα δεν έχουν σταθερές συντεταγμένες, και γι'αυτό η αντικειμενική συνάρτηση του VOS γενικά δεν τοποθετεί τα αντικείμενα ακριβώς στις ιδανικές τους συντεταγμένες. Παρ'όλα αυτά, η κατάσταση με τις σταθερές συντεταγμένες δείχνει καθαρά την τάση του VOS να τοποθετεί τα αντικείμενα κοντά στις ιδανικές τους συντεταγμένες.

Στην εν λόγω εργασία, λύνεται το πρόβλημα περιορισμένης βελτιστοποίησης της ελαχιστοποίησης του (2) υποκείμενου στο (3) σε δύο βήματα. Πρώτα μετατρέπεται το πρόβλημα περιορισμένης βελτιστοποίησης σε ένα πρόβλημα μη περιορισμένης βελτιστοποίησης. Και έπειτα λύνεται το τελευταίο πρόβλημα χρησιμοποιώντας το κουτί εργαλείων βελτιστοποίησης του MATLAB.

3.2.2.4 Concept Map Viewer

Σε αυτή την παράγραφο, συζητούμε συντόμως τον concept map viewer, που είναι ένα applet σε Java για οπτικοποίηση δισδιάστατου concept map. Ο concept map viewer δείχνει την τοποθεσία ενός concept μέσα σε έναν concept map με την εμφάνιση μιας ετικέτας σε εκείνο το σημείο. Αυτή η ετικέτα δείχνει έναν όρο που ταυτοποιεί το concept. Το πρόγραμμα έχει scroll, zoom, και λειτουργία αναζήτησης για να υποστηρίξει την ευρεία εξέταση ενός concept map. Εκτός από την οπτικοποίηση των συσχετίσεων μεταξύ των concepts, το πρόγραμμα επίσης οπτικοποιεί την σημαντικότητα των concepts και την διανομή του ενδιαφέροντος σχετικά με αυτά, μέσα στα τρία πιο σημαντικά υποπεδία του πεδίου CI (που είναι τα νευρωνικά δίκτυα, τα fuzzy systems, και ο εξελικτικός λογισμός (evolutionary computation)). Η σημαντικότητα ενός concept φαίνεται από το μέγεθος της ετικέτας του. Πιο συγκεκριμένα, το μέγεθος της γραμματοσειράς που χρησιμοποιείται στην ετικέτα ενός concept καθορίζεται από μια γραμμική συνάρτηση του τετραγώνου της ρίζας του αριθμού των περιλήψεων στις οποίες εμφανίζεται το concept. Η κατανομή του ενδιαφέροντος για ένα concept πάνω στα υποπεδία του CI συμβολίζεται με το χρώμα της ετικέτας που αντιπροσωπεύει το concept. Το χρώμα συντίθεται από ένα μέρος κόκκινου, πράσινου, και μπλε, κάθε ένα από τα οποία έχει μια τιμή ανάμεσα σε 0 και 255. Θεωρούμε το χρώμα της ετικέτας που αντιπροσωπεύει το concept i. Το κόκκινο, πράσινο, και μπλε συνθετικό του χρώματος δίνονται από τις σχέσεις 6, 7 και 8.

$$r\left(p_i^{\mathrm{FS}}, p_i^{\mathrm{NN}}, p_i^{\mathrm{EC}}; c\right) = c + \frac{p_i^{\mathrm{FS}}}{p_i^{\mathrm{FS}} + p_i^{\mathrm{NN}} + p_i^{\mathrm{EC}}}(255 - c) \quad (6)$$

$$g\left(p_i^{\mathrm{FS}}, p_i^{\mathrm{NN}}, p_i^{\mathrm{EC}}; c\right) = c + \frac{p_i^{\mathrm{NN}}}{p_i^{\mathrm{FS}} + p_i^{\mathrm{NN}} + p_i^{\mathrm{EC}}}(255 - c) \quad (7)$$

$$b\left(p_i^{\mathrm{FS}}, p_i^{\mathrm{NN}}, p_i^{\mathrm{EC}}; c\right) = c + \frac{p_i^{\mathrm{EC}}}{p_i^{\mathrm{FS}} + p_i^{\mathrm{NN}} + p_i^{\mathrm{EC}}}(255 - c) \quad (8)$$

Αντίστοιχα, όπου το p_i^{FS} συμβολίζει το μέρος των περιλήψεων του FUZZIEEE 2006 στις οποίες εμφανίζεται το concept i, το p_i^{NN} συμβολίζει το κλάσμα των περιλήψεων του IJCNN 2006 στις οποίες εμφανίζεται το concept i, και το p_i^{EC}

συμβολίζει το κλάσμα των περιλήψεων του CEC 2006 στις οποίες εμφανίζεται το concept i. Επιπροσθέτως, το c συμβολίζει μια σταθερά που έχει την τιμή 75 στο εν λόγω paper. Χρησιμοποιώντας την (6), (7) και (8), το χρώμα μιας ετικέτας δεν επηρεάζεται από διαφορές στον αριθμό των δημοσιεύσεων που παρουσιάστηκαν σε κάθε ένα από τα τρία υποσυνέδρια του WCCI 2006. (Van Eck et al., 2006)

3.3 Ανάλυση των βημάτων της παρούσας εργασίας

3.3.1 Περιληπτικά

Σε μερική αντιστοιχία με τον πίνακα της προαναφερθείσας εργασίας, για τη δική μας οπτικοποίηση έχουμε τα εξής:

1. Συλλογή των ακατέργαστων δεδομένων – όλες οι περιλήψεις των δημοσιεύσεων τεσσάρων καθηγητών του τμήματος πληροφορικής του ΑΤΕΙΘ (Δέρβος, Σταμάτης, Βίτσας, Διαμαντάρας).
2. Επιλογή του αντικειμένου προς ανάλυση. - Τομείς έρευνας, Concepts, δηλαδή γενικά όροι που σχετίζονται με την πληροφορική ή ανήκουν σ' αυτή.
3. Δημιουργία λεξικού – εξαγωγή κάθε λέξης από τον όγκο των ακατέργαστων δεδομένων, απόδοση βαρών στην καθεμία, και επιλογή αυτών με το μεγαλύτερο βάρος για το τελικό λεξικό.
4. Επινόηση αλγορίθμου (υλοποιείται σε πρόγραμμα Java) για υπολογισμό των συμπτώσεων των όρων του λεξικού μέσα στα abstracts.
5. Τοποθέτηση των αντικειμένων στον δισδιάστατο χώρο, με βάση τις ομοιότητες τους. - Εισαγωγή του πίνακα διανυσμάτων που προκύπτει από το βήμα 4 στον VOS viewer.

Ας σημειωθεί ότι για τους σκοπούς της εργασίας οι λέξεις “δημοσίευση” και “περίληψη”, όταν αναφέρονται σχετιζόμενες με τους τέσσερις καθηγητές του πειράματος, έχουν ταυτόσημη έννοια, αφού για τον καθένα έχουμε συγκεντρώσει μόνο τις περιλήψεις από τις δημοσιεύσεις του, και οι μεν είναι αναγκαστικά ίσες σε πλήθος με τις δε.

3.3.2 Αναλυτικά

Σε αυτό το κομμάτι θα παρουσιαστεί η πορεία, ο συλλογισμός και οι θεωρητικές παραδοχές που έγιναν σε κάθε ένα από τα βήματα της προηγούμενης παραγράφου. Θα εξαιρεθεί το τελευταίο βήμα το οποίο εντάσσεται στο επόμενο κεφάλαιο της εργασίας, «Υλοποίηση - Αποτελέσματα Χρήσης – Αποτίμηση». Θα παρουσιαστεί μόνο μία οπτικοποίηση για να παρουσιαστεί ολοκληρωμένα η διαδικασία.

3.3.2.1 Συλλογή δεδομένων

Αρχικά αυτό που χρειαζόταν, ήταν η συλλογή των ακατέργαστων δεδομένων (raw data). Κατόπιν σύντομης έρευνας στις προσωπικές ιστοσελίδες των καθηγητών, αναζητήθηκαν όλες οι διαθέσιμες δημοσιεύσεις του καθένα, από τις οποίες ήταν ούτως ή άλλως τις περισσότερες φορές διαθέσιμες μόνο οι περιλήψεις (abstracts) αυτών, συνήθως σε μορφή pdf ή ως κείμενο σε ιστοσελίδα (.html).

Τα διαθέσιμα abstracts του κάθε καθηγητή αποθηκεύτηκαν ξεχωριστά το καθένα σε txt αρχεία με τον τίτλο της αντίστοιχης δημοσίευσης ή μέρος αυτού, σε αντίστοιχους καταλόγους με όνομα, το επώνυμο του αντίστοιχου καθηγητή.

3.3.2.2.Αντικείμενο προς ανάλυση / στοιχεία-κλειδιά

Σε αυτό το βήμα γίνεται η επιλογή του αντικειμένου προς ανάλυση - τομείς έρευνας, Concepts, δηλαδή γενικά όροι που σχετίζονται με την πληροφορική ή ανήκουν σ' αυτή. Όπως έχει αναφερθεί ήδη, αυτό που θέλουμε να δούμε είναι τα ερευνητικά πεδία στα οποία δραστηριοποιούνται οι τέσσερις καθηγητές του πειράματος. Επομένως, σε αντίθεση με την προαναφερθείσα εργασία, δεν έχουμε κάποιο συγκεκριμένο πεδίο της επιστήμης της πληροφορικής στο οποίο να θέλουμε να περιοριστούμε.

Στο παρόν πείραμα μας ενδιαφέρει οποιοσδήποτε όρος εμφανίζεται στις δημοσιεύσεις των καθηγητών, και σχετίζεται με την πληροφορική γενικώς. Μπορεί να μην είναι τόσο καίριοι και να μην αντιπροσωπεύσουν όλοι πεδία έρευνας, αλλά

στο σύνολό τους και με την τοποθέτησή στους στο χώρο, μπορούν να αναδείξουν πού κινείται η έρευνα του κάθε καθηγητή, καθώς και η σχέση της με τις έρευνες των υπολοίπων. Επομένως για μας το κάθε concept δεν αντιστοιχεί σε ένα μόνο όρο, αλλά σε ένα σύνολο αυτών ή σε κάποιο cluster που γίνεται εμφανές με την οπτικοποίηση. Αφήνεται στον θεατή, μέσω των βασικών όρων των οποίων η σχέση αποτυπώνεται, να αναγνωρίσει τα concepts μέσα στους χάρτες.

3.3.2.3 Η δημιουργία του λεξικού

Επόμενο βήμα είναι η κατασκευή ενός λεξικού με τους όρους, τους σχετικούς με την επιστήμη της πληροφορικής. Το μέγεθος του λεξικού και το πώς αυτό επηρεάζει το αποτέλεσμα, θα συζητηθεί και θα φανεί μέσω της οπτικοποίησης παρακάτω.

Στο αρχικό πλάνο ήταν να εξαχθούν όλα τα ουσιαστικά από τα κείμενα, να αφαιρεθούν τα stop words και να ταξινομηθούν οι εναπομείναντες όροι με βάσει τα tf-idf βάρη που τους έχουν αποδοθεί. Γι αυτό τον λόγο έγινε επεξεργασία της κάθε περίληψης με τη χρήση του MontyLingua. Το πρόγραμμα δεχόταν ως είσοδο το κείμενο, και παρήγαγε μια έξοδο της μορφής:

([ετικέτα σύνταξης] [λέξη ή φράση]/[ετικέτα γραμματικής] [ετικέτα σύνταξης])
για κάθε λέξη και φράση της εισόδου.

Ακολουθεί ένα παράδειγμα εξόδου του MontyLingua όταν δεν υπάρχει είσοδος, (όπου A# ο αύξων αριθμός μιας πρότασης, αντίστοιχα με την εμφάνιση της στην περίληψη), που περιλαμβάνει ρήματα, ουσιαστικά, επίθετα και τις αντίστοιχες φράσεις όπως τις αναγνώρισε το MontyLingua.

SENTENCE #A DIGEST:
adj_phrases: []
adj_phrases_tagged: []
modifiers: []
modifiers_tagged: []
noun_phrases: []
noun_phrases_tagged: []

parameterized_predicates: []
prep_phrases: []
prep_phrases_tagged: []
verb_arg_structures: []
verb_arg_structures_concise: []
verb_phrases: []
verb_phrases_tagged: []
None
[]

GENERATED SUMMARY: (περίληψη που παράγει το πρόγραμμα)

Για παράδειγμα θέτουμε ως είσοδο την περίληψη της δημοσίευσης του κ. Διαμαντάρα με τίτλο “Asymmetric PCA Neural Networks for Adaptive Blind Source Separation”:

We show that second order cross-coupled Hebbian rule used for asymmetric principal component analysis is capable of blindly and adaptively separating uncorrelated sources. Our method enjoys the following advantages over similar higher-order models such as those performing independent component analysis: 1) the strong independence assumption about the source signals is reduced to the weaker uncorrelation assumption; 2) there is no constraint on the sources PDFs, i.e., we remove the assumption that at most one signal is Gaussian; 3) the higher order statistical optimization methods are replaced with second order methods with no local minima; and 4) the kurtosis of the sources becomes irrelevant. Simulation experiments shows that the model successfully separates source images with kurtoses of different signs

ως έξοδο από το montylingua (για την πρώτη πρόταση) έχουμε:

(NX We/PRP NX) (VX show/VBP VX) that/IN (NX second/JJ order/NN cross-coupled/NNP Hebbian/NNP rule/NN NX) (VX used/VBN VX) for/IN (NX asymmetric/JJ pri
ncipal/JJ component/NN analysis/NN NX) (VX is/VBZ VX) (NX capable/JJ NX)

of/IN blindly/RB (NX and/CC adaptively/NN NX) (VX separating/VBG VX) (NX unco
rrelated/NN sources/NNS NX) ./.

SENTENCE #1 DIGEST:

adj_phrases: ['capable']
adj_phrases_tagged: ['capable/JJ']
modifiers: ['second', 'asymmetric', 'principal', 'capable', 'blindly']
modifiers_tagged: ['second/JJ', 'asymmetric/JJ', 'principal/JJ', 'capable/JJ', 'blindly/RB']
noun_phrases: ['We', 'second order cross-coupled Hebbian rule', 'asymmetric principal component analysis', 'and adaptively', 'uncorrelated sou
rces']
noun_phrases_tagged: ['We/PRP', 'second/JJ order/NN cross-coupled/NNP Hebbian/NNP rule/NN', 'asymmetric/JJ principal/JJ component/NN analysis/NN', 'a
nd/CC adaptively/NN', 'uncorrelated/NN sources/NNS']
parameterized_predicates: [[['show', []], ['We', []], ['that second order cross-coupled Hebbian rule', ['prep=that']]], [['use', ['past_tense']], ['se
cond order cross-coupled Hebbian rule', []], ['for asymmetric principal component analysis', ['prep=for']]], [['be', []], ['asymmetric principal compo
nent analysis', []], ['capable', []]], [['separate', ['past_tense']], ['and adaptively', []], ['uncorrelated source', ['plural']]]]
prep_phrases: ['that second order cross-coupled Hebbian rule', 'for asymmetric principal component analysis']
prep_phrases_tagged: ['that/IN second/JJ order/NN cross-coupled/NNP Hebbian/NNP rule/NN', 'for/IN asymmetric/JJ principal/JJ component/NN analysis/NN
']
verb_arg_structures: [['show/VBP', 'We/PRP', ['that/IN second/JJ order/NN cross-coupled/NNP Hebbian/NNP rule/NN']], ['used/VBN', 'second/JJ order/NN
cross-coupled/NNP Hebbian/NNP rule/NN', ['for/IN asymmetric/JJ principal/JJ component/NN analysis/NN']], ['is/VBZ', 'asymmetric/JJ principal/JJ compon
ent/NN analysis/NN', ['capable/JJ']], ['separating/VBG', 'and/CC adaptively/NN',

['uncorrelated/NN sources/NNS']]]
verb_arg_structures_concise: ['("show" "We" "that second order cross-coupled Hebbian rule")', '("use" "second order cross-coupled Hebbian rule" "for a symmetric principal component analysis")', '("be" "asymmetric principal component analysis" "capable")', '("separate" "and adaptively" "uncorrelated s ource")']
verb_phrases: ['show', 'used', 'is', 'separating']
verb_phrases_tagged: ['show/VBP', 'used/VBN', 'is/VBZ', 'separating/VBG']

GENERATED SUMMARY:
We showed that second order cross-coupled Hebbian rule.

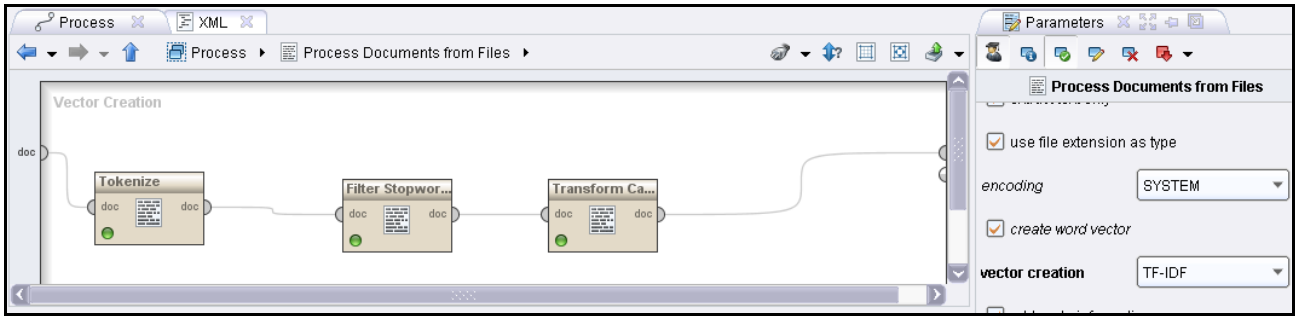
Βλέπουμε ότι το MontyLingua αναγνώρισε τα ρήματα (πχ 'show', 'used', 'is', 'separating'), τα ουσιαστικά, τα επίθετα, κτλ. Επίσης αναγνώρισε πιο περίπλοκες δομές όπως ουσιαστικά με τα επίθετα που τα προσδιορίζουν (πχ. 'We', 'second order cross-coupled Hebbian rule', 'asymmetric principal component analysis', 'and adaptively', 'uncorrelated sources'). Από τη δημιουργημένη σύνοψη φαίνεται ότι η πολυπλοκότητα της πραγματικής γλώσσας ξεπερνά τις δυνατότητες του εργαλείου, αλλά παρόλα αυτά η ανάλυσή που παρέχει είναι ικανοποιητική αφού επιτρέπει να αναδειχθούν οι σημαντικοί όροι του κειμένου.

Κατά αυτόν τον τρόπο προέκυψε ένας όγκος αρχείων ίσων σε πλήθος με αυτόν των αρχείων δημοσιεύσεων, με τη διαφορά ότι τα νέα αρχεία περιλαμβάνουν αυτήν την ανάλυση των περιλήψεων που παρέχει το montyLingua. Τα αρχεία αυτά κρατήθηκαν ξεχωριστά και θα χρησιμοποιηθούν μαζί με τις ακατέργαστες περιλήψεις ως πρώτη ύλη για τον υπολογισμό του tf-idf βάρους του κάθε όρου.

Όπως φαίνεται από το παράδειγμα παραπάνω, η μορφή της εξόδου του montylingua είναι αρκετά περίπλοκη και δύσχρηστη. Μετά από την πρώτη ανάλυση που παρέχει με αυτές τις εξόδους, θα έπρεπε να γραφτεί πρόγραμμα που να αφαιρεί τα stop words και να αποδίδει το tf-idf βάρος σε κάθε λέξη. Θα χρειάζονταν επίσης λειτουργίες ταξινόμησης και μορφοποίησης. Έτσι για να τη δημιουργία του λεξικού χρησιμοποιήθηκε ένα έτοιμο open source εργαλείο, και η συμβολή του montylingua περιορίζεται στη συμπερίληψη των εξόδων του στα

ακατέργαστα δεδομένα, τα οποία χρησιμοποιήθηκαν στο επόμενο βήμα.

Για την εξαγωγή και ταξινόμηση των λέξεων με σκοπό να κρατηθούν οι πιο σημαντικές, χρησιμοποιήθηκε τελικά το RapidMiner, το οποίο αποτελεί πιο ολοκληρωμένη λύση, και διαθέτει όλα τα φίλτρα που χρειάστηκαν αργότερα. Μέσα από τη διεπαφή που διαθέτει, επιλέχθηκε αρχικά η συμβολοποίηση των κειμένων (tokenization), η αλλαγή των γραμμάτων των όρων που ήταν κεφαλαία, σε μικρά (transformation), και ο αποκλεισμός των stop words σύμφωνα με την ενσωματωμένη λίστα που διαθέτει ήδη το πρόγραμμα. Τέλος, επιλέχθηκε, τα διανύσματα που δημιουργούνται για να συμβολίσουν την εμφάνιση κάθε λέξης σε κάθε αρχείο περίληψης, να δημιουργούνται με απόδοση βαρών tf-idf στις λέξεις (Εικόνα 21).



Εικόνα 21 Το εσωτερικό της διεργασίας επεξεργασίας κειμένου του rapid miner.

Με τη χρήση αυτού του προγράμματος, εισήχθησαν για κάθε καθηγητή οι δημοσιεύσεις του και η αντίστοιχη έξοδος του MontyLingua για κάθε μία δημοσίευση, και παράχθηκαν τέσσερις λίστες (μία για κάθε καθηγητή) με ταξινομημένες τις λέξεις κατά φθίνουσα σειρά με βάση το βάρος tf-idf. Από αυτές τις λίστες πήραμε τις πρώτες v λέξεις από την κάθε μία για να φτιάξουμε έναν αρχικό θησαυρό.

Σε αυτό το σημείο πρέπει να αποφασίσουμε δύο πράματα. Πόσο μεγάλος θέλουμε να είναι ο θησαυρός. Με ποιόν τρόπο θα συμμετέχουν οι όροι του κάθε καθηγητή, με βάση κάποιο ποσοστό, στη λίστα.

Το προεπιλεγμένο μέγεθος του θησαυρού, αποφασίστηκε να είναι 100 λέξεις. Στη δημοσίευση στην οποία βασίζεται η εργασία, ο θησαυρός αντιστοιχεί περίπου στο ένα δέκατο (1/10) του συνόλου των περιλήψεων. Αυτό σε εφαρμογή στο δικό μας παράδειγμα θα έδινε ένα αποτέλεσμα 11 λέξεων μιας και το σύνολο των δημοσιεύσεων όλων των καθηγητών είναι 106. Επίσης για μας το concept έχει

διαφορετική έννοια (μπορεί δηλαδή τις περισσότερες φορές να αποτελείται από πολλούς όρους – και εμείς αυτούς οπτικοποιούμε) και επομένως δε μπορεί να προκύψει μια εικόνα, χωρίς να έχουμε αρκετούς όρους οπτικοποιημένους. Για λόγους πειραματισμού εννοείται πως το μέγεθος της λίστας θα μικρύνει, και θα αυξηθεί μέχρις υπερβολής. Τα συγκεκριμένα πειράματα και τα αποτελέσματά τους, περιλαμβάνονται στο επόμενο κεφάλαιο.

Μια λογική σκέψη, η οποία θα εφαρμοστεί ως προεπιλογή στα πειράματα, είναι ότι ο κάθε καθηγητής πρέπει να συμβάλει με διαφορετικό αριθμό περιλήψεων στο δεδομένο σύνολο των 106. Δηλαδή αν π είναι οι περιλήψεις ενός καθηγητή και θ το μέγεθος του θησαυρού του τρέχοντος πειράματος, τότε τ θα είναι οι όροι με το υψηλότερο βάρος tf-idf που θα εξαχθούν από τη λίστα με τις περιλήψεις του εν λόγω καθηγητή, όπου τ = (θ*π)/106. Πιο συγκεκριμένα θα παρατεθούν οι υπολογισμοί για το ποσοστό που «δικαιούται» ο κάθε καθηγητής σύμφωνα με τον κανόνα συμμετοχής που θέσαμε, και το πόσο είναι αυτό για κάποια μεγέθη θησαυρού που θα χρησιμοποιηθούν.

Να σημειωθεί ότι οι αριθμοί δεν είναι απόλυτα ακριβείς γιατί δεν ήταν δυνατό να βρεθούν όλες οι δημοσιεύσεις των καθηγητών, οπότε περιοριζόμαστε σε αυτές που έχουμε, και δεχόμαστε ότι αναλογικά ανταποκρίνονται αρκετά στην πραγματικότητα. Για τον κ. Δέρβο έχουμε 14 περιλήψεις, για τον κ. Διαμαντάρα έχουμε 50 και για τους κυρίους Σταμάτη και Βίτσα, έχουμε 13 και 29 περιλήψεις αντίστοιχα.

Ποσοστά επί του συνόλου των περιλήψεων (106):

Δερβος: 13%

Διαμανταρας: 47%

Βίτσας: 28%

Σταμάτης: 12%

Για τα ανάλογα μεγέθη θησαυρού, δίνονται παρακάτω σε πίνακα, τα πλήθη των λέξεων από τη λίστα του κάθε καθηγητή, που μπορούν να συμμετέχουν στον αντίστοιχο θησαυρό.

Μεγέθη θησαυρού ->	50	150	200	300
Δέρβος	7	20	26	39
Διαμαντάρας	23	70	94	141
Βίτσας	14	42	56	84
Σταμάτης	6	18	24	36

Να σημειωθεί ότι οι αριθμοί είναι στρογγυλοποιημένοι γιατί σχεδόν σε κάθε περίπτωση προκύπτουν δεκαδικά ψηφία. Για μέγεθος θησαυρού 100 και 1000 είναι εντελώς αυτονόητο το πόσοι όροι θα συμμετάσχουν από κάθε καθηγητή.

3.3.2.4 Ο αλγόριθμος που εκτελεί το πρόγραμμα για τη δημιουργία των διανυσμάτων

Θεωρώντας ότι πλέον έχουμε μια λίστα λέξεων (θησαυρό) για τις οποίες πρέπει να διαπιστώσουμε πότε συνυπάρχουν στην κάθε περίληψη, και για να κατασκευάσουμε τα διανύσματα που εκφράζουν τις συμπτώσεις της κάθε μίας με όλες τις υπόλοιπες, δημιουργήθηκε ένα πρόγραμμα σε Java που διαβάζει όλα τα αρχεία με τις περιλήψεις και αναζητά τις λέξεις του θησαυρού, συμπληρώνοντας και γράφοντας τα διανύσματα σε ένα αρχείο. Παρατίθεται πρώτα ο αλγόριθμος σε φυσική γλώσσα μαζί με κάποια σημειογραφία για να διευκολυνθεί η διατύπωση των νοημάτων.

Έχουμε έναν θησαυρό με n όρους.

Οι n όροι του θησαυρού αντιστοιχούν σε έναν αριθμό k από concepts όπου $k \leq n$. (Ένα concept μπορεί να περιγράφεται με παραπάνω από έναν όρους. Επόμενο είναι οι όροι του θησαυρού στην καλύτερη περίπτωση να περιγράφουν ένα διαφορετικό concept ο καθένας που σημαίνει ότι είναι ίδιοι στο πλήθος με τα concepts. Σε κάθε άλλη περίπτωση είναι περισσότεροι.)

Θεωρούμε έναν πίνακα $n \times n$, $C = (c_{ij})$.

Το στοιχείο i,j του πίνακα C είναι ο αριθμός των papers στα οποία υπάρχουν ταυτόχρονα οι όροι του θησαυρού που αντιστοιχούν ο ένας στο στοιχείο i και ο άλλος στο j. (Ο πίνακας C επομένως είναι συμμετρικός ως προς τη διαγώνιο του, αφού C(ij) = C(ji)).

Το στοιχείο C_{ii} είναι ο αριθμός των εμφανίσεων του όρου i δηλαδή ο αριθμός των abstracts στα οποία εμφανίζεται ο όρος.

Ο αριθμός όλων των papers (ή abstracts) συμβολίζεται με m.

Το χρώμα που θα έχει ο όρος στην οθόνη θα εξαρτάται από τον καθηγητή στου οποίου την περίληψη βρέθηκε πριν να τοποθετηθεί στον θησαυρό. Ο όρος μπορεί να εμφανιστεί πάνω από μία φορά αν είχε υψηλό βάρος tf-idf σε δύο και παραπάνω καθηγητές.

Την τοποθέτηση των στοιχείων σε χώρο λίγων διαστάσεων (2 στη προκειμένη περίπτωση) θα την αναλάβει ο VOS viewer, διαβάζοντας ένα αρχείο με τα αντικείμενα (το όνομα του αντικειμένου, και έναν αριθμό που προκύπτει από τον καθηγητή στον οποίο εντοπίστηκε και ο οποίος επηρεάζει το χρώμα της φούσκας που αντιπροσωπεύει το αντικείμενο).

Σύμφωνα με τα παραπάνω που έχουν εξαχθεί από την εργασία (Van Eck et al., 2006) και διατυπώθηκαν ελαφρώς τροποποιημένα, και σύμφωνα με το ζητούμενο της παρούσας εργασίας, κατασκευάστηκε αλγόριθμος που να θεωρεί ως είσοδο όλα τα abstracts που υπάρχουν αποθηκευμένα, τον θησαυρό των λέξεων και να συμπληρώνει τον πίνακα συμπτώσεων C. Ο πίνακας συμπτώσεων αποθηκεύεται σε αρχείο txt. Αποθηκεύεται και ένα ακόμη αρχείο με τους όρους και δίπλα το χρώμα με το οποίο αυτοί θα απεικονίζονται, με συγκεκριμένο τρόπο ώστε να διαβάζονται από τον VOS viewer. Με αυτό τον τρόπο μπορούμε να έχουμε την οπτικοποίηση στην οποία στοχεύουμε.

Τα βήματα του αλγορίθμου είναι τα εξής:

1.Δημιούργησε ένα διάνυσμα V.

2.Δημιούργησε τον πίνακα C.

3.Άνοιξε τον φάκελο με τα abstracts.

4.Άνοιξε το τρέχον abstract.

5.Σύγκρινε την τρέχουσα λέξη του abstract με κάθε όρο.

6.Αν είναι ίδια με κάποιον όρο (ο οποίος δεν έχει εμφανιστεί ακόμη) αποθήκευσε τον αύξοντα αριθμό του όρου* στην τρέχουσα θέση του διανύσματος V.

7.Θέσε ως τρέχουσα θέση του διανύσματος την επόμενη.

8. Θέσε ως τρέχουσα λέξη την επόμενη και πάνε στο βήμα 5 – εκτός αν δεν υπάρχει επόμενη που σημαίνει ότι φτάσαμε στο τέλος του αρχείου.

Μέχρι εδώ αποθηκεύονται σε ένα διάνυσμα οι αύξοντες αριθμοί των όρων που συνυπάρχουν στο τρέχον abstract.

9. Αύξησε κατά 1 την τιμή της θέσης C[V(i),V(i)] για i από 0 έως length(V)-1 . **

Αυξάνουμε κατά 1 τις θέσεις του πίνακα C που αντιστοιχούν στους όρους που βρέθηκαν στο τρέχον abstract και βρίσκονται πάνω στην διαγώνιο του. Αυτό γίνεται ώστε στο τέλος να βρίσκονται στη διαγώνιο του πίνακα οι αριθμοί των περιλήψεων στις οποίες εμφανίστηκε έστω μία φορά ο κάθε όρος i (στοιχείο Cii).

10. Αύξησε κατά 1 την τιμή της θέσης C[V(i),V(j)] για κάθε (i και j) < length(V) && i ≠ j. **

Καταχωρούμε ότι συνέπεσε ο κάθε όρος που εμφανίστηκε στο abstract με όλους τους υπόλοιπους που επίσης εμφανίστηκαν.

11. Μηδένισε το διάνυσμα V (σβήσε τα αποθηκευμένα στοιχεία και θέσε ως τρέχουσα θέση την Νο 0). Θέσε ως τρέχον abstract το επόμενο και πάνε στο βήμα 4 – εκτός αν δεν υπάρχει επόμενο.

12.Γράψε τον πίνακα C σε ένα αρχείο txt**,** χωρίζοντας το κάθε στοιχείο της κάθε σειράς με κόμμα, και κάθε σειρά με αλλαγή γραμμής. ***

* ο αύξων αριθμός του όρου είναι ο αριθμός της θέσης του στη λίστα - θησαυρό

**στην Java η αρίθμηση των δεικτών ενός πίνακα ξεκινάει από το 0 για αυτό αν ένας πίνακας είναι λ θέσεων ο μέγιστος δείκτης έχει τιμή λ-.

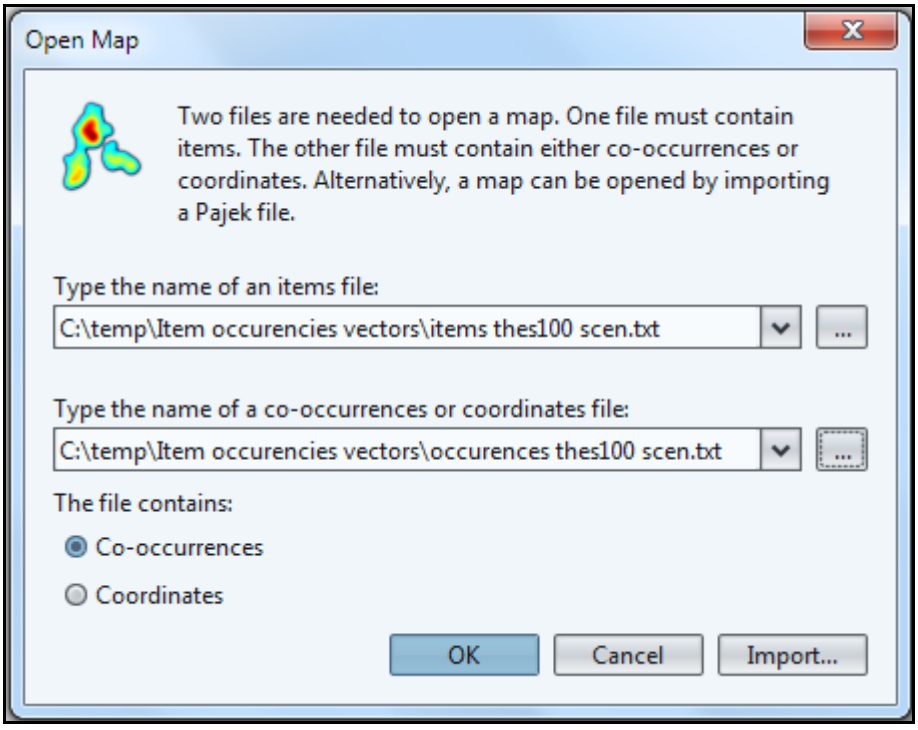
***Το πρόγραμμα διαβάζει επίσης τη λίστα–θησαυρό και γράφει και ένα αρχείο με τους όρους, και δίπλα τους το χρώμα ανάλογα με τον καθηγητή, σε μορφή που να την καταλαβαίνει ο VOS viewer. Αυτή η διαδικασία και πολλές άλλες, φαίνονται διαβάζοντας τον κώδικα, αλλά παραλήφθηκαν από τον αλγόριθμο γιατί

προσθέτουν περιπλοκότητα στην διατύπωση και ξεφεύγουν κάπως από την ουσία.

Με το πέρας της εκτέλεσης τους αλγορίθμου αυτό που μένει, είναι δύο txt αρχεία. Το ένα ας το πούμε “items”, και περιέχει όλα τα αντικείμενα - όρους που θα οπτικοποιηθούν, και το δεύτερο, ας το πούμε “co-occurencies”, περιέχει τα διανύσματα που συμβολίζουν την συνύπαρξη των αντικειμένων. Οι σειρές του πρώτου αρχείου είναι ίσες με τις σειρές και τις στήλες του δεύτερου.

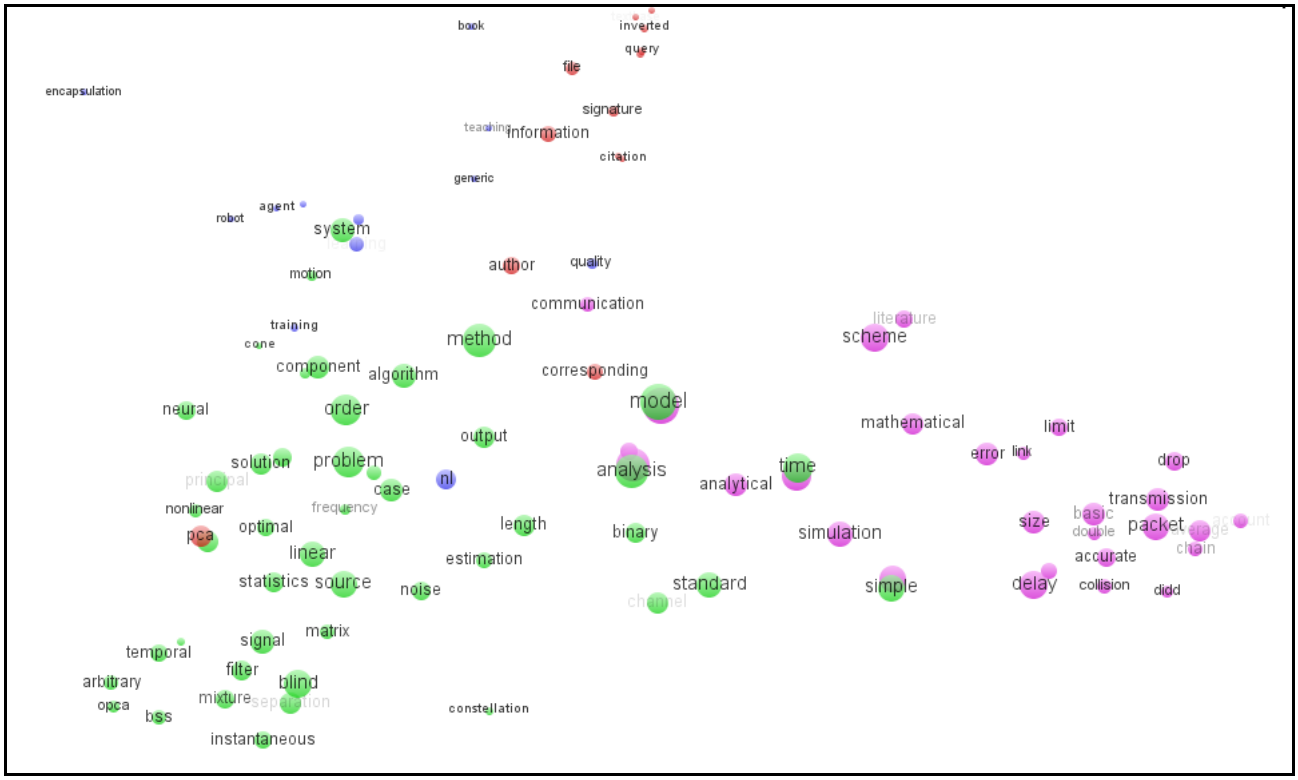
3.3.2.5 Μια πρώτη οπτικοποίηση

Έστω ότι εισάγαμε στο rapid miner τις π περιλήψεις του κάθε καθηγητή, και τις π εξόδους του montylingua. Πήραμε τους όρους του καθένα, ταξινομημένους σύμφωνα με το tf-idf βάρος που τους αποδόθηκε, σε τέσσερις λίστες – μία για τον κάθε καθηγητή. Από αυτές πήραμε τους τ πρώτους από την κάθε μία ανάλογα με τον καθηγητή, και φτιάξαμε μια λίστα από 100 όρους. Το πρόγραμμά μας αναζήτησε κάθε όρο της λίστας μέσα στον όγκο των 106 περιλήψεων και έφτιαξε δύο αρχεία, ένα αρχείο items και ένα αρχείο occurencies. Αυτά τα δύο αρχεία τα εισάγουμε στον VOS viewer (Εικόνα 22).



Εικόνα 22 Εισαγωγή του αρχείου “items” και του αρχείου “occurrences” στον VOS viewer.

Πατώντας “ok” δημιουργείται η εξής εικόνα (Εικόνα 23):



Εικόνα 23 Οπτικοποίηση με 100 λέξεις

3.5 Επίλογος

Παρουσιάσαμε σε αυτό το κεφάλαιο, το πώς εφαρμόζονται αυτά που αναφέρθηκαν στη θεωρία ώστε να επιτευχθεί η ζητούμενη οπτικοποίηση. Φανερώνεται ότι η οπτικοποίηση δεν είναι κάτι στατικό αλλά υπάρχουν άπειρες παράμετροι με τις οποίες μπορεί κανείς να πειραματιστεί ώστε να τροποποιηθεί το αποτέλεσμα, είτε για λόγους έρευνας είτε απλά από περιέργεια. Βλέπουμε ότι η δημοσίευση στην οποία βασιστήκαμε, επηρέασε αρκετά το δρόμο προς το αποτέλεσμα, αλλά παρόλα αυτά, χρειάστηκε σε πολλά σημεία να ακολουθηθεί ένα διαφορετικό μονοπάτι.

ΚΕΦΑΛΑΙΟ 4 - ΥΛΟΠΟΙΗΣΗ- ΑΠΟΤΕΛΕΣΜΑΤΑ ΧΡΗΣΗΣ-ΑΠΟΤΙΜΗΣΗ

4.1 Εισαγωγή

Ακολουθώντας όλα τα βήματα που αναφέρθηκαν, εφαρμόζοντας κάποια κομμάτια της θεωρίας και γράφοντας κώδικα σε Java, οδηγηθήκαμε σε μερικές οπτικοποιήσεις του πεδίου έρευνας τεσσάρων καθηγητών του Τμήματος Πληροφορικής του ΑΤΕΙ Θεσσαλονικής. Σε αυτό το σημείο θα πρέπει να παρατηρήσουμε τη δομή αυτών των εικόνων. Τί βλέπουμε από το στήσιμο των σχημάτων στο χώρο; Τί μας δείχνει το χρώμα τους και ποιά συμπεράσματα προκύπτουν; Στο παρόν κεφάλαιο θα εμβαθύνουμε σε κάποια σημεία, επιχειρώντας να αναλύσουμε αυτό που μας λέει η εικόνα και οι διαφορετικές της οπτικές**,** είτε αυτές προκύπτουν από κάποιες πρόσθετες λειτουργίες του VOS viewer, είτε προκύπτουν από χειροκίνητη τροποποίηση μέσω προγράμματος επεξεργασίας εικόνας, με σκοπό την ανάδειξη των μορφών.

4.2 Υλοποίηση στα κομβικά σημεία (από τα 106 abstracts στη βασική εικόνα)

Κατά την επεξεργασία του θέματος της πτυχιακής, προέκυψαν κάποια ζητήματα που για να διεκπεραιωθούν, απαίτησαν περισσότερη προσοχή και χρόνο. Σε αυτά τα ζητήματα είναι σκόπιμο να σταθούμε ξεχωριστά και πιο αναλυτικά.

4.2.1 Πειράματα – κατασκευη θησαυρών

Επειδή ο όγκος των δεδομένων στα οποία εφαρμόζεται η οπτικοποίηση δεν

αλλάζει, αυτό που διαφοροποιεί τις ποικίλες δοκιμές που πραγματοποιήθηκαν, είναι η λίστα με τις λέξεις που περιλαμβάνονται κάθε φορά στην τελική εικόνα.

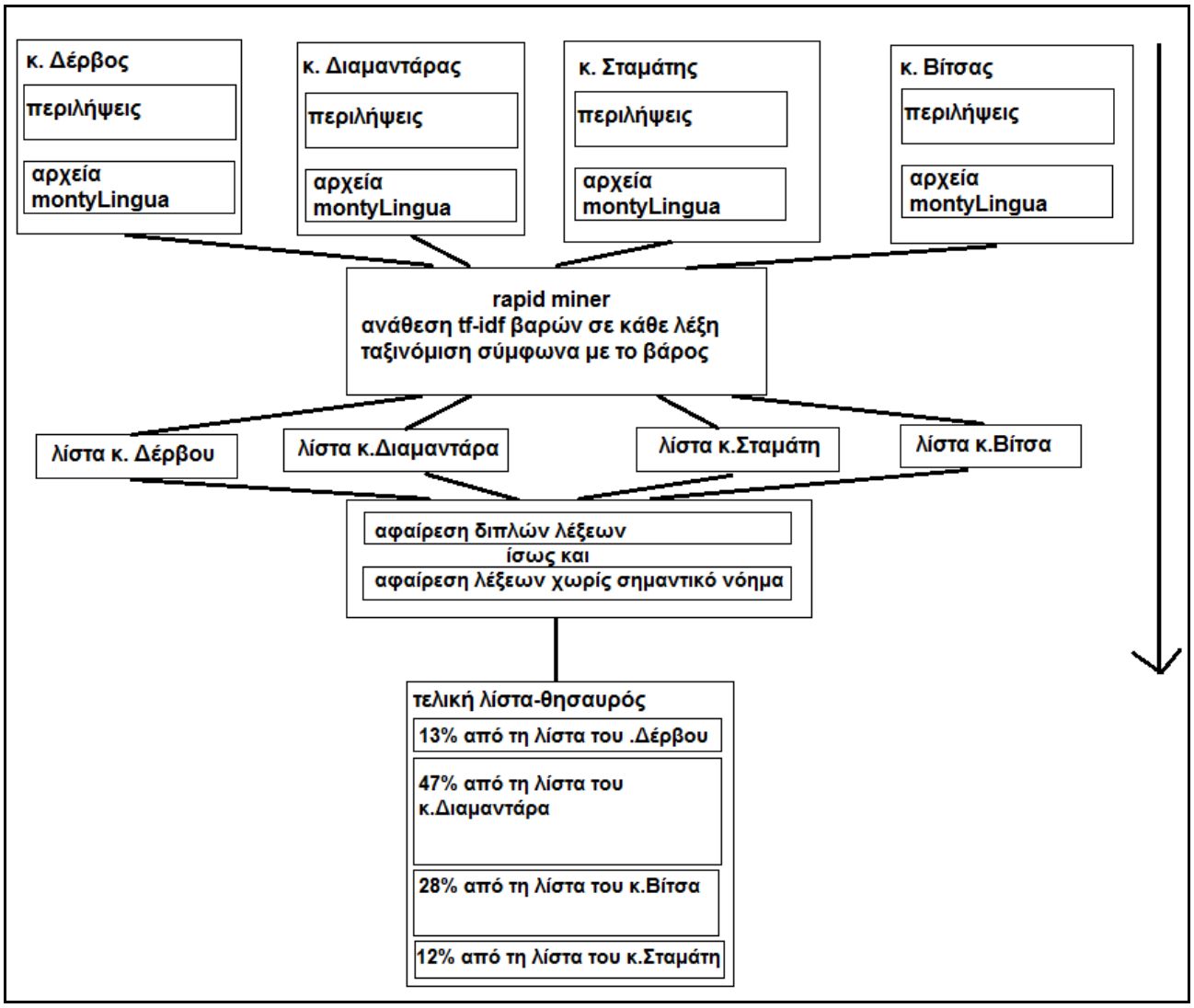
4.2.1.1 Το βασικό πείραμα

Το ορθότερο είναι να ξεκινήσουμε ορίζοντας τα χαρακτηριστικά του βασικού πειράματος. Ή αλλιώς να ορίσουμε τις προδιαγραφές που πρέπει να έχει η λίστα-θησαυρός που μας ενδιαφέρει να χρησιμοποιήσουμε για την παραγωγή της τελικής εικόνας.

Όπως αναφέρεται και στην παράγραφο 3.3.2.3 σχετικά με τον θησαυρό, το βασικό πείραμα θα γίνει χρησιμοποιώντας μια λίστα 100 λέξεων, οι οποίες εξάγονται με ποσοστιαία αναλογία από τις περιλήψεις κάθε καθηγητή αλλά και από τα αρχεία με τις εξόδους του MontyLingua που προέκυψαν από αυτές. Κατά σύμβαση μπορούμε να ονομάσουμε “μεικτές”, τις λίστες που προκύπτουν από τις περιλήψεις και το αποτέλεσμα της επεξεργασίας αυτών από το MontyLingua. Με την ίδια λογική μπορούμε να ονομάσουμε “καθαρές” τις λίστες που προέρχονται μόνο από τον όγκο των περιλήψεων, και “επεξεργασμένες” αυτές που δημιουργήθηκαν με βάση μόνο τα προϊόντα του MontyLingua. (τα αρχεία txt με τις εξόδους του MontyLinngua μπορούμε εν συντομία να τα λέμε “αρχεία MontyLingua”).

Είναι συχνό σχετικά φαινόμενο να εμφανίζονται μέσα στις λίστες, λέξεις που είναι ουσιαστικά ίδιες αλλά η μία στον ενικό και η άλλη στον πληθυντικό. Υπάρχει επίσης η πιθανότητα μία λέξη να έχει υψηλό tf-idf βάρος χωρίς να φέρει κάποια ουσιαστική έννοια για τον κλάδο της πληροφορικής και γενικώς για οποιοδήποτε επιστημονικό πεδίο. Αυτές τις λέξεις μπορούμε να τις αφαιρέσουμε χειροκίνητα από τις λίστες. Αυτός ο αποκλεισμός των λέξεων διαφορετικού αριθμού (πληθυντικού κυρίως) γίνεται σε κάθε περίπτωση. Ας ονομάσουμε όμως “τροποποιημένες” τις λίστες από τις οποίες αφαιρέθηκαν χειροκίνητα λέξεις χωρίς σημασία για την πληροφορική.

Ένα σχήμα που περιγράφει γενικά τη διαδικασία δημιουργίας οποιουδήποτε μεγέθους μεικτής λίστας, είναι το ακόλουθο:



Εικόνα 24 Συνοπτική αναπαράσταση της δημιουργίας του θησαυρού

Η διαδικασία λοιπόν ξεκινά με τον ίδιο τρόπο για κάθε καθηγητή. Εισάγεται στο Rapid Miner ο “μεικτός” όγκος περιλήψεων και των αρχείων MontyLingua, και αποθηκεύεται μια λίστα με όποια διαφορετική λέξη έχει εμφανιστεί μέσα στον όγκο, ταξινομημένη σύμφωνα με το tf-idf βάρος.

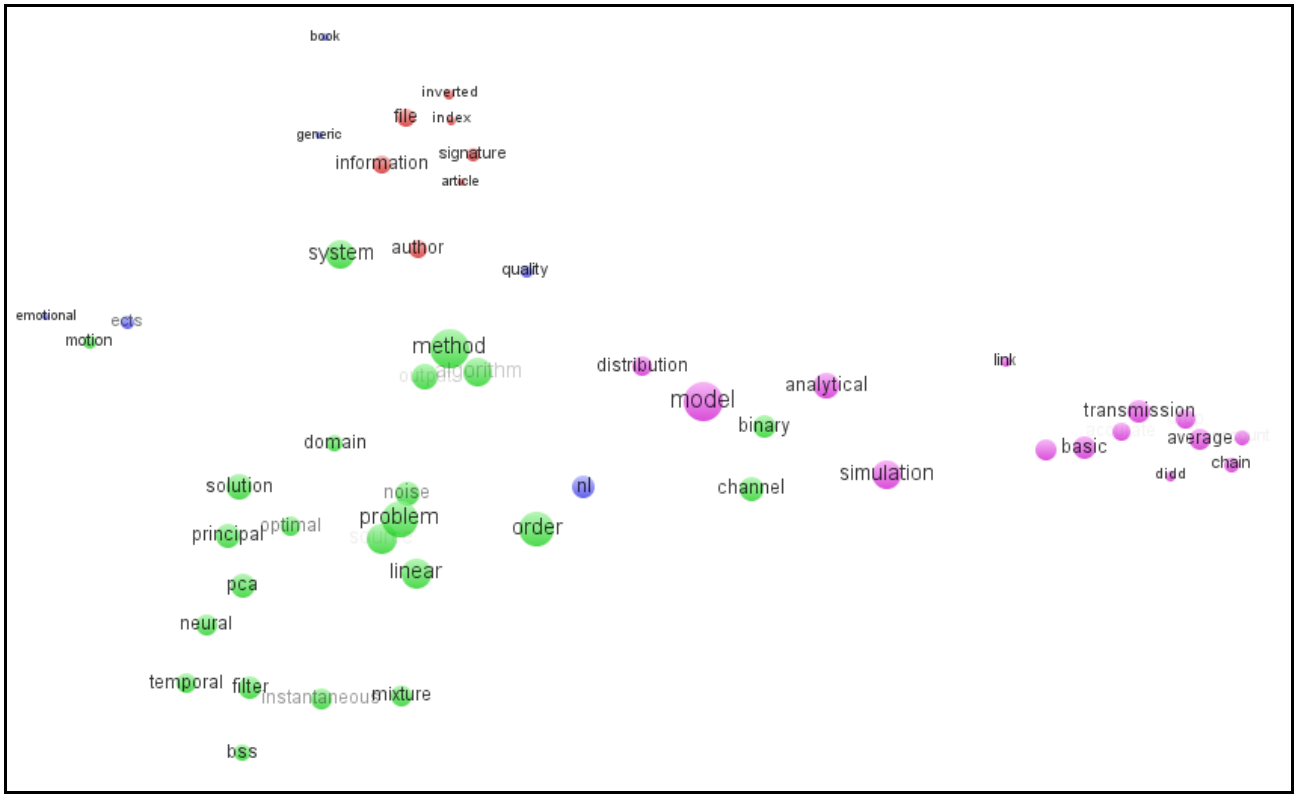
Κοινή αρχή για όλα τα πειράματα επομένως είναι μια ταξινομημένη (με το γνωστό τρόπο) λίστα για κάθε καθηγητή. Θα δείξουμε στη συνέχεια αναλυτικά πώς προκύπτει αυτή η λίστα για κάθε έναν. Οι πλήρεις λίστες θα είναι διαθέσιμες στο ΠΑΡΑΡΤΗΜΑ Β’. Από τις λίστες αφαιρέθηκαν χειροκίνητα οι πληθυντικοί αριθμοί λέξεων που υπήρχαν ήδη στον ενικό. Διαφορετικά, λέξεις που ήταν μόνο στον πληθυντικό, μετατράπηκαν στον ενικό.

Εισάγοντας στον VOS την λίστα με τους 100 όρους που προκύπτει από τη

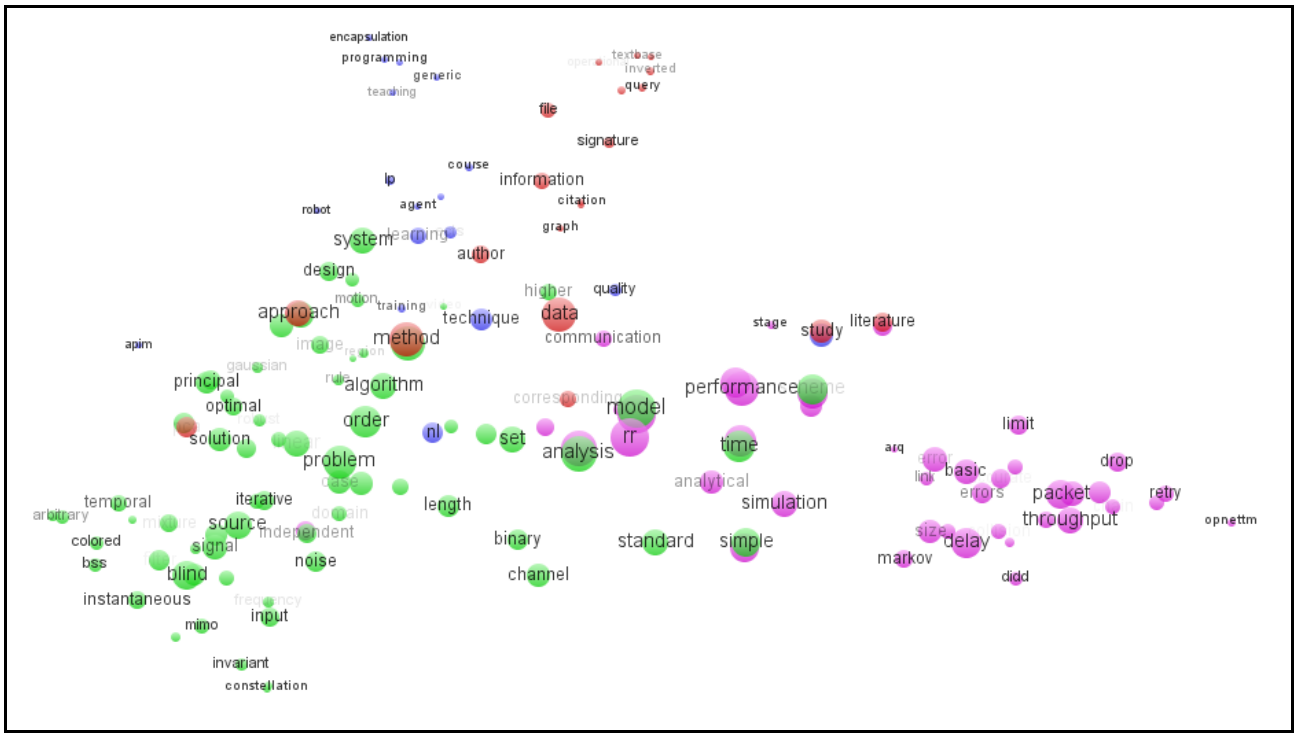
διαδικασία που μόλις περιγράψαμε, μαζί με τα διανύσματα που δημιουργεί το πρόγραμμά μας, παίρνουμε την εικόνα που φαίνεται στην ενότητα 3.3.2.5.

4.2.1.2 Παραλλαγές πειραμάτων (λίστα μεγέθους 50,150,300,200,1000 λέξεων)

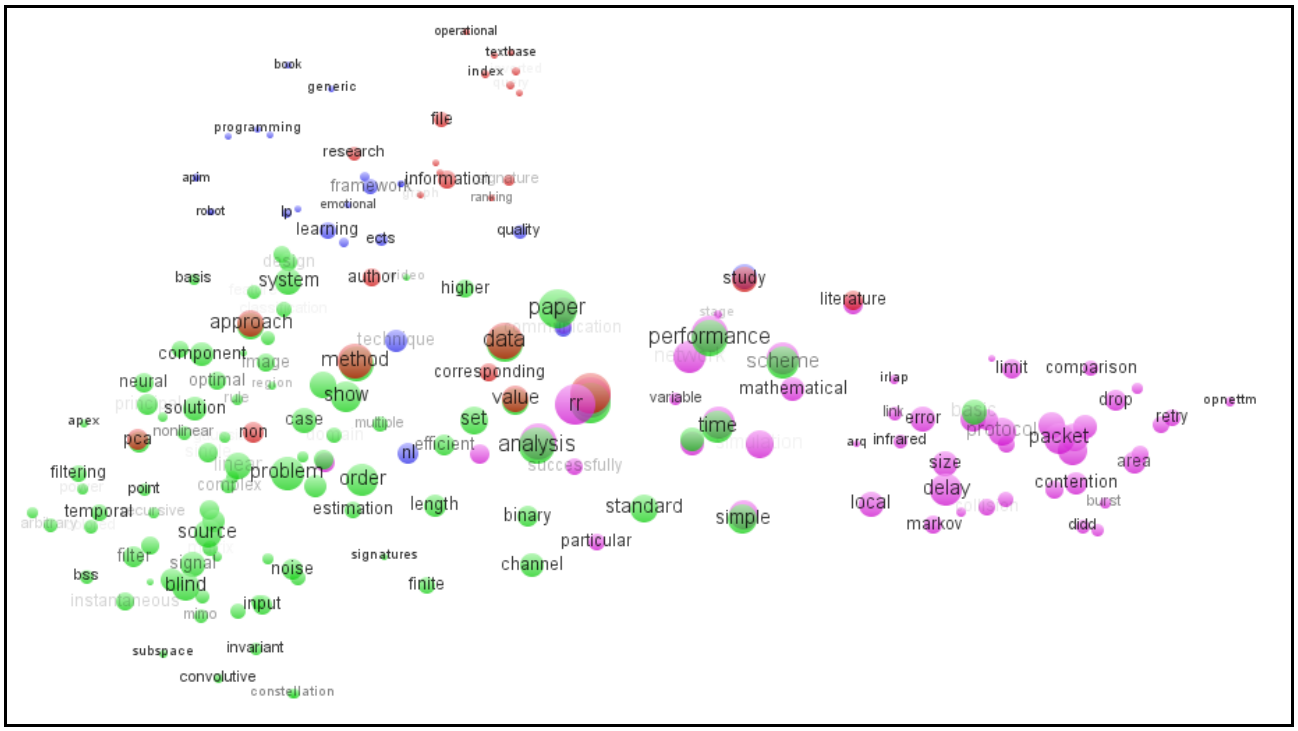
Για να μπορέσουμε να δούμε διαφορετικές εικόνες και να παρατηρήσουμε το πώς αλλάζει η κατανομή των φυσαλίδων στο χώρο, αλλάζουμε το μέγεθος του θησαυρού. Αντιπροσωπευτικά μεγέθη είναι αυτά των 50, 150, 200,300 και 1000 λέξεων. Παρατίθενται οι εικόνες που προκύπτουν όταν εισαχθούν στον VOS viewer οι αντίστοιχες λίστες με τα αντίστοιχα διανύσματα.



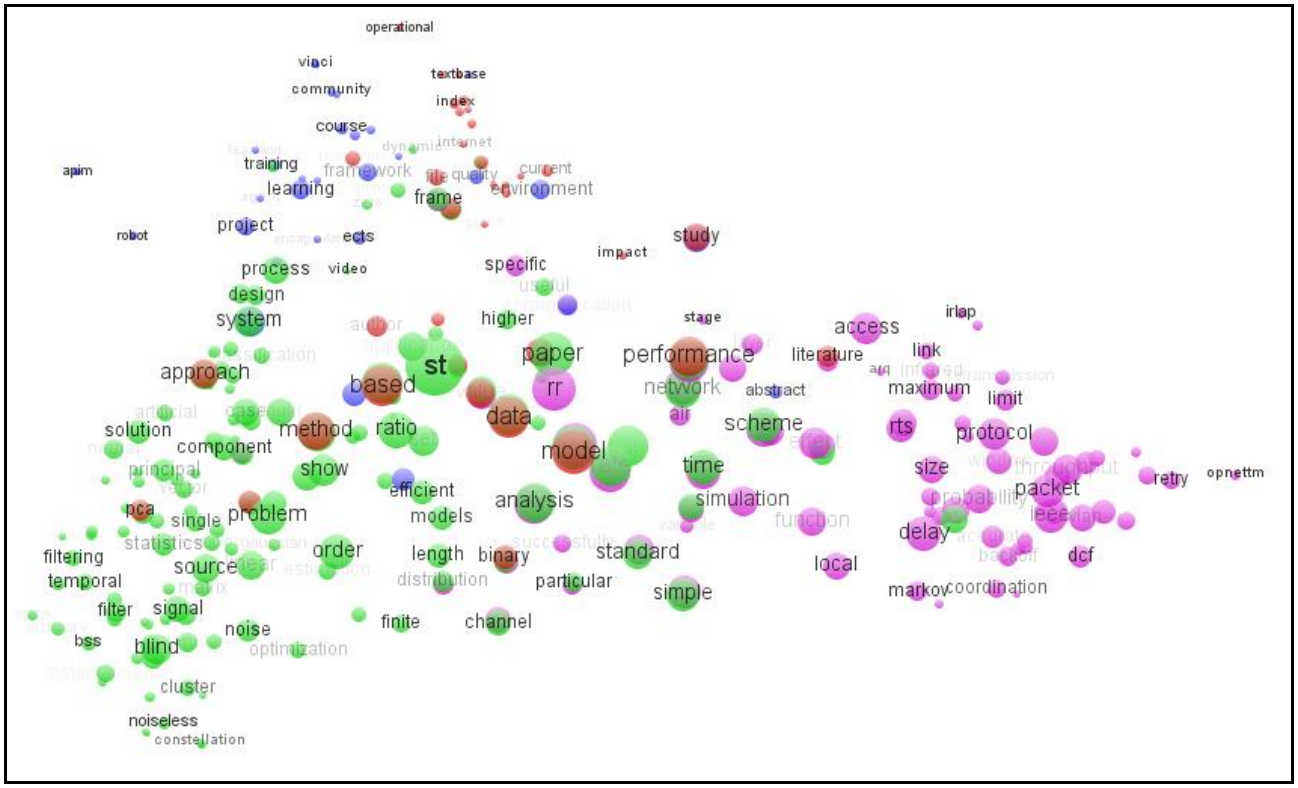
Εικόνα 25 Οπτικοποίηση βασισμένη σε θησαυρό 50 λέξεων



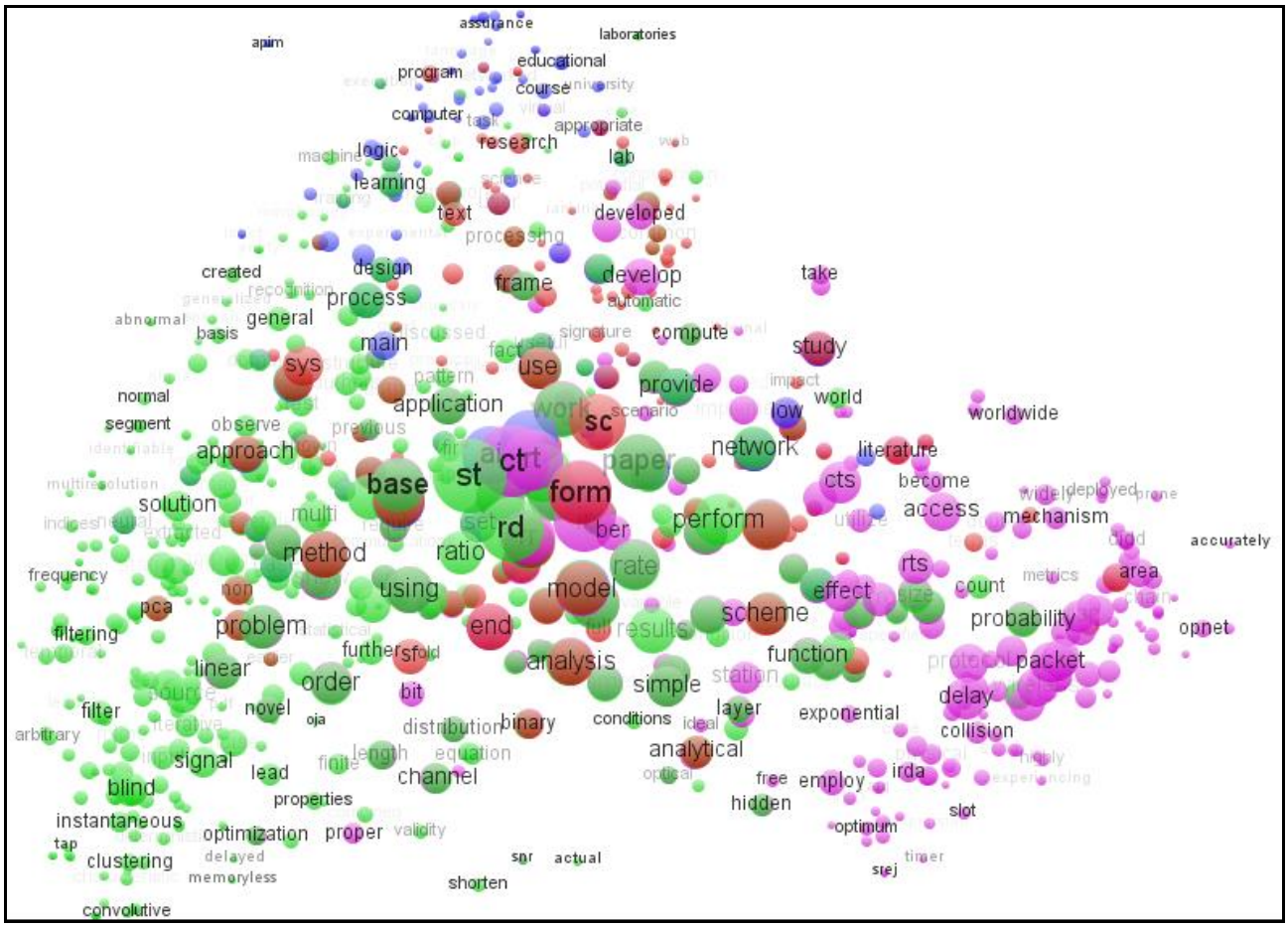
Εικόνα 26 Οπτικοποίηση βασισμένη σε θησαυρό 150 λέξεων



Εικόνα 27 Οπτικοποίηση βασισμένη σε λίστα 200 λέξεων



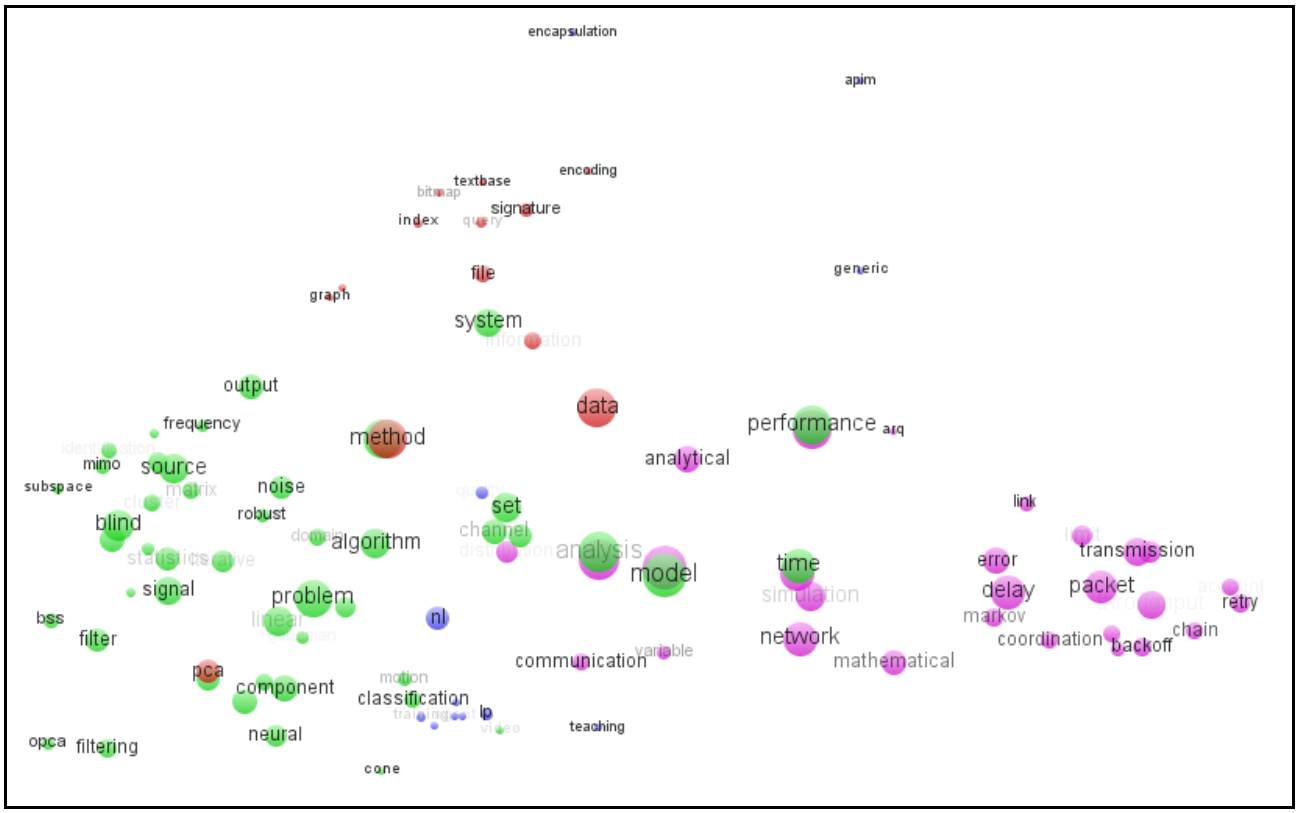
Εικόνα 28 Οπτικοποίηση βασισμένη σε λίστα 300 λέξεων



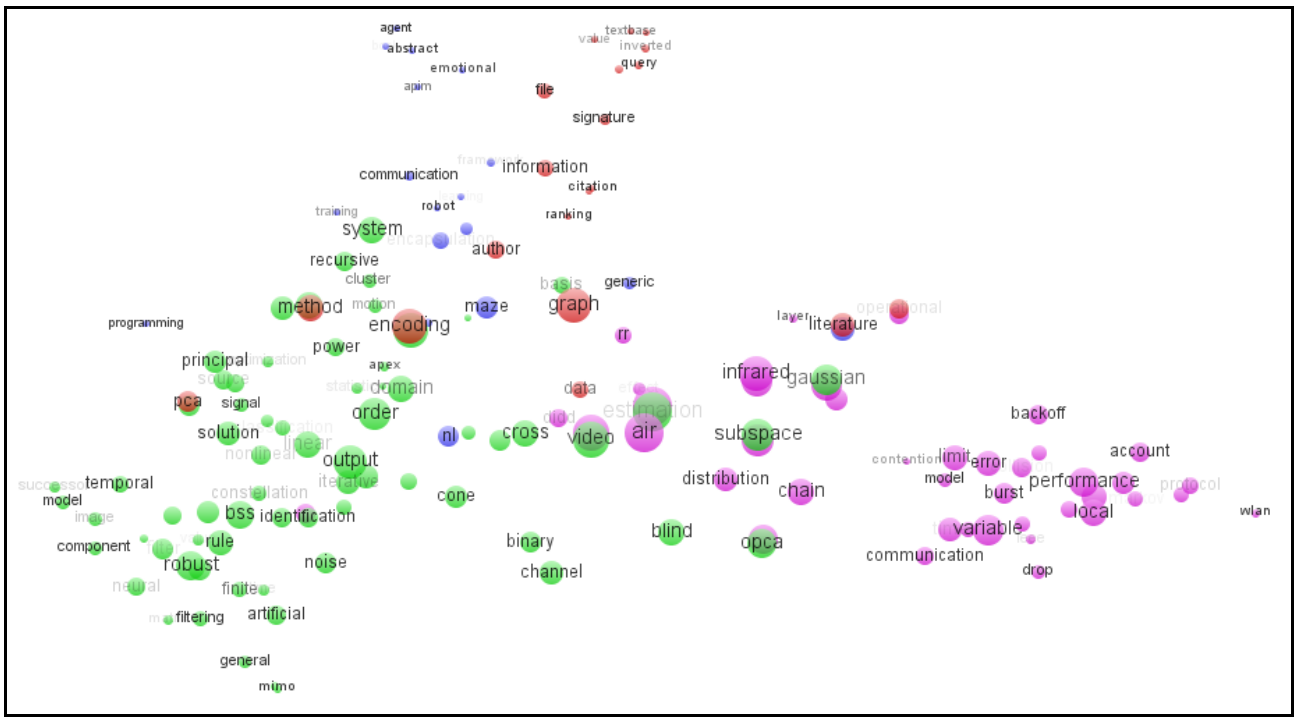
Εικόνα 29 Οπτικοποίηση βασισμένη σε λίστα 1000 λέξεων

4.2.1.3 Βελτίωση θησαυρών (θησαυροί 100 και 150 λέξεων χωρίς άσχετες με την πληροφορική λέξεις)

Για να πάρουμε κάποιες πιο περιεκτικές εικόνες αφαιρέσαμε χειροκίνητα από τις λίστες των 100 και 150 λέξεων, τις λέξεις με υψηλό βάρος tf-idf αλλά που εννοιολογικά δεν ήταν σημαντικές. Αυτές αντικαταστάθηκαν από άλλες με χαμηλότερο tf-idf βάρος, που διαλέχθηκαν όμως γιατί κρίθηκε ότι λογικά έχουν σημασία για την επιστήμη της πληροφορικής.



Εικόνα 30 Οπτικοποίηση βασισμένη σε λίστα 100 διαλεγμένων λέξεων.



Εικόνα 31 Οπτικοποίηση βασισμένη σε λίστα 150 διαλεγμένων λέξεων.

4.2.2 Κώδικας σε Java

Σε αυτήν την ενότητα παρουσιάζονται τα κομμάτια του προγράμματος που κάνουν την ουσιαστική δουλειά, ώστε να δημιουργηθούν και να αποθηκευτούν τα διανύσματα που αντιπροσωπεύουν τις συμπτώσεις των όρων του θησαυρού στα κείμενα, και μετά να εισαχθούν στον VOS viewer.

4.2.2.1 Η κλάση IntVector

Έχει συγγραφεί μία κλάση που υλοποιεί ένα διάνυσμα (Vector) ακεραίων στη Java, γιατί η υπάρχουσα κλάση Vector δεν λειτουργούσε με έναν τρόπο ώστε να εξυπηρετεί εύκολα τους σκοπούς του προγράμματος. Όπως φαίνεται και στον αλγόριθμο που παρουσιάζεται στην παράγραφο 3.3.2.4, η χρήση του διανύσματος είναι καταλυτική για τη λειτουργία του αλγορίθμου. Ακολουθούν τα ονόματα των κλάσεων και μεθόδων με τις υπογραφές τους, ενώ ο πλήρης κώδικας βρίσκεται στο ΠΑΡΑΡΤΗΜΑ Α’.

public class IntVector {

.....

public IntVector() {..... } //Ο προεπιλεγμένος δομιτής

public void add(int val) {.....} // Η μέθοδος αυτή προσθέτει στην τρέχουσα θέση του διανύσματος τον ακέραιο που μπαίνει ως όρισμα.

public int size() {.....} // Η μέθοδος αυτή επιστρέφει το μέγεθος του διανύσματος μια δεδομένη στιγμή.

public int get(int pos) {..... } // Η μέθοδος αυτή επιστρέφει τον ακέραιο που βρίσκεται στη θέση του διανύσματος που δηλώνεται από το όρισμα.

public void showAll() {.....}// Η μέθοδος αυτή τυπώνει στην οθόνη όλους τους ακέραιους που έχει αποθηκευμένους το διάνυσμα μια δεδομένη στιγμή.

public boolean exists(int val){.....}// Η μέθοδος αυτή αναζητά αν υπάρχει καταχωρημένος ο ακέραιος που δηλώνεται στο όρισμα.

}

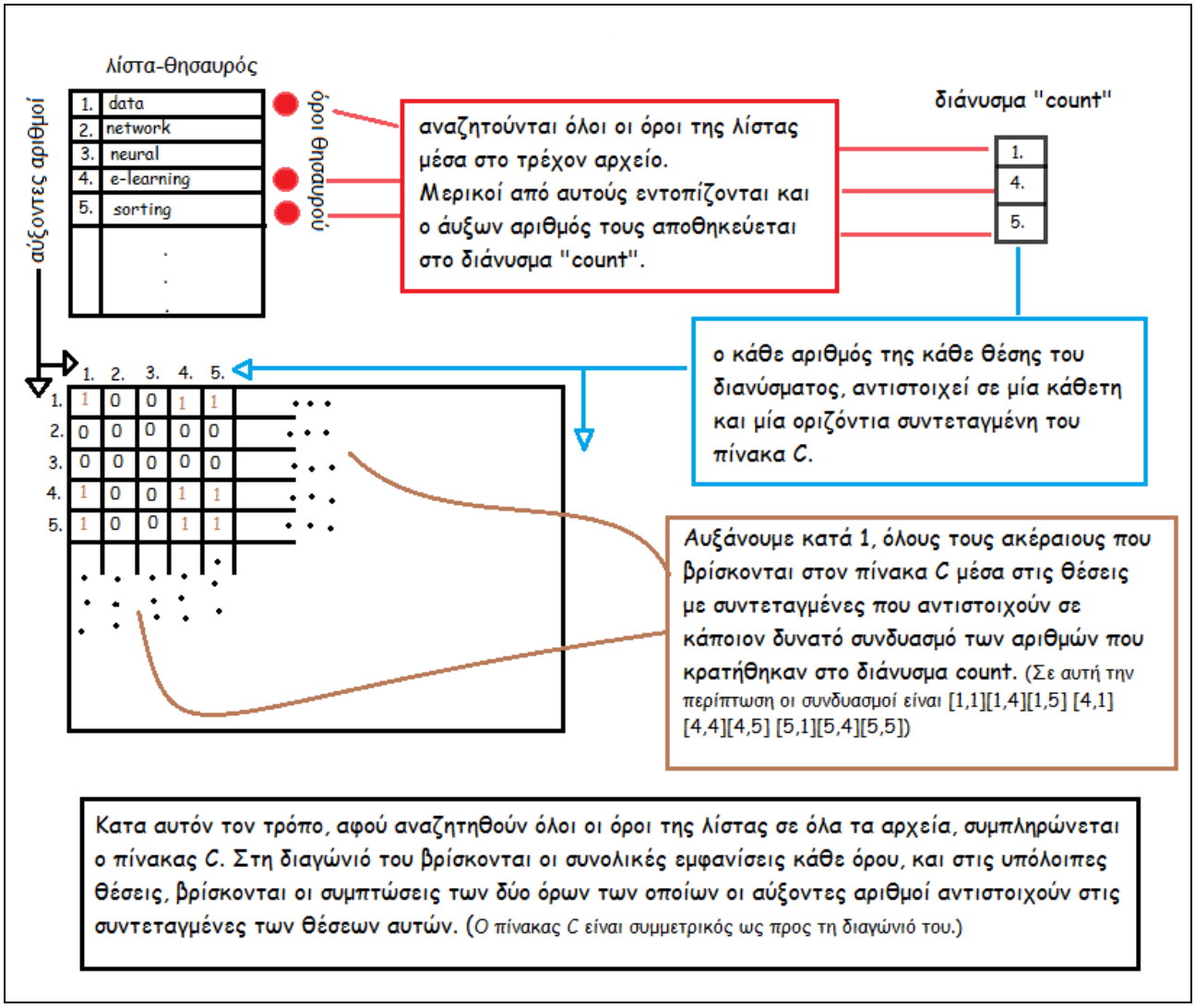
4.2.2.2 Η μέθοδος fileLister

Η μέθοδος fileLister ανοίγει τον κατάλογο που μας ενδιαφέρει, και αναζητά σε ποιά αρχεία εμφανίζεται ο κάθε όρος του θησαυρού, συμπληρώνοντας τον πίνακα διανυσμάτων.

```
public static void fileLister(String path, String[] terms, int[][] C) ... {

…..............................................................


    for (int i = 0; i < listOfFiles.length; i++) {

      if (listOfFiles[i].isFile()) { // αναζητούμε όποιο στοιχείο μέσα στον τρέχοντα κατάλογο είναι αρχείο.

          files = listOfFiles[i].getName();

          if (files.endsWith(".txt") || files.endsWith(".TXT")) { //ελέγχουμε καθένα ξεχωριστά αν είναι αρχείο κειμένου.

….....................................................

              for (int j = 0; j < terms.length; j++) {

                 if (exists(terms[j], files)) { // αν υπάρχει ο τρέχων όρος του θησαυρού (Η μέθοδος “exists” αναλύεται παρακάτω)

                     if (!count.exists(j)) { // και αν δεν έχουμε αποθηκεύσει ακόμη τον αύξοντα αριθμό του στο διάνυσμα “count” (μας ενδιαφέρει να τον αποθηκεύσουμε μόνο μία φορά, γιατί μία είναι αρκετή για να κατοχυρώσει την εμφάνιση του αντίστοιχου όρου στο τρέχον αρχείο περίληψης)

                         count.add(j); // τον προσθέτουμε τώρα

                     }

                 }

              }//for j

              occurenciesFill(C); //καλούμε τη μέθοδο που διαμορφώνει τα διανύσματα, τα οποία αποθηκεύονται στον πίνακα ακεραίων C.

….....................................................................................
```

```
            }//if files
        }//if list
    }//for i
}//fileLister
```

Ακολουθεί ένα σχήμα που απεικονίζει τη διαδικασία σχηματικά για να γίνει πιο κατανοητή στον αναγνώστη.



Εικόνα 32 Σύνοπτική αναπαράσταση του αλγόριθμου συμπλήρωσης του πίνακα διανυσμάτων

4.2.2.3 Η μέθοδος exists

Η μέθοδος exists εξετάζει αν υπάρχει ή όχι ένας όρος σε ένα αρχείο.

```
public static boolean exists(String term, String file) ... {
```

…..

while (sc.hasNext()) { //όσο υπάρχει επόμενη λέξη στο κείμενο

String temp = sc.next(); // πάρε την επόμενη λέξη

if (temp.matches("(?i).*" + term + ".*") && term.length() > 4) { // αν το μήκος της λέξης “term” είναι μεγαλύτερο του 4 και η επόμενη λέξη περιέχει σε οποιοδήποτε σημείο της την λέξη “term” που ψάχνουμε, αγνοώντας αν τα γράμματα είναι κεφαλαία ή μικρά

flag = true; // σημείωσε ότι βρέθηκε

} else if (term.length() < 5 && temp.matches("(?i)" + term)) //αλλιώς αν το μήκος του όρου που αναζητούμε είναι μικρότερο του 5 τότε κοίτα αν είναι ακριβώς ο ίδιος με την επόμενη λέξη.*

{

flag = true; // σημείωσε ότι βρέθηκε

} …….....

}//while

…..

}

*Το σκεπτικό στο σημείο αυτό, είναι να μπορέσουμε αφενός να βρούμε λέξεις οι οποίες έχουν την ίδια έννοια με τον όρο που ψάχνουμε, ώστε να θεωρήσουμε ότι υπάρχει εμφάνιση του όρου αυτού στο κείμενο, και σύμπτωση του όρου με κάποιους άλλους κατ' επέκταση. Αφετέρου θέλουμε αυτό να γίνεται για σχετικά μεγάλες λέξεις, από 5 χαρακτήρες και πάνω, γιατί όταν μια λέξη είναι πολύ μικρή, είναι πολύ πιο πιθανό και εύκολο να αποτελεί μέρος μια σύνθετης λέξης η οποία δε φέρει κάποιο σχετικό νόημα. Είναι δηλαδή ζητούμενο, αν ο όρος που ψάχνουμε υπάρχει στο κείμενο, αλλά μόνο σε μορφή πληθυντικού, να θεωρηθεί ότι βρέθηκε. Γι’αυτό στον πρώτο έλεγχο με χρήση wild cards ελέγχουμε αν ο όρος βρίσκεται στην αρχή, στη μέση ή στο τέλος της επόμενης λέξης. Αυτό δε γίνεται στον δεύτερο έλεγχο γιατί έτσι οι πολύ μικρές λέξεις είχανε τεράστιο αριθμό εμφανίσεων, ο οποίος όμως σύμφωνα με τον σκοπό της οπτικοποίησης μας δημιουργεί πρόβλημα. Με αυτόν τον τρόπο μειώσαμε σε πολύ μεγάλο βαθμό το

πρόβλημα αλλά δεν παύει να υπάρχει σε λίγες περιπτώσεις (μόνο εννοείται για λέξεις που είναι μεγαλύτερες των 5 χαρακτήρων, αλλά παρόλα αυτά αποτελούν συνθετικό μεγαλύτερης λέξης με την οποία δεν έχουν μεγάλη εννοιολογική συγγένεια).

4.2.2.4 Η μέθοδος occurenciesFill

Η μέθοδος occurenciesFill γεμίζει με τους κατάλληλους ακέραιους έναν πίνακα διανυσμάτων.

```
public static void occurenciesFill(int[][] c) {
….....
for (int v = 0; v < count.size(); v++) { //παίρνουμε κάθε αριθμό που έχει αποθηκευτεί στο διάνυσμα “count”
….....
c[count.get(v)][count.get(v)]++; // συμπληρώνουμε τη διαγώνιο του πίνακα C που κάθε στοιχείο της αντιστοιχεί στις συνολικές εμφανίσεις του αντίστοιχου όρου
for (int y = 0; y < count.size(); y++) {
if (!(y == v)) { // για κάθε στοιχείο εκτός της διαγωνίου (που αντιστοιχεί σε κάποιον άλλο όρο)
c[count.get(v)][count.get(y)]++; // αυξάνουμε κατά ένα τις συμπτώσεις του με το τρέχον στοιχείο της διαγωνίου
}
}//for y

}//for v
….....
}//occurenciesFill
```

4.2.2.5 Η μέθοδος arrayTransformer

Η μέθοδος arrayTransformer δέχεται ως όρισμα έναν πίνακα με λέξεις που

στην πράξη είναι η λίστα κάποιου καθηγητή, και έναν αριθμό που καθορίζει σε ποιόν καθηγητή ανήκει η λίστα αυτή.

```
public static String[] arrayTransformer(String[] arr, int val) {
    for (int i = 0; i < arr.length; i++) { // και σε κάθε κελί του πίνακα
        arr[i] = arr[i] + ",," + arr[i] + "," + val; //αλλάζει τη μορφή του ώστε όταν γραφτεί σε αρχείο να είναι αναγνωρίσιμη από τον VOS viewer
    }//for

    return arr;
}
```

Για παράδειγμα, ο πίνακας αρχικά είναι μια στήλη από λέξεις. Μετά από την εκτέλεση της παραπάνω μεθόδου που τον δέχεται ως όρισμα, μεταμορφώνεται σε μια στήλη από Strings της μορφής “λέξη, λέξη, αριθμός”, όπου «λέξη» είναι ένας όρος, και «αριθμός» είναι ένας ακέραιος που αντιπροσωπεύει τον καθηγητή. Ο VOS viewer αναγνωρίζει αυτή τη μορφή. Το αρχείο items δηλαδή που θα εισαχθεί σε αυτόν, πρέπει να έχει σε κάθε σειρά του μια φράση της παραπάνω μορφής.

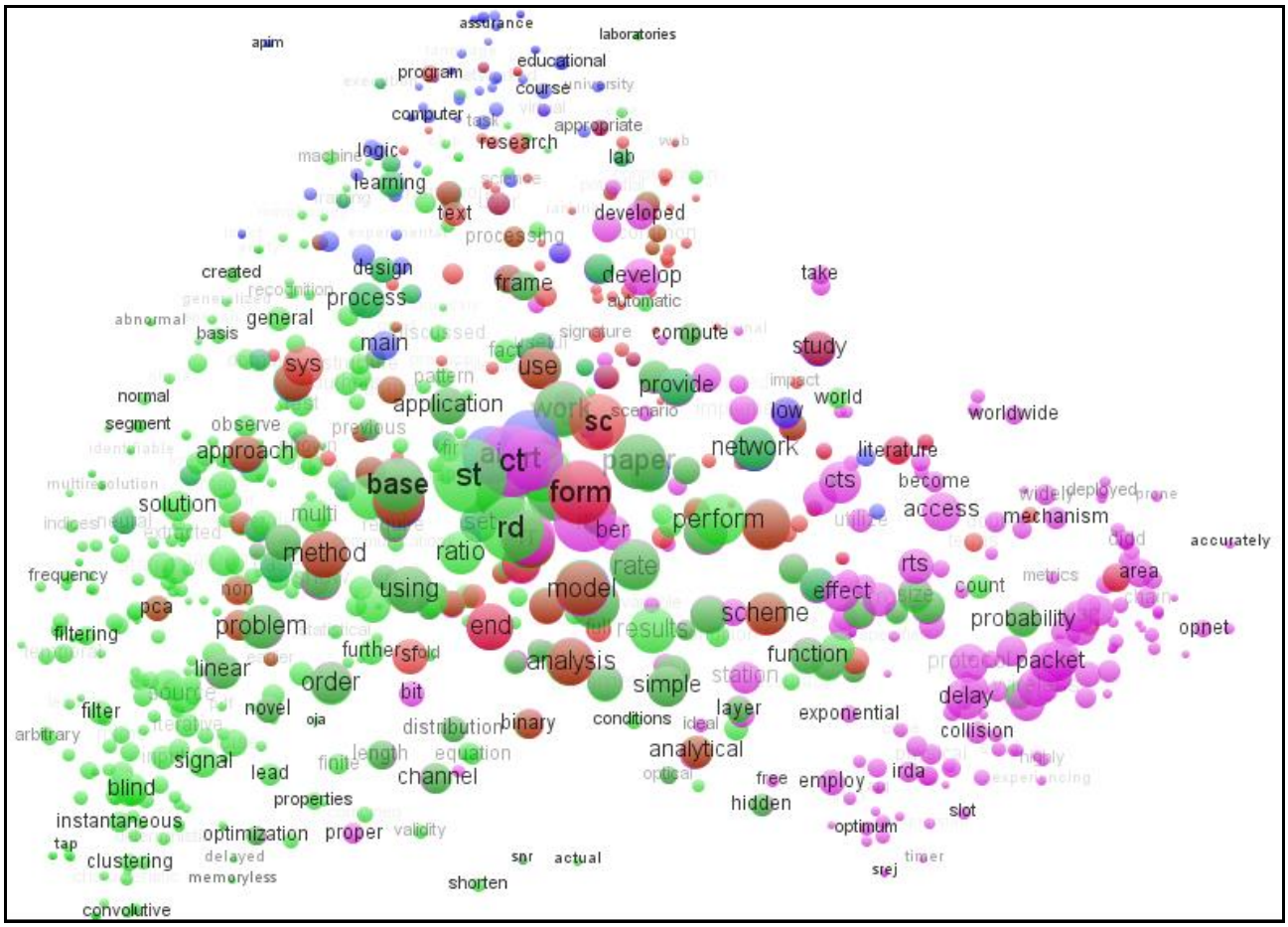
4.3 Τί παρατηρούμε;

Έχοντας δημιουργήσει τις οπτικοποιήσεις της παραγράφου 4.2.1.2, είναι σκόπιμο να αναλύσουμε τη δομή τους. Οι ακόλουθες υποπαράγραφοι, περιλαμβάνουν και σύμφωνα με τον τίτλο τους, τον σχολιασμό για κάποια συγκεκριμένα χαρακτηριστικά των παραπάνω εικόνων.

4.3.1 Συγγένειες

Στην εικόνα 33 βλέπουμε την οπτικοποίηση που προέκυψε από τη μεγαλύτερη λίστα που κατασκευάστηκε, αυτή των 1000 λέξεων. Στη μία γωνία

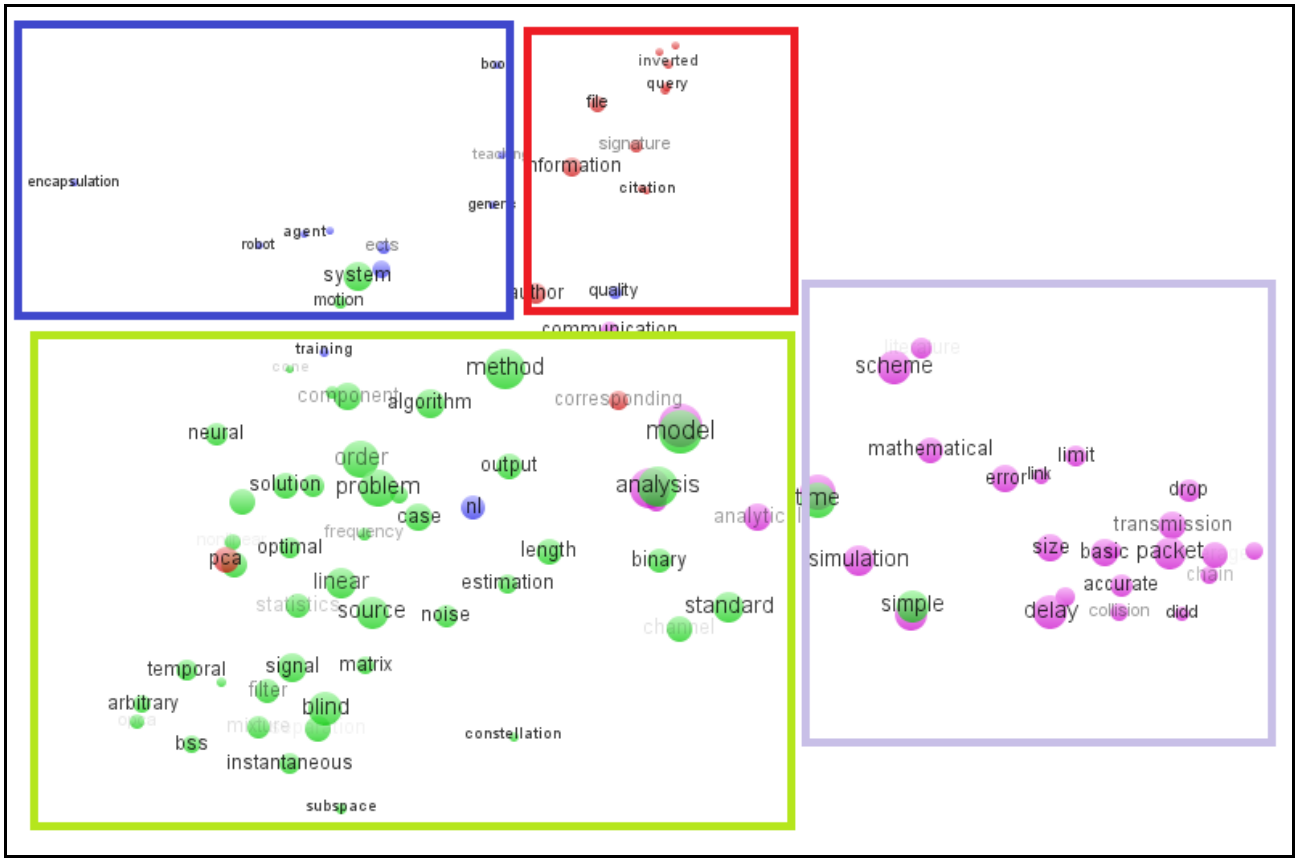
κάτω αριστερά επικρατεί το πράσινο χρώμα που συμβολίζει όρους του κ. Διαμαντάρα. Προς τα πάνω έρχεται κοντά με το μπλε χρώμα του κ. Σταμάτη. Αυτό πιθανώς οφείλεται στην εγγύτητα μεταξύ κάποιων δημοσιεύσεών τους που έχουν ως βάση την τεχνητή νοημοσύνη. Στην κορυφή μένει σε κάποιο σημείο μόνο το μπλε χρώμα και πιο κάτω εντείνεται το κόκκινο του κ. Δέρβου, που τείνει προς το κέντρο, εκεί δηλαδή που συμπέφτουν λίγο πολύ όλα τα χρώματα λόγω των κοινών λέξεων που δε φέρουν ιδιαίτερο νόημα. Στην κάτω δεξιά γωνία βρίσκονται οι όροι του κ. Βίτσα με το μωβ χρώμα, που έχουν να κάνουν με τα δίκτυα υπολογιστών. Βρίσκονται πολύ απόμακροι από το μπλε χρώμα, και περισσότερο κοντά με τα άλλα δύο, με τα οποία μοιράζονται πολλούς κοινούς όρους



Εικόνα 33 Οπτικοποίηση με 1000 λέξεις

4.3.2 Ομαδοποίηση (Clustering) (ή αλλιώς συσταδοποίηση)

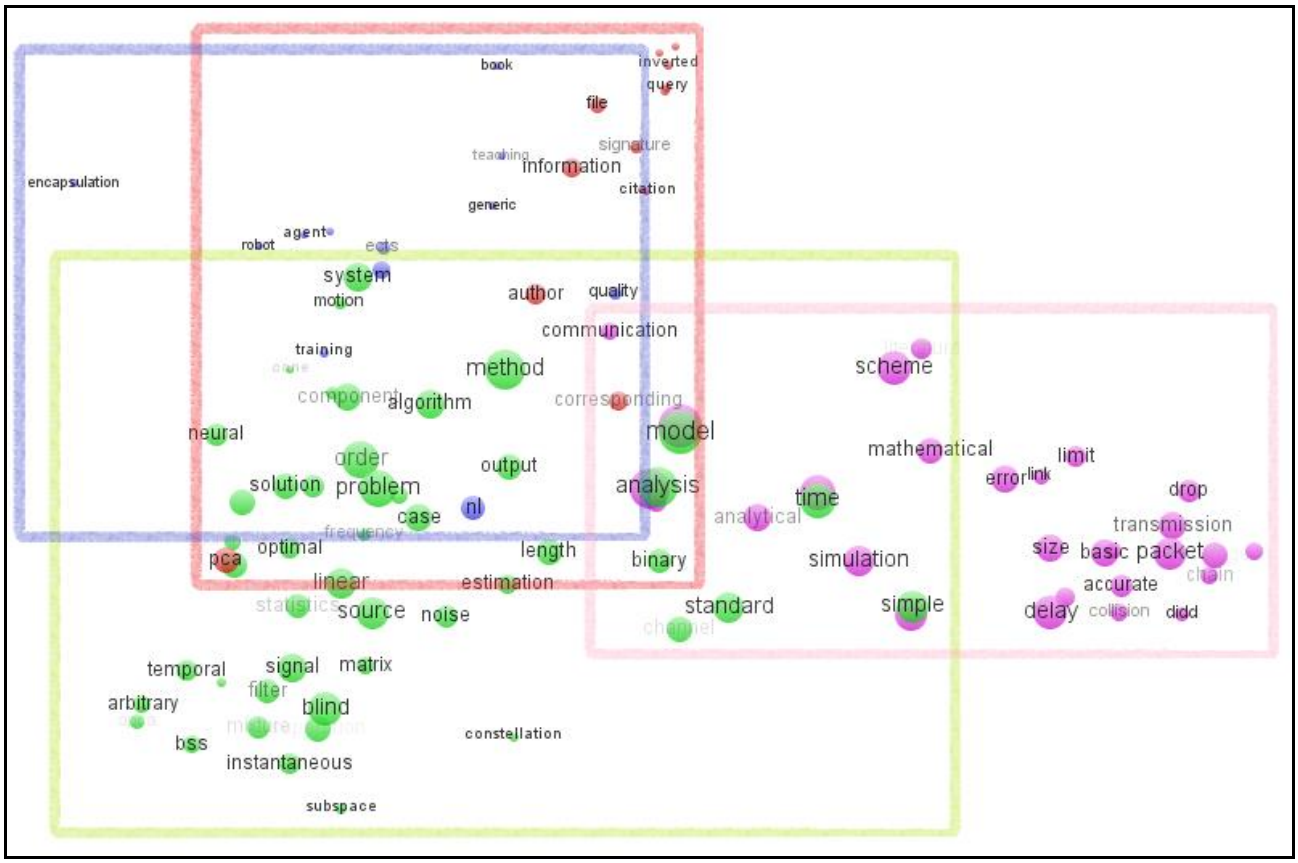
Σε αυτό το σημείο θα χρησιμοποιήσουμε δύο εικόνες, μία σχετικά λίγων λέξεων – συγκεκριμένα 100, και μία που δομήθηκε χρησιμοποιώντας τη λίστα των 300 λέξεων. Αυτό γίνεται για να εντοπίσουμε τυχόν διαφορές ανάμεσα σε μια "φορτωμένη" εικόνα και μια πιο "ελαφριά", αλλά κυρίως γιατί η κύρια δομή της "μεγάλης εικόνας" που αναζητούμε, είναι ίδια και κυριαρχεί σχεδόν σε κάθε περίπτωση. Έτσι δε χρειάζεται να μπούμε σε αυτή τη διαδικασία για κάθε μία εικόνα από αυτές που παράξαμε για λόγους πειραματισμού.



Εικόνα 34 Ομαδοποίηση στην οπτικοποίηση των 100 λέξεων

Στην εικόνα 34, οριοθετήσαμε τον πυρήνα της κάθε ομάδας λέξεων. Κάθε ομάδα χρώματος είναι διαφορετικός καθηγητής. Σε κάθε τετράγωνο δεν περικλείονται μόνο όροι της ίδιας ομάδας, αλλά κυριαρχεί εντός των ορίων ένα μόνο χρώμα.

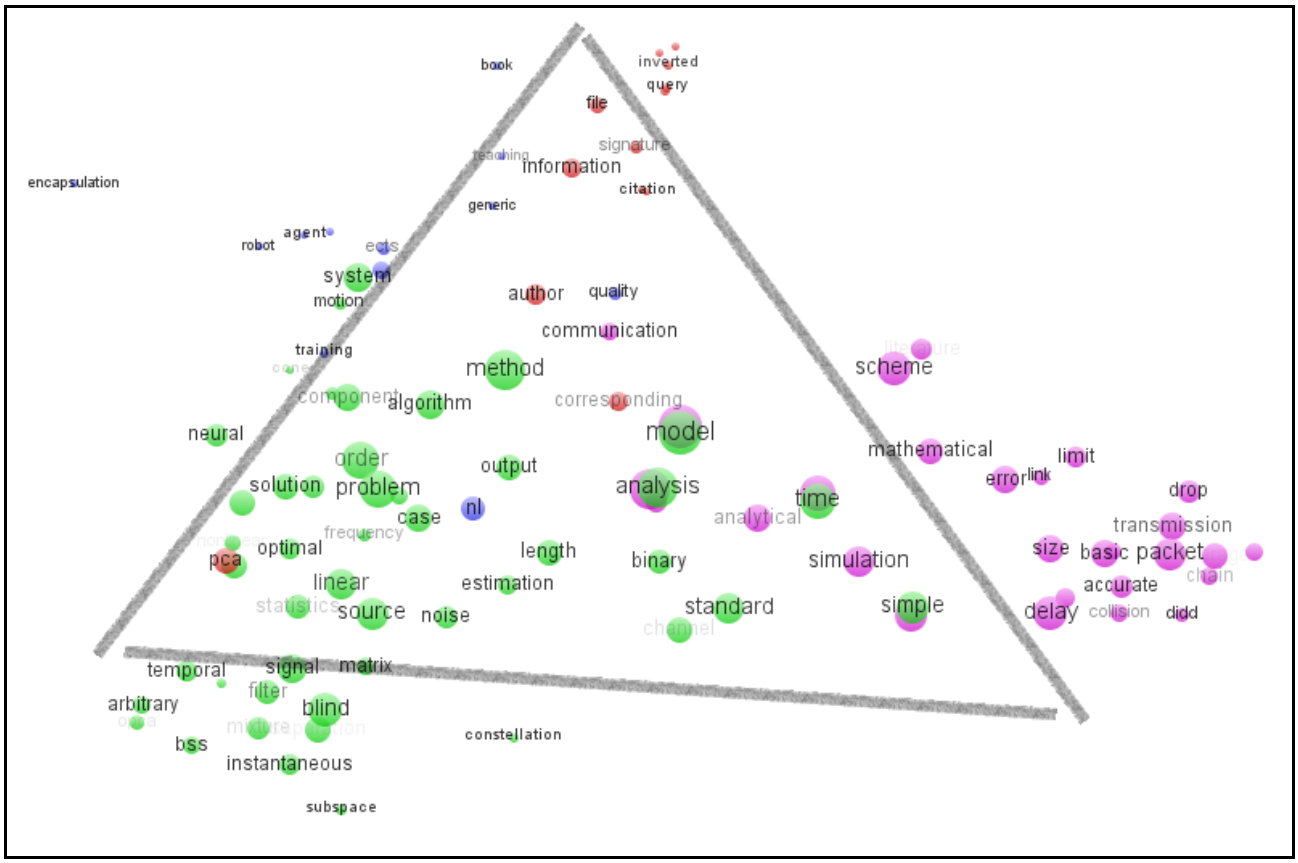
Στην επόμενη εικόνα (35) τα τετράγωνα μεγαλώνουν ώστε να περικλείουν κάθε όρο της ίδιας ομάδας και τα πράγματα περιπλέκονται κάπως. Φαίνεται πλέον πόσο κοντά είναι κάποιες ομάδες (σχεδόν ταυτίζονται) ενώ κάποιες άλλες έχουν αρκετές διαφορές.



Εικόνα 35 Οπτικοποίηση 100 λέξεων με πλαίσια που περικλείουν το σύνολο των λέξεων κάθε ομάδας.

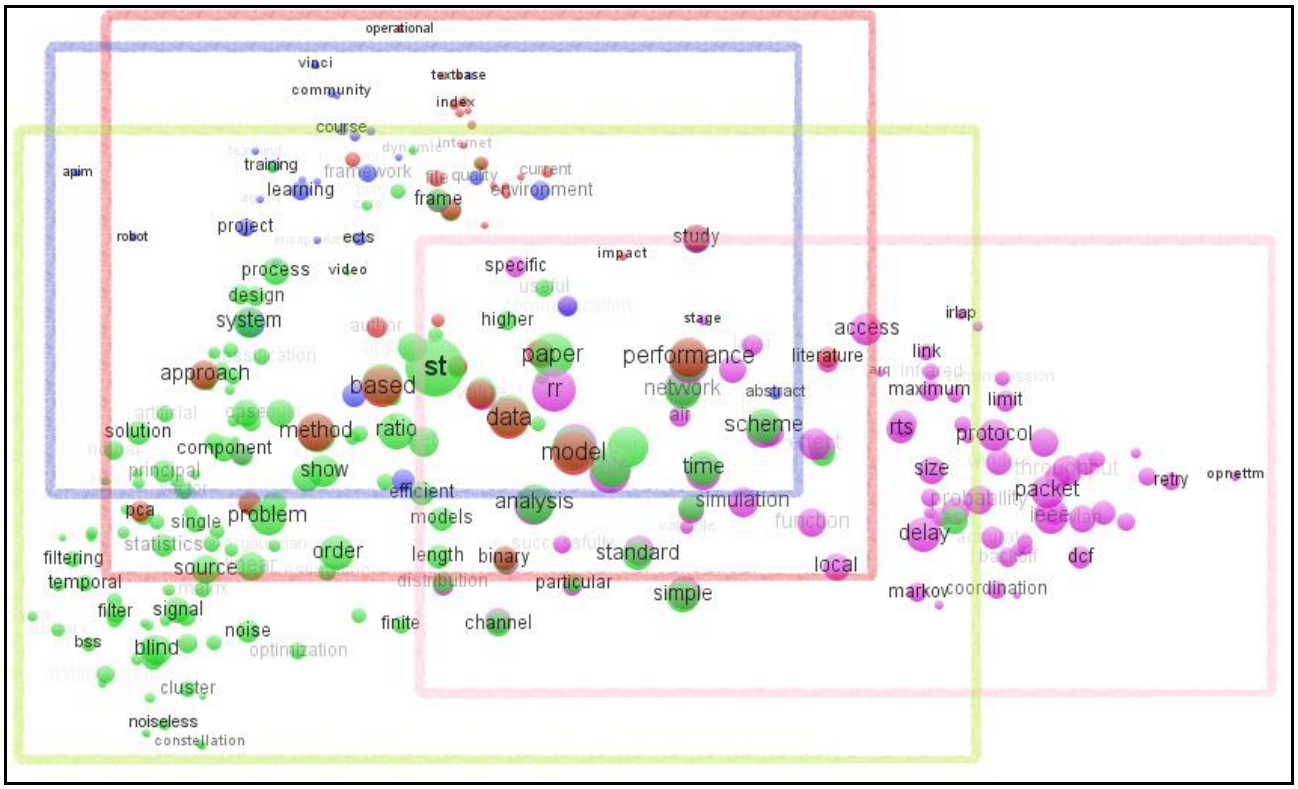
Το γεγονός όμως της “τριγωνικής” μορφής της κατανομής των φυσαλίδων στο χώρο δεν αλλάζει. Θα δούμε παρακάτω παρατηρώντας και άλλες εικόνες ότι μόνο σε μία περίπτωση αλλάζουν οι ισορροπίες τόσο που το τρίγωνο αυτό “χαλάει”.

Σε κάθε εικόνα (πχ εικόνα 36) διακρίνουμε τρεις πόλους έλξης και στη μέση μια πιο έντονη πυκνότητα. Στον ένα πόλο η συντριπτική πλειοψηφία των όρων ανήκουν στον κ. Διαμαντάρα, στον άλλο ανήκουν στον κ. Βίτσα, και στην τρίτη κορυφή βρίσκονται κοντά οι όροι του κ. Σταμάτη και του κ. Δέρβου.

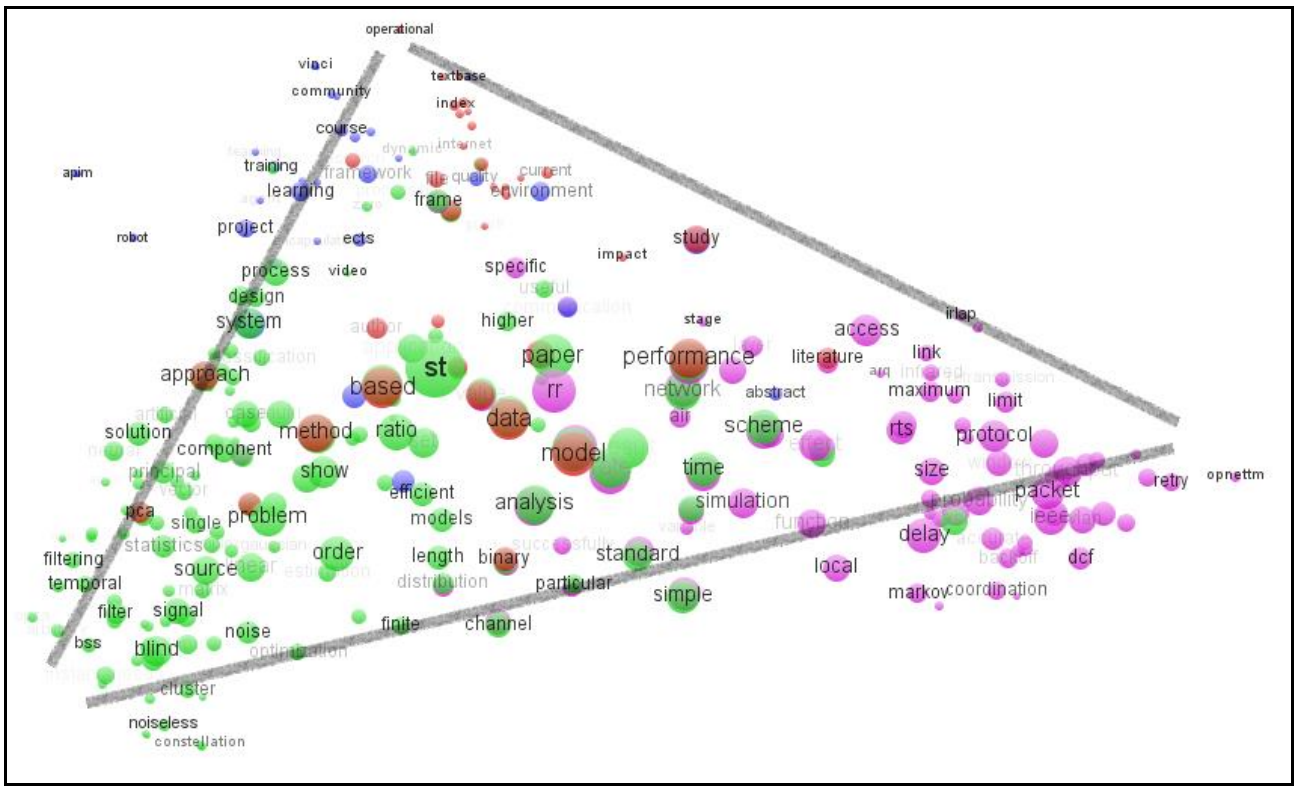


Εικόνα 36 Οπτικοποίηση με 100 λέξεις και περιγραφή της βασικής κατανομής.

Ακολουθεί η εικόνα 37 με τις 300 λέξεις στην οποία φαίνεται το πόσο όμοια είναι η σχέση μεταξύ των διάφορων ομάδων χρωμάτων. Στη δεύτερη εικόνα (38) είναι ξεκάθαρο πως οι τρεις πόλοι εξακολουθούν να υπάρχουν, το ίδιο και η υψηλή συγκέντρωση στο κέντρο του τριγώνου.



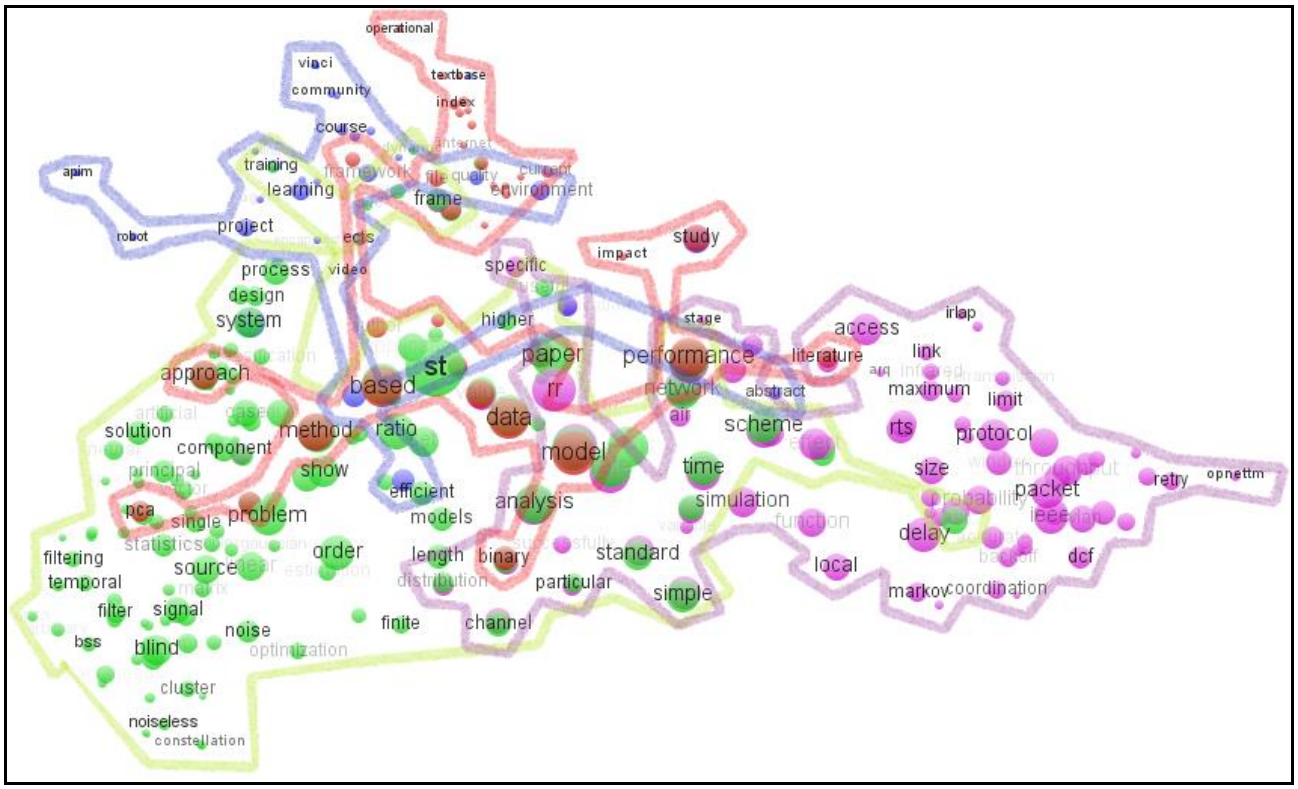
Εικόνα 37 Οπτικοποίηση 300 λέξεων με πλαίσια που περικλείουν το σύνολο των λέξεων κάθε ομάδας.



Εικόνα 38 Οπτικοποίηση με 300 λέξεις και περιγραφή της βασικής κατανομής

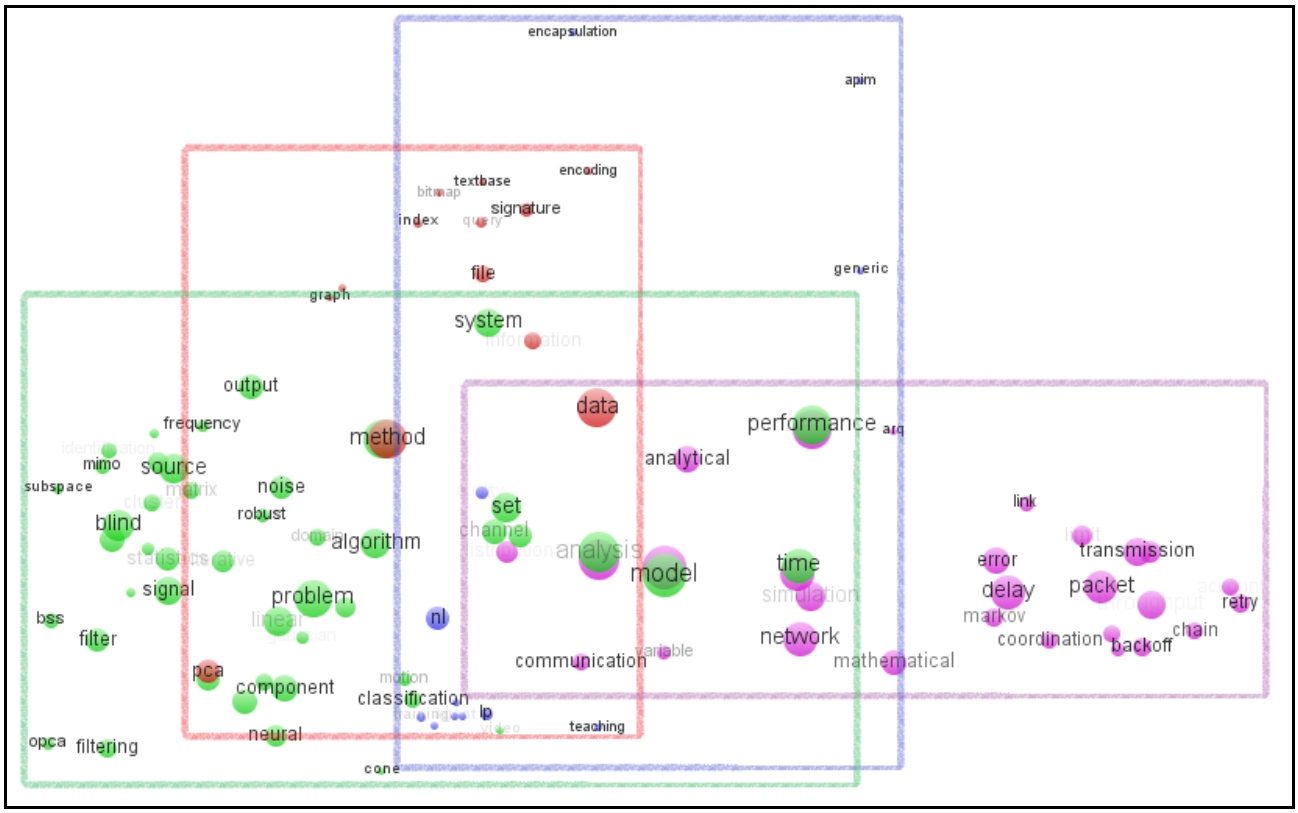
Στην παρακάτω εικόνα (39) είναι κυκλωμένη η κάθε ομάδα χωρίς τη μεγάλη

σπατάλη εμβαδού που συμβαίνει όταν κυκλώνονται οι ομάδες με τετράγωνα. Το αποτέλεσμα είναι πιο μπερδεμένο οπτικά άλλα δείχνει καθαρότερα ποιές ομάδες συμμετέχουν με περισσότερους όρους αφού τα σχήματά τους έχουν μεγαλύτερο εμβαδό, καθώς και ποιό είναι το σημείο συνάντησης – εκεί δηλαδή που διασταυρώνονται οι περισσότερες γραμμές και τείνει να γίνει “μουτζούρα”.



Εικόνα 39 Οπτικοποίηση 300 λέξεων με προσαρμοσμένα πλαίσια που περιλαμβάνουν όλους τους όρους κάθε ομάδας

Στην τελευταία εικόνα (40) αυτής της παραγράφου βλέπουμε την περίπτωση οπτικοποίησης, με χρήση του τροποποιημένου μεικτού θησαυρού. Είναι η μόνη από τις εικόνες που παράξαμε στην οποία η τριγωνική μορφή “καταρρέει” και η πλειοψηφία των όρων συγκεντρώνεται από τη μέση και κάτω της εικόνας.



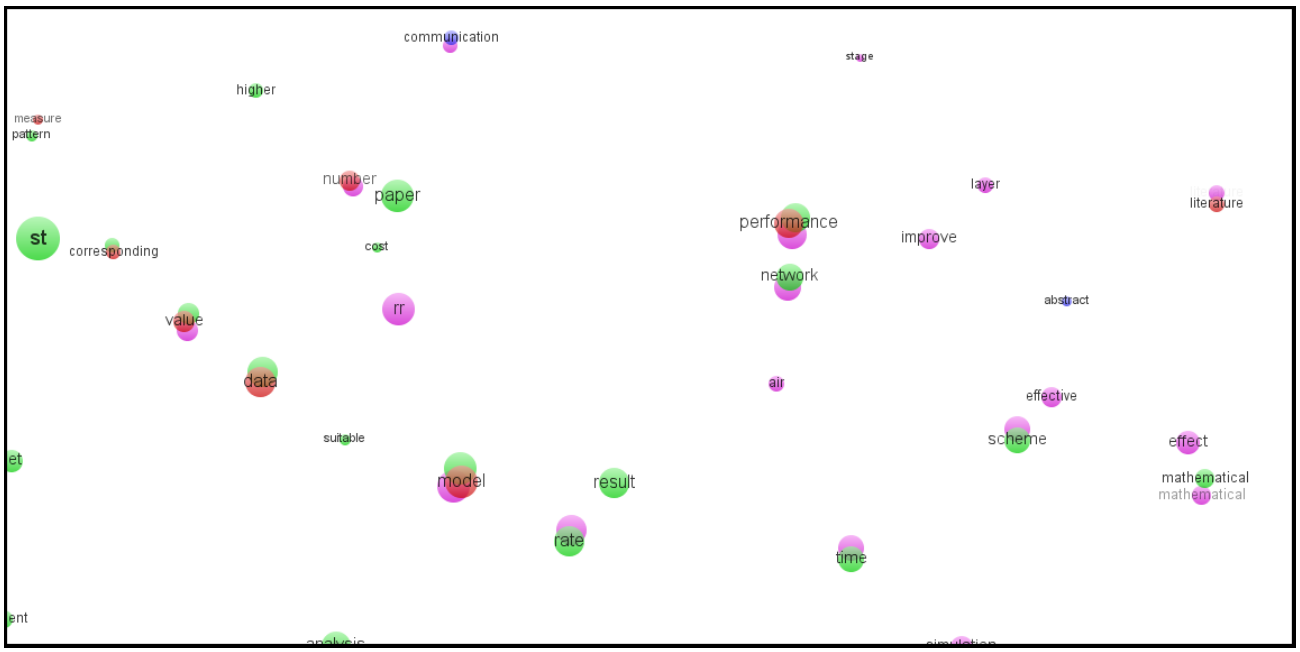
Εικόνα 40 Οπτικοποίηση 100 διαλεγμένων λέξεων με πλαίσια που περικλείουν το συνολο των λέξεων κάθε ομάδας.

4.3.3 Κοινές λέξεις

Στην εικόνα (41) που ακολουθεί βλέπουμε την οπτικοποίηση των 300 λέξεων εστιασμένη σε ένα σημείο με αρκετές λέξεις να είναι κοινές για 2 ή 3 καθηγητές. Κάποιες από αυτές τις λέξεις, όπως οι “number”, “value”, “data” κ.α., είναι όροι γενικού περιεχομένου που ενώ σχετίζονται άμεσα με την επιστήμη της πληροφορικής, δεν παρέχουν πληροφορία ότι οι δημοσιεύσεις που τις περιέχουν, ασχολούνται με κάποιο συγκεκριμένο πεδίο. Σε αντίθεση με αυτές, η λέξη “network” παραπέμπει σε πολύ πιο συγκεκριμένα πεδία. Δεν είναι τυχαίο που είναι κοινή μεταξύ ενός καθηγητή που ασχολείται με δίκτυα υπολογιστών, και ενός άλλου που ασχολείται με νευρωνικά δίκτυα. Λέξεις που μπορούν να θεωρηθούν γενικές, αλλά οριοθετούν το θέμα με την εμφάνιση τους στην περιοχή που απεικονίζεται στην εικόνα 41, είναι οι “performance” και “mathematical”. Είναι πολύ λογικό δηλαδή κάποιος που ασχολείται με δίκτυα υπολογιστών να μελετά την απόδοσή τους, καθώς και κάποιος που μελετά τα νευρωνικά δίκτυα να μελετά την

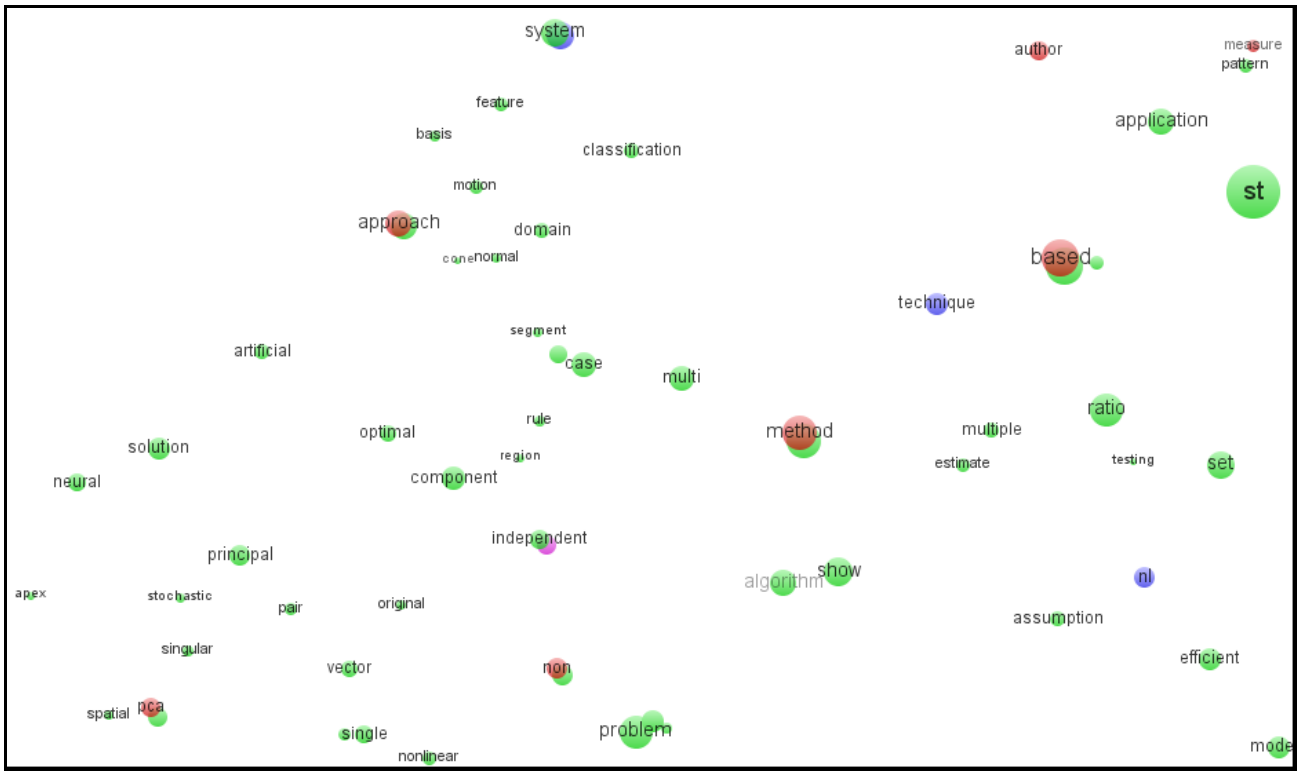
απόδοση των αλγόριθμων που υλοποιούνται. Η μαθηματική έκφραση δε, και στις δύο περιπτώσεις, είναι απαραίτητη.

Παρατηρούμε, επίσης, ότι η λέξη “data” εμφανίζεται στον κ. Δέρβο και στον κ. Διαμαντάρα, η λέξη “model” είναι κοινή μεταξύ των τριών καθηγητών πλην του κ. Σταμάτη κ.ο.κ. Ομοίως συνυπάρχουν λέξεις όπως οι “rate”, “time”, “mathematical”, “performance”, “network”, “communication”, κ.α. Η λέξη network, όπως αναφέρθηκε προηγουμένως, είναι μία από τις λέξεις που ενώνει τον ένα μεγάλο πόλο (του κ. Διαμαντάρα που ασχολείται με τεχνητή νοημοσύνη και νευρωνικά δίκτυα), με τον άλλο μεγάλο πόλο (του κ. Βίτσα που ασχολείται με δίκτυα υπολογιστών).



Εικόνα 41 Συμπτώσεις των λέξεων performance, model, value κ.α στην οπτικοποίηση των 300 λέξεων

Στην ακόλουθη εικόνα (42) που είναι πάλι μια μεγέθυνση της οπτικοποίησης των 300 όρων, βλέπουμε κοινές λέξεις γενικού νοήματος μεταξύ του κ. Δέρβου και του κ. Διαμαντάρα όπως οι “approach”, “method”, “based”, αλλά και ειδικής σημασίας όπως η “pca”. Η λέξη “system” είναι κοινή μεταξύ κ. Διαμαντάρα και κ. Σταμάτη, ενώ η λέξη “independent” κοινή μεταξύ κ. Διαμαντάρα και κ. Βίτσα.



Εικόνα 42 Συμπτώσεις των λέξεων pca, system, independent κ.α στην οπτικοποίηση των 300 λέξεων

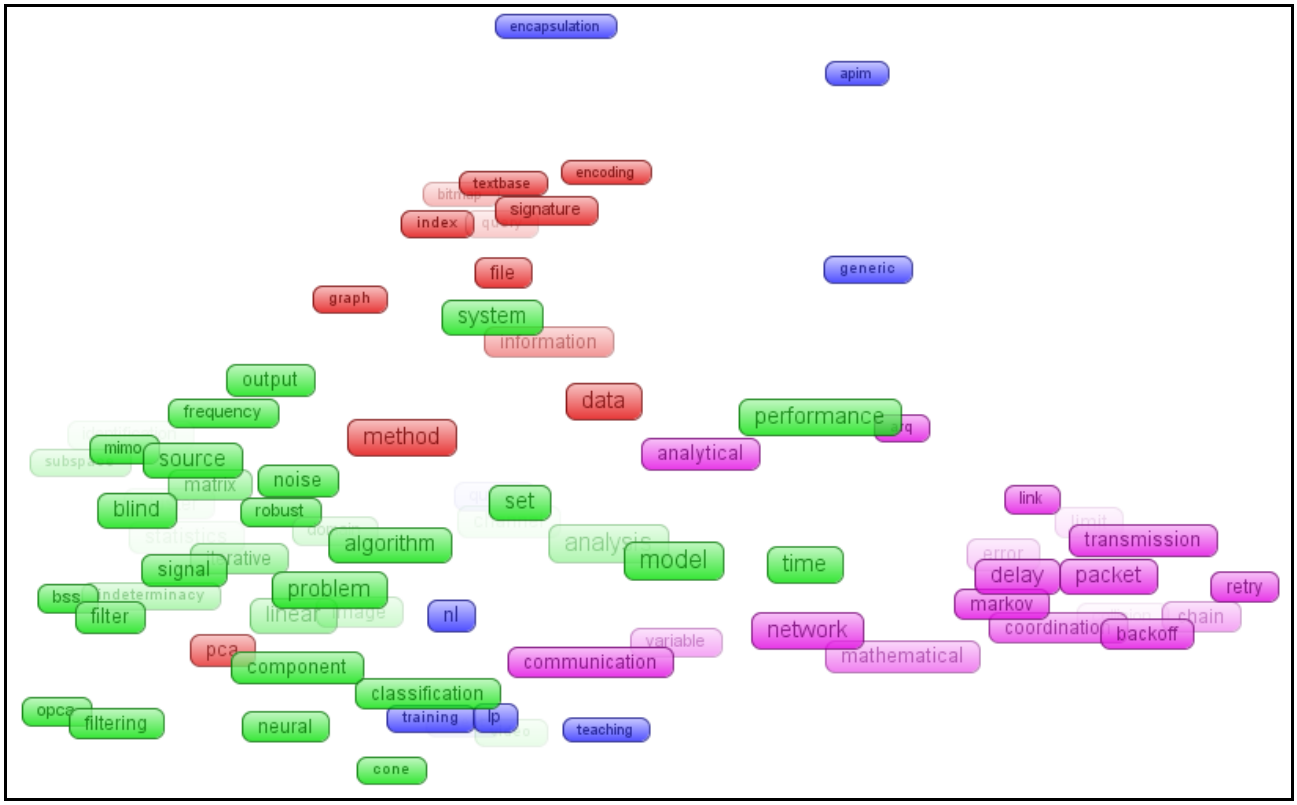
4.4 Παράμετροι και όψεις στον VOS

Στις ακόλουθες παραγράφους παρατίθενται μερικές εικόνες από αυτές που ήδη έχουμε αναλύσει, αλλά μέσα από τις υπόλοιπες όψεις του VOS viewer, αλλάζοντας κάποιες παραμέτρους των διαθέσιμων επιλογών.

4.4.1 Label view

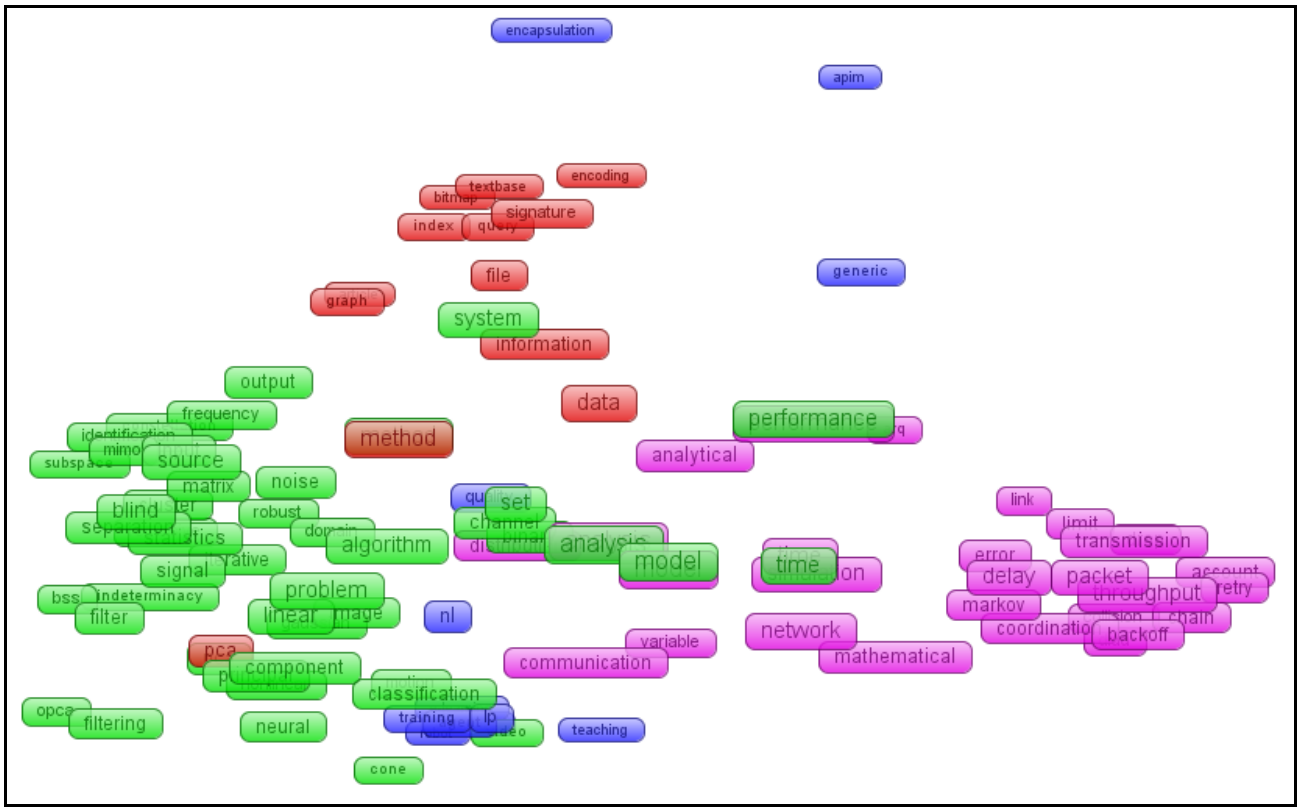
Στο Label View που είναι η προεπιλεγμένη οπτική του VOS viewer, τα αντικείμενα υποδεικνύονται από την ετικέτα τους και από έναν κύκλο. Για κάθε αντικείμενο, το μέγεθος της γραμματοσειράς της ετικέτας του, και το μέγεθος του κύκλου του αντικειμένου εξαρτώνται από το βάρος που του έχει αποδοθεί. Αν έχουν αποδοθεί και χρώματα στα αντικείμενα, κάθε κύκλος αντικειμένου εμφανίζεται στο αντίστοιχό του χρώμα. Από προεπιλογή, για να αποφευχθούν οι αλληλοεπικαλύψεις των ετικετών, μόνο ένα υποσύνολο αυτών είναι ορατό. Η label view είναι χρήσιμη συγκεκριμένα για λεπτομερή εξέταση ενός χάρτη.

Όλες οι εικόνες του προηγούμενου κεφαλαίου χρησιμοποιούν αυτήν την αισθητική με τις φυσαλίδες, οπότε έχει νόημα να δούμε κάποια ελαφρώς αλλαγμένη. Επιλέγουμε από το πάνελ στα δεξιά της κυρίως διεπαφής του VOS viewer, την καρτέλα “Options” και στο πεδίο “Label view”, αλλάζουμε την επιλογή από “Show labels and circles” σε “Show labels and frames”. Αυτή η αλλαγή, για την οπτικοποίηση της τροποποιημένης λίστας των 100 όρων, μας δίνει την εξής εικόνα:



Εικόνα 43 Οπτικοποίηση με 100 διαλεγμένες λέξεις, με ετικέτες και πλαίσια αντί για κύκλους.

Αν στο πεδίο “Labels” της ίδιας καρτέλας ακυρώσουμε την επιλογή “No overlap”, τότε η εικόνα (43) αλλάζει ως εξής:



Εικόνα 44 Οπτικοποίηση με 100 διαλεγμένες λέξεις σε πλαίσια, με επιλογή που επιτρέπει επικαλύψεις

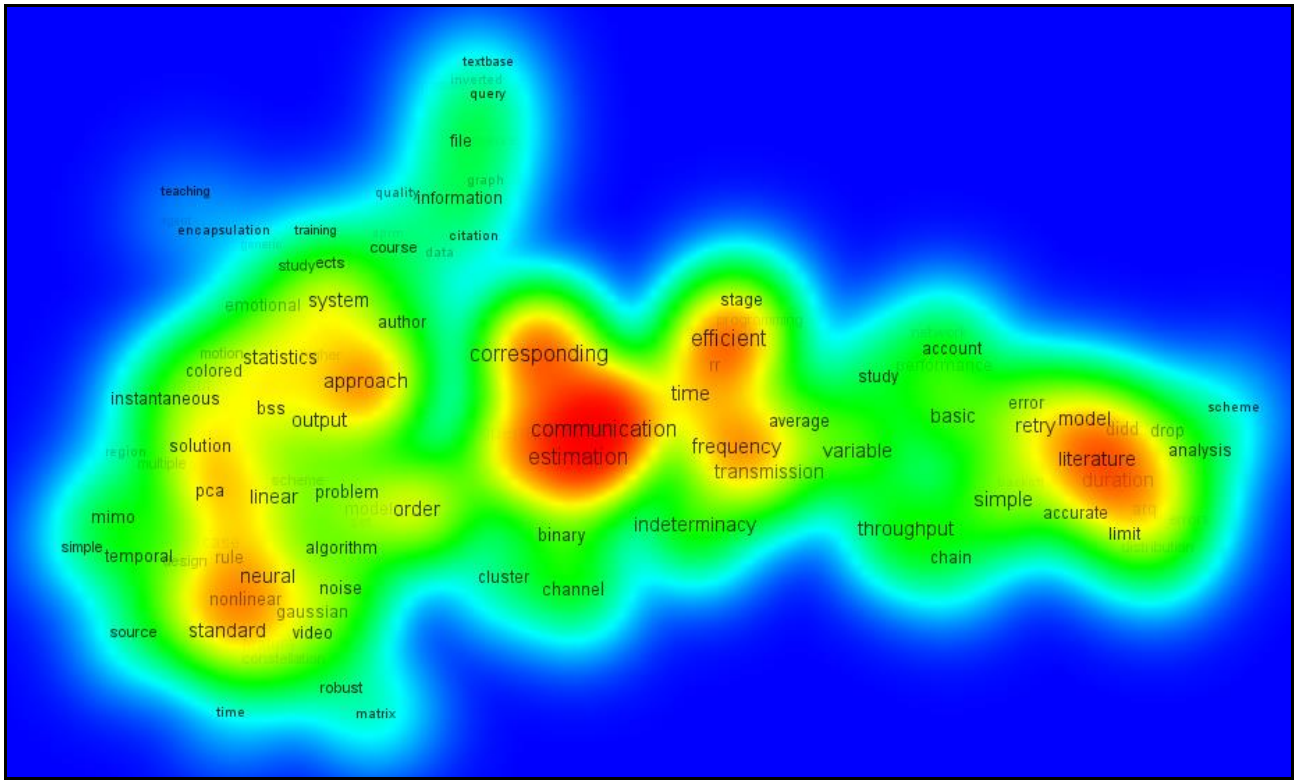
Παρατηρούμε ότι έρχονται στην επιφάνεια και λέξεις που δε φαίνονταν πριν, αφού ουσιαστικά χάνεται η τρίτη διάσταση – δηλαδή το βάθος- στην εικόνα.

4.4.2 Density view

Μια πολύ ενδιαφέρουσα οπτική που προσφέρει ο VOS viewer είναι η Density View. Στη density view, τα αντικείμενα συμβολίζονται με την ετικέτα τους με παρόμοιο τρόπο όπως στη label view. Κάθε σημείο στον χάρτη έχει ένα χρώμα που εξαρτάται από την πυκνότητα των αντικειμένων σε εκείνο το σημείο. Η προεπιλογή για το χρώμα αυτό είναι μεταξύ κόκκινου και μπλε. Όσο μεγαλύτερος είναι ο αριθμός των αντικειμένων στη γειτονιά ενός σημείου και όσο υψηλότερα είναι τα βάρη των αντικειμένων, τόσο το χρώμα του σημείου πλησιάζει στο κόκκινο. Αντιστρόφως, όσο μικρότερος είναι ο αριθμός των αντικειμένων στο γειτονικό χώρο ενός σημείου και όσο χαμηλότερα τα βάρη των αντικειμένων, τόσο το χρώμα του σημείου αυτού πλησιάζει στο μπλε. Η density view είναι χρήσιμη

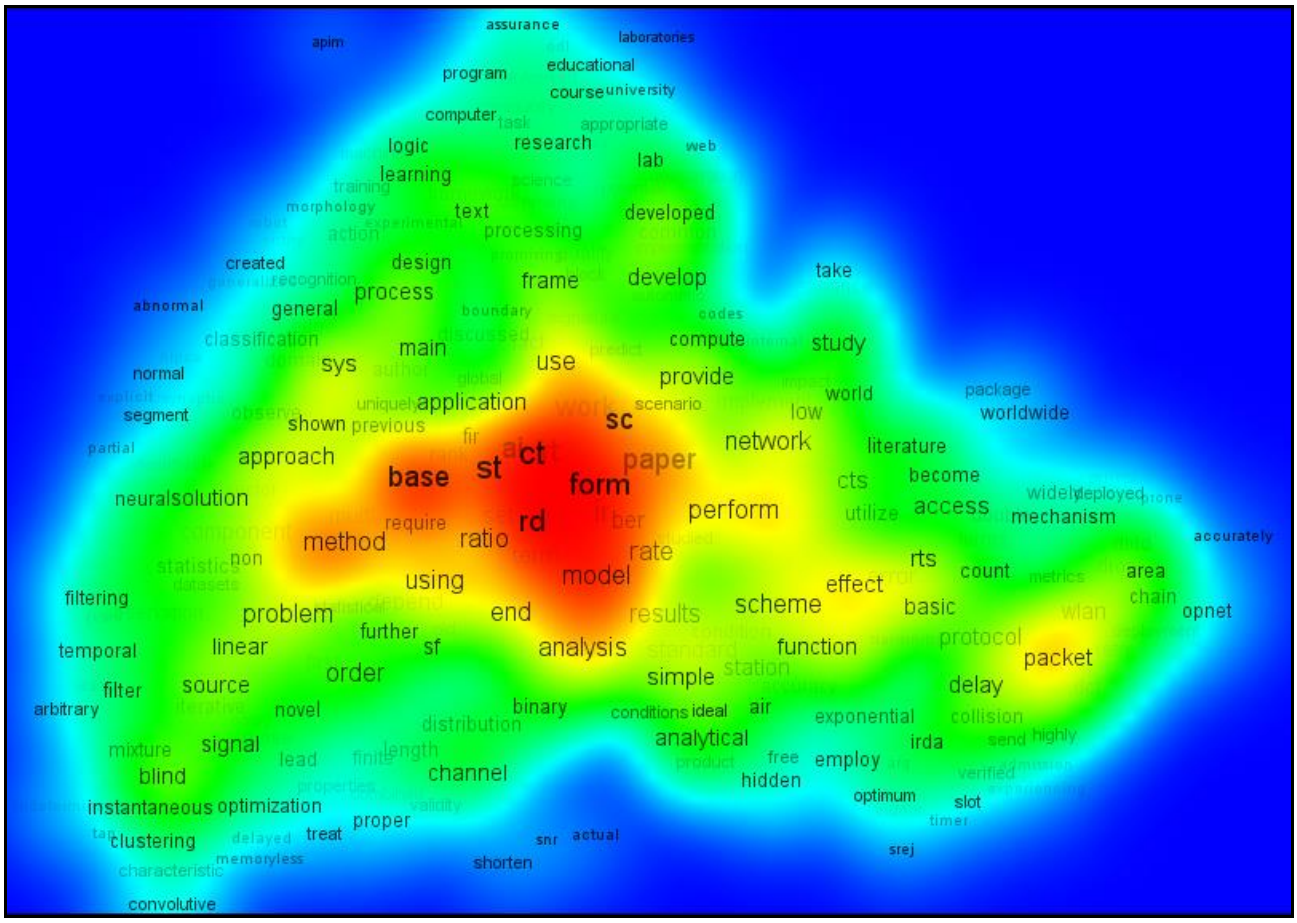
συγκεκριμένα για να πάρουμε μια γενική εποπτεία των σημαντικών περιοχών ενός χάρτη.

Παρακάτω εμφανίζεται η οπτικοποίηση (Εικόνα 45) με χρήση της τροποποιημένης λίστας των 150 λέξεων.



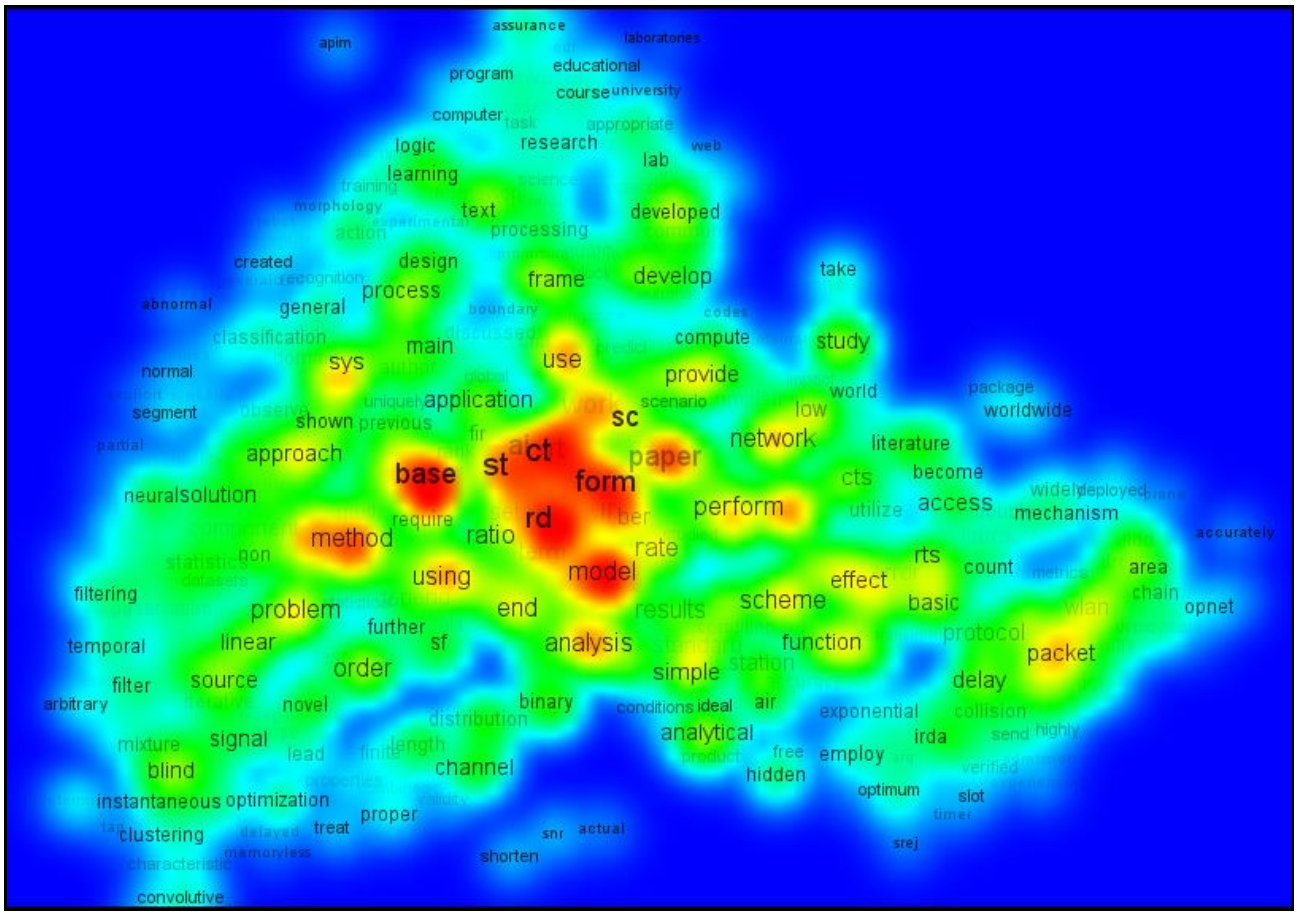
Εικόνα 45 Οπτικοποίηση 150 διαλεγμένων λέξεων σε density view.

Παρακάτω, στην εικόνα (46) που δείχνει το density view της οπτικοποίησης με τις 1.000 λέξεις, παρατηρούμε ότι όταν αυξάνεται σημαντικά ο αριθμός τότε ενοποιείται το κόκκινο χρώμα στο κέντρο, και γύρω του ακολουθεί το κίτρινο, το πράσινο, το γαλάζιο και τέλος το μπλε. Αυτό δε μας αφήνει να διακρίνουμε εύκολα τις χαμηλότερες “κορυφές” της εικόνας.



Εικόνα 46 Οπτικοποίηση 1000 λέξεων σε density view.

Για το πρόβλημα που δημιουργείται στην παραπάνω εικόνα, μια επιλογή είναι να πειράξουμε την κυλιόμενη κουκίδα με τίτλο “kernel width”, στο πεδίο “density view” της καρτέλας Options. Αν μειώσουμε αρκετά το φάρδος του πυρήνα του κάθε όρου το αποτέλεσμα είναι το ακόλουθο (Εικόνα 47).



Εικόνα 47 Οπτικοποίηση 1000 λέξεων σε density view με μειωμένο εύρος πυρήνων (kernel width).

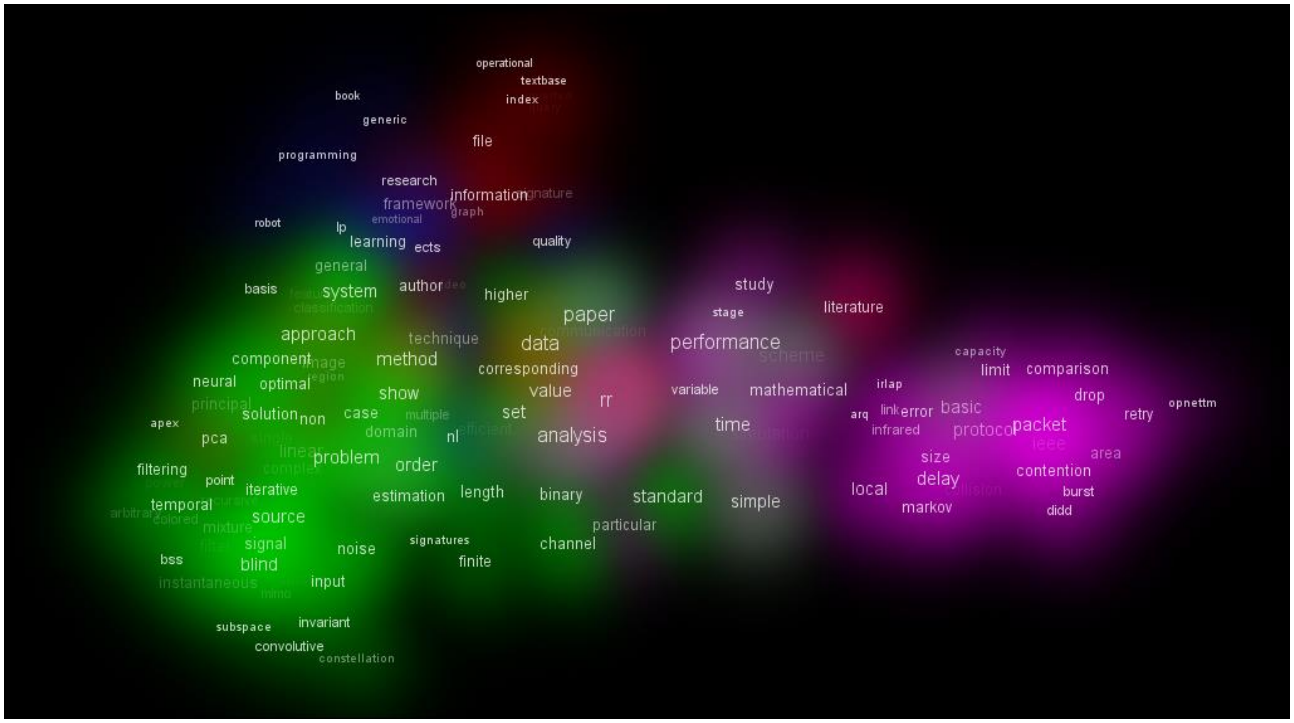
Ο VOS viewer δίνει πολλές επιλογές για αλλαγή χρωμάτων, αλλά σκοπός της ενότητας αυτής είναι να παρουσιάσει τις πιο χρήσιμες για την οπτικοποίηση του συγκεκριμένου προβλήματος. Είναι στην ευχέρεια του αναγνώστη αν επιθυμεί να πειραματιστεί, τροποποιώντας κάθε επιλογή.

4.4.3 Cluster Density view

Το Cluster Density View είναι μια οπτική που δίνει έμφαση στα χρώματα ώστε να διακρίνουμε πιο εύκολα την ομαδοποίηση (clustering). Η cluster density view είναι διαθέσιμη μόνο αν οι ομάδες (clusters) έχουν οριστεί στο αρχείο items. Στην τελευταία στήλη, όπως είδαμε στην παράγραφο 4.2.2.5, μπαίνει ο αριθμός για το χρώμα, που στην περίπτωσή μας, μας επιτρέπει να ξεχωρίζουμε τους καθηγητές. Κάθε αντικείμενο με αυτόν τον τρόπο, ανήκει σε μια ομάδα. Η cluster density view είναι παρόμοια με την κανονική density view εκτός από το ότι η

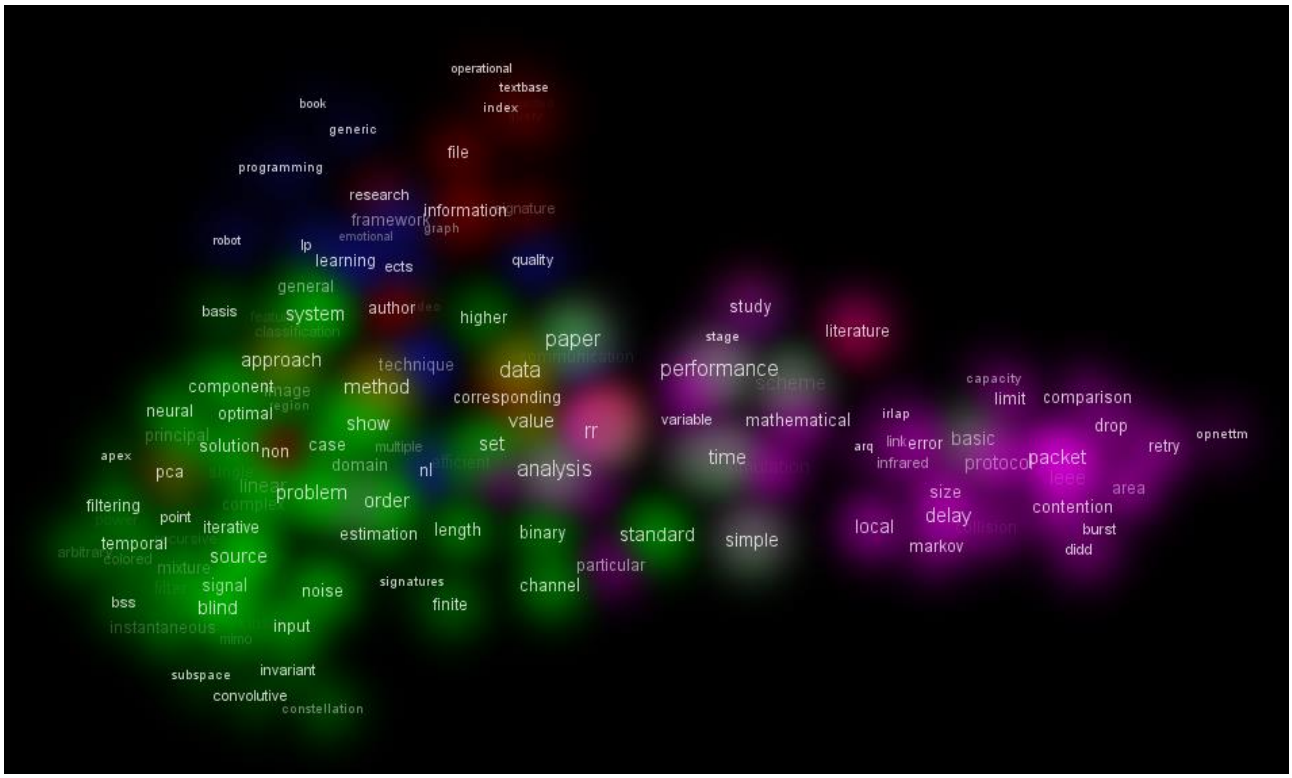
πυκνότητα των αντικειμένων απεικονίζεται χωριστά για κάθε cluster αντικειμένων. Σε αυτήν την οπτική, το χρώμα ενός σημείου στον χάρτη είναι κοντά στο χρώμα ενός συγκεκριμένου cluster, αν υπάρχει μεγάλος αριθμός αντικειμένων στην ίδια περιοχή που να ανήκουν σε αυτό το cluster. Όπως και στην κανονική density view, τα αντικείμενα με μεγάλα βάρη είναι πιο σημαντικά από τα άλλα με χαμηλά βάρη. Η cluster density view είναι χρήσιμη πιο συγκεκριμένα για μια γενική εποπτεία, όλων των ξεχωριστών clusters, των σημαντικών περιοχών του χάρτη.

Ακολουθεί η οπτικοποίηση του πειράματος των 200 λέξεων (Εικόνα 48) σε cluster density view.



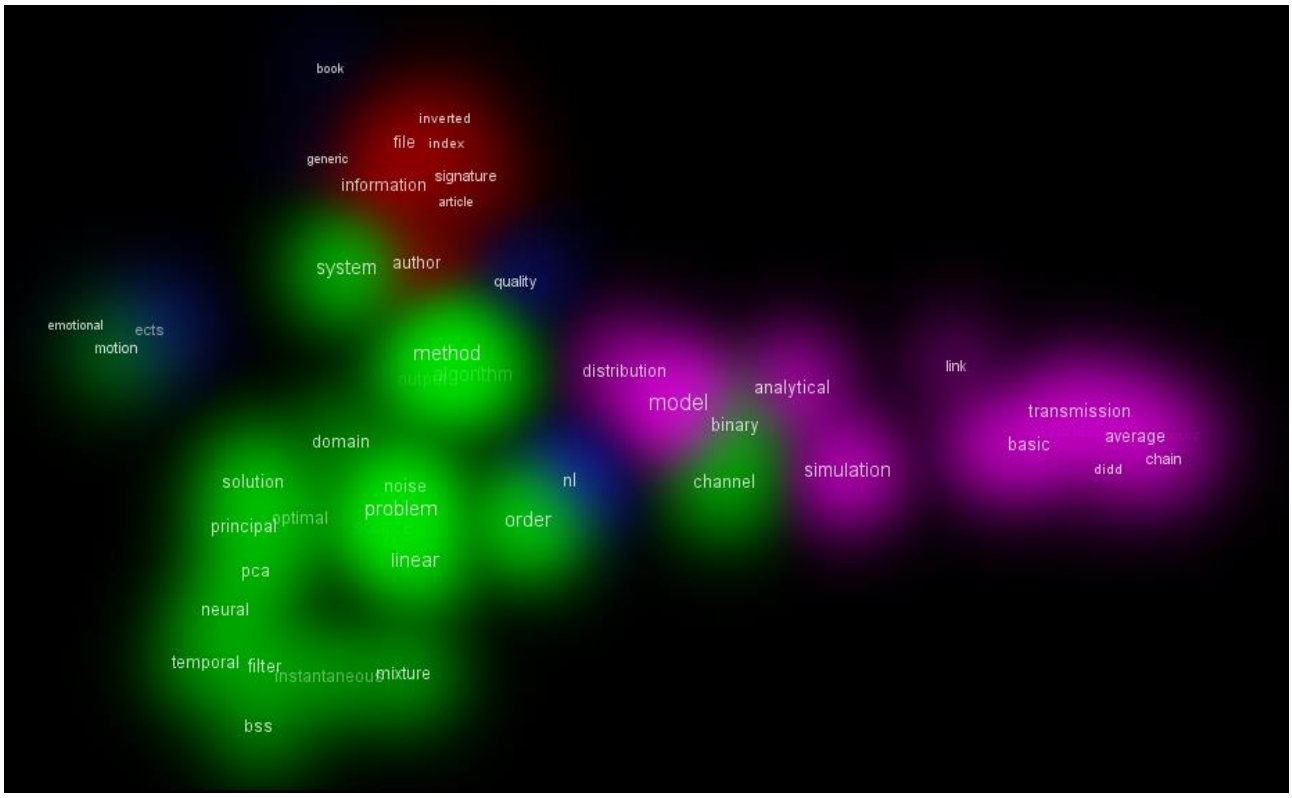
Εικόνα 48 Οπτικοποίηση 200 λέξεων σε cluster density view.

Και πάλι μειώνοντας το kernel width βοηθάμε να ξεκαθαρίσουν τα πράγματα (Εικόνα 49).



Εικόνα 49 Οπτικοποίηση 200 λέξεων με μειωμένο kernel width.

Η μείωση του kernel width δεν είναι αναγκαία για οπτικοποιήσεις με λίγες λέξεις. Παράδειγμα η εικόνα 50 που ακολουθεί με την οπτικοποίηση 50 λέξεων σε cluster density view.

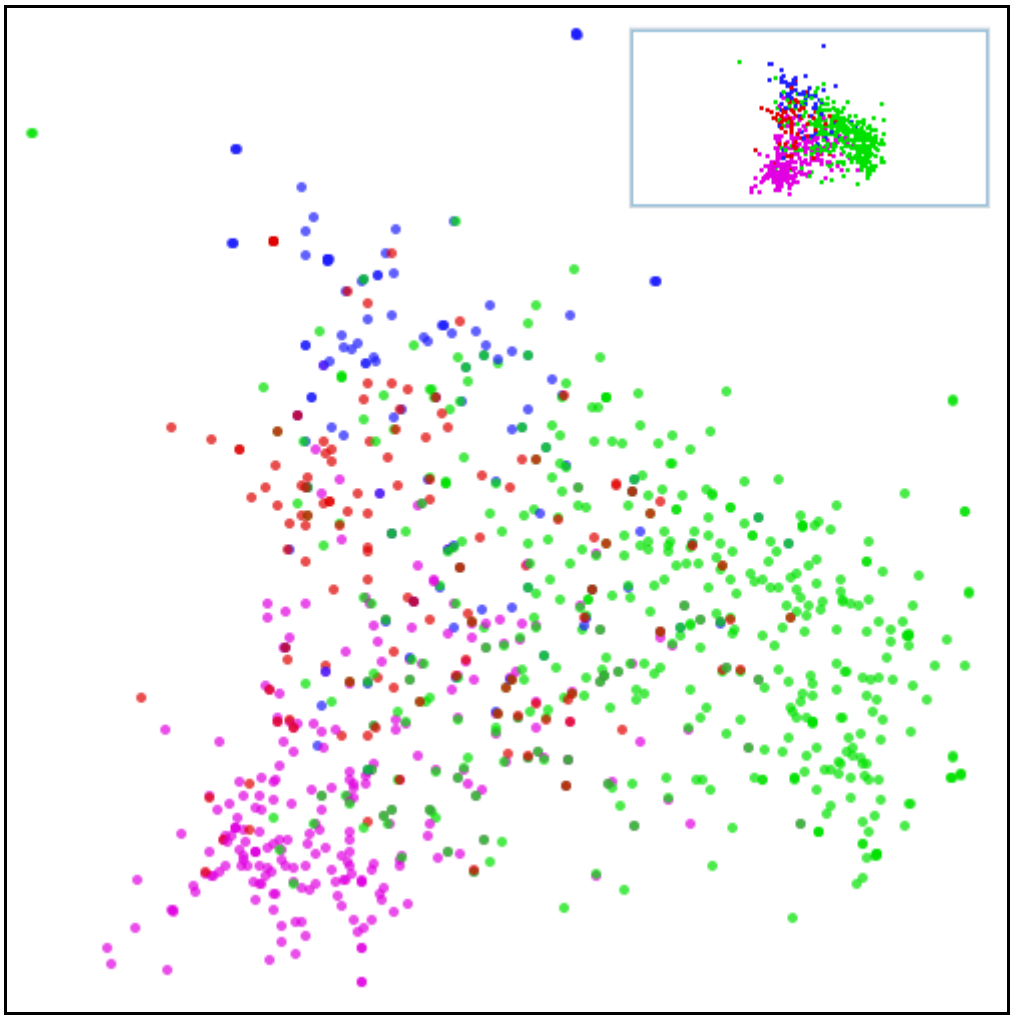


Εικόνα 50 Οπτικοποίηση 50 λέξεων σε cluster density view. Δεν υπάρχει ανάγκη να μειωθεί το kernel width. Οι όροι έχουν χώρο να εκφραστούν χρωματικά χωρίς επικαλύψεις.

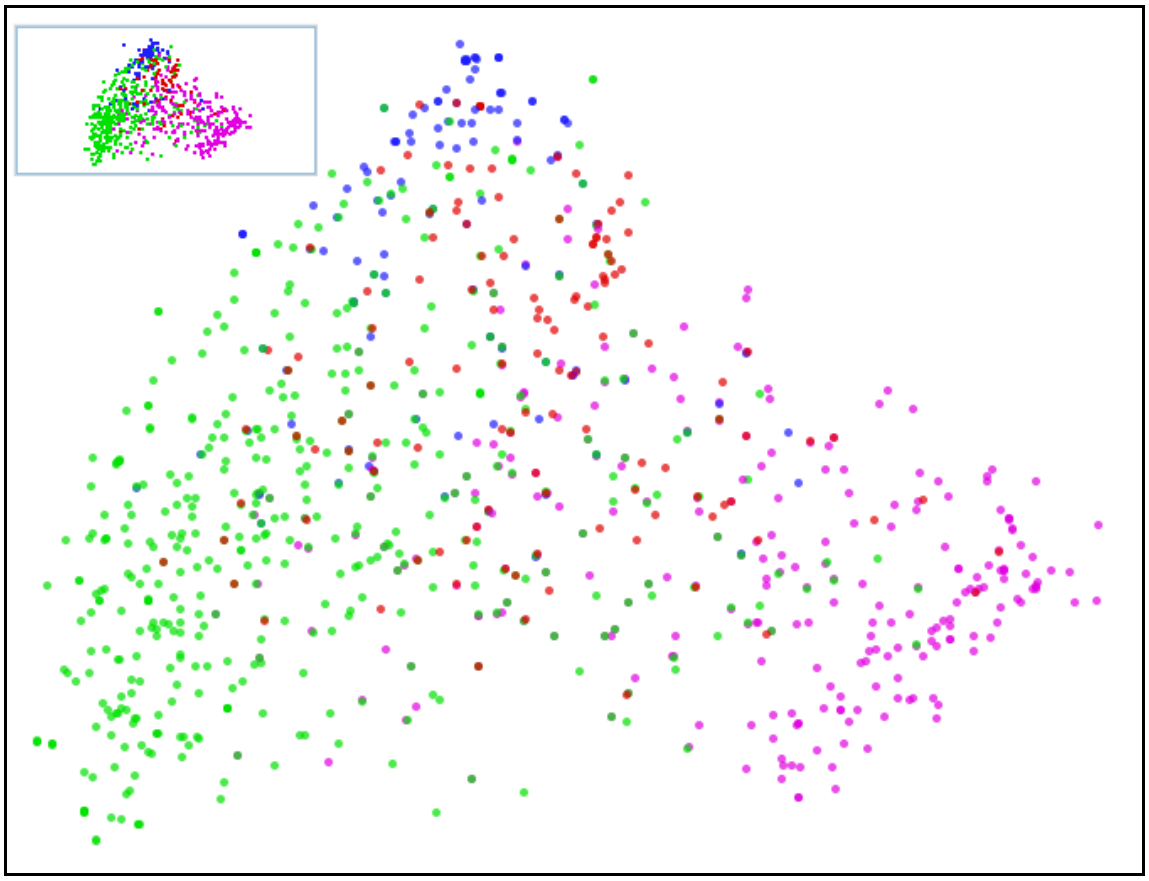
4.4.4 Scatter View

Στην οπτική αυτή, τα αντικείμενα συμβολίζονται με ένα μικρό κύκλο. Αν έχουν αποδοθεί χρώματα στα αντικείμενα, κάθε κύκλος απεικονίζεται στο χρώμα του αντίστοιχου αντικειμένου. Δεν εμφανίζονται οι ετικέτες. Η scatter view είναι χρήσιμη για να πάρουμε μια εικόνα της γενικής δομής ενός χάρτη. Ο τρόπος με τον οποίο φαίνεται ένας χάρτης σε scatter view είναι ο ίδιος με αυτόν που φαίνεται και στο overview panel**.** Το overview panel είναι μια μικρή οθόνη πάνω αριστερά στο κύριο Interface του VOS viewer, στην οποία, μέσα σε πολύ μικρό εμβαδό, εμφανίζεται κάθε αντικείμενο ως χρωματισμένη κουκκίδα.

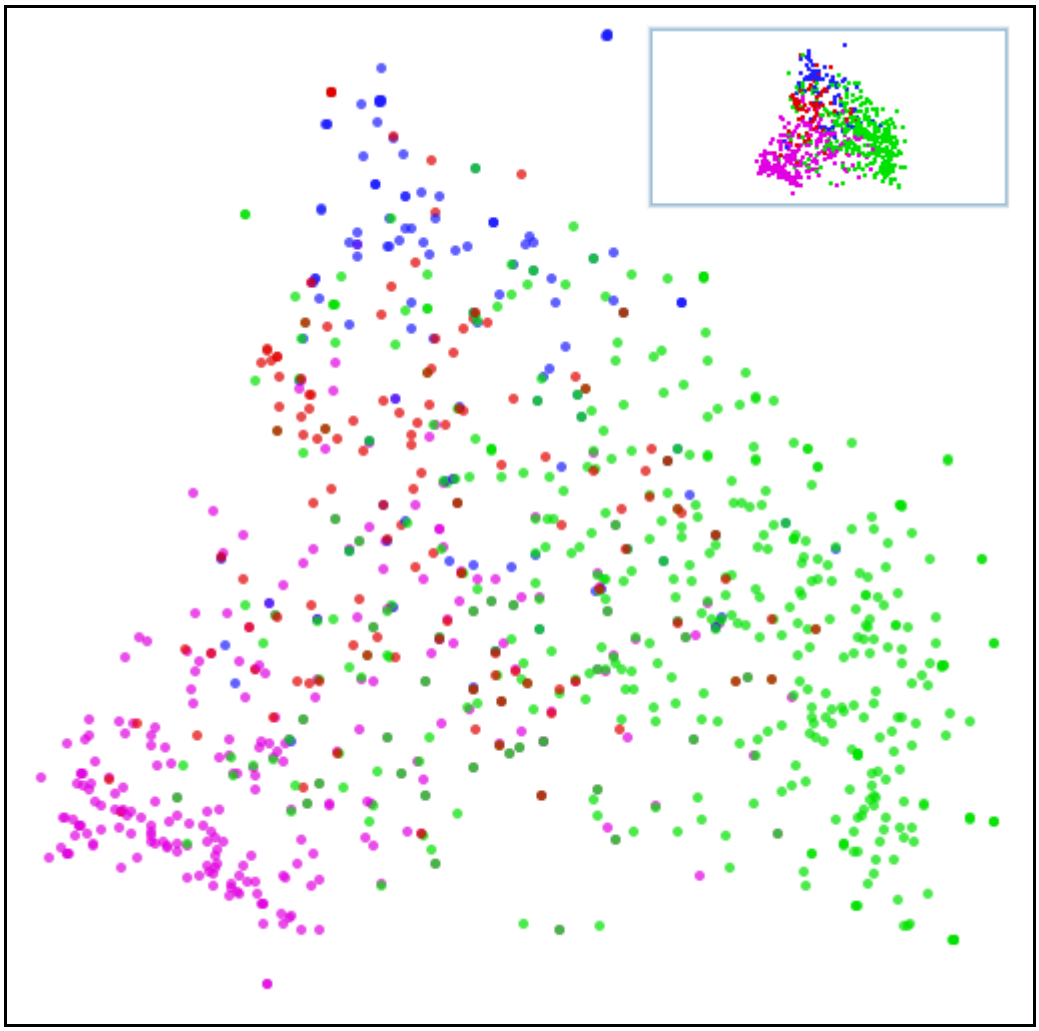
Ακολουθούν τρεις εκδοχές της οπτικοποίησης με τις 1.000 λέξεις σε scatter view (Εικόνες 51, 52 και 53), μία για κάθε ένα similarity measure διαθέσιμο από τον VOS viewer (Στην καρτέλα “options” στο πεδίο “similarities” έχουμε τρεις επιλογές: “similarity measure 1” “similarity measure 2” και “similarity measure 3”)



Εικόνα 51 Οπτικοποίηση 1000 λέξεων σε scatter view και similarity measure 1.



Εικόνα 52 Οπτικοποίηση 1000 λέξεων σε scatter view και similarity measure 2



Εικόνα 53 Οπτικοποίηση 1000 λέξεων σε scatter view και similarity measure 3.

Τα παραδείγματα για κάθε οπτική που παρατίθενται στο VOS viewer manual, βρίσκονται στο ΠΑΡΑΡΤΗΜΑ Γ’.

4.5 Αξιολόγηση – Αποτίμηση αποτελεσμάτων

Από τα διάφορα μεγέθη των λεξικών, αλλά και από τις ποικίλες επιλογές που προσφέρει ο VOS viewer, προέκυψε ανάλογος αριθμός οπτικοποιήσεων. Κάθε μία από αυτές δείχνει κάποιες βασικές πληροφορίες για τα πεδία έρευνας του κάθε καθηγητή και για τις μεταξύ τους σχέσεις, αλλά είναι μερικές που θα μπορούσαν να θεωρηθούν περισσότερο “πετυχημένες”.

Παρότι το βασικό σενάριο ήταν αυτό με τις 100 λέξεις, και αυτό με τις 1000 υπήρχε περισσότερο για λόγους σύγκρισης, το δεύτερο αποτελεί αισθητικά την

καλύτερη οπτικοποίηση γιατί μας δίνει μια εικόνα πολύ πιο “γεμάτη” απ' ότι οι υπόλοιπες οπτικοποιήσεις (η διαφορά ακόμη και με αυτή των 300 λέξεων είναι εξόφθαλμη). Ο μεγάλος όγκος των λέξεων οδηγεί στο να καλύπτονται οι διάφορες περιοχές με χρώμα, και να γίνονται πιο εύκολα αναγνωρίσιμα τα clusters του χάρτη. Επίσης ο μεγάλος όγκος λέξεων, δίνει ευκαιρία και σε λέξεις με μεγάλη σημασία, που όμως αποκλείονταν από τις μικρότερες λίστες λόγω χαμηλού βάρους, να αναδειχθούν στο χάρτη αφού εμφανίζονται σε πολλά papers. Η ύπαρξη δε πολλών λέξεων βοηθά στην καλύτερη περιγραφή του γνωσιακού πεδίου κάθε καθηγητή.

Με βάση τα παραπάνω κριτήρια**,** ως οι επόμενες καλύτερες οπτικοποιήσεις μπορούν να θεωρηθούν αυτές των 200 και 300 λέξεων, μαζί με τις παρελκόμενές τους εκδοχές όπως η density και cluster density view.

Ιδιαίτερο ενδιαφέρον έχουν οι οπτικοποιήσεις που προέκυψαν από τις τροποποιημένες λίστες, και ιδιαίτερα από αυτή με τις 150 λέξεις. Λόγω του ότι οι λέξεις έχουν φιλτραριστεί και έχουν χρησιμοποιηθεί μόνο υψηλού νοήματος λέξεις, η έκφραση των εννοιών είναι πιο αξιόπιστη ακόμη και με λίγες λέξεις, και η σχέση μεταξύ τους αποδίδεται με τρόπο που πλησιάζει την πραγματικότητα.

4.6 Επίλογος

Σε αυτό το κεφάλαιο είδαμε τις οπτικοποιήσεις έτσι όπως προέκυψαν από τις αντίστοιχες λίστες. Αναλύσαμε τις εικόνες, και παράξαμε από αυτές**,** με τη βοήθεια του VOS viewer**,** επιπλέον εικόνες, δίνοντας έμφαση σε συγκεκριμένα χαρακτηριστικά κάθε φορά. Πειραματιζόμενοι με τις διαθέσιμες όψεις του προγράμματος καθώς και με τον τρόπο κατασκευής των θησαυρών, είναι δυνατόν να προκύψουν πολλαπλάσιες εικόνες, οι οποίες θα μας λένε τα ίδια πράγματα με τον ίδιο ή διαφορετικό τρόπο, ίσως και ακόμη περισσότερα. Η οπτικοποίηση**,** όποιο θέμα και να έχει, δεν μπορεί να φτάσει σε τέλμα, αφού η πληροφορία αυξάνεται, τροποποιείται και εξελίσσεται. Μιας και η οπτικοποίηση είναι επιπλέον μια καλλιτεχνική διαδικασία, από αυτό και μόνο είναι σαφές ότι μπορεί να βελτιώνεται χωρίς όρια.

ΚΕΦΑΛΑΙΟ 5 - ΣΥΜΠΕΡΑΣΜΑΤΑ – ΠΡΟΤΑΣΕΙΣ ΓΙΑ ΜΕΛΛΟΝΤΙΚΗ ΕΡΓΑΣΙΑ

5.1 Εισαγωγή

Έχοντας ολοκληρώσει τη βασική διαδικασία οπτικοποίησης και έχοντας παρουσιάσει μερικές παραλλαγές, έπεται να καταγράψουμε κάποια συμπεράσματα, θέτοντας όμως τη βάση και για μελλοντική εργασία. Στο πρώτο μέρος αυτού του κεφαλαίου, θα αναφερθούμε συνοπτικά στην όλη διαδικασία και στο πού αυτή μας οδήγησε. Στο δεύτερο μέρος, παρουσιάζονται κάποιες πιθανές βελτιώσεις που θα μπορούσε να υποστεί η διαδικασία που ακολουθήθηκε, καθώς και μερικά θεωρητικά σενάρια που μπορούν να υλοποιηθούν στο μέλλον για βελτίωση της οπτικοποίησης και για διευκόλυνση της διαδικασίας δημιουργίας της.

5.2 Συμπεράσματα

5.2.1 Το αρχικό πρόβλημα

Αυτό που θέλαμε να πετύχουμε από την αρχή ήταν να οπτικοποιήσουμε το ερευνητικό πεδίο τεσσάρων καθηγητών του ΑΤΕΙ Θεσσαλονίκης. Η εργασία Van Eck, N.J. Et al (2006) αποτέλεσε την κύρια έμπνευση, για το πώς θα ήταν περίπου το τελικό οπτικό αποτέλεσμα. Στόχος μας ήταν, οι λέξεις που να περιγράφουν έννοιες πληροφορικής να τοποθετηθούν στο δισδιάστατο χώρο, ο όγκος της κάθε λέξης - αντικειμένου (σε σχήμα φυσαλίδας) να δηλώνει το πόσα κείμενα την περιλαμβάνουν έστω μία φορά, το χρώμα να δηλώνει τον καθηγητή στου οποίου τα κείμενα βρέθηκε, και φυσικά, η τοποθέτηση στο χώρο να εκφράζει την εγγύτητα ή μη, με τις υπόλοιπες λέξεις, μέσω των συμπτώσεών τους σε ίδια κείμενα.

5.2.2 Σύντομη αναφορά στην αντιμετώπιση

Συλλέξαμε όσες περιλήψεις μπορούσαμε από τους τέσσερις καθηγητές. Παράξαμε άλλα τόσα κείμενα, με τις επεξεργασμένες εκδοχές αυτών μέσω του MontiLingua. Χρησιμοποιήσαμε το Rapid Miner για να εξάγουμε όλες τις λέξεις που εμφανίζονται στα αρχεία αυτά, και να τις ταξινομήσουμε ανάλογα με το tf-idf βάρος που τους αποδόθηκε. Κατασκευάσαμε μια λίστα - λεξικό με λέξεις από κάθε καθηγητή ανάλογα με το πόσες δημοσιεύσεις του χρησιμοποιήσαμε. Γράψαμε ένα πρόγραμμα σε Java που διαβάζει το λεξικό και αναζητά κάθε μία λέξη στον όγκο των δημοσιεύσεων συμπληρώνοντας και αποθηκεύοντας έναν πίνακα διανυσμάτων. Εισάγαμε τον πίνακα διανυσμάτων και το λεξικό στον VOS viewer 1.2 ο οποίος οπτικοποίησε το αποτέλεσμα.

5.2.3 Η θέαση του αποτελέσματος

Εδώ αξίζει να αναφερθούμε λίγο στους πιθανούς θεατές. Ενδέχεται σε κάποιον που δεν έχει σχέση με την πληροφορική, αυτή η οπτικοποίηση να μη προσφέρει τίποτα. Στην καλύτερη περίπτωση μπορεί να την θεωρήσει μια συμπαθητική χρωματιστή εικόνα. Επομένως είναι πιθανό ότι οι εν λόγω εικόνες, δεν αναφέρονται σε όλους αλλά σε κάποιους εξειδικευμένους θεατές. Ακόμη και για τους δεύτερους, θα μπορούσε να είναι μια δυσνόητη εικόνα αν δε ξεκαθαριστεί με κάποιο κείμενο το πλαίσιο στο οποίο εντάσσονται οι εικόνες αυτές.

Έστω λοιπόν ότι κάποιος είναι σχετικός, αλλά δε γνωρίζει τους καθηγητές του τμήματος, και βλέπει τις εικόνες. Τα χρώματα τον βοηθούν να ξεχωρίσει τον κάθε καθηγητή και να υποθέσει περίπου σε τί αντικείμενο κινείται, παρότι οι ομαδοποιήσεις είναι κάπως δυσδιάκριτες. Φυσικά βλέποντας το σύνολο των οπτικοποιήσεων και τις διαφορετικές οπτικές, είναι δυνατόν να διακριθούν και τα δύο clusters που έχουν τους λιγότερους όρους.

Εάν κάποιος έχει άμεση σχέση με το τμήμα, τότε οι εικόνες βγάζουν πολύ περισσότερο νόημα και φαίνεται ότι ανταποκρίνονται σε αυτό που κάνει ο κάθε ένας από τους καθηγητές που συμμετείχαν.

Τελικά καταφέραμε αυτό που θέσαμε ως αρχικό στόχο. Δημιουργήσαμε

χάρτες με χρωματιστές φυσαλίδες που αντιστοιχούν στους όρους του λεξικού, οι εικόνες που προέκυψαν, αποδίδουν ικανοποιητικά το γνωσιακό πεδίο των καθηγητών, ενώ και η σχέση μεταξύ των ομάδων στον χάρτη, φαίνεται να έχει λογική σε σύγκριση με την πραγματικότητα. Το αποτέλεσμα δεν είναι τέλειο και δε μπορεί να είναι. Κάποιες πιθανές βελτιώσεις, που μελλοντικά θα μπορούσαν να αποτελέσουν τα πρώτα βήματα για την αναβάθμιση της οπτικοποίησης, αναφέρονται στις ακόλουθες παραγράφους.

5.3 Προτάσεις για Μελλοντική Εργασία

5.3.1 Νέος VOS viewer

Στις 10 Σεπτεμβρίου του 2011 και ενώ η συγγραφή της παρούσας εργασίας ήταν σε τελικό στάδιο, δημοσιεύτηκε η νέα έκδοση του VOS viewer με αριθμό 1.4.0. Είναι διαθέσιμη για κατέβασμα στην ιστοσελίδα των Nees Jan van Eck και Ludo Waltman (VOS viewer, July 2010). Η σημαντικότερη νέα λειτουργία του είναι η χρήση τεχνικών εξόρυξης κειμένου (text mining techniques) για τη δημιουργία χαρτών που βασίζονται σε μεγάλο όγκο κειμένων. Άλλες νέες δυνατότητες περιλαμβάνουν τη βελτιωμένη λειτουργία για τη δημιουργία νέων χαρτών, επεκτάσεις του τύπου αρχείου του χάρτη, και επιπλέον υποστήριξη για αρχεία Pajek.

Ο νέος VOS viewer δέχεται δύο είδη αρχείων μιας κατηγορίας που την ονομάζει “network”. Η μία κατηγορία που είναι ένα txt αρχείο “πλήρους δικτύου” (full network), είναι ουσιαστικά ίδιο με το αρχείο που κατασκευάσαμε με τα διανύσματα, μόνο που στα αριστερά έχει μία επιπλέον στήλη με τον αύξοντα αριθμό κάθε γραμμής.

Τα άλλα αρχεία, είναι αρχεία κειμένου “αραιού δικτύου” (sparse network), τα οποία έχουν ουσιαστικά την ίδια πληροφορία με τα πλήρους δικτύου, μόνο που η δομή τους είναι πιο «οικονομική». Έχουν τρεις στήλες. Οι δύο πρώτες αντιστοιχούν στις συντεταγμένες του πίνακα διανυσμάτων, και η τρίτη στο περιεχόμενο του αντίστοιχου ζεύγους συντεταγμένων. Η ιδιαιτερότητα του αρχείου

αυτού είναι ότι, όταν μεταξύ των όρων που αντιστοιχούν στις συντεταγμένες δεν υπήρξε σύμπτωση, και επομένως εκείνη η θέση του πίνακα είχε την τιμή 0, το ζεύγος των συντεταγμένων αυτών και μαζί η τιμή 0 που τους αντιστοιχεί, παραλείπεται από το αρχείο αραιού δικτύου.

Τέλος, ο νέος VOS viewer διαβάζει ένα αρχείο κειμένου που περιέχει όλη την πληροφορία για ένα χάρτη, και είναι της μορφής:

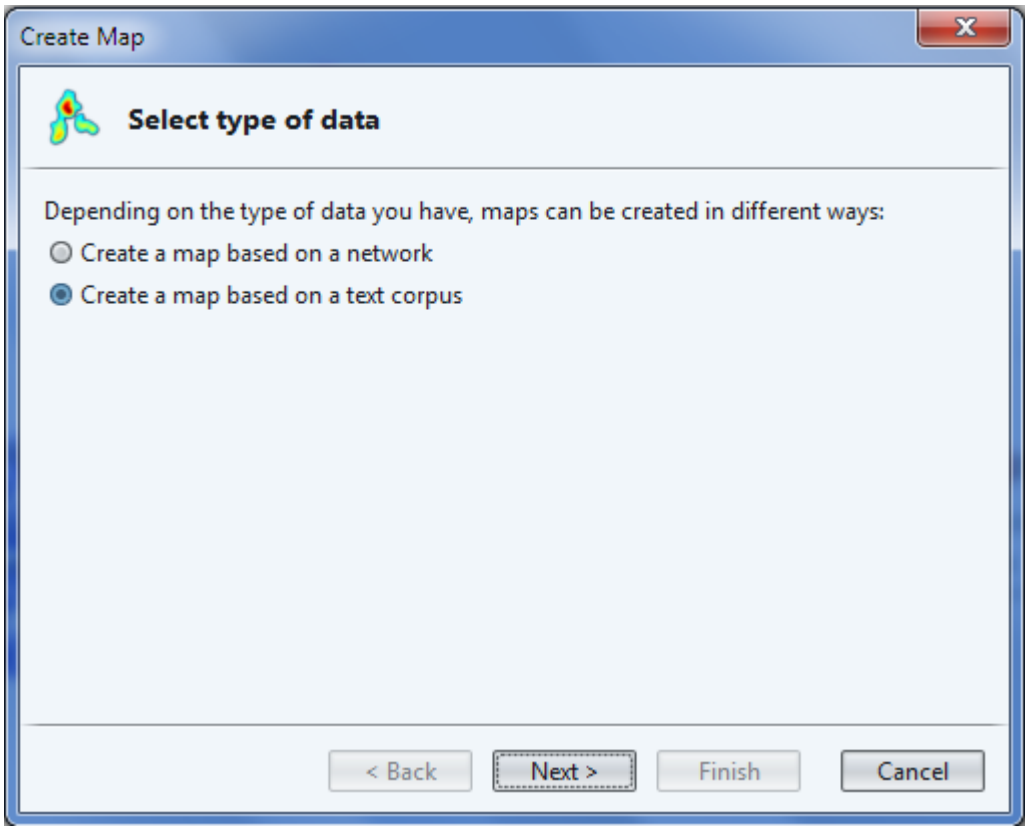
id, label, description, x, y, weight, cluster.

Το περιεχόμενο κάθε γραμμής περιέχει τον αριθμό id της πρώτης στήλης που είναι ουσιαστικά ο αύξων αριθμός της σειράς, την ετικέτα στη 2η στήλη, την περιγραφή στην 3η, τις συντεταγμένες στο δισδιάστατο χώρο στην 4η και 5η στήλη, το βάρος στην έκτη, και έναν ακέραιο που συμβολίζει την ομάδα στην οποία ανήκει ο όρος και κατ'επέκταση το χρώμα με το οποίο θα εμφανιστεί, στην 7η στήλη.

Τα παραπάνω αρχεία ο VOS viewer έχει τη δυνατότητα να τα παράξει, εφόσον δημιουργήσει έναν χάρτη από έναν όγκο κειμένων (text corpora). Πατώντας το κουμπί “create” στην καρτέλα “Action” του νέου VOS viewer, υπάρχει η επιλογή να κατασκευαστεί ένας χάρτης από αρχείο δικτύου, ή από έναν όγκο κειμένων. Με τη δεύτερη επιλογή το πρόγραμμα μας ζητά να εισάγουμε ένα αρχείο κειμένου, στο οποίο κάθε σειρά αντιπροσωπεύει μια δημοσίευση, την περίληψή της, μια παράγραφο ή κομμάτι της, ή τον τίτλο της. Όταν εισαχθεί αυτό το αρχείο, το πρόγραμμα εντοπίζει τους σημαντικούς όρους και παράγει μια οπτικοποίηση.

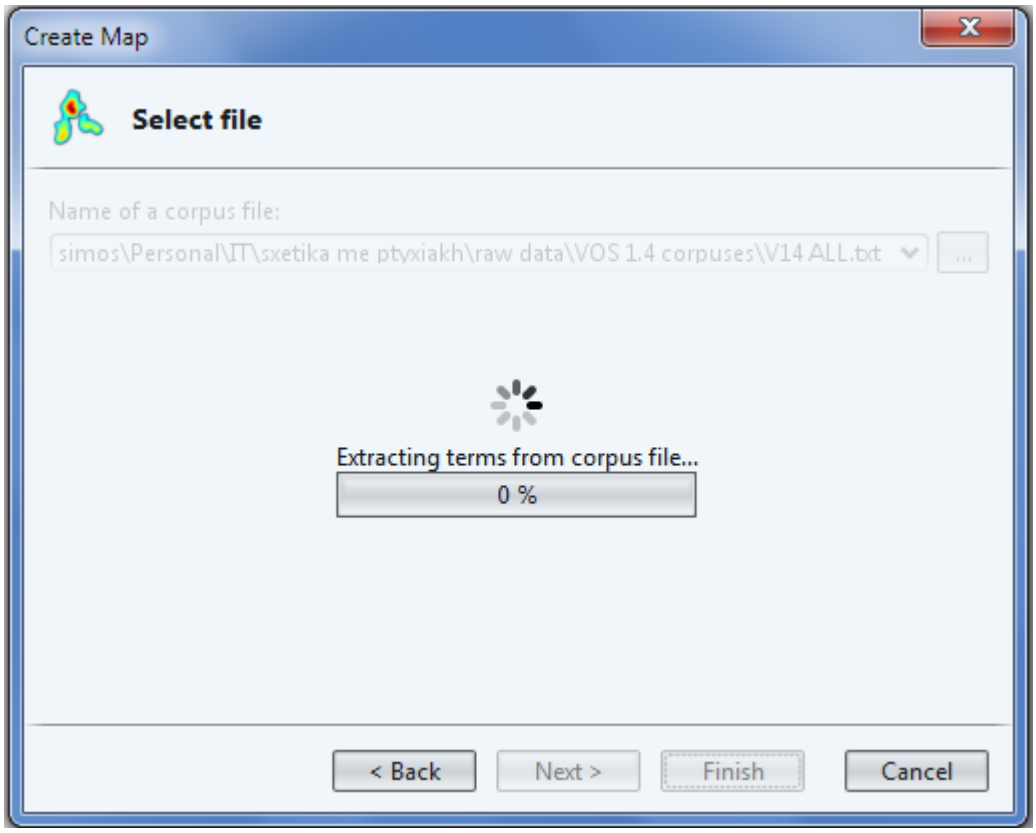
Προκειμένου να δοκιμαστεί το νέο πρόγραμμα, συντάχθηκε ένα αρχείο κειμένου που σε κάθε σειρά του είναι μία περίληψη δημοσίευσης. Η κάθε δημοσίευση του κάθε καθηγητή, εννοείται πως υπάρχει μία μόνο φορά σε κάποια γραμμή του αρχείου. Όταν δίνεται το αρχείο ως είσοδος στον VOS viewer, μεσολαβούν κάποια στάδια μέχρι την τελική εικόνα, τα οποία φαίνονται παρακάτω.

Πρώτα επιλέγουμε τον τρόπο με τον οποίο θα δημιουργήσουμε τον χάρτη. (Στην δική μας περίπτωση από ένα text corpus).



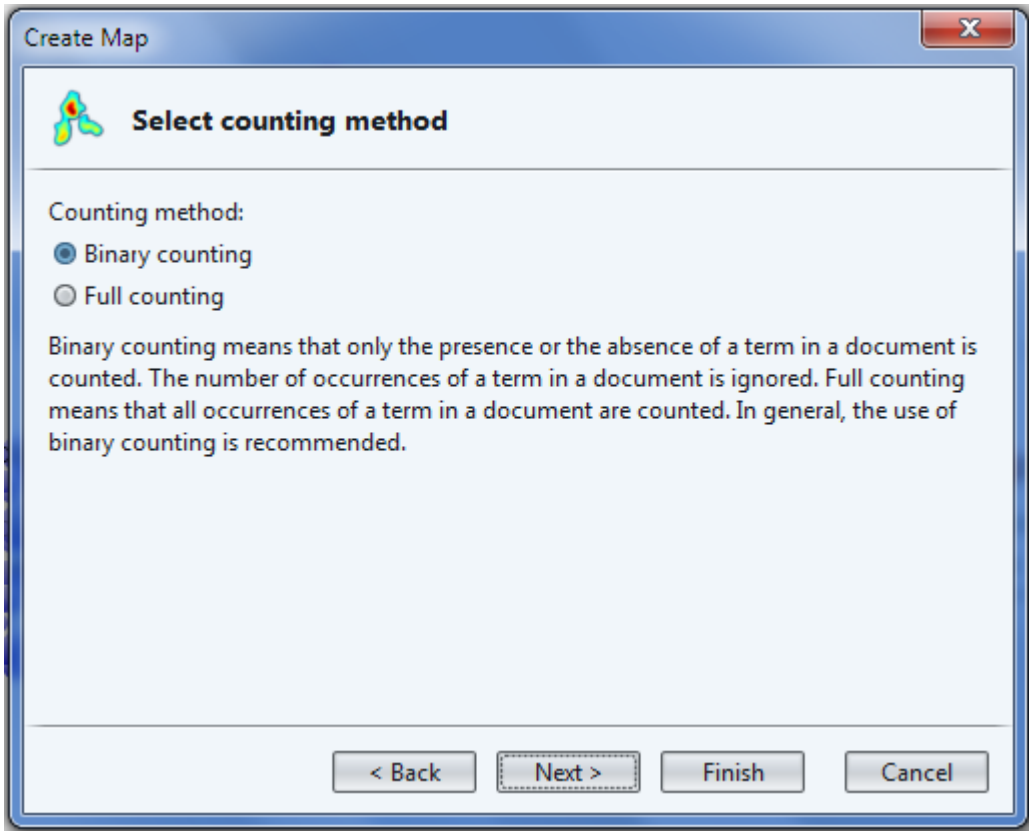
Εικόνα 54 1ο βήμα οπτικοποίησης με τον VOS viewer 1.4.0

Μετά το πρόγραμμα χρειάζεται μερικά δευτερόλεπτα για να εξάγει τους όρους του αρχείου που επιλέξαμε.



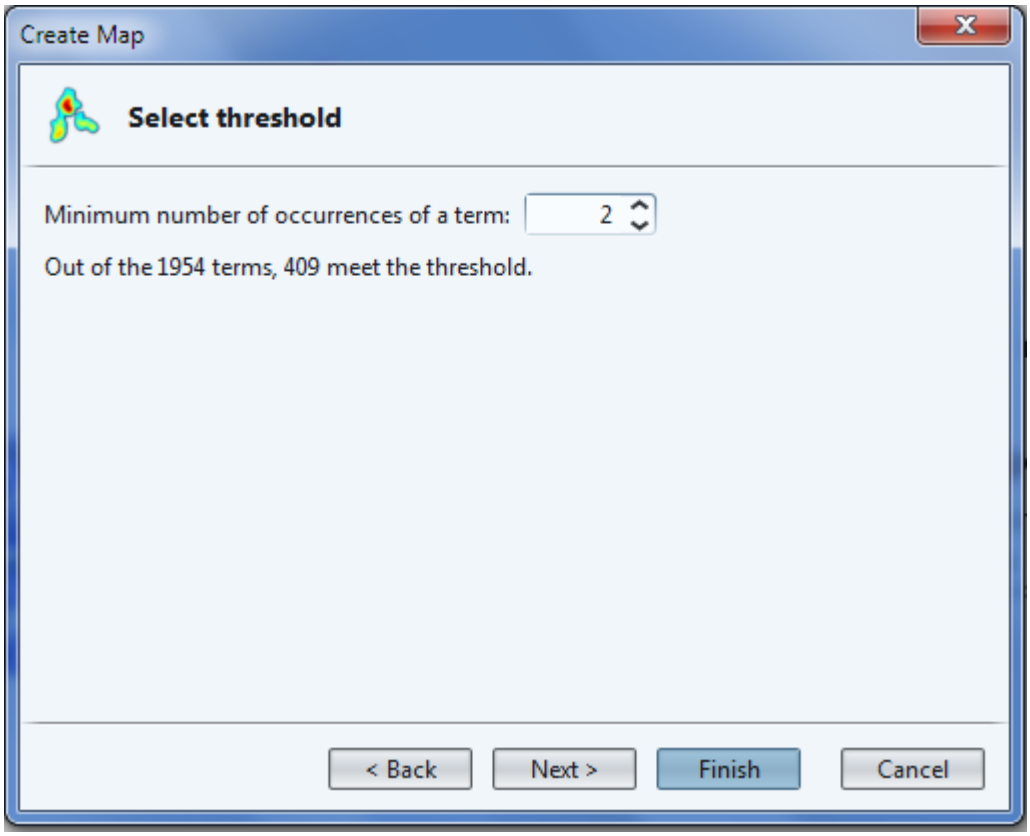
Εικόνα 55 Στιγμιότυπο της οπτικοποίησης με τον VOS viewer 1.4.0

Δεύτερον, επιλέγουμε αν θέλουμε να μετρηθεί η ύπαρξη του εκάστοτε όρου στις περιλήψεις, ή το σύνολο των εμφανίσεών σε κάθε μία. (Επιλέγουμε το προτεινόμενο, όπως φαίνεται στην εικόνα 56)



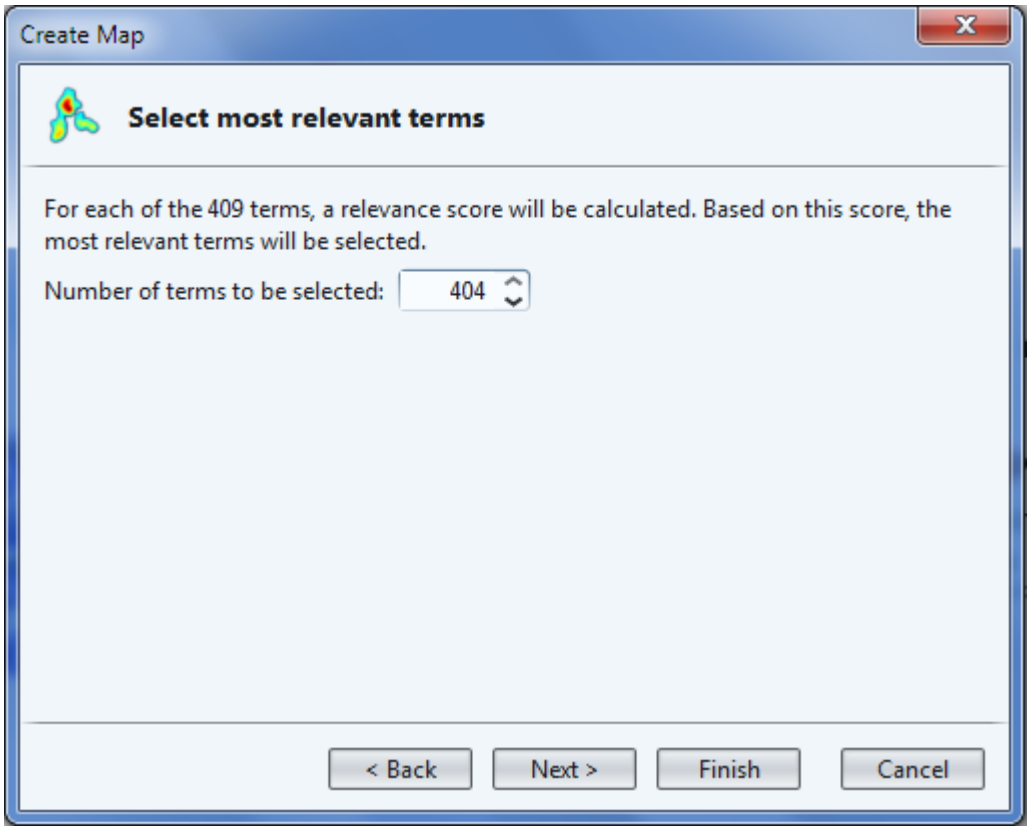
Εικόνα 56 2ο βήμα οπτικοποίησης με τον VOS viewer 1.4.0

Τρίτον, επιλέγουμε το κατώφλι που θα αποκλείσει τους όρους που εμφανίστηκαν μόνο μία φορά.(το κατώφλι το ορίζει ο χρήστης όσο θέλει, και το πρόγραμμα τον ενημερώνει απευθείας για το πόσους όρους έχουν αποκλειστεί.)



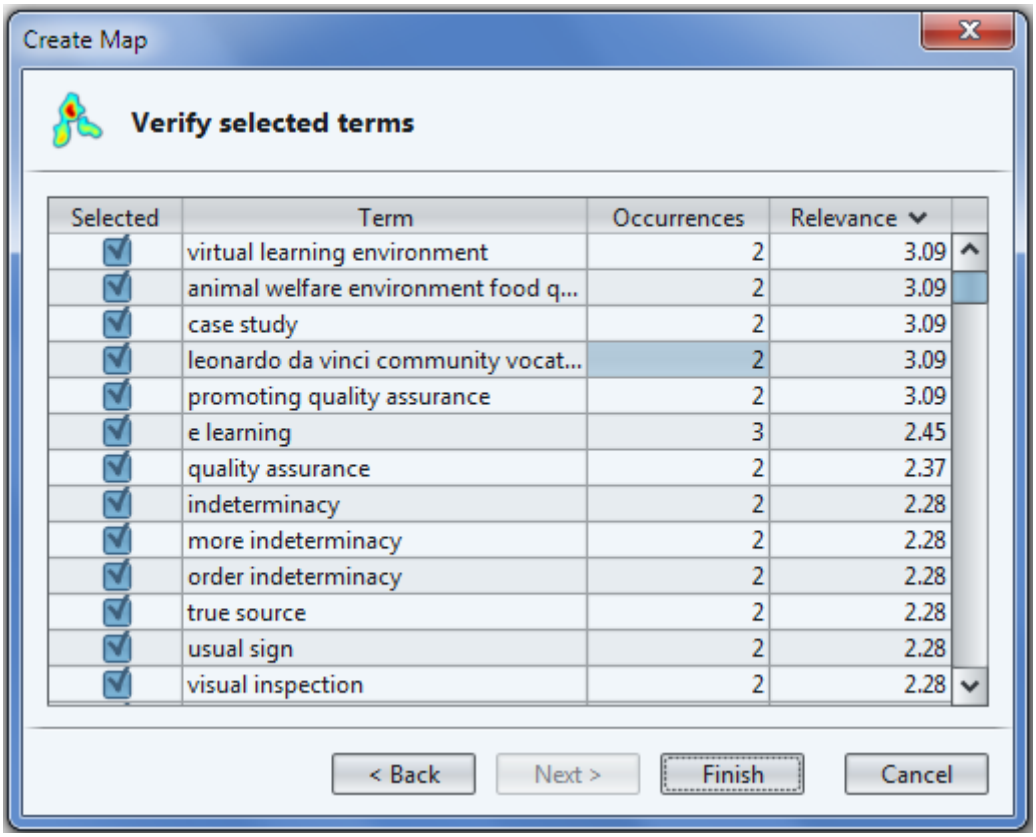
Εικόνα 57 3ο βήμα οπτικοποίησης με τον VOS viewer 1.4.0

Επιλέγουμε πόσους από τους όρους θέλουμε το πρόγραμμα να θεωρήσει σχετικούς. (προεπιλογή είναι η μισοί όροι, αλλά εμείς θέλουμε μια γεμάτη εικόνα)



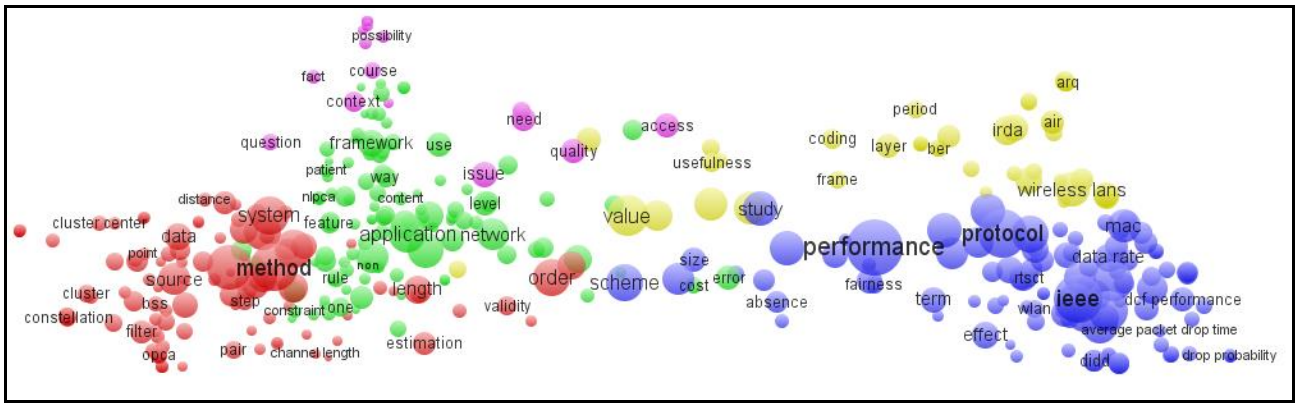
Εικόνα 58 4ο βήμα οπτικοποίησης με τον VOS viewer 1.4.0

Πριν την οπτικοποίηση ο VOS viewer μας παρέχει μια προεπισκόπηση των όρων που επιλέχθηκαν, όπου μπορούμε να αποκλείσουμε όποιον δε μας αρέσει.

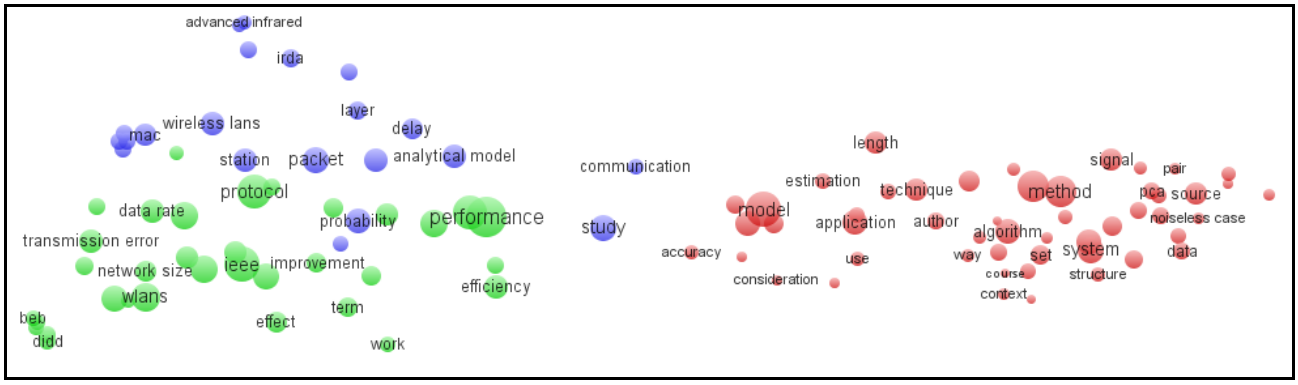


Εικόνα 59 5ο βήμα οπτικοποίησης με τον VOS viewer

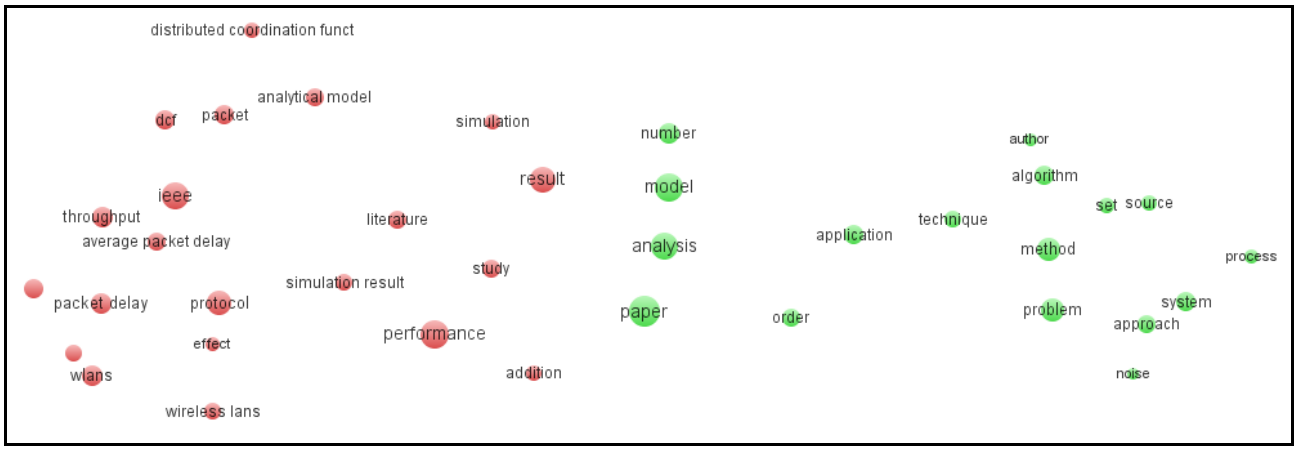
Ακολουθούν οι τέσσερις οπτικοποιήσεις με κατώφλι 2,4,8 και 10 αντίστοιχα.



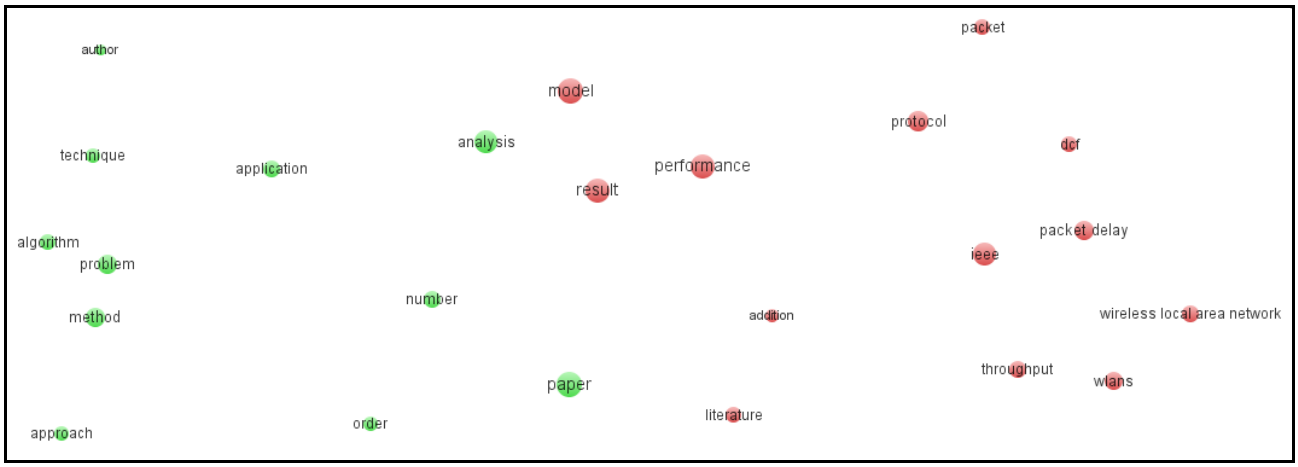
Εικόνα 60 οπτικοποίηση στον νέο VOS viewer με κατώφλι 2



Εικόνα 61 οπτικοποίηση στον νέο VOS viewer με κατώφλι 4



Εικόνα 62 οπτικοποίηση στον νέο VOS viewer με κατώφλι 8



Εικόνα 63 οπτικοποίηση στον νέο VOS viewer με κατώφλι 10

Αυτό που βλέπουμε είναι ότι το πρόγραμμα χωρίζει μόνο του τα clusters, αφού ο χρήστης δεν έχει την επιλογή εκ των προτέρων, παρά μόνο να πειράξει χειροκίνητα τις τιμές στα αρχεία network που μπορεί να αποθηκεύσει μετά την παραπάνω διαδικασία. Αυτό που κυριαρχεί σε κάθε εικόνα είναι οι δύο “πόλοι” στις πλάγιες άκρες του χάρτη. Στον ένα έχουμε μεγάλη συγκέντρωση των όρων του κ. Διαμαντάρα και στον άλλο του κ.Βίτσα. Όσο πιο μεγάλο κατώφλι θέτουμε και επομένως όσο περισσότερες λέξεις απορρίπτονται, τόσο η εικόνα καταλήγει με λιγότερα χρώματα και πιο αραιά διατεταγμένα αντικείμενα.

5.3.2 Οπτικοποίηση με διαφορετικό εργαλείο - Εισαγωγή διάστασης χρόνου

Προφανώς οι εναλλακτικές για τον τρόπο που μπορεί να παραχθεί μια οπτικοποίηση είναι αμέτρητες. Υπάρχει η προοπτική να ξαναγραφτεί ένα πρόγραμμα στο Processing, το οποίο να επιχειρεί να οπτικοποιήσει το ερευνητικό πεδίο του τμήματος Πληροφορικής του ΤΕΙ Θεσσαλονίκης. Οι δυνατότητες του περιβάλλοντος του Processing μπορεί να δώσουν άλλα χαρακτηριστικά στην οπτικοποίηση, από αισθητική και νοηματική πλευρά.

Θα μπορούσε για παράδειγμα να εισαχθεί και η διάσταση του χρόνου στην οπτικοποίηση. Αυτό μπορεί να γίνει με τη χρήση κινούμενων γραφικών, όπου στην οθόνη θα εμφανίζεται ένας μετρητής των ετών λειτουργίας από τότε που ξεκίνησε να υπάρχει το ίδρυμα, και καθώς αυτός αυξάνεται σε πραγματικό χρόνο, να εμφανίζονται νέες έννοιες (concepts) στον χάρτη, οι οποίες με το πέρασμα του χρόνου να αλλάζουν θέση ανάλογα με τη σχέση που προκύπτει μεταξύ τους, καθώς επίσης μέγεθος και χρώμα, ανάλογα με τη συχνότητα εμφανίσεων στις δημοσιεύσεις και το clustering. Θα μπορούσαν επίσης να χάνονται αυτές με τις οποίες δεν ασχολήθηκε κανείς για αρκετό καιρό.

5.3.3 Ανάπτυξη διεπαφής για τη δημιουργία των διανυσμάτων

Όσον αφορά το εργαλείο που συγγράφηκε σε Java για την παραγωγή των διανυσμάτων, θα μπορούσαμε να το φανταστούμε και με ένα interface που να παρέχει τις βασικές λειτουργίες του. Να δίνει τη δυνατότητα να επιλέγει κανείς κάθε φορά τον κατάλογο στον οποίο βρίσκεται ο όγκος των κειμένων προς ανάλυση, αλλά και το αρχείο με τη λίστα των λέξεων προς αναζήτηση. Ίσως η δυνατότητα προεπισκόπησης της λίστας να είναι επίσης χρήσιμη, καθώς και η επεξεργασία της σε πραγματικό χρόνο μέσα από τη διεπιφάνεια.

Μια σημαντική βελτίωση θα ήταν η δυνατότητα να αναγνωρίζει το πρόγραμμα όχι μόνο μεμονωμένες λέξεις αλλά, με μια γραμματική ανάλυση, να εντοπίζει τα ουσιαστικά μαζί με τα επίθετα που τα προσδιορίζουν, ώστε να μπορεί να αναζητήσει σύνθετους όρους μέσα στα κείμενα.

5.3.4 Αλλαγή των ποσοστών συμμετοχής των καθηγητών – παραπάνω καθηγητές-πλήρεις δημοσιεύσεις.

Είδαμε στα περισσότερα πειράματα των προηγούμενων κεφαλαίων, ότι η απόλυτα αναλογική συμμετοχή των καθηγητών στις λίστες έκανε την παρουσία αυτών με τις λιγότερες δημοσιεύσεις, ιδιαίτερα σε οπτικοποιήσεις βασισμένες σε μικρές λίστες, πολύ περιορισμένη. Αυτό δεν είναι απαραίτητα δίκαιο, γιατί το πλήθος των διαφορετικών χρήσιμων όρων μέσα στις δημοσιεύσεις ενός καθηγητή, μπορεί να μην είναι ανάλογο των δημοσιεύσεων αυτών.

Επομένως μια ενδιαφέρουσα δοκιμή θα ήταν να αλλάξουμε το σύστημα συμμετοχής. Αυτό θα μπορούσε να γίνει με το να ορίσουμε από πόσους όρους θα αποτελείται το κομμάτι του θησαυρού, στο οποίο θα συμμετέχει εξίσου ο κάθε καθηγητής, και ταυτόχρονα ποιό θα είναι το υπόλοιπο, στο οποίο οι καθηγητές θα συμμετέχουν αναλογικά με τον αριθμό των δημοσιεύσεών τους. Αυτό με ένα απλό παράδειγμα σημαίνει ότι, αν έχουμε ένα θησαυρό με 100 λέξεις και επιλέξουμε το 80% να είναι το μέρος που οι καθηγητές συμμετέχουν εξίσου, ο κάθε καθηγητής θα συμμετέχει με πάνω από 20 λέξεις στο θησαυρό, ανεξάρτητα από το αν με το προηγούμενο σύστημα θα συμμετείχε με λιγότερες.

Το τμήμα, εκτός από τους τέσσερις καθηγητές που έχουμε αναφέρει, έχει επιπλέον προσωπικό (αναπληρωτές, επίκουροι, εφαρμογών). Θα μπορούσαν να συμπεριληφθούν επιπλέον καθηγητές ώστε να επεκταθεί η οπτικοποίηση, και να

εντοπιστούν νέες ομαδοποιήσεις.

Αν θα υπήρχε η χρονική δυνατότητα, αποθηκεύοντας ολόκληρες τις δημοσιεύσεις των καθηγητών που συμμετέχουν στο πείραμα, θα μπορούσε να εμπλουτιστεί ο θησαυρός με όρους που ίσως δεν φάνηκαν, μόνο με τη χρήση των abstracts των δημοσιεύσεων.

5.4 Επίλογος

Τα βήματα και οι διαδικασίες που πραγματοποιήθηκαν, οδήγησαν σε μία ικανοποιητική οπτικοποίηση του knowledge domain (γνωσιακού πεδίου) των τεσσάρων καθηγητών που συμμετείχαν. Όπως έχει αναφερθεί και σε προηγούμενο κεφάλαιο, μία τέτοια εικόνα μπορεί να εξελίσσεται συνεχώς. Σίγουρα το αποτέλεσμα της εργασίας έχει περιθώρια για επιπλέον βελτίωση. Μια μελλοντική, καλύτερη και πιο αυτοματοποιημένη υλοποίηση θα μπορούσε να εφαρμοστεί και σε άλλα ιδρύματα για παρόμοιο σκοπό.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Ben Fry (2008), Visualizing Data, O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Colin Ware (2004),Information Visualization,Morgan Kaufmann publications, 500 Sansome Street, Suite 400, San Francisco, CA 94111

Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze (2008), Introduction to Information Retrieval, Cambridge University Press.

H. P. F. Peters and A. F. J. van Raan (1993), "Co-word-based science maps of chemical engineering. Part I: representations by direct multidimensional scaling," Research Policy, vol. 22, pp. 23–45.

Julie Steele, Noah Iliinsky (2010), Beautiful Visualization,O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472

Katy Börner, Chaomei Chen, Kevin Boyack: Visualizing Knowledge Domains. In Blaise Cronin (Ed.), Annual Review of Information Science & Technology, Volume 37, Medford, NJ: Information Today, Inc./American Society for Information Science and Technology

Van Eck, N.J., Waltman, L., Van den Berg, J., & Kaymak, U. (2006). Visualizing the WCCI 2006 knowledge domain. In Proceedings of the 2006 IEEE International Conference on Fuzzy Systems (pp. 7862-7869). IEEE Press.

Tf-idf weighting, Cambridge University Press , Ανακτήθηκε τον Ιούλιο του 2011, από το http://nlp.stanford.edu/IR-book/html/htmledition/contents-1.html

VOS viewer, Erasmus University Rotterdam, Ανακτήθηκε τον Ιούλιο του 2011, από το http://www.vosviewer.com/

Prefuse, Ανακτήθηκε τον Ιούλιο του 2011, από το http://prefuse.org/doc/manual/introduction/overview/

GraphViz – Graph visualization software, Ανακτήθηκε τον Οκτώβρη του 2011, από το http://www.graphviz.org/About.php

MontyLingua, MIT, Ανακτήθηκε τον Ιούνιο του 2011, από το http://web.media.mit.edu/~hugo/montylingua/ (June 2011)

Rapid Miner, Ανακτήθηκε τον Οκτώβρη του 2011, από το http://rapid-i.com/

Netbeans, wikipedia, Ανακτήθηκε το Νοέμβρη του 2011, από το http://en.wikipedia.org/wiki/NetBeans

ΠΑΡΑΡΤΗΜΑ Α’

Ακολουθεί, ολόκληρο το πρόγραμμα σε Java.

```
package search;

import java.io.*;
import java.util.*;
import java.lang.Number.*;

public class SearchComplex5 {

   static String path = "E:\\Users\\simos\\Personal\\IT\\sxetika me ptyxiakh\\raw data\\ALL\\";
   //   + "vitsas publications\\vitsas txts\\";
   static IntVector count = new IntVector();
   static File occurenciesFile = new File("C:\\temp\\300 occurencies test.txt");
   static File itemsFile = new File("C:\\temp\\300 items test.txt");
   static File dervosThes = new File(" ");
   static File diamanThes = new File(" ");
   static File stamaThes = new File(" ");
   static File vitsasThes = new File(" ");

   public static void main(String[] args) throws IOException {
      String[] terms1 = null;
      String[] terms2 = null;
      String[] terms3 = null;
      String[] terms4 = null;
      String[] terms = null;
      pathChanger(5);
      terms1 = thesReader(terms1, dervosThes.getAbsolutePath());
      terms2 = thesReader(terms2, diamanThes.getAbsolutePath());
      terms3 = thesReader(terms3, stamaThes.getAbsolutePath());
      terms4 = thesReader(terms4, vitsasThes.getAbsolutePath());


      terms = mergeArrays(terms1, terms2, terms3, terms4);

      int[][] C = new int[terms.length][terms.length];
      String files = null;
      File folder = new File(path);
      File[] listOfFiles = folder.listFiles();

      arrayInitializer(C);
      //      showArray(C);

      //fileLister
      fileLister(path, terms, C);
      //   count.showAll();
      //    showArray(C);
```

```
        terms1 = arrayTransformer(terms1, 1);
        terms2 = arrayTransformer(terms2, 2);
        terms3 = arrayTransformer(terms3, 3);
        terms4 = arrayTransformer(terms4, 5);
        terms = mergeArrays(terms1, terms2, terms3, terms4);
        itemWriter(terms);
        vectorWriter(C);

    }//main

    public static void fileLister(String path, String[] terms, int[][] C) throws
IOException {
        String files = null;
        File folder = new File(path);
        File[] listOfFiles = folder.listFiles();
        for (int i = 0; i < listOfFiles.length; i++) {
            if (listOfFiles[i].isFile()) {
                files = listOfFiles[i].getName();
                if (files.endsWith(".txt") || files.endsWith(".TXT")) {
                    System.out.print(i + ":");
                    System.out.println(files);
                    for (int j = 0; j < terms.length; j++) {
                        if (exists(terms[j], files)) {
                            if (!count.exists(j)) {
                                count.add(j);
                            }
                        }
                    }//for j
                    occurenciesFill(C);

                    count.showAll();

                    count.showAll();

                }//if files
            }//if list
        }//for i
    }//fileLister

    public static boolean exists(String term, String file) throws IOException {
        Scanner sc = new Scanner(new BufferedReader(new FileReader(path +
file)));
        boolean flag = false;

        while (sc.hasNext()) {
            String temp = sc.next();
            if (temp.matches("(?i).*" + term + ".*") && term.length() > 4) {
                flag = true;
            } else if (term.length() < 5 && temp.matches("(?i)" + term)) //for shorter
```

```
terms
            {
                flag = true;                    //which have more posibilities to be part of
            } else {
                continue;                       //larger and more complex term, check if it
maches exactly ignoring case.
            }
        }//while
        sc.close();
        return flag;
    }

    public static void showArray(int[][] arr) {

        System.out.println("------ARRAY----------");
        System.out.println();
        for (int i = 0; i < arr.length; i++) {
            for (int j = 0; j < arr.length; j++) {
                if (arr.length - j == 1) {
                    System.out.print(arr[i][j]);
                } else {
                    System.out.print(arr[i][j] + " , ");
                }
            }//for y
            System.out.println();
        }//for j
    }

    public static void showArray(int[] arr) {
        for (int i = 0; i < arr.length; i++) {

            System.out.print(arr[i] + " , ");

            System.out.println();
        }//for j
    }

    public static void showArray(String[] arr) {
        for (int i = 0; i < arr.length; i++) {

            System.out.println(arr[i]);

            System.out.println("--------------");
        }//for j
    }

    public static void arrayInitializer(int[][] arr) {

        for (int i = 0; i < arr.length; i++) {
            for (int j = 0; j < arr.length; j++) {
```

```
                arr[i][j] = 0;

            }//for y
        }//for j
    }

    public static String[] thesReader(String[] arr, String path) throws
FileNotFoundException {

        Scanner s = new Scanner(new BufferedReader(new FileReader(path)));

        int counter = 0;
        //--word counter
        while (s.hasNext()) {
            s.next();
            counter++;
        }
        arr = new String[counter];

        int i = 0;
        try {
            s = new Scanner(new BufferedReader(new FileReader(path)));

            while (s.hasNext() && i < counter) {
                arr[i++] = s.next();
            }
        } finally {
            if (s != null) {
                s.close();
            }
        }
        return arr;
    }

    static public void vectorWriter(int[][] occurences) {

        BufferedWriter bufferedWriter = null;
        try {

            bufferedWriter = new BufferedWriter(new FileWriter(occurenciesFile));

            String line = null;
            //      String[] lineArray = new String[occurences.length];

            for (int i = 0; i < occurences.length; i++) {
                line = null;
                for (int j = 0; j < occurences.length; j++) {
                    if (i == j); else {
                        occurences[i][j] = occurences[j][i];
                    }
```

```
                if (j == 0) {
                    line = "" + occurences[i][j];

                } else {
                    line = line + "," + occurences[i][j];
                }

            }//for j
            //     lineArray[i] = line;
            //      if (i != (occurences.length - 1)) {
            //          line = line + "\r\n";
            //      }
            if (i != (occurences.length - 1)) {
                bufferedWriter.write(line);
                bufferedWriter.newLine();
            } else {
                bufferedWriter.write(line);
            }

        }//for i
        System.out.println(line);


    } catch (FileNotFoundException ex) {
    } catch (IOException ex) {
    } finally {
        //Close the BufferedWriter
        try {
            if (bufferedWriter != null) {
                bufferedWriter.flush();
                bufferedWriter.close();
            }
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}

public static void itemWriter(String[] arr) {
    BufferedWriter bufferedWriter = null;
    try {

        bufferedWriter = new BufferedWriter(new FileWriter(itemsFile));

        String line = null;
        String[] lineArray = new String[arr.length];

        for (int i = 0; i < arr.length; i++) {
            line = arr[i];
            //          lineArray[i] = line;
```

```
            System.out.println(line);
            if (i != (arr.length - 1)) {
                bufferedWriter.write(line);
                bufferedWriter.newLine();
            } else {
                bufferedWriter.write(line);
            }

        }//for
    } catch (FileNotFoundException ex) {
    } catch (IOException ex) {
    } finally {
        //Close the BufferedWriter
        try {
            if (bufferedWriter != null) {
                bufferedWriter.flush();
                bufferedWriter.close();
            }
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }

}

static public void itemWriter2(String[] arr, int val) {

    BufferedWriter bufferedWriter = null;
    try {

        bufferedWriter = new BufferedWriter(new FileWriter(itemsFile));

        String line = null;
        String[] lineArray = new String[arr.length];

        for (int i = 0; i < arr.length; i++) {
            line = arr[i] + ",," + arr[i] + "," + val;
            //          lineArray[i] = line;
            System.out.println(line);
            if (i != (arr.length - 1)) {
                bufferedWriter.write(line);
                bufferedWriter.newLine();
            } else {
                bufferedWriter.write(line);
            }

        }//for
    } catch (FileNotFoundException ex) {
    } catch (IOException ex) {
    } finally {
```

```
            //Close the BufferedWriter
            try {
               if (bufferedWriter != null) {
                  bufferedWriter.flush();
                  bufferedWriter.close();
               }
            } catch (IOException ex) {
               ex.printStackTrace();
            }
         }
      }//itemWriter2

      static public void setContents(File aFile, String aContents)
            throws FileNotFoundException, IOException {
         if (aFile == null) {
            throw new IllegalArgumentException("File should not be null.");
         }
         if (!aFile.exists()) {
            throw new FileNotFoundException("File does not exist: " + aFile);
         }
         if (!aFile.isFile()) {
            throw new IllegalArgumentException("Should not be a directory: " + aFile);
         }
         if (!aFile.canWrite()) {
            throw new IllegalArgumentException("File cannot be written: " + aFile);
         }

         //use buffering
         Writer output = new BufferedWriter(new FileWriter(aFile));
         try {
            //FileWriter always assumes default encoding is OK!
            output.write(aContents);
         } finally {
            output.close();
         }
      }

      public static String[] mergeArrays(String[] mainArray, String[] addArray1, String[]
addArray2, String[] addArray3) {
         String[] finalArray = new String[mainArray.length + addArray1.length +
addArray2.length + addArray3.length];
         System.arraycopy(mainArray, 0, finalArray, 0, mainArray.length);
         System.arraycopy(addArray1, 0, finalArray, mainArray.length,
addArray1.length);
         System.arraycopy(addArray2, 0, finalArray, mainArray.length +
addArray1.length, addArray2.length);
         System.arraycopy(addArray3, 0, finalArray, mainArray.length +
addArray1.length + addArray2.length, addArray3.length);

         return finalArray;
```

```
    }

    public static String[] arrayTransformer(String[] arr, int val) {

        for (int i = 0; i < arr.length; i++) {
            arr[i] = arr[i] + ",," + arr[i] + "," + val;

        }//for

        return arr;
    }

    public static void pathChanger(int i) {

        switch (i) {

            case 1:
                //THESAURUS SIZE 50
                occurenciesFile = new File("C:\\temp\\occurences thes50 scen.txt");
                itemsFile = new File("C:\\temp\\items thes50 scen.txt");

                dervosThes = new File("E:\\Users\\simos\\Personal\\IT\\sxetika me
ptyxiakh\\thesaurus\\"
                        + "Variated size thesaurus\\50 thesaurus\\derv 7-50 thes.txt");
                diamanThes = new File("E:\\Users\\simos\\Personal\\IT\\sxetika me
ptyxiakh\\thesaurus\\"
                        + "Variated size thesaurus\\50 thesaurus\\diaman 23-50 thes.txt");
                stamaThes = new File("E:\\Users\\simos\\Personal\\IT\\sxetika me
ptyxiakh\\thesaurus\\"
                        + "Variated size thesaurus\\50 thesaurus\\stam 6-50 thes.txt");
                vitsasThes = new File("E:\\Users\\simos\\Personal\\IT\\sxetika me
ptyxiakh\\thesaurus\\"
                        + "Variated size thesaurus\\50 thesaurus\\vitsas 14-50 thes.txt");
                break;

            case 2:
                //THESAURUS SIZE 100
                occurenciesFile = new File("C:\\temp\\occurences thes100 scen.txt");
                itemsFile = new File("C:\\temp\\items thes100 scen.txt");

                dervosThes = new File("E:\\Users\\simos\\Personal\\IT\\sxetika me
ptyxiakh\\thesaurus\\"
                        + "Variated size thesaurus\\100 thesaurus\\derv 13-100 thes.txt");
                diamanThes = new File("E:\\Users\\simos\\Personal\\IT\\sxetika me
ptyxiakh\\thesaurus\\"
                        + "Variated size thesaurus\\100 thesaurus\\diam 47-100 thes.txt");
                stamaThes = new File("E:\\Users\\simos\\Personal\\IT\\sxetika me
ptyxiakh\\thesaurus\\"
                        + "Variated size thesaurus\\100 thesaurus\\stam 12-100 thes.txt");
                vitsasThes = new File("E:\\Users\\simos\\Personal\\IT\\sxetika me
```

```
ptyxiakh\\thesaurus\\"
                + "Variated size thesaurus\\100 thesaurus\\vitsas 28-100 thes.txt");
            break;
        case 3:
            //THESAURUS SIZE 150
            occurenciesFile = new File("C:\\temp\\occurences thes150 scen.txt");
            itemsFile = new File("C:\\temp\\items thes150 scen.txt");

            dervosThes = new File("E:\\Users\\simos\\Personal\\IT\\sxetika me
ptyxiakh\\thesaurus\\"
                + "Variated size thesaurus\\150 thesaurus\\derv 20-150 thes.txt");
            diamanThes = new File("E:\\Users\\simos\\Personal\\IT\\sxetika me
ptyxiakh\\thesaurus\\"
                + "Variated size thesaurus\\150 thesaurus\\diaman 70-150
thes.txt");
            stamaThes = new File("E:\\Users\\simos\\Personal\\IT\\sxetika me
ptyxiakh\\thesaurus\\"
                + "Variated size thesaurus\\150 thesaurus\\stam 18-150 thes.txt");
            vitsasThes = new File("E:\\Users\\simos\\Personal\\IT\\sxetika me
ptyxiakh\\thesaurus\\"
                + "Variated size thesaurus\\150 thesaurus\\vitsas 42-150 thes.txt");
            break;
        case 4:
            //THESAURUS SIZE 200
            occurenciesFile = new File("C:\\temp\\occurences thes200 scen.txt");
            itemsFile = new File("C:\\temp\\items thes200 scen.txt");

            dervosThes = new File("E:\\Users\\simos\\Personal\\IT\\sxetika me
ptyxiakh\\thesaurus\\"
                + "Variated size thesaurus\\200 thesaurus\\derv 26-200 thes.txt");
            diamanThes = new File("E:\\Users\\simos\\Personal\\IT\\sxetika me
ptyxiakh\\thesaurus\\"
                + "Variated size thesaurus\\200 thesaurus\\diaman 94-200
thes.txt");
            stamaThes = new File("E:\\Users\\simos\\Personal\\IT\\sxetika me
ptyxiakh\\thesaurus\\"
                + "Variated size thesaurus\\200 thesaurus\\stam 24-200 thes.txt");
            vitsasThes = new File("E:\\Users\\simos\\Personal\\IT\\sxetika me
ptyxiakh\\thesaurus\\"
                + "Variated size thesaurus\\200 thesaurus\\vitsas 56-200 thes.txt");
            break;
        case 5:
            //THESAURUS SIZE 300
            occurenciesFile = new File("C:\\temp\\occurences thes300 scen.txt");
            itemsFile = new File("C:\\temp\\items thes300 scen.txt");

            dervosThes = new File("E:\\Users\\simos\\Personal\\IT\\sxetika me
ptyxiakh\\thesaurus\\"
                + "Variated size thesaurus\\300 thesaurus\\derv 39-300 thes.txt");
            diamanThes = new File("E:\\Users\\simos\\Personal\\IT\\sxetika me
```

```
ptyxiakh\\thesaurus\\"
                + "Variated size thesaurus\\300 thesaurus\\diaman 141-300
thes.txt");
            stamaThes = new File("E:\\Users\\simos\\Personal\\IT\\sxetika me
ptyxiakh\\thesaurus\\"
                + "Variated size thesaurus\\300 thesaurus\\stam 36-300 thes.txt");
            vitsasThes = new File("E:\\Users\\simos\\Personal\\IT\\sxetika me
ptyxiakh\\thesaurus\\"
                + "Variated size thesaurus\\300 thesaurus\\vitsas 84-300 thes.txt");
            break;
        case 6:
            //ALL WORDS - mpa oxi, 1000 lekseis isws einai ok gia mia ypervolikh
apeikonhsh....
            occurenciesFile = new File("C:\\temp\\occurences thes1000 scen.txt");
            itemsFile = new File("C:\\temp\\items thes1000 scen.txt");

            dervosThes = new File("E:\\Users\\simos\\Personal\\IT\\sxetika me
ptyxiakh\\thesaurus\\"
                + "Variated size thesaurus\\1000 thesaurus\\derv 130-1000
thes.txt");
            diamanThes = new File("E:\\Users\\simos\\Personal\\IT\\sxetika me
ptyxiakh\\thesaurus\\"
                + "Variated size thesaurus\\1000 thesaurus\\diam 470-1000
thes.txt");
            stamaThes = new File("E:\\Users\\simos\\Personal\\IT\\sxetika me
ptyxiakh\\thesaurus\\"
                + "Variated size thesaurus\\1000 thesaurus\\stam 120-1000
thes.txt");
            vitsasThes = new File("E:\\Users\\simos\\Personal\\IT\\sxetika me
ptyxiakh\\thesaurus\\"
                + "Variated size thesaurus\\1000 thesaurus\\vitsas 280-1000
thes.txt");
            break;
        case 7:
            //Manually corrected thesaurus 100
            occurenciesFile = new File("C:\\temp\\occurences man thes100
scen.txt");
            itemsFile = new File("C:\\temp\\items man thes100 scen.txt");

            dervosThes = new File("E:\\Users\\simos\\Personal\\IT\\sxetika me
ptyxiakh\\thesaurus\\"
                + "Manually corrected thesaurus\\manual 100 thes\\derv manual
100 thes.txt");
            diamanThes = new File("E:\\Users\\simos\\Personal\\IT\\sxetika me
ptyxiakh\\thesaurus\\"
                + "Manually corrected thesaurus\\manual 100 thes\\diam manual
100 thes.txt");
            stamaThes = new File("E:\\Users\\simos\\Personal\\IT\\sxetika me
ptyxiakh\\thesaurus\\"
                + "Manually corrected thesaurus\\manual 100 thes\\stam manual
```

```
100 thes.txt");
            vitsasThes = new File("E:\\Users\\simos\\Personal\\IT\\sxetika me
ptyxiakh\\thesaurus\\"
                    + "Manually corrected thesaurus\\manual 100 thes\\vitsas manual
100 thes.txt");
            break;
         case 8:
            //Manually corrected thesaurus 150
            occurenciesFile = new File("C:\\temp\\occurences man thes150
scen.txt");
            itemsFile = new File("C:\\temp\\items man thes150 scen.txt");

            dervosThes = new File("E:\\Users\\simos\\Personal\\IT\\sxetika me
ptyxiakh\\thesaurus\\"
                    + "Manually corrected thesaurus\\manual 150 thes\\derv manual
150 thes.txt");
            diamanThes = new File("E:\\Users\\simos\\Personal\\IT\\sxetika me
ptyxiakh\\thesaurus\\"
                    + "Manually corrected thesaurus\\manual 150 thes\\diam manual
150 thes.txt");
            stamaThes = new File("E:\\Users\\simos\\Personal\\IT\\sxetika me
ptyxiakh\\thesaurus\\"
                    + "Manually corrected thesaurus\\manual 150 thes\\stam manual
150 thes.txt");
            vitsasThes = new File("E:\\Users\\simos\\Personal\\IT\\sxetika me
ptyxiakh\\thesaurus\\"
                    + "Manually corrected thesaurus\\manual 150 thes\\vitsas manual
150 thes.txt");
            break;
         default:
            return;
      }
   }

   public static void occurenciesFill(int[][] c) {

      count.showAll();
//------At LAST!! we fill in the diagonal of the array with
//       the number of documents in which each term occured.
      for (int v = 0; v < count.size(); v++) {
         System.out.println(v);
         System.out.println("!----!");
         System.out.println(count.get(v));
         System.out.println("______");
         c[count.get(v)][count.get(v)]++;
         //      System.out.println(c[g][g]);
         for (int y = 0; y < count.size(); y++) {
            if (!(y == v)) {
               c[count.get(v)][count.get(y)]++;
            }
```

```
            }//for y

        }//for v

        count = new IntVector();

    }//occurenciesFill
}//class
```

	Δέρβος	Διαμαντάρας	Σταμάτης	Βίτσας
Λίστα με 50 όρους	index article signature inverted author information file	temporal pca filter linear noise solution optimal system order method motion mixture channel binary principal instantaneous problem neural output source algorithm bss domain	nl book ects quality generic emotional	analytical distribution didd drop account link model transmission simulation average basic chain accurate mathematical

	Δέρβος	Διαμαντάρας	Σταμάτης	Βίτσας
Λίστα με 100 όρους	index article signature inverted author information file citation textbase query bitmap pca corresponding	temporal pca filter linear noise solution optimal system order method motion mixture channel binary principal instantaneous problem neural output source algorithm bss domain single component frequency nonlinear arbitrary separation time signal model indeterminacy subspace rule estimation standard length matrix simple case statistics analysis blind cone constellation opca	nl book ects quality generic emotional learning encapsulation agent robot training teaching	analytical distribution didd drop account link model transmission simulation average basic chain accurate mathematical literature error size simple markov collision analysis packet double limit time delay communication scheme

	Δέρβος	Διαμαντάρας		Σταμάτης	Βίτσας
Λίστα 150 όρων	index article signature inverted author information file citation textbase query bitmap pca corresponding study approach data literature method graph operational	temporal pca filter linear noise solution optimal system order method motion mixture channel binary principal instantaneous problem neural output source algorithm bss domain single component frequency nonlinear arbitrary separation time signal model indeterminacy subspace rule	estimation standard length matrix simple case statistics analysis blind cone constellation opca iterative colored approach video robust feature scheme design mimo complex independent efficient successor region multiple image set input invariant cluster gaussian recursive higher	nl book ects quality generic emotional learning encapsulation agent robot training teaching study apim course lp technique programming	analytical distribution didd drop account link model transmission simulation average basic chain accurate mathematical literature error size simple markov collision analysis packet double limit time delay communication scheme throughput retry errors performance arq duration backoff network independent variable coordination rr stage opnettm

	Δέρβος	Διαμαντάρας		Σταμάτης	Βίτσας	
Λίστα 200 όρων	index article signature inverted author information file citation textbase query bitmap pca corresponding study approach data literature method graph operational encoding ranking non value model research	temporal pca filter linear noise solution optimal system order method motion mixture channel binary principal instantaneous problem neural output source algorithm bss domain single component frequency nonlinear arbitrary separation time signal model indeterminacy subspace rule estimation standard length matrix simple case statistics analysis blind cone constellation opca	iterative colored approach video robust feature scheme design mimo complex independent efficient successor region multiple image set input invariant cluster gaussian recursive higher observation filtering classification identification performance data show apex unknown stochastic power cross artificial general finite optimization value paper signatures basis basic convolutive analytical point	nl book ects quality generic emotional learning encapsulation agent robot training teaching study apim course lp technique programming framework communication maze vinci welfare parallel	analytical distribution didd drop account link model transmission simulation average basic chain accurate mathematical literature error size simple markov collision analysis packet double limit time delay communication scheme	throughput retry performance arq duration backoff network independent variable coordination rr stage opnettm local protocol burst infrared number area contention ieee capacity irlap beb package particular successfully comparison

	Δέρβος	Διαμαντάρας				Σταμάτης	Βίτσας	
Λίστα 300 όρων	index article signature inverted author information file citation textbase query bitmap pca corresponding study approach data literature method graph operational encoding ranking non value model research impact binary credibility measure indexing relational current nnp datum performance based number internet	temporal pca filter linear noise solution optimal system order method motion mixture channel binary principal instantaneous problem neural output source algorithm bss domain single component frequency nonlinear arbitrary separation time signal model indeterminacy subspace rule estimation	standard length matrix simple case statistics analysis blind cone constellation opca iterative colored approach video robust feature scheme design mimo complex independent efficient successor region multiple image set input invariant cluster gaussian recursive higher observation	filtering classification identification performance data show apex unknown stochastic power cross artificial general finite optimization value paper signature basis basic convolutive analytical models point svd capable network process vector application datum based dynamic frame segment	geometric normal mathematical st spatial rate pair computationally corresponding information noiseless ratio distribution multi particular processing non involve jjr result nxn core zero cost tap testing suitable training covariance useful pattern singular estimate assumption original	nl book ects quality generic emotional learning encapsulation agent robot training teaching study apim course lp technique programming framework communication maze vinci welfare parallel abstract community pilot project leonardo vocational educational parallelism odl environment networked system	analytical distribution didd drop account link model transmission simulation average basic chain accurate mathematical literature error size simple markov collision analysis packet double limit time delay communication scheme throughput retry performance arq duration backoff network independent variable coordination rr stage opnettm local	protocol burst infrared number area contention ieee capacity irlap beb package particular successfully comparison effect window air specific layer wlan dcf value access rts mbit frame binary effective probability function rate retransmission resetting worldwide channel physical maximum datum standard improve admission send

	Δέρβος	Διαμαντάρας					Σταμάτης	Βίτσας		
Λίστα 1000 όρων >>	index	temporal	svd	edge	conventional	vast	nl	analytical	medium	considerably
	article	pca	capable	framework	trivial	execution	book	distribution	initial	saturation
	signature	filter	network	improvement	confidence	opc	ects	didd	environment	handshake
	inverted	linear	process	random	hebbian	digital	quality	drop	efficiency	verified
	author	noise	vector	total	quantizer	hold	generic	account	develop	become
	information	solution	application	combination	upca	morphological	emotional	link	data	derive
	file	optimal	datum	delayed	key	rd	learning	model	linear	taking
	citation	system	based	resolution	klt	reduction	encapsulation	transmission	adjustment	validate
	textbase	order	dynamic	asymmetric	morphology	uniform	agent	simulation	elementary	introduce
	query	method	frame	components	shot	fact	robot	average	ideal	prp
	bitmap	motion	segment	machine	specificity	boundary	training	basic	suitable	paper
	pca	mixture	geometric	centers	context	reduced	teaching	chain	addition	uncertain
	corresponding	channel	normal	uniquely	diversity	similar	study	accurate	length	great
	study	binary	mathematical	provide	nlpca	smaller	apim	mathematical	control	proper
	approach	principal	st	recognition	edges	constant	course	literature	order	gbn
	data	instantaneous	spatial	deterministic	computational	constraints	lp	error	station	require
	literature	problem	rate	extraction	measurement	detection	technique	size	study	growth
	method	neural	pair	patient	perform	implementation	programming	simple	use	market
	graph	output	corresponding	reconstruction	scene	occur	framework	markov	simulative	metrics
	operational	source	information	equation	shorten	statistical	communication	collision	free	accurately
	encoding	algorithm	noiseless	equal	entity	translation	maze	analysis	popular	request
	ranking	bss	ratio	further	knowledge	world	vinci	packet	various	deployed
	non	domain	distribution	theoretical	sensitivity	care	welfare	double	exponential	deployment
	value	single	multi	mixing	validity	correspondence	parallel	limit	explore	problem
	model	component	particular	demonstrate	index	object	abstract	time	rt	corresponding
	research	frequency	processing	number	memory	predecessor	community	delay	proposed	highly
	impact	nonlinear	non	observed	operation	rigid	pilot	communication	easy	improvement
	binary	arbitrary	involve	wdt	dependent	compression	project	scheme	previous	establish
	credibility	separation	jjr	nnp	eigenvalue	describe	leonardo	throughput	modern	extensive
	measure	time	result	soc	perspective	determined	vocational	retry	conditional	reach
	indexing	signal	nxn	camera	conceptual	distance	educational	performance	method	effectiveness
	relational	model	core	explicit	earlier	hidden	parallelism	arq	prone	ct
	current	indeterminacy	zero	abnormal	solve	smallest	odl	duration	lower	utilized
	nnp	subspace	cost	classical	systolic	synaptic	environment	backoff	ber	jjr
	datum	rule	tap	multichannel	decomposition	weight	networked	network	take	application
	performance	estimation	testing	dtc	good	calculate	system	independent	transmitted	employ
	based	standard	suitable	accurate	matrices	conditions	qa	variable	compute	give
	number	length	training	intensity	computation	generalized	case	coordination	parameter	developed
	internet	matrix	covariance	criterion	efficiency	markov	decision	rr	wlans	mac
	management	simple	useful	mtc	characteristic	measure	language	stage	result	appropriate
	web	case	pattern	partial	environment	speech	fcm	opnettm	wireless	cas
	sf	statistics	singular	programming	identify	speed	organisational	local	srej	computation
	mining	analysis	estimate	spurious	pos	viterbi	knowledge	protocol	equal	destination
	text	blind	assumption	unsymmetric	structure	blindly	da	burst	calculation	internal

Λίστα 1000 όρων >>	system	cone	original	view	compared	interconnection	number	infrared	ir	realtime
	earlier	constellation	prewhitening	mapping	equivalent	numerical	action	number	optimum	unfair
	fundamental	opca	correct	pdf	treat	presence	food	area	established	receiver
	factor	iterative	step	cdma	previously	torus	distance	contention	increment	fairness
	database	colored	clustering	effective	property	compare	logic	ieee	decrement	higher
	technique	approach	simulation	typical	oriented	name	institution	capacity	widely	experiencing
	jjr	sources	work	additive	outputs	created	template	irlap	calculate	predict
	uai	video	technique	approximation	successful	preceded	animal	beb	cts	scenario
	calculation	robust	systems	distances	achieve	properties	artificial	package	presented	specified
	library	feature	novel	inspection	given	resolve	model	particular	virtual	degrade
	analysis	scheme	using	layer	mix	support	network	successfully	transmitter	expression
	scheme	design	oja	overdetermined	tested	yield	student	comparison	bianchi	bit
	codes	mimo	users	previous	introduce	leads	domain	effect	respect	quality
	knowledge	complex	chip	shortening	require	addition	apply	window	comprehensive	avoidance
	clear	independent	dp	local	techniques	communications	expert	air	term	indoor
	category	efficient	occlusion	specific	thus	datasets	virtual	specific	low	cope
	user	successor	quantization	various	jjs	simulations	user	layer	implement	show
	environment	region	submatrices	common	author	address	specific	wlan	evaluate	years
	output	multiple	visual	rank	wiener	compute	effective	dcf	transmit	becomes
	name	image	level	theory	transformation	determine	low	value	indicate	extremely
	extreme	set	usual	overall	separability	known	current	access	using	experience
	life	input	test	svm	autocorrelation	derived	prolog	rts	capable	consider
	tree	invariant	deflation	coupled	circuit	extracted	execution	mbit	end	studied
	unified	cluster	indices	size	count	learning	data	frame	turn	hidden
	valid	gaussian	deconvolution	phase	greek	combined	design	binary	commercial	utilize
	handling	recursive	idea	artificially	illumination	separating	experimental	effective	optical	association
	sense	higher	correlation	patterns	linguistic	demonstrated	independent	probability	legacy	wdt
	respect	observation	noisy	stationary	identifiable	extend	lab	function	novel	encounter
	sc	filtering	transform	condition	imagery	combine	appropriate	rate	opnet	multimedia
	sys	classification	unit	convex	luminance	required	intelligence	retransmission	tremendous	potential
	block	identification	function	discussed	pixel	md	assurance	resetting	enhance	useful
	higher	performance	separate	spectra	rotation	vbg	approach	worldwide	extend	arrival
	identifier	data	sign	consider	segmented	vbp	various	channel	given	inter
	good	show	spectrum	dataset	separable	intractable	fuzzy	physical	introduced	previously
	candidate	apex	example	representation	transition	odd	synthetic	maximum	identify	purpose
	false	unknown	parameter	depend	automatic	practical	architecture	datum	validated	selection
	limited	stochastic	procedure	due	column	actual	computer	standard	implemented	threshold
	paradigm	power	extension	observe	density	additional	context	improve	base	form
	common	cross	lead	obtain	differences	digitalization	process	admission	rbs	infinite
	hybrid	artificial	propose	fast	fir	distant	e-learning	send	provide	reliable
	term	general	proposed	shown	greater	global	description	beneficial	distribute	importance
	cdc	finite	use	authors	lab	impulse	farm	work	distributed	technology
	end	optimization	cd	presented	laboratory	kalman	intelligent	irda	snr	jjs
	future	value	outliers	results	likely	memoryless	ai	based	necessary	essential
	place	paper	accuracy	vbd	magged	mismatch	cost	propose	successful	meaningful
	queries	basis	prediction	base	pam	modeling	multiprocessor	algorithm	optimization	minimum
	overhead	basic	ccu	available	probability	old	task	cw	timer	carry

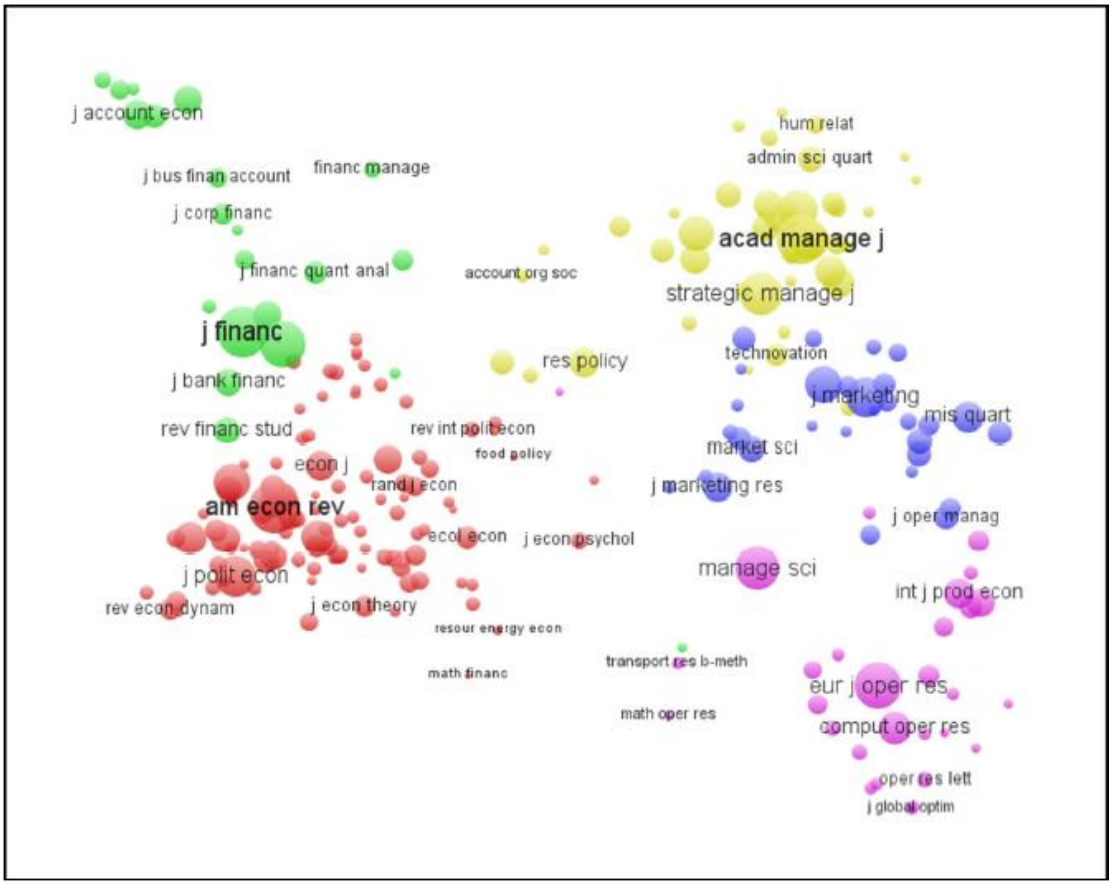
Λίστα 1000 όρων <<.	analytical storage form coding traditional comparison target similar communication collection usefulness larger latter processing use appropriate consideration convective expert hail hellenic implementation procedure program suppression full optimal practice terms loss promising compressed popular table efficiency shown variation framework drop	convolutive analytical point	npca laboratories array	lateral multiresolution topology	product satisfactory snr	optical ptcc response	set information preliminary state srv multiple pedagogical view university support tool presented overview rpm face meta useful education evaluation main provide based science critical neural types flexible level interaction	realistic mechanism overhead	slot curve	named deploy

	Δέρβος	Διαμαντάρας		Σταμάτης	Βίτσας
Λίστα με 100 διαλεγμένες λέξεις	index article signature information file textbase query bitmap pca data method graph encoding	pca filter linear noise system method motion channel binary principal problem neural output source algorithm bss domain component frequency nonlinear separation time signal model	indeterminacy subspace matrix statistics analysis blind cone constellation opca iterative video robust mimo image set input cluster gaussian recursive filtering classification identification performance	nl quality generic emotional e-learning encapsulation agent robot training teaching apim lp	analytical distribution didd drop account link model transmission simulation chain mathematical error markov collision analysis packet limit time delay communication throughput retry performance arq backoff network variable coordination

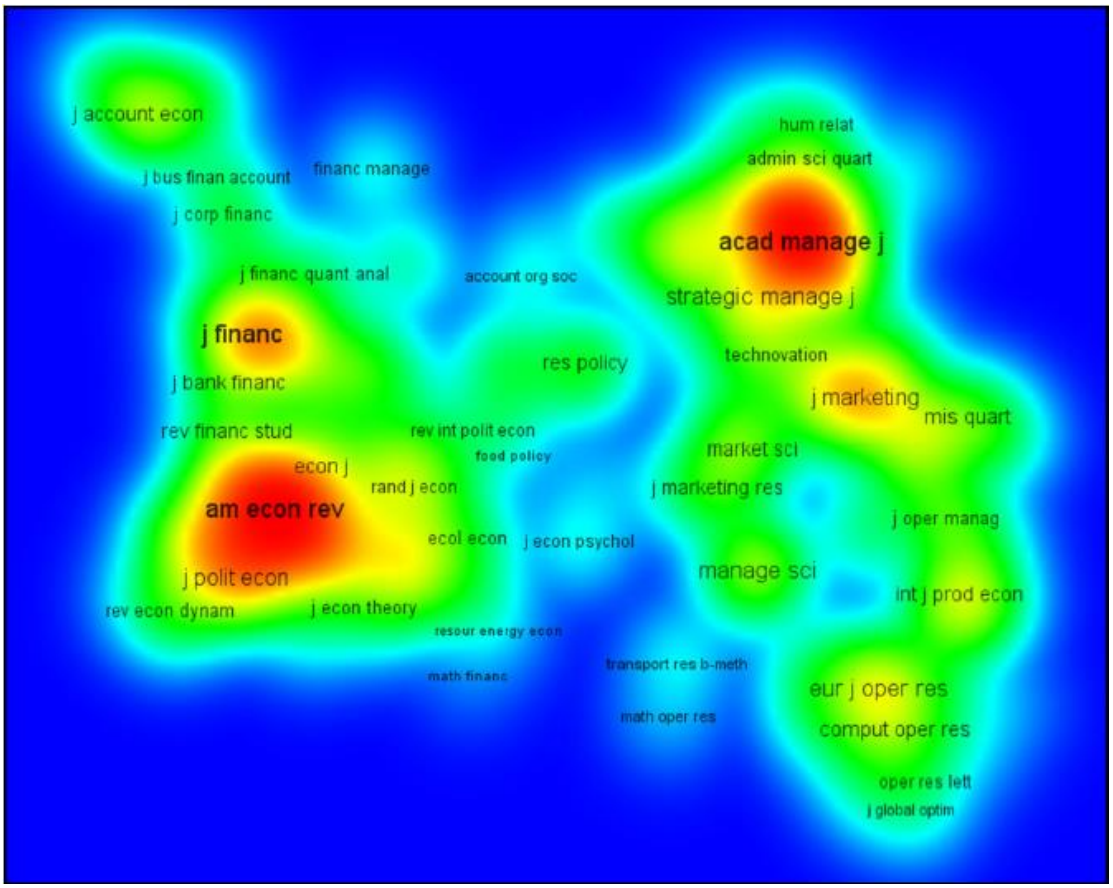
	Δέρβος	Διαμαντάρας		Σταμάτης	Βίτσας
Λίστα με 150 διαλεγμένες λέξεις	index article signature inverted author information file citation textbase query bitmap pca data literature method graph operational encoding ranking value	temporal pca filter linear noise solution optimal system order method motion mixture channel binary principal neural output source algorithm bss domain component frequency nonlinear separation time signal model indeterminacy subspace rule estimation standard matrix statistics	analysis blind cone constellation opca iterative approach video robust design mimo successor region image set input invariant cluster gaussian recursive filtering classification identification performance data apex stochastic power cross artificial general finite optimization value basis	150 διαλεγμένους	index article signature inverted author information file citation textbase query bitmap pca data literature method graph operational encoding ranking value

ΠΑΡΑΡΤΗΜΑ Γ’

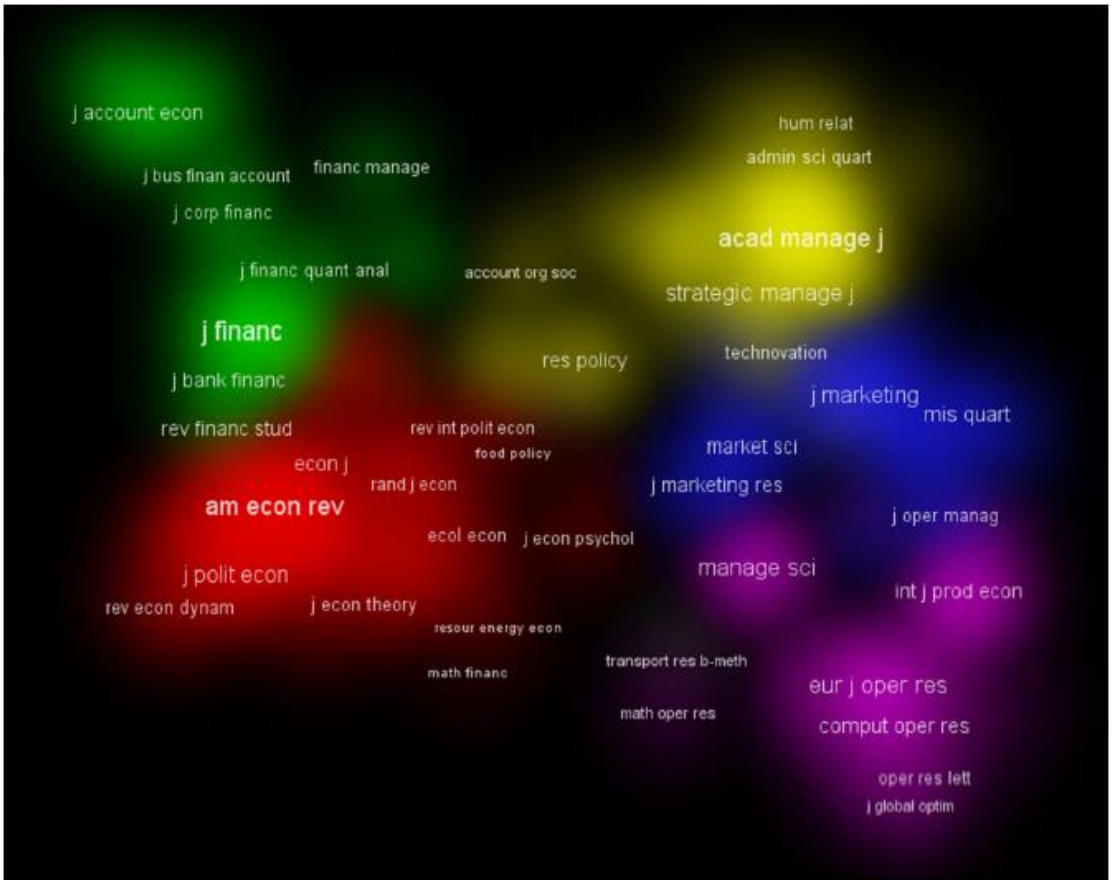
Ακολουθούν τα παραδείγματα του εγχειριδίου του VOS viewer.



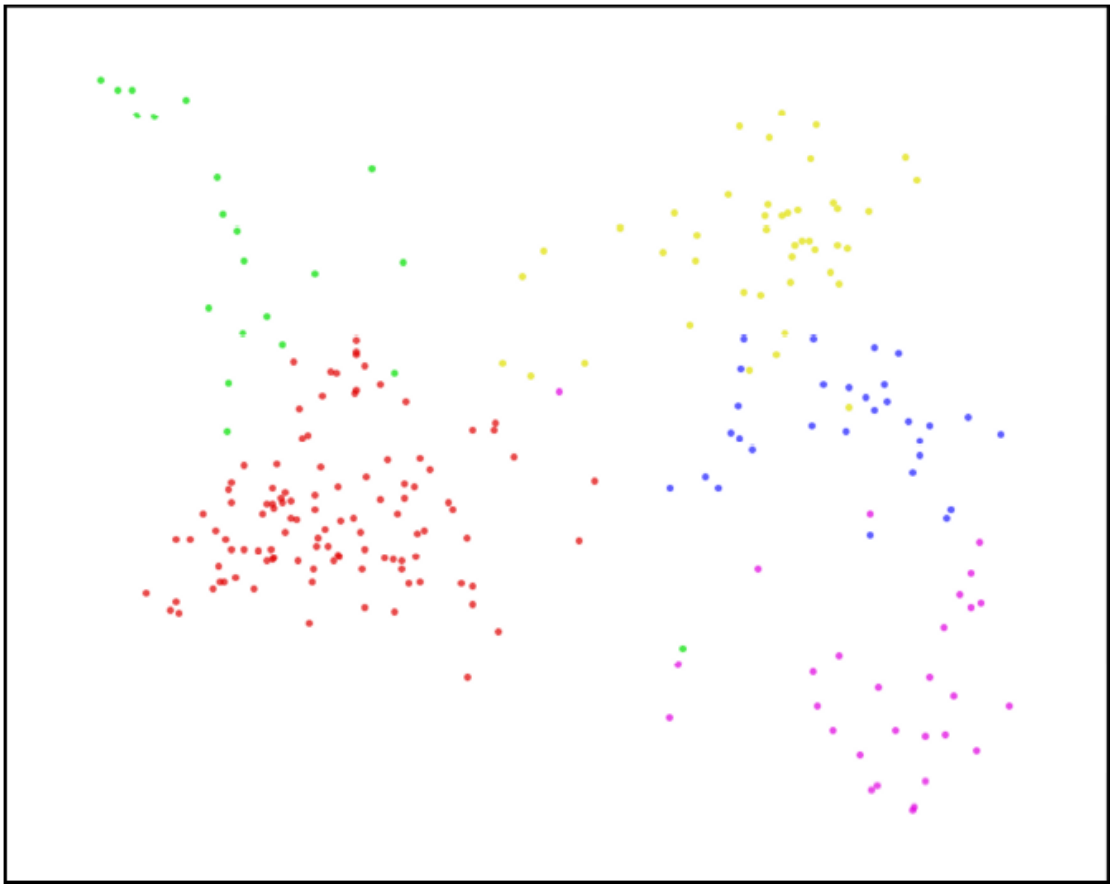
Εικόνα 64 Label view



Εικόνα 65 density view



Εικόνα 66 cluster density view



Εικόνα 67 scatter view