ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ
ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Υλοποίηση Ιεραρχικής Ομαδοποίησης χρησιμοποιώντας Μαρκοβιανές Τεχνικές

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Δημήτριος Χ. Σκούτας

Επιβλέπων: Θ. Βαρβαρίγου
Αν. Καθηγήτρια Ε.Μ.Π.

Αθήνα, Σεπτέμβριος 2003

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Υλοποίηση Ιεραρχικής Ομαδοποίησης χρησιμοποιώντας Μαρκοβιανές Τεχνικές

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Δημήτριος Χ. Σκούτας

Επιβλέπων: Θ. Βαρβαρίγου
Αν. Καθηγήτρια Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 24η Σεπτεμβρίου 2003

---	---	---
Θ. Βαρβαρίγου	Ε. Πρωτονοτάριος	Ε. Καγιάφας
Αν. Καθηγήτρια Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Σεπτέμβριος 2003

Δημήτριος Χ. Σκούτας

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

Περίληψη

Η τεράστια αύξηση που σημειώνεται καθημερινά στον όγκο της πληροφορίας που υπάρχει διαθέσιμη σε ηλεκτρονική μορφή στο Διαδίκτυο, σε ψηφιακές βιβλιοθήκες, σε βάσεις δεδομένων κλπ. σε όλο τον κόσμο καθιστά αναγκαία την ανάπτυξη και χρησιμοποίηση αυτοματοποιημένων και όσο το δυνατόν πιο «έξυπνων» τεχνικών και αλγορίθμων για την αναζήτηση και γενικά τη διαχείρηση και εκμετάλευση αυτής της πληροφορίας.

Η εργασία είναι χωρισμένη σε 5 ενότητες. Στο 1^{ο} μέρος γίνεται μία εισαγωγή στο χώρο της εξόρυξης δεδομένων. Βλέπουμε τι σημαίνει ο όρος εξόρυξη δεδομένων, που βρίσκει εφαρμογές, με ποια άλλα επιστημονικά πεδία συνδέεται και παρουσιάζουμε συνοπτικά κάποιες βασικές έννοιες και τεχνικές που συναντά κανείς στο χώρο αυτό.

Το 2^{ο} μέρος αναφέρεται στο πρόβλημα της ομαδοποίησης και ταξινόμησης αντικειμένων. Συγκεκριμένα μελετάμε τους τύπους δεδομένων που μπορεί να συναντήσει κανείς, μερικές από τις πιο βασικές και συνηθισμένες μετρικές που χρησιμοποιούνται για τον προσδιορισμό της απόστασης μεταξύ αντικειμένων και τέλος τις κυριότερες κατηγορίες αλγορίθμων ομαδοποίησης.

Το 3^{ο} μέρος αποτελεί μία εισαγωγή στις Μαρκοβιανές αλυσίδες. Περιγράφουμε τα βασικά χαρακτηριστικά και τις ιδιότητες των Μαρκοβιανών αλυσίδων, ώστε να δούμε στη συνέχεια με ποιο τρόπο θα μπορούσαν να χρησιμοποιηθούν στο πρόβλημα της ομαδοποίησης και ταξινόμησης αντικειμένων.

Στο 4^{ο} μέρος παρουσιάζεται το πρόβλημα της κατηγοριοποίησης κειμένων. Γίνεται μία σύντομη αναφορά σε τεχνικές που έχουν χρησιμοποιηθεί στο πρόβλημα αυτό και κατόπιν μελετάμε τον τρόπο αναπαράστασης και προσδιορισμού της απόστασης μεταξύ κειμένων, χρησιμοποιώντας και ιδιότητες των Μαρκοβιανών αλυσίδων. Τέλος παρουσιάζουμε μία εφαρμογή που υλοποιεί αυτές τις τεχνικές, καθώς και τα αποτελέσματα που προέκυψαν από την εφαρμογή της σε ένα δοκιμαστικό σύνολο κειμένων.

Λέξεις-κλειδιά

Εξόρυξη δεδομένων, ομαδοποίηση, ταξινόμηση, μετρικές, Μαρκοβιανές αλυσίδες, κατηγοριοποίηση κειμένων

Abstract

The enormous increase in the volume of information that is available in electronic form in the Internet, in digital libraries, in databases and so on all over the world renders necessary the development and usage of automated and as "intelligent" as possible techniques and algorithms for the search and generally the management of this information.

This thesis is separated in 5 chapters. The 1st part is an introduction in the field of data mining. We see what the term data mining means, its applications, with which scientific fields it is related and we concisely present some basic concepts and techniques that are used in this field.

The 2nd part refers to the problem of clustering and classification of objects. Particularly we study the data types that someone may have to deal with, some of the most basic and common metrics that are used for the determination of distance between objects and finally the main categories of clustering algorithms.

The 3rd part serves as an introduction to Markov chains. We describe the basic characteristics and properties of Markov chains, in order to examine in what way they could be used in the problem of clustering and classification of objects.

The 4th part presents the problem of text categorization. We start with a short report in techniques that have been used in this problem and then we study the way of representation and determination of distance between texts, using also some properties of Markov chains. Finally we present an application that implements these techniques, as well as the results from applying it on a set of texts.

Keywords

Data mining, clustering, classification, metrics, Markov chains, text categorization

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον κ. Νίκο Παπαδάκη και τον κ. Κώστα Ραφτόπουλο τόσο για το χρόνο που μου αφιέρωσαν και την πολύτιμη βοήθεια που μου προσέφεραν με τις γνώσεις και τις υποδείξεις τους όσο και για την ιδιαίτερα φιλική τους συμπεριφορά και για τη δημιουργία ενός ευχάριστου και εποικοδομητικού κλίματος για έρευνα και συνεργασία.

Περιεχόμενα

Μέρος 1ο: Εξόρυξη Δεδομένων

1.1 Τι ονομάζουμε εξόρυξη δεδομένων

Με τη ραγδαία ανάπτυξη των υπολογιστών τόσο σε επίπεδο υλικού όσο και λογισμικού, που έχει σημειωθεί τα τελευταία χρόνια, έχει καταστεί δυνατή η συλλογή και αποθήκευση τεράστιου όγκου πληροφοριών σε πολλούς τομείς. Οι –μεγάλες κυρίως– επιχειρήσεις συγκεντρώνουν και αποθηκεύουν τεράστιες ποσότητες δεδομένων που αφορούν τις διάφορες δραστηριότητές τους. Και αυτό δε συμβαίνει μόνο στον κόσμο των επιχειρήσεων. Πολύ μεγάλο όγκο δεδομένων μπορεί επίσης να διαθέτουν τα νοσοκομεία (ιατρικό παρελθόν ασθενών και άλλα ιατρικά δεδομένα), οι δικαστικές αρχές (ποινικά μητρώα, δικογραφίες), βιβλιοθήκες, υπουργεία κλπ. Ωστόσο τα δεδομένα αυτά, στη μορφή που είναι, απλώς δηλώνουν διάφορα γεγονότα. Μάλιστα όσο ευκολότερη γίνεται η συλλογή και η αποθήκευση των δεδομένων τόσο αυξάνει το πλήθος τους και κατά συνέπεια τόσο δυσκολότερη γίνεται η κατανόηση και ανάλυσή τους. Είναι εύλογο ότι αν κανείς διαθέτει μια τόσο μεγάλη συλλογή δεδομένων θα επιθυμεί να τη χρησιμοποιήσει για να εξάγει, αν και όποτε είναι δυνατό, πιο χρήσιμες πληροφορίες και να τα εκμεταλλευτεί με ένα πιο γόνιμο και παραγωγικό τρόπο. Πιο συγκεκριμένα, αυτό που συνήθως επιθυμούμε και προσπαθούμε να εξάγουμε από τα δεδομένα είναι συσχετίσεις ανάμεσα σε διάφορα γεγονότα ή γενικούς κανόνες με τους οποίους μπορούμε είτε να εξηγήσουμε είτε να προβλέψουμε κάποια γεγονότα. Η προσπάθεια αυτή απαιτεί προφανώς μια αυτόματη διαδικασία, δηλαδή επινόηση και χρησιμοποίηση διαφόρων τεχνικών και αλγορίθμων και χρήση υπολογιστών. (Θα ήταν αδιανόητο να προσπαθεί κάποιος να βγάλει χρήσιμα συμπεράσματα απλά κοιτώντας σε μία βάση δεδομένων που περιέχει μερικά εκατομμύρια εγγραφές με δεκάδες πεδία η κάθε μία). Αυτή η διαδικασία ονομάζεται *Εξόρυξη Δεδομένων* (*Data Mining*).

Ένας ορισμός που έχει δοθεί για την εξόρυξη δεδομένων (γνωστή και ως ανακάλυψη γνώσης σε βάσεις δεδομένων –*Knowledge Discovery in Databases* (KDD)) είναι ο εξής: Ονομάζουμε Εξόρυξη Δεδομένων τη μη-τετριμμένη εξαγωγή έμμεσης, προηγουμένως άγνωστης, και ενδεχομένως χρήσιμης πληροφορίας από τα δεδομένα. Η εξόρυξη δεδομένων χρησιμοποιεί τεχνικές μάθησης (machine learning), στατιστικής και οπτικής απεικόνισης, για να ανακαλύψει πληροφορία κρυμμένη στα δεδομένα και να την παρουσιάσει σε μία μορφή που μπορεί να γίνει εύκολα κατανοητή από τον άνθρωπο.

Γενικά, η επιδίωξη μιας εργασίας εξόρυξης δεδομένων είναι η κατασκευή ενός μοντέλου που θα έχει έναν από τους εξής δύο ρόλους: επεξηγηματικό ή προβλεπτικό. Στην πρώτη περίπτωση το μοντέλο θα μπορεί δηλαδή να χρησιμοποιηθεί για να εντοπίσει και να ερμηνεύσει συσχετίσεις και αλληλεπιδράσεις ανάμεσα σε γεγονότα, παράγοντες που επηρεάζουν την εμφάνιση κάποιου γεγονότος κλπ. Στη δεύτερη περίπτωση ο στόχος είναι η πρόβλεψη κάποιων γεγονότων, χωρίς να ενδιαφέρει απαραίτητα ο εντοπισμός των παραγόντων που μπορεί να επηρεάζουν την εμφάνισή τους.

Λαμβάνοντας υπόψη τον όγκο των δεδομένων που πρέπει κανείς να επεξεργαστεί, όταν κάνει εξόρυξη δεδομένων, είναι προφανές πόσο αποτελεσματικής σημασίας είναι στον τομέα αυτό η πρόοδος στο χώρο των υπολογιστών. Βασικοί παράγοντες που έχουν συμβάλει στη δυνατότητα εφαρμογής τέτοιου είδους τεχνικών στην πράξη είναι η αύξηση της χωρητι-

κότητας των αποθηκευτικών μέσων, η αύξηση της ταχύτητας επεξεργασίας, η βελτίωση της αρχιτεκτονικής των συστημάτων παράλληλης επεξεργασίας και η βελτίωση των συστημάτων διαχείρισης βάσεων δεδομένων.

Η εξόρυξη δεδομένων εκμεταλλεύεται επίσης τις προόδους στους τομείς της τεχνητής νοημοσύνης (artificial intelligence) και της στατιστικής. Και οι δύο αυτοί τομείς έχουν συμβάλει στην επίλυση προβλημάτων αναγνώρισης προτύπων (*pattern recognition*) και ταξινόμησης (classification) και έχουν συνεισφέρει στην κατανόηση και την εφαρμογή των νευρωνικών δικτύων (neural networks) και των δέντρων απόφασης (decision trees).

1.2 Εφαρμογές

Η εξόρυξη δεδομένων γίνεται όλο και πιο δημοφιλής λόγω της ουσιαστικής συμβολής που μπορεί να έχει σε ένα πλήθος εφαρμογών. Στον κόσμο των επιχειρήσεων μπορεί να χρησιμοποιηθεί για να περιορίσει τις δαπάνες καθώς επίσης και να συμβάλει στην αύξηση του εισοδήματος. Πολλές επιχειρήσεις χρησιμοποιούν την εξόρυξη δεδομένων για την καλύτερη διαχείριση όλων των φάσεων του κύκλου ζωής των πελατών, συμπεριλαμβανομένων της απόκτησης νέων πελατών, της αύξησης του εισοδήματος από τους υπάρχοντες πελάτες, και της διατήρησης των καλών πελατών. Με τον καθορισμό των χαρακτηριστικών των καλών πελατών (*profiling*) μπορεί μια επιχείρηση να στοχεύσει σε κοινό με παρόμοια χαρακτηριστικά. Με τη σκιαγράφηση των πελατών που έχουν αγοράσει ένα ιδιαίτερο προϊόν μπορεί να στρέψει την προσοχή σε παρόμοιους πελάτες που δεν έχουν αγοράσει ακόμα εκείνο το προϊόν (*cross-selling*). Σκιαγραφώντας πελάτες που έχουν φύγει, μια επιχείρηση μπορεί να ενεργήσει για να διατηρήσει τους πελάτες που επίσης είναι πιθανό να φύγουν, διότι είναι συνήθως πολύ λιγότερο ακριβό να διατηρηθεί ένας πελάτης από το να αποκτηθεί νέος.

Η εξόρυξη δεδομένων είναι χρήσιμη για ένα αρκετά ευρύ φάσμα βιομηχανιών. Εταιρίες τηλεπικοινωνιών, ασφαλιστικές εταιρείες και επιχειρήσεις με πιστωτικές κάρτες χρησιμοποιούν την εξόρυξη δεδομένων για να ανιχνεύσουν ψευδή χρήση των υπηρεσιών τους και απάτες. Σε αλυσίδες εμπορικών καταστημάτων η εξόρυξη δεδομένων μπορεί να βοηθήσει στην επιλογή των προϊόντων που είναι προτιμότερο να τοποθετηθούν σε κάθε κατάστημα ή ακόμα και στο πού να τοποθετηθούν μέσα στο ίδιο κατάστημα.

Σημαντική συμβολή μπορεί να υπάρξει και στο χώρο της ιατρικής: η εξόρυξη δεδομένων μπορεί να χρησιμοποιηθεί για να προβλέψει την αποτελεσματικότητα χειρουργικών επεμβάσεων, ιατρικών δοκιμών ή φαρμάκων. Οι φαρμακευτικές εταιρίες μπορούν να χρησιμοποιήσουν τεχνικές εξόρυξης δεδομένων σε μεγάλες βάσεις δεδομένων από χημικές ενώσεις, για να εντοπίσουν ουσίες υποψήφιες για κατασκευή φαρμάκων.

Φυσικά πρέπει, όπως πάντα, να αποφεύγονται οι υπερβολές και να μην έχει κανείς ψευδαισθήσεις για τα αποτελέσματα που μπορεί να επιτύχει με την εξόρυξη δεδομένων. Δεν είναι για παράδειγμα δυνατό μια τέτοια τεχνική να χρησιμοποιηθεί για τον πλήρη καθορισμό της στρατηγικής μίας επιχείρησης, αντικαθιστώντας τη σκέψη, το σχεδιασμό και τη λήψη αποφάσεων. Αντίθετα πρόκειται για μια τεχνική που λειτουργεί βοηθητικά και συμπληρωματικά σε σχέση με αυτά. Πρέπει όποιος τη χρησιμοποιεί να γνωρίζει καλά τον τομέα του, να κατανοεί καλά τα δεδομένα, να μπορεί να ερμηνεύσει τα διάφορα συμπεράσματα που μπορεί να προκύψουν από τη διαδικασία και να μπορεί να ελέγξει αν οι διάφορες διαπιστώσεις και προβλέψεις όντως επαληθεύονται στον πραγματικό κόσμο. Για παράδειγμα ο εντοπισμός κάποιας συσχέτισης ανάμεσα σε δύο γεγονότα δε σημαίνει απαραίτητα την ύπαρξη κάποιας αιτιώδους σχέσης ανάμεσά τους. Συνεπώς πολύ βασικές προϋποθέσεις για

την επιτυχία μιας εργασίας εξόρυξης δεδομένων είναι: α) Να είναι διαθέσιμο ένα πολύ μεγάλο και κατάλληλο σύνολο δεδομένων (δηλαδή πολύ προσεκτική συλλογή και προετοιμασία των δεδομένων) και β) Να υπάρχει πολύ καλή γνώση και κατανόηση της περιοχής του προβλήματος (δηλαδή δυνατότητα σωστής ερμηνείας, κατανόησης και επαλήθευσης των αποτελεσμάτων).

Βέβαια και αυτός ο τομέας της επιστήμης έχει την κοινωνική του πλευρά. Η συλλογή και χρήση δεδομένων που αφορούν ανθρώπινες δραστηριότητες και συναλλαγές μπορεί να εγείρει νομικά και ηθικά προβλήματα, όπως ο σεβασμός της ιδιωτικής ζωής κλπ. Πολλοί άνθρωποι νιώθουν –τουλάχιστον– άβολα με τη σκέψη ότι η κυβέρνηση ή κάποιες εταιρίες ή οργανισμοί παρακολουθούν –ή έχουν τη δυνατότητα να παρακολουθούν– δραστηριότητες της προσωπικής τους ζωής, όπως αγορές προϊόντων, τηλεφωνικές συνδιαλέξεις, περιηγήσεις στο Internet κλπ., ανεξάρτητα από το αν έχουν κάτι να κρύψουν ή όχι. Ωστόσο τα θέματα αυτά, όπως και γενικά οποιοδήποτε θέμα, είναι καλό να αντιμετωπίζονται νηφάλια και ορθολογικά και να γίνει προσπάθεια για τη λήψη μέτρων και τη δημιουργία θεσμών που θα προστατεύουν την ιδιωτική ζωή των πολιτών και γενικά θα διευθετούν τέτοιου είδους ζητήματα που μπορεί να προκύψουν αντί να αποτελούν αιτία ή αφορμή για άκριτη και προκατειλημμένη επίθεση εναντίον της επιστήμης και της τεχνολογίας. Άλλωστε όπως –δυστυχώς πολύ σωστά– αναφέρει σε ένα έργο του ο *Karl Popper*, «Δεν υπάρχει τίποτα κάτω από τον ήλιο που να μην μπορεί να υποστεί κατάχρηση και που να μην την έχει υποστεί».

1.3 Σχέση με άλλους τομείς

Παρόλο που η εξόρυξη δεδομένων αποτελεί εξέλιξη της στατιστικής, ενός τομέα με μακρόχρονη ιστορία, ο όρος «εξόρυξη δεδομένων» εισήχθη σχετικά πρόσφατα, στη δεκαετία του 1990.

Η εξόρυξη δεδομένων έχει τις ρίζες της σε τρεις τομείς. Ο παλαιότερος από αυτούς είναι η κλασική Στατιστική. Χωρίς τη στατιστική, δεν θα υπήρχε εξόρυξη δεδομένων, καθώς η στατιστική αποτελεί τη βάση στην οποία στηρίζονται οι περισσότερες τεχνολογίες που χρησιμοποιεί η εξόρυξη δεδομένων. Η στατιστική περιλαμβάνει πολλές έννοιες (πχ. regression analysis, standard distribution, standard deviation, standard variance, discriminant analysis, cluster analysis, confidence intervals κλπ.), που χρησιμοποιούνται για τη μελέτη των δεδομένων και των σχέσεων ανάμεσα στα δεδομένα και αποτελούν δομικά στοιχεία για τεχνικές εξόρυξης δεδομένων.

Ωστόσο η εξόρυξη δεδομένων και η στατιστική ανάλυση παρά τις πολλές τους ομοιότητες δεν αποτελούν την ίδια διαδικασία. Στην περίπτωση της στατιστικής ανάλυσης κάποιος έχει μία ιδέα, προαίσθημα ή υπόθεση και αυτό που προσπαθεί να κάνει είναι να συγκεντρώσει δεδομένα σχετικά με το πρόβλημα και να ελέγξει αν τα δεδομένα αυτά συνηγορούν υπέρ ή κατά αυτής της υπόθεσης και σε ποιο βαθμό. Αντίθετα, στην περίπτωση της εξόρυξης δεδομένων, κάποιος διαθέτει ένα πλήθος δεδομένων σχετικά με κάποιο πρόβλημα και προσπαθεί να τα μελετήσει, για να δει τι συμπεράσματα και υποθέσεις μπορούν να προκύψουν. Δηλαδή στην πρώτη περίπτωση εναπόκειται στον άνθρωπο να σκεφτεί πιθανές ιδέες και υποθέσεις προς έλεγχο και ακόμα και αν βρεθεί μία τέτοια υπόθεση και διαπιστωθεί ότι τα δεδομένα την υποστηρίζουν, αυτό δεν αποκλείει την περίπτωση να υπάρχει και κάποια άλλη εναλλακτική υπόθεση η οποία να υποστηρίζεται εξίσου ή και ακόμα περισσότερο από τα ίδια δεδομένα. Από την άλλη πλευρά, με την εξόρυξη δεδομένων

είναι δυνατό να ανακαλύψει κανείς όλες τις υποθέσεις που είναι δυνατό να υποστηριχθούν από τα διαθέσιμα δεδομένα αλλά μετά εναπόκειται στον άνθρωπο να εξετάσει ποιες από αυτές έχουν πρακτική αξία. Όπως εύστοχα έχει ειπωθεί, «με τη στατιστική ανάλυση προσπαθείς να βρεις αυτό για το οποίο ψάχνεις, ενώ με την εξόρυξη δεδομένων ψάχνεις για το τι μπορείς να βρεις».

Ο δεύτερος τομέας είναι η *Τεχνητή Νοημοσύνη* (*Artificial Intelligence – AI*). Η Τεχνητή Νοημοσύνη, σε αντίθεση με τη στατιστική, στηρίζεται σε ευριστικές μεθόδους (heuristics) και προσπαθεί να εφαρμόσει στα στατιστικά προβλήματα μεθόδους επεξεργασίας που μοιάζουν με τον ανθρώπινο τρόπο σκέψης (human-thought-like processing). Επειδή αυτή η προσέγγιση απαιτεί τεράστια υπολογιστική ισχύ, δεν ήταν εύκολη η πρακτική εφαρμογή της μέχρι τις αρχές της δεκαετίας του ’80.

Ο τρίτος τομέας είναι η μηχανική μάθηση (*machine learning*), η οποία ουσιαστικά μπορεί να θεωρηθεί ότι αποτελεί ένωση της στατιστικής και της Τεχνητής Νοημοσύνης. Ο τομέας αυτός συνδυάζει θεμελιακές έννοιες της στατιστικής με αλγορίθμους της Τεχνητής Νοημοσύνης με σκοπό να παρέχει στα προγράμματα υπολογιστών τη δυνατότητα να «μαθαίνουν» από τα διαθέσιμα δεδομένα, δηλαδή να είναι σε θέση να παίρνουν διαφορετικές αποφάσεις σε διαφορετικές περιπτώσεις ανάλογα με τις ιδιότητες που έχουν βρει στα δεδομένα που έχουν μελετήσει.

Η εξόρυξη δεδομένων συνδέεται προφανώς και με τον τομέα των βάσεων και αποθηκών δεδομένων (*data warehouses*). Πολύ συχνά τα δεδομένα στα οποία θα εφαρμοστεί η διαδικασία βρίσκονται σε μία αποθήκη δεδομένων. Το γεγονός αυτό μπορεί να είναι χρήσιμο για το λόγο ότι τα δεδομένα πριν τοποθετούν σε μια αποθήκη δεδομένων έχουν υποστεί «καθαρισμό» και είναι πιθανόν η προετοιμασία αυτή να είναι αρκετή, ώστε να ξεκινήσει η εξόρυξη. Ωστόσο αυτό δε συμβαίνει πάντα και γενικά σε καμία περίπτωση δεν μπορεί να ειπωθεί ότι η τοποθέτηση των δεδομένων σε μία αποθήκη δεδομένων αποτελεί προϋπόθεση για την εξόρυξη δεδομένων.

Η OLAP (On-Line Analytical Processing) είναι μέρος του φάσματος των εργαλείων υποστήριξης αποφάσεων. Τα παραδοσιακά εργαλεία ερωτήσεων (query tools) περιγράφουν τι είναι σε μια βάση δεδομένων. Η OLAP προχωρεί παραπάνω, γιατί μπορεί να απαντήσει γιατί ορισμένα πράγματα ισχύουν. Ο χρήστης διαμορφώνει μια υπόθεση για μια σχέση και την ελέγχει με μια σειρά ερωτήσεων πάνω στα δεδομένα. Όταν όμως οι προς ανάλυση μεταβλητές είναι δεκάδες ή ακόμα και εκατοντάδες, γίνεται πολύ πιο δύσκολο και χρονοβόρο να βρεθεί μια καλή υπόθεση. Αλλά ακόμα και αν επιβεβαιωθεί κάποια υπόθεση δεν μπορούμε να είμαστε βέβαιοι ότι δεν υπάρχει μια ακόμα καλύτερη εξήγηση από αυτή που βρέθηκε. Η εξόρυξη δεδομένων διαφέρει από την OLAP στο ότι αντί να προσπαθεί να επιβεβαιώσει ή να διαψεύσει υποθέσεις που έχουν ήδη διατυπωθεί, χρησιμοποιεί τα ίδια τα δεδομένα για να αποκαλύψει διάφορα πρότυπα (patterns).

1.4 Στάδια της διαδικασίας

Η εξόρυξη δεδομένων είναι μία πολύπλοκη διαδικασία που αποτελείται από αρκετά στάδια, τα οποία μπορούμε να πούμε ότι σε γενικές γραμμές είναι τα εξής:

- συλλογή των δεδομένων
- προετοιμασία των δεδομένων
- κατασκευή του μοντέλου

- επαλήθευση και ερμηνεία των συμπερασμάτων

Πριν την κατασκευή ενός μοντέλου για προβλέψεις είναι απαραίτητο να γίνει μια προσεκτική προετοιμασία, μελέτη και κατανόηση των δεδομένων. Στο στάδιο αυτό μπορεί να γίνει μια στατιστική επεξεργασία των δεδομένων, δηλαδή ο υπολογισμός διάφορων χαρακτηριστικών στατιστικών μεγεθών, όπως μέσοι όροι, αποκλίσεις κλπ. Επίσης συχνά βοηθά η χρήση τεχνικών για οπτική αναπαράσταση των δεδομένων, γιατί μπορεί να διευκολύνει πολύ τον εντοπισμό κάποιων συσχετίσεων, προτύπων, εξαιρέσεων κλπ. ανάμεσα στα δεδομένα και να οδηγήσει στη διατύπωση κάποιων υποθέσεων και ιδεών που μπορεί να κατευθύνουν καλύτερα την περαιτέρω διαδικασία. Στο σκοπό αυτό, δηλαδή την ανακάλυψη συσχετίσεων και προτύπων ανάμεσα στα δεδομένα και γενικά την καλύτερη κατανόησή τους, μπορεί να συμβάλουν και τεχνικές ομαδοποίησης των δεδομένων (*clustering*) ή τεχνικές ανάλυσης συνδέσμων (*link analysis*). Οι δύο πιο κοινές προσεγγίσεις στην ανάλυση συνδέσμων είναι η ανακάλυψη συσχετίσεων (association discovery) και η ανακάλυψη ακολουθιών (sequence discovery). Αν, για παράδειγμα, επιθυμούμε να ελέγξουμε αν υπάρχει κάποια συσχέτιση ανάμεσα σε δύο γεγονότα Α και Β (πχ. κάποιος που αγόρασε το προϊόν Α να αγοράσει και το προϊόν Β) μπορούμε να ελέγξουμε τους εξής δύο παράγοντες: α) τη συχνότητα με την οποία τα δύο αυτά γεγονότα εμφανίζονται μαζί στο σύνολο των περιπτώσεων που έχουν καταγραφεί στη βάση δεδομένων (*support*) και β) τη σχετική τους πιθανότητα, δηλαδή με δεδομένο το ένα από τα δύο ποια είναι η πιθανότητα να συμβεί το δεύτερο (*confidence*).

Χρειάζεται επίσης πολύ προσοχή και σε άλλους παράγοντες, όπως στον τρόπο με τον οποίο χειρίζεται κανείς τους λεγόμενους «*outliers*», δηλαδή κάποιες –λίγες– περιπτώσεις που είναι «πάρα πολύ» διαφορετικές από τις περιπτώσεις που γενικά έχουν καταγραφεί (συνήθως πρόκειται για εξαιρέσεις ή λάθη κατά την καταγραφή των δεδομένων), στον τρόπο που κωδικοποιεί τα δεδομένα, καθώς και στην επιλογή των δεδομένων που θα περιλάβει για την κατασκευή του μοντέλου. Επίσης χρειάζονται κάποιες γνώσεις στατιστικής, γιατί βοηθούν στην παραμετροποίηση του μοντέλου με τέτοιο τρόπο ώστε να αυξάνεται η ακρίβεια και η ταχύτητά του.

Μετά την ολοκλήρωση της συλλογής και προετοιμασίας των δεδομένων ακολουθεί το στάδιο της κατασκευής του μοντέλου. Όπως αναφέρθηκε παραπάνω, ο ρόλος του μοντέλου είναι γενικά είτε επεξηγηματικός είτε προβλεπτικός. Συγκεκριμένα χρησιμοποιούμε τα δεδομένα που έχουμε καταγράψει από καταστάσεις που έχουν συμβεί στο παρελθόν, για να κατασκευάσουμε ένα μοντέλο και κατόπιν ερμηνεύουμε συσχετίσεις ή κάνουμε προβλέψεις βάσει αυτού του μοντέλου. Αυτή η ιδέα βέβαια, δηλαδή η χρησιμοποίηση της εμπειρίας του παρελθόντος για την κατανόηση ή την πρόβλεψη καταστάσεων, είναι κάτι που οι άνθρωποι εφάρμοζαν πολύ πριν και από την εμφάνιση των υπολογιστών. Μερικές από τις συνηθέστερα χρησιμοποιούμενες τεχνικές στο στάδιο αυτό είναι:

- Neural Networks (Νευρωνικά δίκτυα)
- Decision trees (Δένδρα αποφάσεων)
- Nearest neighbor method (Μέθοδος του κοντινότερου γείτονα)
- Genetic algorithms (Γενετικοί αλγόριθμοι)

Στις επόμενες παραγράφους παρουσιάζονται τα κύρια σημεία των τεχνικών αυτών.

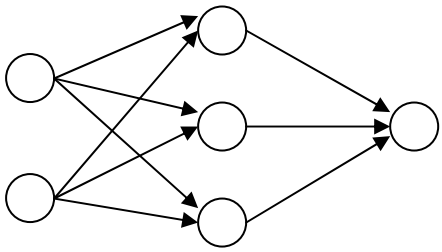
Προφανώς η κατασκευή του μοντέλου δεν είναι αρκετή, διότι πρέπει να επαληθευθεί αν το μοντέλο δουλεύει καλά. Αλλά για την επαλήθευση του μοντέλου δεν μπορούν να χρησιμοποιηθούν τα υπάρχοντα δεδομένα, αφού πάνω σε αυτά στηρίχθηκε η κατασκευή του και επομένως είναι αυτονόητο να τα επαληθεύει. Πρέπει να υπάρχει ένα άλλο σύνολο δεδομένων και για αυτό μία τεχνική που χρησιμοποιείται συνήθως είναι τα διαθέσιμα

δεδομένα να χωρίζονται σε δύο σύνολα και από αυτά το ένα να χρησιμοποιείται για την κατασκευή του μοντέλου, ενώ το άλλο για την επαλήθευσή του.

1.5 Neural Networks

Τα νευρωνικά δίκτυα, τα οποία προέρχονται από το χώρο της Τεχνητής Νοημοσύνης, παρουσιάζουν ιδιαίτερο ενδιαφέρον επειδή μπορούν να χρησιμοποιηθούν αποτελεσματικά για τη μοντελοποίηση μεγάλων και σύνθετων προβλημάτων στα οποία μπορεί να υπάρχουν εκατοντάδες μεταβλητές που αλληλεπιδρούν μεταξύ τους.

Όπως φαίνεται στο παρακάτω σχήμα, ένα νευρωνικό δίκτυο αρχίζει με ένα επίπεδο εισόδου (input layer), όπου κάθε κόμβος αντιστοιχεί σε μία μεταβλητή. Οι κόμβοι αυτού του επιπέδου συνδέονται με διάφορους κόμβους σε ένα εσωτερικό, κρυμμένο, επίπεδο (hidden layer). Κάθε κόμβος του αρχικού επιπέδου είναι συνδεδεμένος με κάθε κόμβο στο κρυμμένο επίπεδο. Οι κόμβοι στο κρυμμένο επίπεδο μπορούν να συνδεθούν είτε με κόμβους σε ένα επόμενο κρυμμένο επίπεδο είτε με κόμβους σε ένα επίπεδο εξόδου (output layer). Το επίπεδο εξόδου αποτελείται από τόσους κόμβους όσες οι τιμές που αναμένουμε να δώσει το μοντέλο στην έξοδο.



Σχήμα 1: Ένα απλό νευρωνικό δίκτυο με δύο εισόδους, ένα κρυμμένο επίπεδο και μία έξοδο

Κάθε ακμή μεταξύ δύο κόμβων του δικτύου χαρακτηρίζεται από μία τιμή, που παίζει το ρόλο του βάρους στους υπολογισμούς που γίνονται (στο σχήμα δεν έχουμε σημειώσει τα βάρη αυτά στις ακμές για λόγους ευκρίνειας). Κάθε κόμβος (εκτός από εκείνους του επιπέδου εισόδου) δέχεται ως είσοδο ένα σύνολο από τιμές (μία από κάθε κόμβο του αμέσως προηγούμενου επιπέδου). Πολλαπλασιάζει κάθε μία από τις τιμές αυτές με το αντίστοιχο βάρος, προσθέτει τα γινόμενα αυτά και στο άθροισμα εφαρμόζει κάποια συνάρτηση. Τέλος περνά το αποτέλεσμα στους κόμβους του επόμενου επιπέδου. Τα βάρη που θα τοποθετηθούν στις ακμές αποτελούν τις άγνωστες παραμέτρους του μοντέλου και προσδιορίζονται με μια διαδικασία εκπαίδευσης (training). Το πλήθος των κόμβων, το πλήθος των εσωτερικών επιπέδων και ο τρόπος σύνδεσης αποτελούν την αρχιτεκτονική ή την τοπολογία του νευρωνικού δικτύου. Στην επιλογή αυτών των χαρακτηριστικών μπορούν να συμβάλουν ορισμένοι παράγοντες αλλά συχνά χρειάζεται κανείς να πειραματιστεί. Μία μέθοδος για την εκπαίδευση του δικτύου, δηλαδή για τον προσδιορισμό των βαρών, μπορεί να περιγραφεί με συντομία ως εξής: Ξεκινάμε έχοντας ένα σύνολο από δεδομένα που θα χρησιμοποιήσουμε για τη διαδικασία εκπαίδευσης (training set) και επιλέγουμε κάποιες αρχικές τιμές για τα βάρη. Επιλέγουμε δεδομένα από το training set ως εισόδους και ακολουθούμε τη διαδικασία υπολογισμών που αναφέρθηκε παραπάνω μέχρι να πάρουμε την τιμή στην έξοδο. Συγκρίνοντας την τιμή αυτή με την πραγματική τιμή, δηλαδή την αντίστοιχη τιμή από το training set, βρίσκουμε το σφάλμα. Κατόπιν κατανέμουμε το σφάλμα αυτό στους κόμβους

ανάλογα με τα βάρη των ακμών και επαναλαμβάνουμε τη διαδικασία μέχρι να ελαχιστοποιήσουμε το σφάλμα.

Εξαιτίας του μεγάλου πλήθους των παραμέτρων ενός νευρωνικού δικτύου (που είναι τόσο περισσότερες όσο περισσότερα είναι τα ενδιάμεσα επίπεδα και όσο περισσότεροι είναι οι κόμβοι σε κάθε επίπεδο), είναι δυνατό ένα νευρωνικό δίκτυο, αν είναι αρκετά μεγάλο, να ταιριάξει τελικά σε οποιοδήποτε σύνολο δεδομένων αν επαναλάβουμε τη διαδικασία εκπαίδευσης μέχρι να φτάσουμε σε σύγκλιση. Ο σκοπός όμως προφανώς δεν είναι να φτιάξουμε ένα δίκτυο που θα κάνει σωστές προβλέψεις στα δεδομένα του συνόλου δοκιμής αλλά και σε άλλα σύνολα δεδομένων. Για να αποφύγουμε αυτόν τον κίνδυνο του υπερβολικού ταιριάσματος του δικτύου στα δοκιμαστικά δεδομένα, είναι συχνά καλό να τερματίσουμε τη διαδικασία εκπαίδευσης πριν φτάσουμε σε σύγκλιση, δηλαδή σε ένα πιο πρώιμο σημείο ακόμα και αν το σφάλμα συνεχίζει να μειώνεται μετά από το σημείο αυτό. Ένας τρόπος να εντοπίσουμε ποιο θα ήταν το κατάλληλο αυτό σημείο είναι να έχουμε εκτός από το σύνολο δοκιμής και ένα δεύτερο σύνολο δεδομένων και κατά τη διάρκεια της διαδικασίας εκπαίδευσης να ελέγχουμε περιοδικά το σφάλμα και στο δεύτερο αυτό σύνολο και όταν παρατηρήσουμε ότι το σφάλμα σε αυτό το σύνολο αρχίζει να αυξάνεται να σταματήσουμε τη διαδικασία ακόμα και αν το σφάλμα στο training set εξακολουθεί να φθίνει.

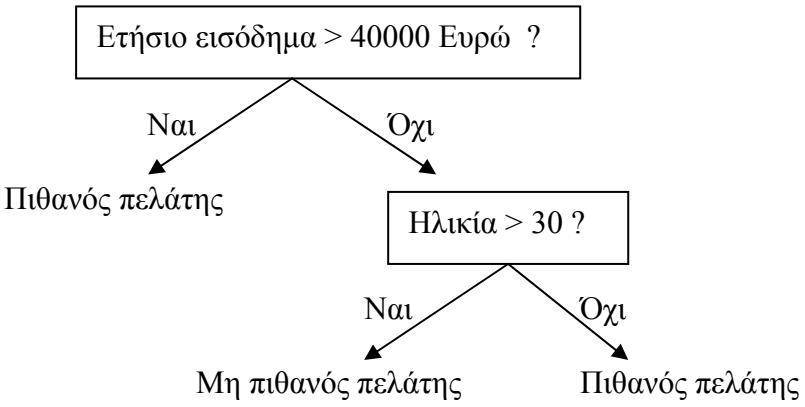
Ένα άλλο μειονέκτημα του μεγάλου πλήθους παραμέτρων είναι το γεγονός ότι καθίσταται πολύ δύσκολη και συνήθως αδύνατη η απόδοση ερμηνείας σε αυτές και είναι αναγκασμένος κανείς να βλέπει το νευρωνικό δίκτυο σαν ένα «μαύρο κουτί». Μάλιστα είναι δυνατό δύο ή περισσότερα διαφορετικά σύνολα παραμέτρων να οδηγούν σε ίδια αποτελέσματα. Ωστόσο υπάρχουν πολλές εφαρμογές στις οποίες αυτό δεν αποτελεί πρόβλημα. Πραγματικά τα νευρωνικά δίκτυα βρίσκουν πολλές εφαρμογές σε διάφορους τομείς, όπως για παράδειγμα στον τομέα της αναγνώρισης προτύπων, για αναγνώριση ομιλίας, εικόνας, γραφικού χαρακτήρα κλπ.

Επίσης στα νευρωνικά δίκτυα απαιτείται τα δεδομένα εισόδου να είναι σε αριθμητική μορφή επομένως αν αυτό δεν ισχύει θα πρέπει να προηγηθεί ένα στάδιο μετατροπής των δεδομένων στη μορφή αυτή.

Ένα βασικό τους πλεονέκτημα είναι ότι, επειδή οι υπολογισμοί σε κάθε κόμβο ενός επιπέδου μπορούν να γίνουν ανεξάρτητα από τους υπόλοιπους κόμβους του ίδιου επιπέδου, είναι εύκολη η υλοποίηση του δικτύου με τρόπο που να μπορεί να λειτουργήσει σε ένα μεγάλο σύστημα παράλληλης επεξεργασίας και να εκμεταλλευτεί έτσι τις μεγάλες υπολογιστικές δυνατότητες που προσφέρουν τα συστήματα αυτά. Επίσης, παρόλο που η διαδικασία εκπαίδευσης απαιτεί πολλή προσοχή και πολύ χρόνο, από τη στιγμή που θα ολοκληρωθεί μπορεί το δίκτυο να παρέχει αποτελέσματα αρκετά γρήγορα.

1.6 Decision Trees

Ένα δένδρο απόφασης (decision tree) είναι ένα μοντέλο για προβλέψεις, το οποίο όπως φανερώνει και το όνομά του έχει τη μορφή δένδρου. Κάθε διακλάδωση του δένδρου παριστάνει μία ερώτηση, η οποία έχει σαν αποτέλεσμα να χωρίζονται τα δεδομένα σε δύο ή περισσότερα υποσύνολα και η διαδικασία αυτή συνεχίζεται μέχρι να φτάσουμε στα φύλλα. Έτσι τα φύλλα αποτελούν στην ουσία μία διαμέριση του αρχικού συνόλου δεδομένων. Αυτό μπορεί να γίνει πιο σαφές με το παρακάτω, απλό, σχήμα:



Σχήμα 2: Παράδειγμα δένδρου απόφασης

Το δένδρο του παραπάνω σχήματος είναι κατασκευασμένο με τέτοιο τρόπο (δηλαδή οι ερωτήσεις έχουν τέτοια μορφή) ώστε να χωρίζει ένα σύνολο ατόμων σε δύο υποσύνολα (πιθανός πελάτης – μη πιθανός πελάτης) που α) είναι ξένα μεταξύ τους (δεν είναι δυνατό το ίδιο άτομο να ανήκει συγχρόνως και στα δύο) και β) η ένωσή τους δίνει το αρχικό σύνολο (κάθε άτομο του αρχικού συνόλου είτε θα είναι πιθανός πελάτης είτε όχι).

Ένα πλεονέκτημα των δένδρων απόφασης είναι ότι εύκολα μπορεί να κατανοήσει κανείς τον τρόπο κατασκευής τους και να τα χρησιμοποιήσει για να κάνει ερμηνείες –σε αντίθεση για παράδειγμα με ένα νευρωνικό δίκτυο το οποίο, όπως αναφέραμε, το βλέπει κανείς περισσότερο σαν ένα «μαύρο κουτί». Στο παραπάνω σχήμα για παράδειγμα, εύκολα μπορεί κανείς να διατυπώσει το συμπέρασμα ότι κάποιος που έχει ετήσιο εισόδημα μικρότερο από 40000 ευρώ και είναι κάτω των 30 ετών δεν αποτελεί πιθανό πελάτη –και συνεπώς δεν υπάρχει λόγος να αποτελεί στόχο της διαφημιστικής του εκστρατείας– ενώ οι υπόλοιπες περιπτώσεις αποτελούν πιθανούς πελάτες. Επίσης δεν απαιτούν πολύ μεγάλη προετοιμασία των δεδομένων και μπορούν να χρησιμοποιηθούν τόσο για την εξερεύνηση των δεδομένων όσο και για προβλέψεις. Εξάλλου η δενδρική μορφή του μοντέλου αποτελεί σημαντικό πλεονέκτημα, γιατί τα δένδρα αποτελούν μια από τις σημαντικότερες και πιο μελετημένες δομές δεδομένων και υπάρχουν πάρα πολλοί αλγόριθμοι για τη διαχείρισή τους.

Η κύρια δυσκολία στην κατασκευή του δένδρου είναι η ανεύρεση των κατάλληλων ερωτήσεων για κάθε διακλάδωση. Κάποιοι αλγόριθμοι χρησιμοποιούν ευριστικές (heuristic) τεχνικές για την επιλογή των κατάλληλων ερωτήσεων ή ακόμα δοκιμάζουν τις διάφορες πιθανές περιπτώσεις και τελικά επιλέγουν αυτή που οδηγεί στην καλύτερη διαμέριση. Μία άλλη απόφαση που πρέπει να ληφθεί είναι το σημείο στο οποίο θα σταματήσει το μεγάλωμα του δένδρου. Αν αφήσουμε το δένδρο να μεγαλώσει πάρα πολύ –τελικά θα καταλήξει στην κατάσταση όπου κάθε φύλλο περιέχει ένα και μόνο άτομο– αφενός θα επιβαρυνθούμε με πολύ μεγάλο υπολογιστικό κόστος και αφετέρου το μοντέλο χάνει το νόημά του. Από την άλλη πλευρά το δένδρο χρειάζεται να μεγαλώσει αρκετά, για να υπάρχει ομοιογένεια στα φύλλα του. Επίσης, όπως περιγράφτηκε και στην περίπτωση των νευρωνικών δικτύων, υπάρχει και εδώ το πρόβλημα του υπερβολικού ταιριάσματος στα δεδομένα (overfitting the training data).

Ένας από τους πρώτους αλγόριθμους που χρησιμοποιούν δένδρα απόφασης είναι ο ID3, ο οποίος εισήχθη από τον J. Ross Quinlan στα τέλη του 1970 και χρησιμοποιήθηκε αρχικά για επιλογή στρατηγικής σε παιχνίδια όπως το σκάκι. Στον ID3 η επιλογή των ερωτήσεων γίνεται με βάση το κέρδος σε πληροφορία που επιτυγχάνεται αν γίνει ο υπό εξέταση διαχωρισμός. Το κέρδος αυτό αντιπροσωπεύει τη διαφορά στο ποσό πληροφορίας που απαιτείται για μία σωστή πρόβλεψη πριν και μετά το διαχωρισμό. Πιο συγκεκριμένα το

κέρδος ορίζεται ως η διαφορά ανάμεσα στην εντροπία του συνόλου των δεδομένων πριν το διαχωρισμό και στο άθροισμα της εντροπίας των υποσυνόλων που προκύπτουν μετά το διαχωρισμό. Αν το κέρδος αυτό είναι σημαντικό, τότε γίνεται ο διαχωρισμός.

1.7 K-Nearest Neighbor

Συχνά όταν αναζητούμε τη λύση ενός προβλήματος, προσπαθούμε να το ανάγουμε σε παρόμοια προβλήματα των οποίων τη λύση γνωρίζουμε. Αυτή είναι η γενική ιδέα στη μέθοδο K-Nearest Neighbor. Η μέθοδος αυτή λειτουργεί ως εξής: Έχουμε κάποιες κατηγορίες K_1, K_2, …, K_v και ένα σύνολο Σ από αντικείμενα για καθένα από τα οποία γνωρίζουμε σε ποια από αυτές τις κατηγορίες ανήκει. Το ερώτημα είναι σε ποια από αυτές τις κατηγορίες θα πρέπει να τοποθετηθεί ένα νέο αντικείμενο Χ. Για να απαντήσουμε, βρίσκουμε τα k αντικείμενα του συνόλου Σ που μοιάζουν περισσότερο με το αντικείμενο Χ και αν τα περισσότερα από αυτά ανήκουν στην κατηγορία K_μ, τότε τοποθετούμε και το Χ στην κατηγορία αυτή. Ο λόγος για τον οποίο λαμβάνουμε υπόψη μας τους k «κοντινότερους γείτονες» και όχι μόνο τον ένα πιο κοντινό είναι διότι υπάρχει περίπτωση αυτός ο ένας να αποτελεί εξαίρεση ή να έχει γίνει κάποιο λάθος στην καταγραφή του οπότε θα μας οδηγούσε σε λάθος συμπέρασμα. Δηλαδή μπορούμε να χρησιμοποιήσουμε αυτή τη μέθοδο είτε για να κατηγοριοποιήσουμε ένα νέο αντικείμενο είτε για να προβλέψουμε τη συμπεριφορά ενός νέου αντικειμένου, με βάση την πληροφορία που διαθέτουμε για παρόμοια με αυτό αντικείμενα. Η σιγουριά (confidence) που μπορούμε να έχουμε για την κατηγοριοποίηση ή την πρόβλεψη που κάνουμε είναι τόσο μεγαλύτερη όσο πιο «κοντινοί» είναι οι «γείτονες», δηλαδή όσο περισσότερο μοιάζει το αντικείμενο αυτό με τα γνωστά μας αντικείμενα και επίσης όσο περισσότεροι από τους «γείτονες» ανήκουν στην ίδια κατηγορία. Για παράδειγμα, αν κάποιος ζει σε μία γειτονιά όπου το 80% των ανθρώπων μιλάει ελληνικά, μπορούμε να κάνουμε την πρόβλεψη ότι και αυτός μιλάει ελληνικά. Αν το 90% των ανθρώπων της γειτονιάς μιλάει ελληνικά μπορούμε να διατυπώσουμε την ίδια υπόθεση με ακόμα περισσότερη βεβαιότητα. Επίσης μπορούμε να είμαστε περισσότερο αισιόδοξοι για την ορθότητα της πρόβλεψής μας αν οι πληροφορίες που έχουμε αφορούν άτομα της ίδιας γειτονιάς από ότι αν αφορούν άτομα της ίδιας πόλης.

Όπως προκύπτει από την παραπάνω περιγραφή της μεθόδου, για να την εφαρμόσουμε είναι απαραίτητο προηγουμένως να ορίσουμε ένα μέτρο ομοιότητας ανάμεσα στα αντικείμενα που μας ενδιαφέρουν, δηλαδή μια «συνάρτηση απόστασης». Αυτό αποτελεί και την κυριότερη δυσκολία στη χρήση της μεθόδου, ειδικά στην περίπτωση που κάποιες από τις μεταβλητές που περιγράφουν τα αντικείμενα δε βρίσκονται σε αριθμητική μορφή.

Ένα μειονέκτημα της μεθόδου είναι το υψηλό υπολογιστικό κόστους που συνεπάγεται, διότι κάθε φορά που θέλουμε να κατηγοριοποιήσουμε ή να κάνουμε μία πρόβλεψη για ένα καινούριο αντικείμενο, πρέπει να υπολογίσουμε την «απόσταση» αυτού του αντικειμένου από κάθε ένα από τα γνωστά αντικείμενα, προκειμένου να βρούμε τους k «κοντινότερους γείτονές» του. Αντίθετα, στην περίπτωση των νευρωνικών δικτύων και των δένδρων αποφάσεων, όπως είδαμε, από τη στιγμή που θα κατασκευασθεί το μοντέλο είναι πολύ εύκολο να εφαρμοσθεί σε καινούρια δεδομένα.

1.8 Γενετικοί αλγόριθμοι

Οι γενετικοί αλγόριθμοι (genetic algorithms) είναι εμπνευσμένοι από τη θεωρία της εξέλιξης του Δαρβίνου. Η γενική ιδέα είναι ότι τα προβλήματα μπορούν να λυθούν με μια εξελικτική διαδικασία, μέσω της οποίας η λύση συνεχώς βελτιώνεται μέχρι να βρεθεί η καλύτερη δυνατή.

Στα κύτταρα των ζωντανών οργανισμών υπάρχουν τα χρωμοσώματα, τα οποία αποτελούνται από γονίδια (τμήματα DNA), κάθε ένα από τα οποία περιγράφει κάποιο χαρακτηριστικό του οργανισμού (πχ. το χρώμα των ματιών). Κατά τη διαδικασία της αναπαραγωγής, τμήματα από τα χρωμοσώματα των γονιών συνδυάζονται, για να δημιουργήσουν τα χρωμοσώματα του παιδιού. Αυτό λέγεται επανασυνδυασμός (recombination) ή διασταύρωση (crossover). Με αυτό τον τρόπο οι γονείς κληροδοτούν τα χαρακτηριστικά τους στα παιδιά τους. Η εξέλιξη συνίσταται στο ότι εκείνοι που αναπαράγονται, και κατά συνέπεια μεταβιβάζουν τα χαρακτηριστικά τους στις επόμενες γενιές, είναι εκείνα τα άτομα του είδους που κατάφεραν να επιβιώσουν, δηλαδή τα χαρακτηριστικά που συμβάλλουν στην επιβίωση είναι πιθανότερο να περάσουν στις επόμενες γενιές. Για τον ίδιο λόγο, ένα παιδί που έχει γονείς μουσικούς είναι πιθανότερο να εκδηλώσει ταλέντο και κλίση στη μουσική. Επίσης κατά τη διαδικασία της αναπαραγωγής είναι δυνατό να συμβεί και μετάλλαξη (mutation), δηλαδή κάποια μικρή αλλαγή σε κάποια τμήματα του DNA.

Για να φανεί πώς τα παραπάνω χρησιμοποιούνται στους γενετικούς αλγόριθμους, μπορούμε να θεωρήσουμε το ακόλουθο πρόβλημα. Έστω ότι θέλουμε να κατασκευάσουμε ένα νευρωνικό δίκτυο και να το χρησιμοποιήσουμε για προβλέψεις. Αν υποθέσουμε ότι έχουμε αποφασίσει για την τοπολογία του δικτύου, αυτό που μένει να επιλέξουμε είναι τα βάρη που θα τοποθετήσουμε στους συνδέσμους. Για κάθε διαφορετική επιλογή βαρών προκύπτει και ένα διαφορετικό δίκτυο, που μπορεί να είναι περισσότερο ή λιγότερο κατάλληλο για το σκοπό μας. Δεν υπάρχει κάποιος αλγόριθμος που να μας οδηγεί στον απευθείας υπολογισμό των βέλτιστων βαρών. Επιπλέον, αν και είναι εύκολο να ελέγξουμε την καταλληλότητα ενός δεδομένου δικτύου, δεν είναι δυνατό να κατασκευάσουμε όλα τα δυνατά δίκτυα και να τα δοκιμάσουμε ένα προς ένα προκειμένου να βρούμε το πιο κατάλληλο. Μπορούμε όμως να ακολουθήσουμε την εξής διαδικασία: Ξεκινάμε κατασκευάζοντας διάφορα δίκτυα, δηλαδή ένα σύνολο πιθανών λύσεων (πληθυσμός - population). Κάθε δίκτυο μπορεί να περιγραφεί από έναν πίνακα-γραμμή που περιέχει τα βάρη των συνδέσμων του δικτύου με μία καθορισμένη σειρά. Αυτός ο πίνακας αποτελεί το «χρωμόσωμα» του δικτύου. Κατόπιν μπορούμε να επιλέξουμε από τον πληθυσμό τις καλύτερες λύσεις και να τις συνδυάσουμε, με τις διαδικασίες της διασταύρωσης και της μετάλλαξης που αναφέρθηκαν παραπάνω, για να δημιουργήσουμε έναν καινούριο πληθυσμό. Για παράδειγμα, αν δύο καλές λύσεις από τον αρχικό πληθυσμό περιγράφονται από τις ακολουθίες βαρών (w_1, w_2, …, w_k, w_{k+1}, …, w_{n-1}, w_n) και (w_1', w_2', …, w_k', w_{k+1}', …, w_{n-1}', w_n') αντίστοιχα, τότε μπορούμε με διασταύρωση να πάρουμε τη λύση (w_1, w_2, …, w_k, w_{k+1}', …, w_{n-1}', w_n'), στην οποία μπορούμε να εφαρμόσουμε και μετάλλαξη, δηλαδή να αλλάξουμε τυχαία κάποια ή κάποιες τιμές. Μπορούμε να ελπίζουμε ότι εφόσον αυτή η λύση προήλθε από συνδυασμό των χαρακτηριστικών δύο καλών λύσεων, θα αποτελεί πιθανόν μια ακόμα καλύτερη λύση. Βέβαια είναι δυνατό να είναι χειρότερη από τις αρχικές, οπότε για να αποφύγουμε τον κίνδυνο χειροτέρευσης των λύσεων, ένας τρόπος είναι μετά τη δημιουργία του νέου πληθυσμού να προσθέτουμε σε αυτόν και αυτούσιες τις καλύτερες λύσεις του προηγούμενου πληθυσμού. Γενικά πάντως ακολουθώντας μια τέτοια διαδικασία, είναι δυνατό να επιτύχουμε μία εξέλιξη (βελτίωση) των λύσεων.

Βέβαια η παραπάνω περιγραφή απλώς σκιαγραφεί τη διαδικασία. Στην πράξη υπάρχουν αρκετές λεπτομέρειες και παράμετροι που πρέπει να καθοριστούν, όπως το πώς θα κωδικοποιούνται οι λύσεις σε μορφή «χρωμοσωμάτων» (αυτό εξαρτάται από τη φύση του προβλήματος), πώς ακριβώς θα γίνεται η διαδικασία της διασταύρωσης, δηλαδή πώς θα επιλέγονται οι «γονείς» και σε πόσα και ποια τμήματα θα χωρίζονται τα χρωμοσώματά τους, ώστε να συνδυαστούν και να δώσουν τα χρωμοσώματα των «παιδιών», πόσο συχνά θα συμβαίνει μετάλλαξη, αν και πόσες λύσεις από έναν πληθυσμό θα περνούν αυτούσιες στον επόμενο, πότε θα σταματά η διαδικασία κλπ.

Μέρος 2ο: Ομαδοποίηση και Ταξινόμηση

2.1 Τι ονομάζουμε ομαδοποίηση και ταξινόμηση

Με τον όρο *ομαδοποίηση* (*clustering*) εννοούμε τη διαδικασία κατά την οποία έχουμε στη διάθεσή μας ένα σύνολο αντικειμένων που περιγράφονται από διάφορες ιδιότητες (μεταβλητές) και θέλουμε να ερευνήσουμε τη δομή αυτού του συνόλου και να διαπιστώσουμε αν τα αντικείμενα αυτά μπορούν από τη φύση τους να χωριστούν σε ομάδες (classes, clusters), με τέτοιο τρόπο ώστε τα αντικείμενα της ίδιας ομάδας να μοιάζουν το ένα με το άλλο όσο το δυνατόν περισσότερο ενώ τα αντικείμενα που ανήκουν σε διαφορετικές ομάδες να διαφέρουν όσο το δυνατόν περισσότερο. (Ο όρος cluster analysis χρησιμοποιήθηκε για πρώτη φορά από τον Tryon, το 1939). Συνεπώς τα δύο κριτήρια που θέλουμε να ικανοποιούν οι ομάδες που θα κατασκευάσουμε όταν κάνουμε ομαδοποίηση είναι: α) «*εσωτερική ομοιογένεια*» και β) «*εξωτερική απομόνωση*». Πρέπει πάντως να επισημάνουμε ότι δεν είναι πάντα δυνατό τα αντικείμενα να χωριστούν σε ομάδες με τρόπο που να ικανοποιεί τα παραπάνω δύο κριτήρια. Αντίθετα είναι δυνατό η προσπάθειά μας να καταλήξει σε αυτή ακριβώς τη διαπίστωση. Αν ωστόσο υπάρχουν πράγματι τέτοιες ομάδες, ο σκοπός αυτής της διαδικασίας είναι η ανακάλυψή τους.

Με τον όρο *ταξινόμηση* (*classification*) εννοούμε τη διαδικασία κατά την οποία έχουμε ένα αντικείμενο και ένα σύνολο από κατηγορίες (classes) και προσπαθούμε να αποφασίσουμε σε ποια από αυτές τις κατηγορίες ταιριάζει καλύτερα το συγκεκριμένο αντικείμενο.

Η ομαδοποίηση και η ταξινόμηση είναι παρόμοιες διαδικασίες. Και στις δύο περιπτώσεις επιδιώκουμε να έχουμε ομάδες από παρόμοια μεταξύ τους αντικείμενα. Η διαφορά είναι ότι στην περίπτωση της ομαδοποίησης είναι αρχικά άγνωστο πόσες και ποιες θα είναι οι ομάδες στις οποίες θα χωριστούν τα αντικείμενα αλλά αντίθετα αποτελεί και αυτό μέρος του ερωτήματος που πρέπει να απαντηθεί. Δηλαδή αφήνουμε το σύστημα να εξετάσει τα αντικείμενα, να τα συγκρίνει μεταξύ τους και να κάνει μια ομαδοποίηση που θα ικανοποιεί τα δύο βασικά κριτήρια που έχουμε αναφέρει και κατόπιν βλέπουμε πόσες ομάδες έχουν σχηματιστεί και προσπαθούμε να κατανοήσουμε το νόημά τους, δηλαδή τι αντιπροσωπεύει κάθε ομάδα -αν βέβαια μπορεί να αποδοθεί σε αυτό μια φυσική σημασία. Για το λόγο αυτό η ομαδοποίηση χαρακτηρίζεται ως «*unsupervised*», δηλαδή χωρίς επίβλεψη, διαδικασία. Αντίθετα στην περίπτωση της ταξινόμησης έχουμε αποφασίσει εκ των προτέρων πόσες και ποιες είναι οι κατηγορίες που μας ενδιαφέρουν και έχουμε τοποθετήσει σε κάθε μία από αυτές έναν αριθμό αντικειμένων, ώστε να υποδείξουμε στο σύστημα τι είδους αντικείμενα ταιριάζουν σε κάθε κατηγορία. Υπάρχει δηλαδή ένα στάδιο εκπαίδευσης (*training*) του συστήματος. Για το λόγο αυτό η ταξινόμηση χαρακτηρίζεται ως «*supervised*», δηλαδή με επίβλεψη, διαδικασία.

2.2 Εφαρμογές

Ο άνθρωπος από πολύ νωρίς επιδίδονταν στην προσπάθεια να ταξινομεί αντικείμενα σε μη προκαθορισμένες ομάδες με τρόπο που τα αντικείμενα κάθε ομάδας να μοιάζουν, με κάποια έννοια, μεταξύ τους. Η σύγκριση και η ταξινόμηση των δεδομένων είναι ένα θεμελιώδες στάδιο στην οργάνωση της πληροφορίας με σκοπό την καλύτερη κατανόησή της. Με τη μεγάλη ανάπτυξη των υπολογιστών έχει αυξηθεί σημαντικά το ενδιαφέρον για αυτόματη ταξινόμηση και υπάρχουν σημαντικές εφαρμογές σε πολλά και διαφορετικά μεταξύ τους πεδία, όπως η ψυχολογία, η βιολογία, η ιατρική, η αναγνώριση προτύπων, η έρευνα αγοράς κλπ.

Ακόμα και στην καθημερινή ζωή συναντούμε άπειρα παραδείγματα ομαδοποίησης και ταξινόμησης. Τα προϊόντα σε ένα πολυκατάστημα ομαδοποιούνται, ώστε να διευκολυνθούν οι αγοραστές στην αναζήτηση και επιλογή τους. Ταξινομούμε τις υποχρεώσεις μας σε εκείνες που πρέπει να εκπληρωθούν άμεσα και σε εκείνες που μπορούν να περιμένουν, ώστε να οργανώσουμε καλύτερα το πρόγραμμά μας και να είμαστε περισσότερο αποτελεσματικοί και συνεπείς κλπ.

Στο χώρο της εξόρυξης δεδομένων οι διαδικασίες της ομαδοποίησης και της ταξινόμησης έχουν αποφασιστική συμβολή. Όπως αναφέρθηκε στο $1^{ο}$ μέρος, καθοριστικός παράγοντας για την επιτυχή έκβαση μιας εργασίας εξόρυξης δεδομένων είναι η προσεκτική προετοιμασία, μελέτη και κατανόηση των δεδομένων. Συνεπώς μέθοδοι, όπως η ομαδοποίηση και η ταξινόμηση, που κατασκευάζουν μια περίληψη των δεδομένων μπορούν να βοηθήσουν στην ανεύρεση σημαντικών συσχετίσεων ανάμεσά τους. Αν είναι πράγματι δυνατή η δημιουργία ομάδων από παρόμοια αντικείμενα, τότε μπορούμε να ονομάσουμε αυτές τις ομάδες και να έχουμε μια περίληψη των ιδιοτήτων τους, επιτυγχάνοντας έτσι την αποτελεσματική οργάνωση των δεδομένων και την εύκολη και γρήγορη αναζήτηση και ανάκτηση της διαθέσιμης πληροφορίας. Επιπλέον η οργάνωση των δεδομένων και η ανεύρεση συσχετίσεων ανάμεσά τους μπορεί να οδηγήσει στη διατύπωση υποθέσεων και προβλέψεων, η εγκυρότητα των οποίων είναι δυνατό να ελεγχθεί με την εφαρμογή τους σε αντίστοιχα σύνολα δεδομένων. Επειδή πάντως συμβαίνει συχνά να υπάρχουν περισσότερες από μία δυνατές ταξινομήσεις για το ίδιο σύνολο δεδομένων, ανάλογα με τα κριτήρια που θα χρησιμοποιηθούν, είναι πολύ σημαντικό ο ερευνητής να επιλέξει προσεκτικά τα κριτήρια εκείνα που τον ενδιαφέρουν, δηλαδή τις μεταβλητές που θα χρησιμοποιηθούν για την περιγραφή κάθε αντικειμένου, ώστε η ταξινόμηση που θα πραγματοποιήσει να είναι αποτελεσματική και χρήσιμη για την εξαγωγή περαιτέρω πληροφοριών και συμπερασμάτων.

2.3 Τύποι δεδομένων

Τα αντικείμενα που θέλουμε να χωρίσουμε σε ομάδες περιγράφονται από διάφορες μεταβλητές (variables, attributes). Υπάρχουν μεταβλητές διαφόρων τύπων.

Ένας τύπος είναι οι «*numeric variables*». Οι μεταβλητές αυτού του τύπου έχουν ως τιμές πραγματικούς αριθμούς. Παραδείγματα τέτοιων μεταβλητών είναι το εισόδημα ενός ατόμου, η ηλικία του κλπ.

Δύο άλλα είδη μεταβλητών είναι οι «*nominal variables*» και οι «*ordinal variables*». Σε αυτή την κατηγορία ανήκουν οι μεταβλητές που παίρνουν τιμές από ένα πεπερασμένο

σύνολο καταστάσεων. Παραδείγματα τέτοιων μεταβλητών είναι ο βαθμός ενός αξιωματικού, το χρώμα ματιών ενός ανθρώπου κλπ. Η διαφορά ανάμεσα στις nominal και στις ordinal μεταβλητές είναι η εξής: Στις nominal μεταβλητές δεν υπάρχει κάποια έννοια διάταξης ανάμεσα στις δυνατές καταστάσεις. Δηλαδή αν S_1 και S_2 είναι δύο καταστάσεις που μπορεί να πάρει μια nominal μεταβλητή, τότε θα ισχύει είτε $S_1 = S_2$ είτε $S_1 \neq S_2$, δεν έχει δηλαδή νόημα να πούμε $S_1 < S_2$ ή $S_1 > S_2$. Αντίθετα στις ordinal μεταβλητές υπάρχει διάταξη ανάμεσα στις καταστάσεις. Έτσι αν $S_1 \neq S_2$, τότε $S_1 < S_2$ ή $S_1 > S_2$. Παράδειγμα στην πρώτη περίπτωση είναι το χρώμα ματιών, ενώ στη δεύτερη ο βαθμός ενός αξιωματικού. Το γεγονός αυτό πρέπει να λαμβάνεται υπόψη κατά τον υπολογισμό της ομοιότητας ή διαφοράς ανάμεσα στα αντικείμενα.

Μία ειδική περίπτωση της παραπάνω κατηγορίας μεταβλητών είναι οι «*binary variables*», οι οποίες παίρνουν τιμές από ένα σύνολο δύο διαφορετικών καταστάσεων, όπως για παράδειγμα το φύλο ενός ανθρώπου. Σε ορισμένες περιπτώσεις μπορούμε να θεωρούμε τη μία από τις δύο καταστάσεις ως προτιμώμενη ή default, ενώ σε άλλες να θεωρούμε και τις δύο ισοδύναμες (true alternatives).

Μία ακόμη κατηγορία μεταβλητών είναι οι «*conditionally present variables*». Πρόκειται για μεταβλητές, η ύπαρξη των οποίων εξαρτάται από την τιμή κάποιας άλλης μεταβλητής. Για παράδειγμα αν τα αντικείμενα που μελετάμε είναι φυτά, τότε είναι δυνατό για ένα φυτό να υπάρχει η μεταβλητή «χρώμα πετάλων» μόνο αν η μεταβλητή «ύπαρξη πετάλων» έχει τιμή «ναι». Και σε αυτή την περίπτωση χρειάζεται προσοχή ως προς τον καθορισμό του βαθμού ομοιότητας ή διαφοράς ανάμεσα στα αντικείμενα. Στο παραπάνω παράδειγμα, θα θέλαμε πιθανόν ένα φυτό που έχει πέταλα κάποιου χρώματος να θεωρηθεί περισσότερο όμοιο με ένα άλλο το οποίο έχει επίσης πέταλα αλλά διαφορετικού χρώματος από ότι με ένα τρίτο που δεν έχει καθόλου πέταλα.

2.4 Μετρικές

Όπως έχει ήδη αναφερθεί, όταν κάνουμε ομαδοποίηση ή ταξινόμηση, η τοποθέτηση των αντικειμένων σε ομάδες γίνεται με τέτοιο τρόπο, ώστε τα αντικείμενα της ίδιας ομάδας να μοιάζουν όσο γίνεται περισσότερο μεταξύ τους, ενώ αντικείμενα που ανήκουν σε διαφορετικές ομάδες να διαφέρουν όσο γίνεται περισσότερο. Είναι προφανές λοιπόν ότι για να γίνει μια τέτοια διαδικασία είναι απαραίτητο προηγουμένως να έχει καθοριστεί ένας τρόπος για τον προσδιορισμό του βαθμού ομοιότητας ή διαφοράς ανάμεσα σε δύο αντικείμενα, δηλαδή μια *συνάρτηση απόστασης*. Έχοντας καθορίσει μια συνάρτηση απόστασης, βρίσκουμε την απόσταση κάθε αντικειμένου από κάθε άλλο και έτσι κατασκευάζουμε έναν πίνακα που περιέχει τιμές «ομοιότητας» ή «διαφοράς» για κάθε ζεύγος αντικειμένων. Η συνάρτηση που θα μας δίνει την τιμή της ομοιότητας ή διαφοράς για κάθε ζεύγος αντικειμένων θέλουμε συνήθως να ικανοποιεί τις σχέσεις:

1) $d_{ij} \geq 0$, για κάθε ζεύγος αντικειμένων i, j.
2) $d_{ii} = 0$, για κάθε αντικείμενο i.
3) $d_{ij} = d_{ji}$, για κάθε ζεύγος αντικειμένων i, j.

Αν πληρούνται αυτές οι προϋποθέσεις τότε οδηγούμαστε στην κατασκευή ενός πάνω ή κάτω τριγωνικού πίνακα που θα περιέχει συνολικά $n(n-1)/2$ στοιχεία, όπου n το πλήθος των αντικειμένων. Πρέπει να σημειωθεί ότι υπάρχουν κάποια προβλήματα στα οποία η τρίτη

συνθήκη δεν είναι επιθυμητό να ισχύει. Κατά τη διαδικασία της μετατροπής του αρχικού συνόλου δεδομένων σε έναν πίνακα ομοιοτήτων ή διαφορών είναι δυνατό να χαθεί πολύτιμη πληροφορία, συνεπώς είναι πολύ σημαντικό να δοθεί προσοχή στην κατάλληλη επιλογή της συνάρτησης που θα χρησιμοποιηθεί σε κάθε περίπτωση.

Ανάλογα με τον τύπο των μεταβλητών επιλέγουμε και τη συνάρτηση με βάση την οποία θα υπολογίσουμε το βαθμό διαφοράς ή ομοιότητας. Στην περίπτωση των binary variables, έστω ότι έχουμε δύο αντικείμενα που περιγράφονται από τέτοιου τύπου μεταβλητές, που έχουν μία από τις καταστάσεις «+» ή «-» και έστω ότι a μεταβλητές εμφανίζονται και στα δύο αντικείμενα στην κατάσταση «+», b μεταβλητές εμφανίζονται στο πρώτο αντικείμενο με την κατάσταση «+» και στο άλλο με «-», c μεταβλητές εμφανίζονται στο πρώτο με «-» και στο δεύτερο με «+» και τέλος d μεταβλητές εμφανίζονται και στα δύο με την κατάσταση «-». Δύο συχνά χρησιμοποιούμενες συναρτήσεις που δίνουν ένα μέτρο της ομοιότητας και της διαφοράς (με s συμβολίζουμε την ομοιότητα και με d τη διαφορά) είναι οι εξής:

1) Simple matching coefficient:

$$s = \frac{a+d}{a+b+c+d}, \qquad d = \frac{b+c}{a+b+c+d}$$

2) Jaccard's coefficient:

$$s = \frac{a}{a+b+c}, \qquad d = \frac{b+c}{a+b+c}$$

Η διαφορά ανάμεσα στις δύο αυτές συναρτήσεις είναι στον τρόπο με τον οποίο χειρίζονται την περίπτωση στην οποία μία δυαδική μεταβλητή εμφανίζονται και στα δύο αντικείμενα με την τιμή «-». Στις περιπτώσεις που οι δύο καταστάσεις είναι ισοδύναμες, είναι πιο κατάλληλο να χρησιμοποιήσουμε την πρώτη συνάρτηση, ενώ σε περιπτώσεις που θέλουμε να δώσουμε έμφαση σε μία από τις δύο καταστάσεις, είναι πιο κατάλληλο να επιλέξουμε τη δεύτερη.

Όταν έχουμε μία nominal variable, μπορούμε να την ανάγουμε σε δυαδικές μεταβλητές και να χρησιμοποιήσουμε μία από τις παραπάνω συναρτήσεις για τον υπολογισμό της ομοιότητας ή διαφοράς. Η αναγωγή μπορεί να γίνει με τον εξής τρόπο: Αν η μεταβλητή παίρνει τιμές από ένα σύνολο s διαφορετικών καταστάσεων, τότε μπορούμε να την αντικαταστήσουμε με s δυαδικές μεταβλητές, κάθε μια από τις οποίες θα αντιστοιχεί σε μία κατάσταση. Η δυαδική μεταβλητή που αντιστοιχεί στην κατάσταση την οποία είχε ως τιμή η αρχική μεταβλητή θα έχει την τιμή «+», ενώ όλες οι υπόλοιπες μεταβλητές θα έχουν την τιμή «-».

Μία αντίστοιχη προσέγγιση μπορεί να χρησιμοποιηθεί και όταν έχουμε ordinal variables. Μία ordinal variable με s καταστάσεις θα μπορούσε να αντικατασταθεί από s-1 binary variables που θα παίρνουν τιμές ως εξής: αν η αρχική μεταβλητή είχε ως τιμή την κατάσταση m, τότε οι δυαδικές μεταβλητές που αντιστοιχούν στις πρώτες m-1 καταστάσεις θα έχουν την τιμή «+», ενώ οι υπόλοιπες παίρνουν την τιμή «-».

Στην περίπτωση των numeric variables χρησιμοποιούνται συνήθως οι ακόλουθες συναρτήσεις για την έκφραση της διαφοράς ανάμεσα σε δύο αντικείμενα a και b:

- Euclidean distance: $d_{ab} = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}$

- Squared Euclidean distance: $d_{ab} = \sum_{i=1}^{n}(a_i - b_i)^2$
- City-block (Manhattan) distance: $d_{ab} = \sum_{i=1}^{n}|a_i - b_i|$
- Chebychev distance: $d_{ab} = \max|a_i - b_i|$

όπου n το πλήθος των μεταβλητών που περιγράφουν κάθε αντικείμενο. Αν θέλουμε να δώσουμε διαφορετική βαρύτητα στις διάφορες μεταβλητές, μπορούμε να γενικεύσουμε τους παραπάνω τύπους ως εξής:

- $d_{ab} = \sqrt{\sum_{i=1}^{n} w_i(a_i - b_i)^2}$
- $d_{ab} = \sum_{i=1}^{n} w_i(a_i - b_i)^2$
- $d_{ab} = \sum_{i=1}^{n} w_i|a_i - b_i|$
- $d_{ab} = \max(w_i|a_i - b_i|)$

όπου w κατάλληλα επιλεγμένα βάρη. Οι τρεις πρώτες συναρτήσεις είναι ειδικές περιπτώσεις της παρακάτω συνάρτησης (Minkowski metrics):

$$d_{ab} = \left\{\sum_{i=1}^{n} w_i|a_i - b_i|^{\lambda}\right\}^{1/\lambda}$$

Η κατάλληλη επιλογή της παραμέτρου λ εξαρτάται από την έμφαση που θέλουμε να δώσουμε σε μεγάλες διαφορές σε μία μεταβλητή. Όσο μεγαλύτερη είναι η τιμή του λ, τόσο περισσότερο επηρεάζεται το τελικό αποτέλεσμα από μία μεγάλη διαφορά στην τιμή μίας μεταβλητής. Ένας εναλλακτικός τρόπος για να προσδιορίσουμε τα κατάλληλα βάρη στην περίπτωση που οι μεταβλητές παίρνουν θετικές τιμές είναι να χρησιμοποιήσουμε τη συνάρτηση:

$$d_{ab} = \sum_{i=1}^{n}\frac{|a_i - b_i|}{a_i + b_i}$$

Επίσης μία συνάρτηση απόστασης που είναι ιδιαίτερα χρήσιμη, όταν έχουμε να κάνουμε με categorical variables, δηλαδή μεταβλητές που οι τιμές τους δεν είναι αριθμοί, είναι η εξής:

Percent disagreement: $\frac{number \quad of \quad a_i \neq b_i}{n}$

Τέλος μία πάρα πολύ σημαντική συνάρτηση απόστασης είναι η λεγόμενη Mahalanobis distance, η οποία οφείλει το όνομά της στον P. C. Mahalanobis, ο οποίος και την εισήγαγε το 1936. Η Mahalanobis distance για δύο αντικείμενα $x = (x_1, x_2, \ldots, x_n)$ και $y = (y_1, y_2, \ldots, y_n)$ ορίζεται ως εξής:

$$d_{xy} = \sqrt{(x-y)C^{-1}(x-y)^T}$$

όπου C ο αντίστοιχος covariance matrix. Αυτή η συνάρτηση απόστασης είναι πολύ σημαντική, διότι υπερέχει έναντι της Ευκλείδειας απόστασης σε δύο πολύ βασικά σημεία:

α) Λαμβάνει υπόψη τη μεταβλητότητα (variance) κάθε μεταβλητής. Όταν χρησιμοποιούμε την Ευκλείδεια απόσταση για τη σύγκριση δύο αντικειμένων, μεταχειριζόμαστε ισότιμα όλες τις μεταβλητές που περιγράφουν αυτά τα αντικείμενα. Ωστόσο συμβαίνει σε πολλές περιπτώσεις οι τιμές κάθε μεταβλητής να παρουσιάζουν μεγαλύτερες ή μικρότερες διακυμάνσεις από τις τιμές των υπόλοιπων μεταβλητών. Τότε είναι προτιμότερο όσο μεγαλύτερες διακυμάνσεις παρουσιάζουν οι τιμές μίας μεταβλητής τόσο λιγότερο να λαμβάνεται υπόψη η διαφορά των δύο αντικειμένων στη μεταβλητή αυτή και αντίστροφα. Δηλαδή αν δύο αντικείμενα διαφέρουν για παράδειγμα κατά 5 σε μία μεταβλητή της οποίας οι τιμές κυμαίνονται από 1 έως 100 δε θα δοθεί σε αυτό τόση έμφαση όση θα δίνονταν αν οι τιμές αυτής της μεταβλητής κυμαίνονταν από 1 έως 10.

β) Λαμβάνει υπόψη τις συσχετίσεις (correlation) που πιθανόν να υπάρχουν ανάμεσα στις μεταβλητές. Στην περίπτωση που κάποιες μεταβλητές είναι συσχετισμένες μεταξύ τους, είναι προτιμότερο να δίνεται μικρότερη έμφαση στις διαφορές που παρουσιάζουν τα δύο αντικείμενα σε αυτές τις μεταβλητές. Για παράδειγμα, αν δύο άνθρωποι διαφέρουν στο ύψος, τότε είναι φυσιολογικό ότι για να είναι εξίσου λεπτοί αναμένουμε να έχουν μία αντίστοιχη διαφορά και στο βάρος και όχι το ίδιο βάρος.

Το τίμημα που υπάρχει για τη χρησιμοποίηση της Mahalanobis distance είναι το υπολογιστικό κόστος. Ο υπολογισμός της Ευκλείδειας απόστασης μπορεί να γίνει αρκετά εύκολα, αλλά στην περίπτωση της Mahalanobis distance απαιτείται, όπως φαίνεται από τον παραπάνω τύπο, η εύρεση του covariance matrix και μάλιστα και του αντιστρόφου του. Σε περιπτώσεις που το πλήθος των μεταβλητών που χρησιμοποιούνται για την περιγραφή των αντικειμένων είναι σχετικά μικρό το υπολογιστικό κόστος δεν είναι ιδιαίτερα μεγάλο αλλά αυξάνει πολύ γρήγορα καθώς μεγαλώνει το πλήθος των μεταβλητών.

2.5 Κατηγορίες αλγορίθμων

Οι βασικές κατηγορίες αλγορίθμων για ομαδοποίηση αντικειμένων είναι οι εξής:

- partitioning methods
- hierarchical methods
- density-based methods
- grid-based methods
- model-based methods
- clumping methods
- geometrical methods

Στις partitioning methods ο σκοπός είναι να χωριστεί το σύνολο των αντικειμένων σε έναν καθορισμένο αριθμό ξένων μεταξύ τους ομάδων (partitions), δηλαδή έτσι ώστε κάθε αντικείμενο να ανήκει σε μία και μόνο ομάδα. Το πλήθος των ομάδων αυτών καθορίζεται από τον ερευνητή και συνήθως γίνονται δοκιμές για διάφορες τιμές αυτής της παραμέτρου, ώστε να βρεθεί η τιμή εκείνη που οδηγεί σε μία ταξινόμηση που είναι βέλτιστη ως προς κάποια κριτήρια. Μία δυνατότητα είναι να δοκιμάσουμε όλες τις δυνατές περιπτώσεις, ώστε να δούμε ποια είναι η βέλτιστη. Φυσικά κάτι τέτοιο έχει πολύ μεγάλο και συχνά

απαγορευτικό υπολογιστικό κόστος. Για το λόγο αυτό συχνά χρησιμοποιούνται heuristic methods, όπως ο αλγόριθμος k-means, στον οποίο κάθε cluster αντιπροσωπεύεται από το κέντρο του και ο k-medoids, στον οποίο κάθε cluster αντιπροσωπεύεται από ένα από τα μέλη του.

Μία δεύτερη κατηγορία είναι οι hierarchical methods. Συχνά είναι ενδιαφέρον να ερευνηθεί η δομή των δεδομένων σε διάφορα επίπεδα και πώς οι ομάδες σχετίζονται μεταξύ τους. Τέτοιες συσχετίσεις συνήθως αναπαρίστανται με τη βοήθεια διαγραμμάτων που έχουν τη μορφή δένδρου ή με φωλιασμένα σύνολα ομάδων (nested set of partitions). Όταν χωρίσουμε το δένδρο σε κάποιο επίπεδο, δημιουργούνται ξένες μεταξύ τους ομάδες. Το χαρακτηριστικό αυτών των μεθόδων είναι ότι οι ομάδες που προκύπτουν από το χωρισμό του δένδρου σε ένα συγκεκριμένο επίπεδο περιέχονται εξ ολοκλήρου στις ομάδες που προκύπτουν από το χωρισμό του δένδρου σε ένα υψηλότερο επίπεδο. Όσο ανεβαίνουμε στην ιεραρχία, δηλαδή όσο πλησιάζουμε προς τη ρίζα του δένδρου, τόσο περισσότερο αυξάνεται η γενικότητα, δηλαδή τόσο λιγότερο μοιάζουν μεταξύ τους τα αντικείμενα που ανήκουν στην ίδια ομάδα. Δηλαδή υπάρχει ένα tradeoff ανάμεσα στο πλήθος των ομάδων και στην ομοιογένειά τους. Οι ιεραρχικές μέθοδοι μας παρέχουν λοιπόν το πλεονέκτημα ότι δε χρειάζεται να αποφασίσουμε εκ των προτέρων για το πλήθος των clusters, όπως συμβαίνει στις partitioning methods, αλλά αντίθετα μπορούμε να φτιάξουμε μία ιεραρχία από ομάδες και μετά να επιλέξουμε το επίπεδο που μας ικανοποιεί περισσότερο τόσο ως προς το πλήθος των ομάδων όσο και ως προς την ομοιογένειά τους. Μπορούμε να διακρίνουμε τις hierarchical methods σε agglomerative και divisive. Στις agglomerative ξεκινάμε στο πρώτο βήμα έχοντας τόσα clusters όσο και το πλήθος των αντικειμένων που μελετάμε –κάθε cluster περιέχει ένα αντικείμενο– και σε κάθε βήμα συγχωνεύουμε τα clusters που απέχουν λιγότερο μεταξύ τους ώσπου να φτάσουμε τελικά σε ένα cluster που θα περιέχει όλα τα αντικείμενα. Στις divisive μεθόδους ακολουθούμε την αντίστροφη ακριβώς πορεία.

Στις density-based methods η διαδικασία βασίζεται σε «connectivity and density functions». Στις μεθόδους αυτές προσπαθούμε να δημιουργήσουμε clusters εντοπίζοντας «πυκνές» περιοχές αντικειμένων στο χώρο. Χρησιμοποιούμε δύο παραμέτρους: Eps, που είναι η μέγιστη ακτίνα μιας «γειτονιάς» και MinPts, που είναι ο ελάχιστος αριθμός σημείων στη γειτονιά ενός σημείου. Με βάση τις παραμέτρους αυτές ορίζουμε το σύνολο NEps(p) = {q belongs to D | dist(p,q) ≤ Eps}. Επίσης χρησιμοποιείται η εξής ορολογία: Ένα αντικείμενο p χαρακτηρίζεται ως «directly-density reachable» από το αντικείμενο q, αν α) το p ανήκει στο NEps(q) και β) NEps(q) ≥ MinPts. Η δεύτερη συνθήκη αναφέρεται ως «core point condition». Ένα αντικείμενο p χαρακτηρίζεται ως «density reachable» από το αντικείμενο q, αν υπάρχει μια αλυσίδα σημείων p_1, …, p_n, όπου p_1 = q και p_n = p τέτοια ώστε το σημείο p_{i+1} να είναι directly-density reachable από το p_i. Τέλος ένα αντικείμενο p χαρακτηρίζεται ως «density connected» στο αντικείμενο q, αν υπάρχει ένα σημείο r, τέτοιο ώστε τόσο το p όσο και το q να είναι density reachable από το r. Ένα cluster λοιπόν ορίζεται ως ένας μέγιστος αριθμός από density connected σημεία. Το πλεονέκτημα αυτών των μεθόδων είναι ότι μπορούν να ανακαλύψουν clusters τυχαίου σχήματος. Ένας γνωστός αλγόριθμος αυτής της κατηγορίας είναι ο DBSCAN.

Στις grid-based methods ο χώρος των αντικειμένων χωρίζεται σε ένα πεπερασμένο πλήθος κελιών (cells), σχηματίζοντας έτσι μια δομή με μορφή πλέγματος. Ορισμένοι ενδιαφέροντες αλγόριθμοι αυτής της κατηγορίας είναι οι STING, WaveCluster και CLIQUE.

Στις model-based methods υποθέτουμε ένα μοντέλο για κάθε cluster και προσπαθούμε να βρούμε το καλύτερο ταίριασμα (best fit) ανάμεσα στα δεδομένα και στο επιλεγμένο μοντέλο. Συχνά οι μέθοδοι αυτής της κατηγορίας χρησιμοποιούν τεχνολογία από τα πεδία της στατιστικής, της τεχνητής νοημοσύνης ή των νευρωνικών δικτύων. Γνωστοί αλγόριθμοι αυτής της κατηγορίας είναι οι Autoclass, Denclue, Cobweb κλπ.

Μία ακόμα κατηγορία είναι οι clumping methods. Στις παραπάνω περιπτώσεις είχαμε υποθέσει ότι κάθε αντικείμενο πρέπει να ανήκει σε μία και μόνο ομάδα. Υπάρχουν όμως πολλές περιπτώσεις στις οποίες ο περιορισμός αυτός δεν είναι επιθυμητός. Στις clumping methods επιτρέπεται να υπάρχει επικάλυψη μεταξύ των ομάδων, δηλαδή είναι επιτρεπτό ένα αντικείμενο να ανήκει σε περισσότερες από μία ομάδες. Σε τέτοιες περιπτώσεις μπορούμε να κάνουμε fuzzy clustering. Σε αυτές τις μεθόδους ο αριθμός των ομάδων g αποφασίζεται εκ των προτέρων και επίσης υπάρχει μια «membership function» y_{im}, που εκφράζει το βαθμό στον οποίο ένα αντικείμενο i ανήκει στην ομάδα m (βαθμός συμμετοχής - degree of belongingness). Υποθέτουμε ότι ισχύουν οι περιορισμοί $y_{im} \geq 0$ και $\sum_{m=1}^{g} y_{im} = 1$. Σκοπός είναι να προσδιορίσουμε τις βέλτιστες, ως προς κάποιο κριτήριο, τιμές για τα y_{im}.

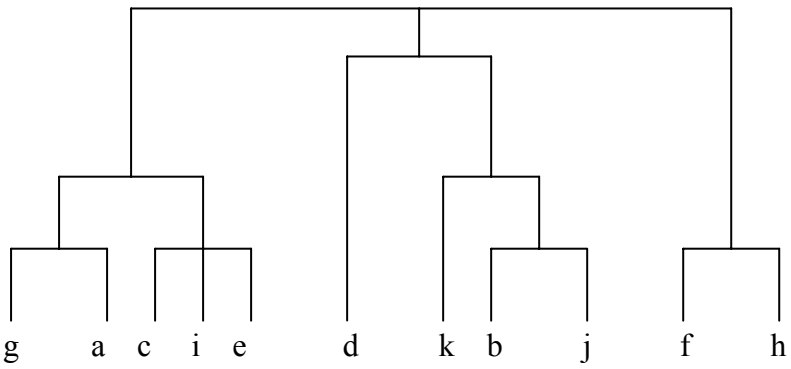
Στις geometrical methods κάθε αντικείμενο αναπαρίσταται από ένα σημείο στο δισδιάστατο ή τρισδιάστατο συνήθως χώρο, με τέτοιο τρόπο ώστε αντικείμενα που είναι παρόμοια μεταξύ τους να αναπαρίστανται από σημεία που είναι κοντά το ένα στο άλλο. Έτσι η οπτική αυτή αναπαράσταση καθιστά δυνατή την ανίχνευση της ύπαρξης ή μη ομάδων στα δεδομένα.

Όπως φαίνεται και από την παραπάνω περιγραφή, οι ιδιότητες των διάφορων αλγορίθμων ομαδοποίησης μπορούν να συνοψιστούν ως εξής:

- Hierarchical ή flat: Στην πρώτη περίπτωση ο αλγόριθμος οδηγεί στη δημιουργία μιας ιεραρχίας ομάδων, δηλαδή μιας πολυεπίπεδης δομής που μπορεί να αναπαρασταθεί με τη μορφή δένδρου (όσο πιο κοντά στη ρίζα του δένδρου βρισκόμαστε τόσο μεγαλύτερη είναι η γενικότητα), ενώ στη δεύτερη περίπτωση όλες οι ομάδες βρίσκονται σε ένα επίπεδο.
- Iterative: Ο αλγόριθμος ξεκινά με ένα αρχικό σύνολο από clusters και σε κάθε επανάληψη προσπαθεί να βελτιώσει τις ομάδες ανακατανέμοντας τα αντικείμενα.
- Hard ή soft: Στην πρώτη περίπτωση κάθε αντικείμενο ανήκει σε κάποια ομάδα με πιθανότητα 0 η 1 (δηλαδή είτε ανήκει είτε δεν ανήκει), ενώ στη δεύτερη περίπτωση επιτρέπονται και ενδιάμεσες τιμές.
- Disjunctive: Αν ο αλγόριθμος έχει αυτή την ιδιότητα, τότε κάθε αντικείμενο μπορεί να ανήκει σε περισσότερες από μία ομάδες.

Στα παρακάτω σχήματα δίνονται κάποια παραδείγματα, ώστε να γίνουν καλύτερα κατανοητές οι παραπάνω ιδιότητες:

I)



Σχήμα 3: 1ο παράδειγμα ομαδοποίησης

Η ομαδοποίηση του παραπάνω σχήματος έχει τις εξής ιδιότητες:
α) hard: Κάθε αντικείμενο ανήκει σε κάποια ομάδα με πιθανότητα 0 ή 1.
β) hierarchical: Υπάρχουν 4 επίπεδα. Στο πρώτο επίπεδο, ξεκινώντας από τη ρίζα του δένδρου, έχουμε 3 ομάδες, που περιέχουν τα αντικείμενα {g, a, c, i, e}, {d, k, b, j} και {f, h} αντίστοιχα. Στο δεύτερο επίπεδο έχουμε 4 ομάδες: {g, a, c, i, e}, {d}, {k, b, j} και {f, h}. Μπορούμε να παρατηρήσουμε ότι οι ομάδες {d} και {k, b, j} προήλθαν από τη διάσπαση της ομάδας {d, k, b, j}. Δηλαδή δεν μπορεί μία ομάδα ενός επιπέδου να περιέχει αντικείμενα από περισσότερες από μία ομάδες υψηλότερων επιπέδων (διαφορετικά δε θα είχαμε δενδρική δομή). Στο τρίτο επίπεδο έχουμε 6 ομάδες: {g, a}, {c, i, e}, {d}, {k}, {b, j} και {f, h}. Τέλος στο τέταρτο επίπεδο έχουμε 11 ομάδες, που η κάθε μία αποτελείται από ένα αντικείμενο.
γ) non-disjunctive: Κάθε αντικείμενο ανήκει σε μια και μόνο ομάδα.

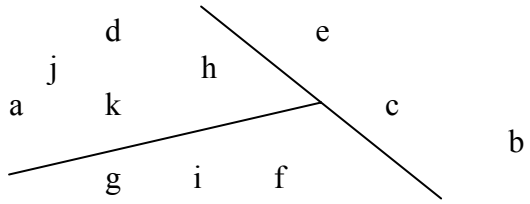
II)

	Ομάδα 1	***Ομάδα 2***	***Ομάδα 3***
a	0.4	0.1	0.5
b	0.1	0.8	0.1
c	0.3	0.3	0.4
d	0.1	0.1	0.8
e	0.4	0.2	0.4
f	0.1	0.4	0.5
g	0.7	0.2	0.1
h	0.5	0.4	0.1

Σχήμα 4: 2^{ο} παράδειγμα ομαδοποίησης

Η ομαδοποίηση του παραπάνω σχήματος έχει τις εξής ιδιότητες:
α) soft: Κάθε αντικείμενο ανήκει σε κάποια ομάδα με ορισμένη πιθανότητα που δεν παίρνει μόνο τις τιμές 0 ή 1. Για παράδειγμα, το αντικείμενο a ανήκει στην ομάδα 1 με πιθανότητα 40%, στην ομάδα 2 με πιθανότητα 10% και στην ομάδα 3 με πιθανότητα 50%. Αυτές οι πιθανότητες έχουν βέβαια άθροισμα 1.
β) flat: Δεν υπάρχει κάποια ιεραρχία από ομάδες.
γ) disjunctive: Κάθε αντικείμενο μπορεί να ανήκει σε περισσότερες από μία ομάδες.

III)

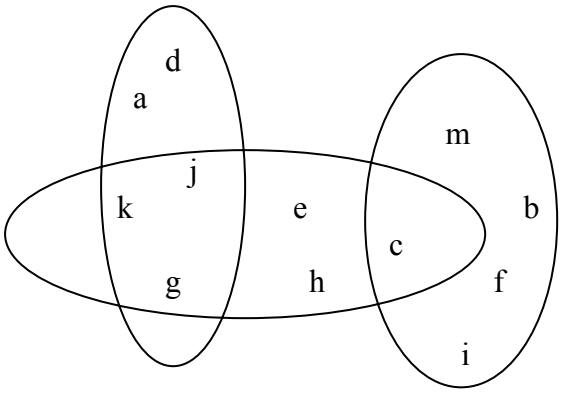


Σχήμα 5: 3^{ο} παράδειγμα ομαδοποίησης

Η ομαδοποίηση του παραπάνω σχήματος έχει τις εξής ιδιότητες:
α) hard: Κάθε αντικείμενο ανήκει σε κάποια ομάδα με πιθανότητα 0 ή 1.
β) flat: Δεν υπάρχει κάποια ιεραρχία από ομάδες.

γ) non-disjunctive: Κάθε αντικείμενο ανήκει σε μια και μόνο ομάδα.

IV)



Σχήμα 6: 4ο παράδειγμα ομαδοποίησης

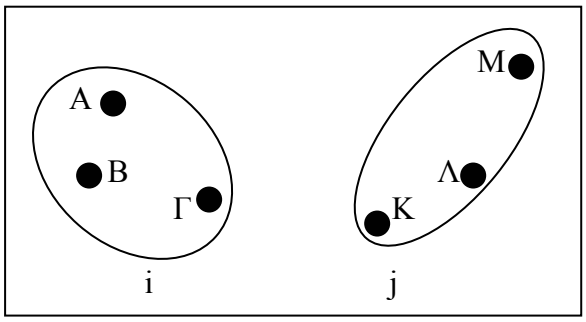
Η ομαδοποίηση του παραπάνω σχήματος έχει τις εξής ιδιότητες:
α) hard: Κάθε αντικείμενο ανήκει σε κάποια ομάδα με πιθανότητα 0 ή 1. Ακόμα και τα αντικείμενα που ανήκουν σε δυο ομάδες, δεν ανήκουν με πιθανότητα πχ. 50% στην κάθε μια αλλά με πιθανότητα 100% στην κάθε μια.
β) flat: Δεν υπάρχει κάποια ιεραρχία από ομάδες.
γ) disjunctive: Κάθε αντικείμενο μπορεί να ανήκει σε περισσότερες από μία ομάδες. Για παράδειγμα το αντικείμενο k ανήκει τόσο στην ομάδα που παριστάνεται στο σχήμα από την αριστερή κατακόρυφη έλλειψη όσο και στην ομάδα που παριστάνεται από την οριζόντια έλλειψη.

2.6 Κανόνες σύνδεσης

Όπως αναφέρθηκε και στην προηγούμενη παράγραφο, πολλοί αλγόριθμοι ομαδοποίησης λειτουργούν ως εξής: Ξεκινούν στο πρώτο βήμα θεωρώντας ότι κάθε αντικείμενο ανήκει στη δική του ομάδα, δηλαδή έχουμε αρχικά τόσες ομάδες όσα και τα αντικείμενο και σε κάθε επόμενο βήμα συγχωνεύουν τις ομάδες που απέχουν μεταξύ τους απόσταση μικρότερη από κάποιο όριο. Έτσι προοδευτικά αυξάνεται το πλήθος των αντικειμένων που περιέχουν οι ομάδες και μειώνεται ο αριθμός των ομάδων, ώσπου να φτάσουμε στην επιθυμητή τιμή. Είναι προφανές ότι αυτή η διαδικασία απαιτεί έναν τρόπο προσδιορισμού της απόστασης μεταξύ των ομάδων. Αυτό που έχουμε στη διάθεσή μας είναι μία συνάρτηση για τον προσδιορισμό της απόστασης μεταξύ δύο αντικειμένων. Μπορούμε να εκμεταλλευτούμε το γεγονός αυτό αν βρούμε έναν τρόπο να ανάγουμε την απόσταση μεταξύ δύο ομάδων σε απόσταση μεταξύ δύο αντικειμένων. Έτσι το πρόβλημα του προσδιορισμού της απόστασης δύο ομάδων ανάγεται στο πρόβλημα της επιλογής των αντικειμένων των δύο ομάδων των οποίων η απόσταση θα θεωρηθεί ως η απόσταση των ομάδων. Οι πιο συνηθισμένοι τρόποι επιλογής των αντικειμένων είναι οι εξής:

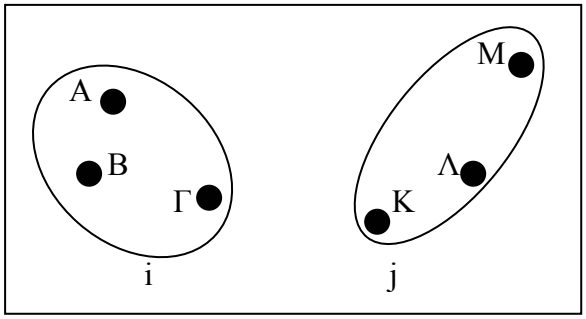
- Single linkage (nearest neighbor): Υπολογίζουμε την απόσταση κάθε αντικειμένου της μίας ομάδας από κάθε αντικείμενο της άλλης ομάδας και λαμβάνουμε ως

απόσταση των δύο ομάδων τη μικρότερη από αυτές τις αποστάσεις. Δηλαδή ως απόσταση των δύο ομάδων θεωρείται η απόσταση των δύο κοντινότερων αντικειμένων τους. Για παράδειγμα, η απόσταση των δύο ομάδων i και j στο παρακάτω σχήμα είναι, σύμφωνα με αυτή τη μέθοδο, η απόσταση των αντικει-μένων Γ και Κ. Η μέθοδος αυτή τείνει να δημιουργεί ομάδες που μοιάζουν με μακριές αλυσίδες.



Σχήμα 7: single linkage: $d_{ij} = d_{ΓΚ}$

- Complete linkage (furthest neighbor): Υπολογίζουμε την απόσταση κάθε αντικειμένου της μίας ομάδας από κάθε αντικείμενο της άλλης ομάδας και λαμβάνουμε ως απόσταση των δύο ομάδων τη μεγαλύτερη από αυτές τις αποστάσεις. Δηλαδή ως απόσταση των δύο ομάδων θεωρείται η απόσταση των δύο μακρινότερων αντικειμένων τους. Για παράδειγμα, η απόσταση των δύο ομάδων στο παρακάτω σχήμα είναι, σύμφωνα με αυτή τη μέθοδο, η απόσταση των αντικειμένων Β και Μ.



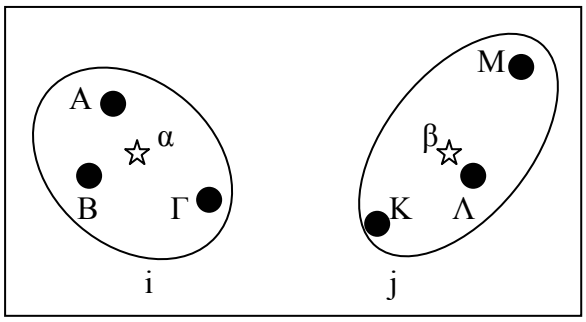
Σχήμα 8: complete linkage: $d_{ij} = d_{BM}$

- Average linkage ή Unweighted pair-group average: Υπολογίζουμε την απόσταση κάθε αντικειμένου της μίας ομάδας από κάθε αντικείμενο της άλλης ομάδας και λαμβάνουμε ως απόσταση των δύο ομάδων το μέσο όρο αυτών των αποστάσεων. Η μέθοδος αυτή μπορούμε να πούμε ότι συνδυάζει τις δύο προηγούμενες μεθόδους.

- Weighted pair-group average: Η μέθοδος αυτή είναι παραλλαγή της προηγούμενης μεθόδου. Είναι ίδια με εκείνη, με μόνη διαφορά ότι λαμβάνει υπόψη το μέγεθος των ομάδων, δηλαδή το πλήθος των αντικειμένων κάθε ομάδας χρησιμοποιείται ως βάρος στους υπολογισμούς. Για το λόγο αυτό η μέθοδος αυτή μπορεί να δώσει

καλύτερα αποτελέσματα σε περιπτώσεις που το πλήθος των αντικειμένων αλλάζει πολύ από ομάδα σε ομάδα.

- Unweighted pair-group centroid: Κάθε ομάδα αντιπροσωπεύεται από το «κέντρο βάρους» της οπότε η απόσταση δύο ομάδων είναι η απόσταση των κέντρων τους, όπως φαίνεται στο παρακάτω σχήμα.



Σχήμα 9: unweighted pair-group centroid: $d_{ij} = d_{\alpha\beta}$

- Weighted pair-group centroid (median): Όπως και προηγουμένως, η μέθοδος αυτή αποτελεί παραλλαγή της παραπάνω μεθόδου, ώστε να λαμβάνεται υπόψη το μέγεθος των ομάδων. Δηλαδή το πλήθος των αντικειμένων κάθε ομάδας χρησιμοποιείται και πάλι ως βάρος στους υπολογισμούς.

- Ward's method: Πρόκειται για μια μέθοδο αρκετά διαφορετική από τις προηγούμενες. Η γενική ιδέα εδώ είναι ότι η τοποθέτηση των αντικειμένων στις ομάδες γίνεται με τέτοιο τρόπο ώστε να ελαχιστοποιείται το άθροισμα των τετραγώνων των αποκλίσεων των τιμών των αντικειμένων από το μέσο όρο της ομάδας.

2.7 Ο αλγόριθμος k-means

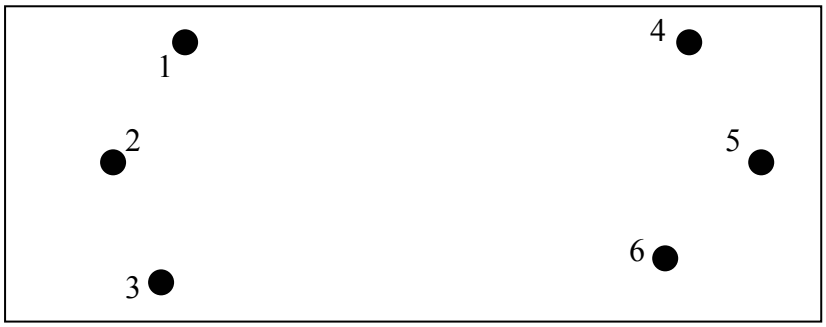
Ο αλγόριθμος k-means ανήκει στην κατηγορία των partitioning methods –και συγκεκριμένα στις heuristic methods. Το βασικό του χαρακτηριστικό είναι ότι κάθε cluster αντιπροσωπεύεται από το «κέντρο» του. Με δεδομένο το πλήθος k των ομάδων, ο αλγόριθμος περιλαμβάνει 4 στάδια:

1) Χωρίζουμε το σύνολο των αντικειμένων σε k μη κενά υποσύνολα.
2) Χρησιμοποιώντας μια συνάρτηση μέσης τιμής προσδιορίζουμε το κέντρο κάθε cluster.
3) Επανατοποθετούμε κάθε αντικείμενο σε εκείνο το cluster, στο κέντρο του οποίου βρίσκεται πιο κοντά.
4) Επαναλαμβάνουμε τα βήματα (2) και (3) μέχρι να μην υπάρχουν πλέον άλλες επανατοποθετήσεις.

Όπως φαίνεται από τα παραπάνω βήματα, κάθε cluster που δημιουργείται με αυτή την προσέγγιση έχει ένα *centroid* (κέντρο). Ένα centroid είναι ένα φανταστικό αντικείμενο που ανήκει στο cluster και οι τιμές των συντεταγμένων του είναι ο μέσος όρος των τιμών των συντεταγμένων όλων των αντικειμένων που ανήκουν στο cluster. Έτσι, όταν θέλουμε να αποφασίσουμε σε ποιο cluster θα τοποθετηθεί ένα καινούριο αντικείμενο, υπολογίζουμε την απόσταση αυτού του αντικειμένου από κάθε centroid και τοποθετούμε το αντικείμενο σε εκείνο το cluster, του οποίου το centroid ήταν το πλησιέστερο στο αντικείμενο.

Επίσης πρέπει να επισημάνουμε ότι απαιτείται ο εκ των προτέρων προσδιορισμός του πλήθους των ομάδων. Για το λόγο αυτό είναι χρήσιμο να έχουμε από πριν μια ιδέα ή υπόθεση για το ποιο θα ήταν το κατάλληλο πλήθος ομάδων. Διαφορετικά θα πρέπει να θεωρήσουμε το πλήθος των ομάδων ως παράμετρο και να κάνουμε διάφορες δοκιμές μέχρι να καταλλήξουμε σε μια ικανοποιητική ομαδοποίηση.

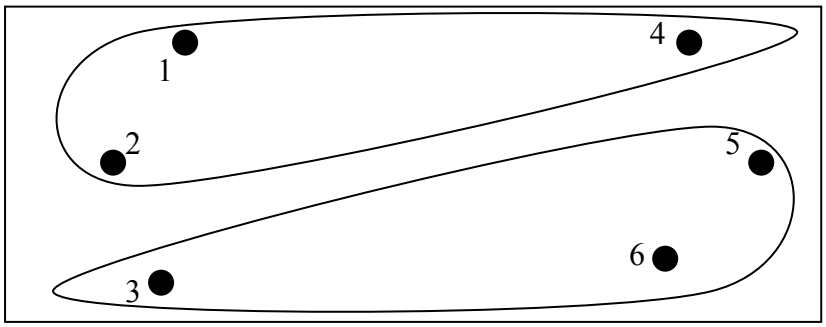
Για παράδειγμα, έστω ότι έχουμε 6 αντικείμενα, που παριστάνονται στο παρακάτω σχήμα με τα σημεία 1, 2, 3, 4, 5 και 6 και θέλουμε να τα χωρίσουμε σε k = 2 ομάδες. Υποθέτουμε ότι η τοποθέτηση των σημείων στο σχήμα έχει γίνει έτσι ώστε η απόσταση δύο σημείων να αποτελεί μέτρο του βαθμού ομοιότητάς τους, δηλαδή όσο περισσότερο απέχουν δύο σημεία τόσο πιο διαφορετικά είναι τα αντικείμενα που αντιπροσωπεύουν και αντίστροφα.



Σχήμα 10: Παράδειγμα k-means, αρχικά σημεία

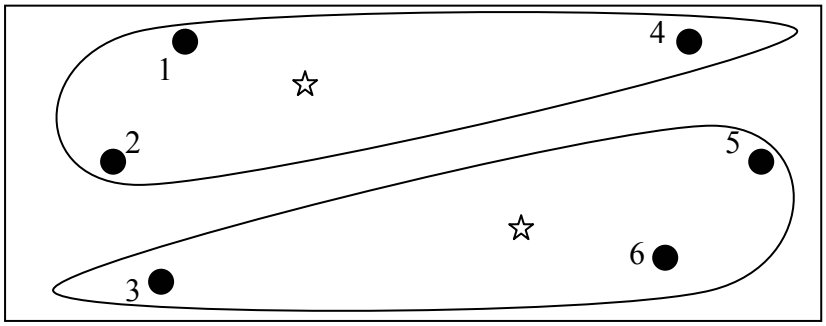
Σύμφωνα με την παραπάνω περιγραφή του αλγορίθμου, η ομαδοποίηση μπορεί να επιτευχθεί ακολουθώντας τα παρακάτω βήματα:

Βήμα 1: Θέλουμε να δημιουργήσουμε 2 ομάδες, οπότε χωρίζουμε τα αντικείμενα σε δύο υποσύνολα με τυχαίο τρόπο. Έστω ότι επιλέγουμε για τη μία ομάδα τα αντικείμενα 1, 2 και 4 και για τη δεύτερη τα αντικείμενα 3, 5 και 6. Τότε το σχήμα αποκτά την παρακάτω μορφή:



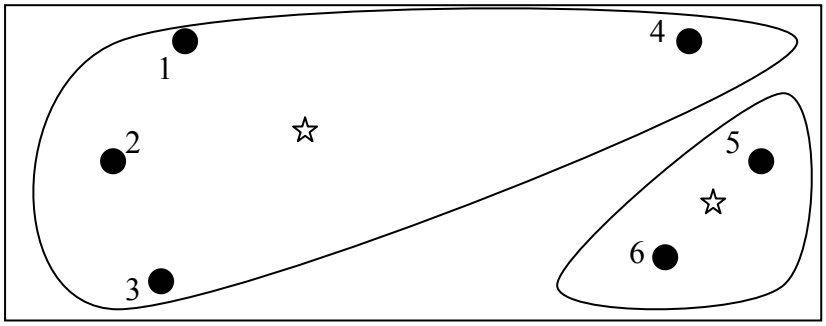
Σχήμα 11: Παράδειγμα k-means, βήμα 1

Βήμα 2: Υπολογίζουμε το κέντρο κάθε ομάδας (σημειώνεται στο σχήμα με το αστεράκι):



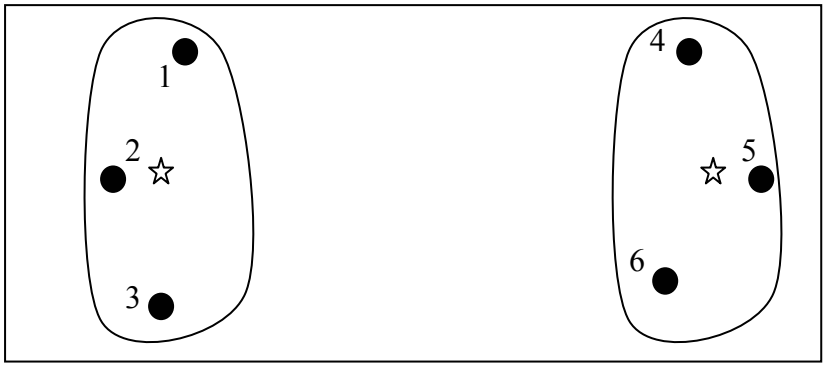
Σχήμα 12: Παράδειγμα k-means, βήμα 2

Βήμα 3: Τα αντικείμενα 1 και 2 βρίσκονται πλησιέστερα στο κέντρο της δικής τους ομάδας από ότι στο κέντρο της άλλης ομάδας, οπότε παραμένουν στην ομάδα αυτή. Ωστόσο το αντικείμενο 3 βρίσκεται πλησιέστερα στο κέντρο της ομάδας {1, 2, 4} από ότι στο κέντρο της δικής του ομάδας, επομένως το τοποθετούμε στην άλλη ομάδα και υπολογίζουμε τα νέα κέντρα:



Σχήμα 13: Παράδειγμα k-means, βήμα 3

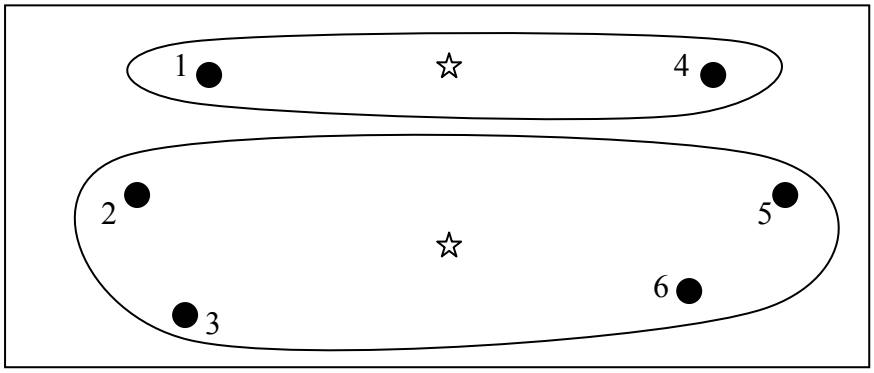
Βήμα 4: Το αντικείμενο 4 βρίσκεται πλησιέστερα στο κέντρο της ομάδας {5, 6} από ότι στο κέντρο της δικής του ομάδας, επομένως το τοποθετούμε στην ομάδα αυτή και υπολογίζουμε τα νέα κέντρα:



Σχήμα 14: Παράδειγμα k-means, βήμα 4

Βήμα 5: Τώρα κάθε αντικείμενο βρίσκεται πλησιέστερα στο κέντρο της δικής του ομάδας από ότι στο κέντρο της άλλης ομάδας, επομένως δε χρειάζονται άλλες επανατοποθετήσεις και η διαδικασία έχει ολοκληρωθεί.

Χρειάζεται μεγάλη προσοχή στο γεγονός ότι η αρχική επιλογή των ομάδων μπορεί να επηρεάσει σημαντικά την τελική έκβαση της διαδικασίας. Για να φανεί αυτό, θεωρούμε και πάλι το σχήμα του παραπάνω παραδείγματος, αλλά έστω ότι τώρα ξεκινάμε με τις ομάδες {1, 4} και {2, 3, 5, 6} αντί για τις ομάδες {1, 2, 4} και {3, 5, 6} που χρησιμοποιήσαμε προηγουμένως. Δηλαδή το σχήμα έχει αρχικά τη μορφή:



Σχήμα 15: Παράδειγμα k-means, εναλλακτικό βήμα 1

Τώρα κάθε αντικείμενο βρίσκεται πλησιέστερα στο κέντρο της δικής του ομάδας από ότι στο κέντρο της άλλης ομάδας, επομένως δε χρειάζονται επανατοποθετήσεις και η διαδικασία έχει ήδη ολοκληρωθεί, δηλαδή παίρνουμε τις ομάδες {1, 4} και {2, 3, 5, 6}. Ωστόσο η πρώτη ομαδοποίηση που επιτύχαμε είναι προφανώς καλύτερη από τη δεύτερη, με την έννοια ότι οι ομάδες στην πρώτη περίπτωση είναι και πιο ομοιογενείς και πιο απομακρυσμένες από ότι στη δεύτερη.

Τα πλεονεκτήματα του αλγορίθμου k-means είναι ότι είναι αρκετά αποδοτικός (efficient) και συχνά τερματίζει σε μια τοπικά βέλτιστη λύση (local optimum). Μπορεί μάλιστα να βρεθεί και global optimum χρησιμοποιώντας ορισμένες τεχνικές, όπως deterministic annealing και genetic algorithms.

Μειονεκτήματα του αλγορίθμου είναι ότι απαιτεί τον προσδιορισμό μίας συνάρτησης μέσης τιμής για τον υπολογισμό του κέντρου κάθε cluster (επομένως υπάρχει επιπρόσθετη δυσκολία σε περιπτώσεις που δεν έχουμε numeric data αλλά categorical), απαιτεί τον εκ των προτέρων προσδιορισμό του πλήθους k των clusters και παρουσιάζει δυσκολίες στην αντιμετώπιση του θορύβου και των outliers.

Μία παραλλαγή του k-means είναι ο αλγόριθμος k-medoids, στον οποίο το κάθε cluster αντιπροσωπεύεται όχι από το κέντρο του αλλά από κάποιο αντικείμενο του cluster. Στον αλγόριθμο αυτό το πρώτο βήμα είναι για κάθε cluster να επιλέξουμε με κάποιο τρόπο ένα αντιπροσωπευτικό αντικείμενο, που καλείται medoid. Ξεκινώντας από αυτό το αρχικό σύνολο από medoids, σε κάθε βήμα αντικαθιστούμε κάποιο medoid με ένα άλλο σημείο του cluster, αν με τον τρόπο αυτό βελτιώνεται η συνολική απόσταση των παραγόμενων clusters. Η μέθοδος αυτή λειτουργεί καλά για μικρά σύνολα δεδομένων αλλά δεν αποδίδει πολύ καλά όταν το πλήθος των δεδομένων αυξάνει πολύ.

Μία άλλη παραλλαγή του k-means είναι ο fuzzy c-means. Η διαφορά τους είναι ότι ο k-means χρησιμοποιείται για hard clustering, ενώ ο fuzzy c-means για soft clustering.

Δηλαδή, ενώ ο k-means προσπαθεί να προσδιορίσει σε ποια ομάδα θα τοποθετηθεί κάθε αντικείμενο, ο fuzzy c-means προσδιορίζει το βαθμό στον οποίο κάθε αντικείμενο ταιριάζει σε κάθε μία από τις ομάδες.

2.8 Ο αλγόριθμος DBSCAN

Ο αλγόριθμος DBSCAN ανήκει στην κατηγορία των density-based methods και χρησιμοποιεί τις παραμέτρους και τους όρους που είχαν αναφερθεί σε προηγούμενη ενότητα για τις μεθόδους αυτής της κατηγορίας. Η διαδικασία περιλαμβάνει τα εξής βήματα:

1. Επιλέγουμε τυχαία ένα σημείο p.
2. Υπολογίζουμε όλα τα σημεία που είναι density-reachable από το p.
3. Αν το p είναι core point, δηλαδή ικανοποιεί τη συνθήκη «core point condition» που είχαμε ορίσει στην αντίστοιχη παράγραφο προηγούμενης ενότητας, τότε έχουμε δημιουργήσει ένα cluster. Αντίθετα αν δεν υπάρχουν αρκετά σημεία που να είναι density-reachable από το p, τότε το p είναι border point και η διαδικασία συνεχίζεται με το επόμενο σημείο από το σύνολο που διαθέτουμε.
4. Επαναλαμβάνουμε τα παραπάνω βήματα μέχρι να εξετάσουμε όλα τα σημεία της βάσης δεδομένων.

2.9 Αξιολόγηση μιας ομαδοποίησης

Όπως έχει γίνει φανερό από όσα περιγράφηκαν μέχρι αυτό το σημείο, δεν υπάρχει μία μοναδική προσέγγιση, ένας μοναδικός αλγόριθμος, που να δίνει κάθε φορά την καλύτερη δυνατή ομαδοποίηση των διαθέσιμων αντικειμένων. Αντίθετα υπάρχει ένα πλήθος τεχνικών που μπορεί κανείς να επιλέξει και οι οποίες ανάλογα με τη φύση και τις απαιτήσεις του προβλήματος και των δεδομένων είναι δυνατό να οδηγήσουν σε περισσότερο ή λιγότερο ικανοποιητικά αποτελέσματα. Συνεπώς χρειάζεται συνήθως να πειραματιστεί κανείς αρκετά και να δοκιμάσει διάφορες μεθόδους. Το πρόβλημα λοιπόν που ανακύπτει είναι πώς μπορεί κάποιος να συγκρίνει τα αποτελέσματα που θα προκύψουν από διαφορετικούς αλγόριθμους, δηλαδή ποιο είναι το κριτήριο για να πούμε ότι μία ομαδοποίηση είναι καλύτερη από μία άλλη.

Συνήθως υπάρχουν δύο ειδών τεχνικές που χρησιμοποιούνται για να αξιολογηθεί το πόσο καλή είναι μία ομαδοποίηση. Οι τεχνικές του πρώτου είδους μπορούν να χρησιμοποιθούν για τη σύγκριση δύο ομαδοποιήσεων χωρίς να απαιτείται αναφορά σε εξωτερική γνώση –γι' αυτό οι τεχνικές αυτού του τύπου αναφέρονται ως internal quality measure. Όπως έχει ήδη αναφερθεί, ο σκοπός κατά τη διαδικασία της ομαδοποίησης είναι ο χωρισμός των αντικειμένων σε ομάδες με τέτοιο τρόπο ώστε τα αντικείμενα που ανήκουν στην ίδια ομάδα να μοιάζουν όσο το δυνατό περισσότερο μεταξύ τους, ενώ τα αντικείμενα που ανήκουν σε διαφοερτικές ομάδες να διαφέρουν όσο το δυνατό περισσότερο. Ακριβώς αυτό είναι το κριτήριο που χρησιμοποιούν οι τεχνικές αυτού του τύπου για να αξιολογήσουν πόσο καλή είναι μία ομαδοποίηση. Δηλαδή αν έχουμε ένα σύνολο αντικειμένων και

καταφέρουμε με χρήση δύο διαφορετικών αλγορίθμων να τα χωρίσουμε σε ομάδες με δύο διαφορετικούς τρόπους, τότε καλύτερος τρόπος μπορεί να θεωρηθεί εκείνος που ικανοποιεί σε μεγαλύτερο βαθμό το παραπάνω κριτήριο.

Αντίθετα στη δεύτερη περίπτωση απαιτείται εξωτερική γνώση (external quality measure), δηλαδή μπορούμε να αξιολογήσουμε μία ομαδοποίηση αν γνωρίζουμε ήδη με κάποιο τρόπο σε ποια κατηγορία ανήκει –ή σε ποια κατηγορία θα θέλαμε να ανήκει– κάθε αντικείμενο. Έτσι εφόσον έχουμε κάποιο εξωτερικό κριτήριο που μπορεί να καθορίσει σε ποια ομάδα είναι ορθό να τοποθετηθεί κάθε αντικείμενο, μπορούμε να αξιολογήσουμε την επίδοση της μεθόδου βλέποντας απλά σε ποιο βαθμό απέδωσε τα αναμενόμενα αποτελέσματα. Δύο συχνά χρησιμοποιούμενες external measures είναι η entropy και η F-measure.

Η entropy λειτουργεί ως εξής: Έστω CS μία ομαδοποίηση (clustering solution). Για κάθε ομάδα j που έχει σχηματιστεί, υπολογίζουμε τις πιθανότητες p_{ij}, οι οποίες είναι οι πιθανότητες κάποιο αντικείμενο της ομάδας αυτής να ανήκει στην πραγματικότητα σε κάποια άλλη ομάδα i. Έτσι η εντροπία κάθε ομάδας υπολογίζεται από τη σχέση:

$$E_j = -\sum_i p_{ij} \log(p_{ij})$$

Συνεπώς η συνολική εντροπία για αυτή την ομαδοποίηση, αν λάβουμε υπόψη και το πλήθος των αντικειμένων που περιέχει κάθε ομάδα, δίνεται από τη σχέση:

$$E_{CS} = \sum_{j=1}^{m} \frac{n_j \cdot E_j}{n}$$

όπου n_j είναι το πλήθος των αντικειμένων που περιέχονται στην ομάδα j, m είναι το πλήθος των ομάδων και n ο συνολικός αριθμός των αντικειμένων. Όσο μεγαλύτερη είναι η εντροπία μίας ομαδοποίησης τόσο καλύτερη αυτή θεωρείται.

Η F-measure χρησιμοποιεί τις έννοιες recall και precision από το χώρο της information retrieval. Έστω ότι έχουμε κάνει μία ομαδοποίηση και

- α είναι το πλήθος των αντικειμένων που ορθά έχουν τοποθετηθεί στην ομάδα i
- b είναι το πλήθος των αντικειμένων που έχουν τοποθετηθεί στην ομάδα i, χωρίς όμως κανονικά να ανήκουν σε αυτή
- c είναι το πλήθος των αντικειμένων που δεν έχουν τοποθετηθεί στην ομάδα i, αλλά κανονικά ανήκουν σε αυτή
- d είναι το πλήθος των αντικειμένων που ορθά δεν έχουν τοποθετηθεί στην ομάδα i

Μπορούμε τώρα να ορίσουμε τις εξής έννοιες:

- Recall = $\frac{a}{a+c}$
- Precision = $\frac{a}{a+b}$
- Accuracy = $\frac{a+d}{a+b+c+d}$

Δηλαδή

- η Recall απαντάει στο ερώτημα «Πόσα από τα αντικείμενα που ανήκουν κανονικά στην ομάδα i τοποθετήθηκαν πράγματι σε αυτή;»

- η Precision απαντάει στο ερώτημα «Πόσα από τα αντικείμενα που τοποθετήσαμε στην ομάδα i ανήκουν πράγματι σε αυτή;»
- η Accuracy απαντάει στο ερώτημα «Για πόσα από τα αντικείμενα που υπάρχουν συνολικά λήφθηκε σωστή απόφαση ως προς το αν ανήκουν ή όχι στην ομάδα i;»

Η F-measure συνυπολογίζει τις ποσότητες recall και precision μέσω της σχέσης:

$$F = \frac{2 \cdot \text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}}$$

Μέρος 3ο: Μαρκοβιανές Αλυσίδες

3.1 Εισαγωγή

Στο χώρο της θεωρίας πιθανοτήτων και της στατιστικής είναι συχνά βολικό να σκεφτόμαστε ότι ένα πείραμα και οι πιθανές εκβάσεις του (ή όμοια ένα σύστημα και οι πιθανές καταστάσεις του) ορίζουν ένα χώρο και τα σημεία του. Σε κάθε στοιχειώδη δυνατή έκβαση του πειράματος μπορούμε να αντιστοιχήσουμε ένα σημείο που ονομάζεται δείγμα (sample point) και το συμβολίζουμε με s. Το σύνολο των δειγμάτων {s}, που αποτελεί ουσιαστικά το σύνολο όλων των δυνατών εκβάσεων του πειράματος, ονομάζεται χώρος δειγμάτων (sample space) και μπορούμε να το συμβολίζουμε με S.. Ας θεωρήσουμε για παράδειγμα ένα απλό πείραμα, που περιλαμβάνει τη ρίψη ενός ζαριού. Υπάρχουν 6 πιθανά αποτελέσματα, δηλαδή να προκύψει ο αριθμός 1, 2, 3, 4, 5 ή 6. Αντιστοιχώντας ένα δείγμα σε κάθε μία από αυτές τις δυνατές εκβάσεις του πειράματος, έχουμε ένα χώρο δειγμάτων που αποτελείται από 6 δείγματα. Ένα γεγονός μπορεί να αντιστοιχεί σε ένα μοναδικό δείγμα ή σε ένα σύνολο δειγμάτων. Για παράδειγμα, το γεγονός «εμφάνιση του αριθμού 2» αντιστοιχεί σε ένα μοναδικό δείγμα, ενώ το γεγονός «εμφάνιση άρτιου αριθμού» αντιστοιχεί σε ένα σύνολο τριών δειγμάτων: {«εμφάνιση 2», «εμφάνιση 4», «εμφάνιση 6»}. Μπορούμε να θεωρήσουμε την έκβαση ενός τέτοιου πειράματος σαν μία μεταβλητή που μπορεί να «περιφέρεται» στο σύνολο των δειγμάτων και της οποίας η τιμή καθορίζεται από το πείραμα. Μία συνάρτηση της οποίας το πεδίο ορισμού είναι ένας χώρος δειγμάτων και το πεδίο τιμών είναι κάποιο σύνολο πραγματικών αριθμών ονομάζεται *τυχαία μεταβλητή* (*random variable*). Έτσι, όταν η έκβαση του πειράματος είναι s, η τυχαία μεταβλητή συμβολίζεται με X(s) ή απλά X. Στο παραπάνω παράδειγμα της ρίψης ζαριού, αν το δείγμα k παριστάνει το γεγονός να εμφανιστεί ο αριθμός k κατά τη ρίψη του ζαριού, η συνάρτηση X(k)=k είναι μία τυχαία μεταβλητή, που η τιμή της ισούται με τον αριθμό που εμφανίζεται κάθε φορά που ρίπτεται το ζάρι. Στο συγκεκριμένο παράδειγμα η τυχαία μεταβλητή λαμβάνει μόνο ένα διακριτό σύνολο τιμών. Γι' αυτό το λόγο λέμε ότι αυτή η τυχαία μεταβλητή είναι μία διακριτή τυχαία μεταβλητή (*discrete random variable*). Δηλαδή μία τυχαία μεταβλητή X λέγεται διακριτή τυχαία μεταβλητή, εάν η X μπορεί να πάρει μόνο ένα πεπερασμένο αριθμό τιμών σε οποιοδήποτε πεπερασμένο διάστημα παρατήρησης. Αντίθετα, αν μία τυχαία μεταβλητή X μπορεί να πάρει οποιαδήποτε τιμή σε ένα διάστημα παρατήρησης, τότε ονομάζεται συνεχής τυχαία μεταβλητή (*continuous random variable*).

Ας υποθέσουμε τώρα ότι έχουμε να μελετήσουμε διάφορα τυχαία σήματα, όπως σήματα φωνής, σήματα τηλεόρασης κλπ. Τέτοιου είδους σήματα έχουν δύο ιδιότητες: α) είναι χρονικές συναρτήσεις που ορίζονται σε κάποιο διάστημα παρατήρησης και β) είναι τυχαία, με την έννοια ότι πριν τη διενέργεια ενός πειράματος δεν είναι δυνατό να περιγράψουμε ακριβώς τις κυματομορφές που θα παρατηρηθούν. Συνεπώς, περιγράφοντας τυχαία σήματα, βρίσκουμε ότι κάθε δείγμα στο χώρο δειγμάτων μας είναι μία χρονική συνάρτηση. Ο χώρος δειγμάτων ή το σύνολο που περιλαμβάνει τις χρονικές συναρτήσεις ονομάζεται *τυχαία διαδικασία ή στοχαστική ανέλιξη* (*random or stochastic process*). Η διαφορά δηλαδή ανάμεσα σε μία τυχαία μεταβλητή και μία στοχαστική ανέλιξη είναι ότι για μια τυχαία μεταβλητή η έκβαση ενός πειράματος απεικονίζεται με έναν αριθμό, ενώ για μια στοχαστική ανέλιξη η έκβαση ενός πειράματος απεικονίζεται με μία χρονική συνάρτηση.

3.2 Μαρκοβιανές αλυσίδες διακριτού χρόνου

Έχοντας περιγράψει τι ονομάζουμε τυχαίες μεταβλητές και στοχαστικές ανελίξεις, μπορούμε να μιλήσουμε για Μαρκοβιανές αλυσίδες. Μία *Μαρκοβιανή αλυσίδα* (*Markov chain*) (η ονομασία προέρχεται από τον Ρώσο μαθηματικό Andrei Markov) είναι μία στοχαστική ανέλιξη (stochastic process) η οποία ικανοποιεί επιπλέον την εξής ιδιότητα: Η μελλοντική εξέλιξη του φαινομένου εξαρτάται από το παρελθόν μόνο μέσω της παρούσας κατάστασης. Δηλαδή η πιθανότητα του συστήματος να βρεθεί σε κάποια κατάσταση στο μέλλον εξαρτάται μόνο από την κατάσταση στην οποία βρίσκεται το σύστημα τώρα και όχι από το πώς έφτασε σε αυτή. Συνεπώς η γνώση της πιο πρόσφατης κατάστασης του συστήματος καθιστά άχρηστη τη γνώση προηγούμενων καταστάσεων. Γι' αυτό λέμε ότι το σύστημα είναι «χωρίς μνήμη» (*memoryless*). Η ιδιότητα αυτή είναι γνωστή ως Markov property (Μαρκοβιανή ιδιότητα).

Αρχικά θα μιλήσουμε για Μαρκοβιανές αλυσίδες διακριτού χρόνου (*Discrete Time Markov Chains - DTMC*). Στην περίπτωση αυτή η αλυσίδα αποτελείται από μία ακολουθία τυχαίων μεταβλητών X_1, X_2, X_3, ..., οι οποίες λαμβάνουν τιμές από κάποιο χώρο καταστάσεων S (state space). Τόσο ο χώρος καταστάσεων όσο και οι χρονικές στιγμές παρατήρησης έχουν διακριτές τιμές (discrete time range και discrete state space). Με X_n συμβολίζουμε την κατάσταση του συστήματος τη στιγμή n. Έτσι η Μαρκοβιανή ιδιότητα μπορεί να γραφτεί στην περίπτωση αυτή ως εξής:

$$\Pr\{X_{n+1} = x_{n+1} \mid X_n = x_n, X_{n-1} = x_{n-1}, \ldots, X_0 = x_0 \} = \Pr\{X_{n+1} = x_{n+1} \mid X_n = x_n \}$$

όπου με Pr{ A | B } συμβολίζουμε την πιθανότητα να συμβεί το γεγονός A δεδομένου ότι έχει συμβεί το γεγονός B (υπό συνθήκη πιθανότητα).

Μία παρατήρηση που αξίζει να σημειωθεί είναι η εξής: Η πιθανότητα να βρίσκεται το σύστημα σε μία κατάσταση i κάποια στιγμή t δεν είναι απαραίτητα ανεξάρτητη από τη χρονική στιγμή t που παρατηρούμε το σύστημα. Δηλαδή είναι δυνατό η τιμή της X_n να εξαρτάται από το n. Αν συμβαίνει αυτό, η Μαρκοβιανή διαδικασία λέγεται μη-ομογενής (inhomogeneous). Αντίθετα αν η πιθανότητα να βρίσκεται το σύστημα σε κάποια κατάσταση είναι ανεξάρτητη από τη στιγμή της παρατήρησης, τότε η Μαρκοβιανή διαδικασία λέγεται ομογενής (homogeneous). Στις Μαρκοβιανές αλυσίδες δεχόμαστε ότι ισχύει η δεύτερη περίπτωση.

Αν με p_{ij} συμβολίζουμε την πιθανότητα να μεταβεί το σύστημα από την κατάσταση i στην κατάσταση j, δηλαδή

$$p_{ij} = \Pr\{X_{n+1} = j \mid X_n = i \}, i, j \in S$$

τότε ο πίνακας P με στοιχεία τα p_{ij} λέγεται *πίνακας μεταβάσεων* (*transition matrix*) του συστήματος. Δηλαδή κάθε στοιχείο p_{ij} αυτού του πίνακα απαντά στο ερώτημα: «Έστω ότι το σύστημα βρίσκεται στην κατάσταση i. Ποια είναι η πιθανότητα το σύστημα να βρεθεί μετά από 1 μετάβαση στην κατάσταση j;». Προφανώς ο P είναι ένας τετραγωνικός N x N πίνακας, όπου N το πλήθος των καταστάσεων στις οποίες είναι δυνατό να βρεθεί το σύστημα.

Αντίστοιχη σημασία όμως έχουν και οι διάφορες δυνάμεις αυτού του πίνακα. Έτσι, για παράδειγμα, τα στοιχεία του P^2, που μπορούμε να τα συμβολίσουμε με $p_{ij}^{(2)}$ δίνουν τις πιθανότητες για μεταβάσεις 2 βημάτων, δηλαδή

$$p_{ij}^{(2)} = \Pr\{X_{n+2} = j \mid X_n = i \}, i, j \in S$$

ή με πιο απλά λόγια απαντούν στο ερώτημα: «Έστω ότι το σύστημα βρίσκεται στην κατάσταση i. Ποια είναι η πιθανότητα το σύστημα να βρεθεί μετά από 2 μεταβάσεις στην κατάσταση j;». Γενικεύοντας, η k-οστή δύναμη του P εκφράζει τις πιθανότητες για μεταβάσεις k βημάτων, δηλαδή

$$p_{ij}^{(k)} = \Pr\{X_{n+k} = j \mid X_n = i\}, \; i, j \in S$$

Είναι πολύ σημαντικό να σημειώσουμε ότι τα στοιχεία του πίνακα P –και κατ' επέκταση και των δυνάμεων του P– εφόσον πρόκειται για πιθανότητες, έχουν τις εξής δύο ιδιότητες:

a) Κάθε στοιχείο είναι μη-αρνητικό, δηλαδή

$$p_{ij} \geq 0, \qquad \forall \; i, j \in S.$$

b) Τα στοιχεία κάθε γραμμής του πίνακα έχουν άθροισμα ίσο με τη μονάδα, δηλαδή

$$\sum_{j \in S} p_{ij} = 1, \qquad \forall \; i \in S.$$

Κάθε πίνακας που ικανοποιεί τις δύο παραπάνω συνθήκες λέγεται στοχαστικός (stochastic matrix).

Για να γίνουν περισσότερο κατανοητά όλα αυτά, ας φανταστούμε το ακόλουθο παράδειγμα: Έχουμε στον υπολογιστή μας 3 μουσικά κομμάτια, έστω Α, Β και Γ, και έχουμε γράψει ένα πρόγραμμα για την αναπαραγωγή τους, το οποίο λειτουργεί ως εξής: Αρχικά επιλέγει τυχαία ένα από τα κομμάτια Α, Β ή Γ και ξεκινάει την αναπαραγωγή του. Μόλις ολοκληρωθεί η αναπαραγωγή αυτού του κομματιού, το πρόγραμμα επιλέγει τυχαία έναν ακέραιο αριθμό από το 1 έως το 10 (είναι πολύ εύκολο με τη βοήθεια μιας γεννήτριας τυχαίων αριθμών να παράγουμε ακέραιους αριθμούς ομοιόμορφα κατανεμημένους στο διάστημα [1, 10]). Αν ο αριθμός που επιλέχθηκε είναι ένας από τους 1, 2 ή 3, τότε η αναπαραγωγή συνεχίζεται με το αμέσως προηγούμενο κομμάτι από αυτό που μόλις τελείωσε (θεωρούμε ότι η σειρά των κομματιών είναι Α, Β, Γ). Αν ο αριθμός είναι το 4 ή το 5, τότε επαναλαμβάνεται το ίδιο κομμάτι. Διαφορετικά, δηλαδή αν ο αριθμός είναι ένας από τους 6, 7, 8, 9 ή 10, η αναπαραγωγή συνεχίζεται με το αμέσως επόμενο στη σειρά κομμάτι.

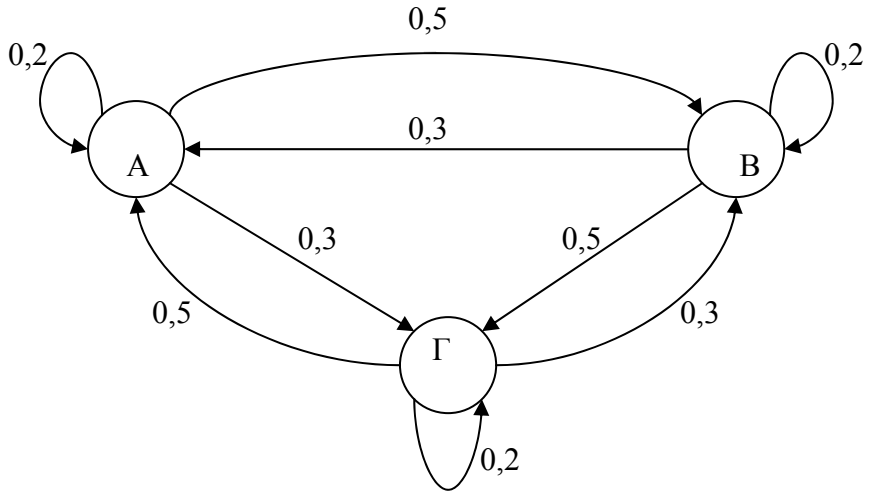
Έστω τώρα ότι αναζητούμε έναν τρόπο να περιγράψουμε τη λειτουργία αυτού του συστήματος. Όπως προκύπτει από την παραπάνω περιγραφή, χρειαζόμαστε δύο ειδών πληροφορίες:

α) Σε ποια κατάσταση βρίσκεται το σύστημα κάθε στιγμή –και κατ' επέκταση το σύνολο των καταστάσεων στις οποίες είναι δυνατό να βρεθεί το σύστημα. Στη συγκεκριμένη περίπτωση μπορούμε να θεωρήσουμε ότι η κατάσταση στην οποία βρίσκεται το σύστημα αντιστοιχεί στο μουσικό κομμάτι που αναπαράγεται τη δεδομένη στιγμή, οπότε έχουμε τις καταστάσεις Α, Β και Γ.

β) Τον τρόπο με τον οποίο γίνεται η μετάβαση από μία κατάσταση σε μία άλλη. Συγκεκριμένα, για κάθε κατάσταση στην οποία μπορεί να βρεθεί το σύστημα, θέλουμε να γνωρίζουμε ποιες καταστάσεις είναι δυνατό να ακολουθήσουν και με ποια πιθανότητα. Στην περίπτωση του παραπάνω παραδείγματος, ο κανόνας που χρησιμοποιείται για την επιλογή των μουσικών κομματιών είναι τέτοιος ώστε αν βρισκόμαστε για παράδειγμα στο κομμάτι Α,

τότε υπάρχει πιθανότητα 30% να ακούσουμε αμέσως μετά το κομμάτι Γ, 20% να ακούσουμε ξανά το κομμάτι Α και 50% να ακούσουμε το κομμάτι Β.

Ένας κομψός και εύκολα κατανοητός τρόπος για να αναπαρασταθούν όλα αυτά είναι με τη βοήθεια του παρακάτω κατευθυνόμενου γράφου με βάρη (weighted, directed graph):



Σχήμα 16: Παράδειγμα Μαρκοβιανής αλυσίδας

Όπως βλέπουμε, ο γράφος αποτελείται από 3 κόμβους, καθένας από τους οποίους αντιστοιχεί σε μία από τις καταστάσεις στις οποίες είναι δυνατό να βρεθεί το σύστημα. Η ύπαρξη ακμής από κάποιον κόμβο i σε κάποιον κόμβο j σημαίνει ότι είναι δυνατό το σύστημα να μεταβεί από την κατάσταση i στην κατάσταση j. Μάλιστα δίπλα σε κάθε ακμή σημειώνεται ένας αριθμός που ισούται με την πιθανότητα να συμβεί η αντίστοιχη μετάβαση. Μπορούμε εύκολα να παρατηρήσουμε ότι η πιθανότητα να ακούσουμε στο μέλλον κάποιο κομμάτι εξαρτάται μόνο από το κομμάτι που αναπαράγεται αυτή τη στιγμή και όχι από το ποια κομμάτια είχαν αναπαραχθεί προηγουμένως. Για παράδειγμα, αν βρισκόμαστε στην κατάσταση Α, η πιθανότητα να ακολουθήσει η κατάσταση Β είναι 0,5 είτε η προηγούμενη κατάσταση ήταν η Α είτε η Β είτε η Γ. Το μοντέλο αυτό είναι λοιπόν μία Μαρκοβιανή αλυσίδα.

Ο πίνακας μεταβάσεων για αυτό το παράδειγμα είναι ο εξής:

$$P = \begin{bmatrix} 0,2 & 0,5 & 0,3 \\ 0,3 & 0,2 & 0,5 \\ 0,5 & 0,3 & 0,2 \end{bmatrix}$$

Όπως ήταν αναμενόμενο, κάθε στοιχείο του πίνακα είναι μη-αρνητικό και τα στοιχεία κάθε γραμμής έχουν άθροισμα 1.

Έστω τώρα ότι θέλουμε να απαντήσουμε το ερώτημα: «Αν το κομμάτι που αναπαράγεται αυτή τη στιγμή είναι το Α, ποια είναι η πιθανότητα το μεθεπόμενο κομμάτι να είναι το Β;». Για να συμβεί το γεγονός αυτό, πρέπει να συμβεί ένας από τους ακόλουθους συνδυασμούς μεταβάσεων:

a) Α → Α και Α → Β ή
b) Α → Β και Β → Β ή
c) Α → Γ και Γ → Β

Άρα η ζητούμενη πιθανότητα είναι

(υπενθυμίζουμε ότι Pr(A $\vee$ B) = Pr(A) + Pr(B) και Pr(A $\wedge$ B) = Pr(A) * Pr(B)):

$Pr(A \overset{(2)}{\rightarrow} \Gamma)$ = Pr(A→A) * Pr(A→B) + Pr(A→B) * Pr(B→B) + Pr(A→Γ) * Pr(Γ→B) = 0,2 * 0,5 + 0,5 * 0,2 + 0,3 * 0,3 = 0,29.

Ας υπολογίσουμε τώρα τον πίνακα P^2. Είναι:

$$P^2 = P * P = \begin{bmatrix} 0,2 & 0,5 & 0,3 \\ 0,3 & 0,2 & 0,5 \\ 0,5 & 0,3 & 0,2 \end{bmatrix} * \begin{bmatrix} 0,2 & 0,5 & 0,3 \\ 0,3 & 0,2 & 0,5 \\ 0,5 & 0,3 & 0,2 \end{bmatrix} = \begin{bmatrix} 0,34 & 0,29 & 0,37 \\ 0,37 & 0.34 & 0,29 \\ 0,29 & 0,37 & 0,34 \end{bmatrix}$$

Πράγματι βλέπουμε ότι το στοιχείο που βρίσκεται στην πρώτη γραμμή και δεύτερη στήλη του P^2 ισούται –όπως αναμενόταν– με 0,29 , δηλαδή όση και η πιθανότητα να μεταβούμε με 2 βήματα από την κατάσταση Α στην κατάσταση Β. Με τον ίδιο τρόπο μπορούμε εύκολα να επαληθεύσουμε ότι το ίδιο ισχύει για κάθε συνδυασμό καταστάσεων και μάλιστα και για υψηλότερες δυνάμεις του P, δηλαδή για μεταβάσεις με μεγαλύτερο αριθμό βημάτων.

Ας δούμε τώρα όμως τι συμβαίνει με τον πίνακα P^k, όταν το k τείνει στο άπειρο. Κατ' αντιστοιχία με τα προηγούμενα, το στοιχείο [i,j] του πίνακα P^k, για k→∞ είναι η απάντηση στην ερώτηση: «Έστω ότι το σύστημα βρίσκεται στην κατάσταση i. Ποια είναι η πιθανότητα το σύστημα να βρεθεί μετά από «άπειρες» μεταβάσεις στην κατάσταση j;». Το γεγονός ότι θεωρούμε άπειρο πλήθος μεταβάσεων συνεπάγεται ότι ουσιαστικά δεν έχει σημασία από ποια κατάσταση ξεκινάει το σύστημα. Δηλαδή η πιθανότητα να βρεθεί να βρεθεί το σύστημα σε κάποια κατάσταση j μετά από άπειρο πλήθος μεταβάσεων είναι ίδια για όλες τις δυνατές αρχικές καταστάσεις i. Αυτό σημαίνει ότι κατά τον υπολογισμό των δυνάμεων του P όσο αυξάνεται ο εκθέτης οι γραμμές του πίνακα τείνουν να ταυτιστούν και η αυτή η ακολουθία πινάκων θα συγκλίνει τελικά σε έναν πίνακα του οποίου όλες οι γραμμές θα είναι ίδιες μεταξύ τους.

Αν δοκιμάσουμε να υπολογίσουμε μερικές δυνάμεις του πίνακα μεταβάσεων του παραπάνω παραδείγματος, διαπιστώνουμε ότι η παραπάνω παρατήρηση επαληθεύεται (μάλιστα ο συγκεκριμένος πίνακας τυχαίνει να συγκλίνει εξαιρετικά γρήγορα):

$$P = \begin{bmatrix} 0,2 & 0,5 & 0,3 \\ 0,3 & 0,2 & 0,5 \\ 0,5 & 0,3 & 0,2 \end{bmatrix}, \quad P^2 = \begin{bmatrix} 0,34 & 0,29 & 0,37 \\ 0,37 & 0.34 & 0,29 \\ 0,29 & 0,37 & 0,34 \end{bmatrix}, \quad P^3 = \begin{bmatrix} 0,340 & 0,339 & 0,321 \\ 0,321 & 0.340 & 0,339 \\ 0,339 & 0,321 & 0,340 \end{bmatrix},$$

$$P^4 = \begin{bmatrix} 0,3302 & 0,3341 & 0,3357 \\ 0,3357 & 0,3302 & 0,3341 \\ 0,3341 & 0,3357 & 0,3302 \end{bmatrix}, \quad P^5 = \begin{bmatrix} 0,3341 & 0,3326 & 0,3332 \\ 0,3332 & 0,3341 & 0,3326 \\ 0,3326 & 0,3332 & 0,3341 \end{bmatrix},$$

$$P^6 = \begin{bmatrix} 0,3332 & 0,3336 & 0,3332 \\ 0,3332 & 0,3332 & 0,3336 \\ 0,3336 & 0,3332 & 0,3332 \end{bmatrix}, \quad P^7 = \begin{bmatrix} 0,3333 & 0,3333 & 0,3334 \\ 0,3334 & 0,3333 & 0,3333 \\ 0,3333 & 0,3334 & 0,3333 \end{bmatrix},$$

$$P^8 = \begin{bmatrix} 0,3333 & 0,3333 & 0,3333 \\ 0,3333 & 0,3333 & 0,3333 \\ 0,3333 & 0,3333 & 0,3333 \end{bmatrix}$$

Αν ονομάσουμε π κάποια από αυτές τις ίδιες γραμμές, τότε έχουμε έναν πίνακα-γραμμή, τα στοιχεία του οποίου εκφράζουν ουσιαστικά την πιθανότητα να βρίσκεται το σύστημα σε μια δεδομένη κατάσταση κάποια χρονική στιγμή (the probability of finding the system in a given state at a particular time). Στο παράδειγμά μας είναι

$$\pi = [0{,}3333 \quad 0{,}3333 \quad 0{,}3333]$$

που σημαίνει ότι το σύστημα μπορεί να βρίσκεται σε κάθε μία από τις καταστάσεις Α, Β, Γ με πιθανότητα 33,33%. (Στο παράδειγμα που χρησιμοποιήσαμε τυχαίνει οι 3 καταστάσεις να είναι ισοπίθανες. Γενικά βέβαια αυτό δεν ισχύει.) Εναλλακτικά, λέγεται μερικές φορές για τον πίνακα π ότι εκφράζει το ποσοστό του χρόνου που το σύστημα «ξοδεύει» σε κάθε κατάσταση (the proportion of the time that the system spends in each state).

Προφανώς και για τον π ισχύουν οι δύο ιδιότητες που αναφέραμε παραπάνω, δηλαδή

a) κάθε στοιχείο του είναι μη-αρνητικό

$$\pi_i \geq 0, \qquad \forall\, i \in S.$$

b) τα στοιχεία του έχουν άθροισμα ίσο με τη μονάδα

$$\sum_{i \in S} \pi_i = 1$$

Ο πίνακας π ονομάζεται *invariant ή equilibrium ή stationary probability distribution* και γενικά ορίζεται ως η λύση της εξίσωσης

$$\pi P = \pi$$

Πράγματι στο παράδειγμά μας μπορούμε εύκολα να διαπιστώσουμε ότι ο π επαληθεύει αυτή την εξίσωση:

$$\pi P = \begin{bmatrix} 0{,}3333 & 0{,}3333 & 0{,}3333 \end{bmatrix} \begin{bmatrix} 0{,}2 & 0{,}5 & 0{,}3 \\ 0{,}3 & 0{,}2 & 0{,}5 \\ 0{,}5 & 0{,}3 & 0{,}2 \end{bmatrix} = \begin{bmatrix} 0{,}3333 & 0{,}3333 & 0{,}3333 \end{bmatrix} = \pi$$

Στο σημείο αυτό εγείρονται δύο ερωτήματα:

a) Έχει πάντα λύση η εξίσωση π P = π; Δηλαδή υπάρχει πάντα ο πίνακας π;
b) Είναι δυνατό η εξίσωση π P = π να έχει περισσότερες από μία λύσεις, δηλαδή ο πίνακας π να μην είναι μοναδικός;

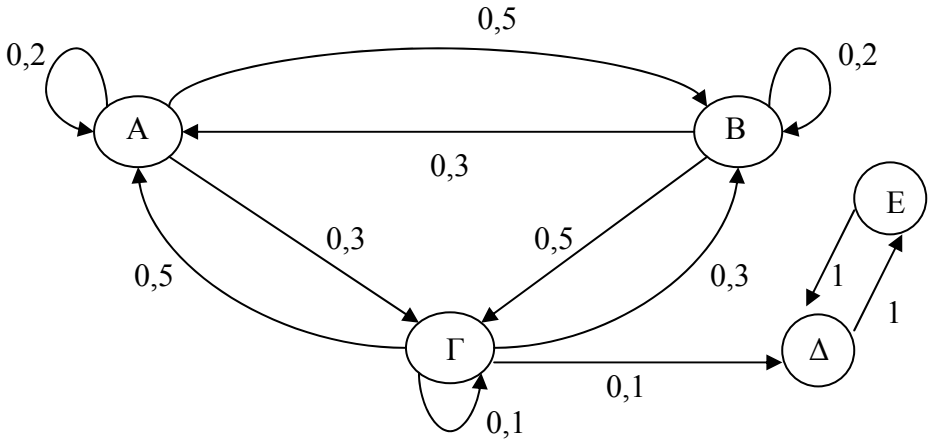
Πριν απαντήσουμε σε αυτά τα ερωτήματα, χρειάζεται να εισάγουμε κάποια επιπλέον χαρακτηριστικά των αλυσίδων Markov.

Μία αλυσίδα Markov λέμε ότι είναι *irreducible* αν είναι δυνατό από οποιαδήποτε κατάσταση να μεταβούμε, με ένα πεπερασμένο αριθμό βημάτων, σε οποιαδήποτε άλλη κατάσταση. Δηλαδή, πιο τυπικά, μία αλυσίδα Markov λέμε ότι είναι irreducible αν για κάθε i, j ∫ S υπάρχει n τέτοιο ώστε να ισχύει

$$p_{ij}^{(n)} > 0.$$

Η αλυσίδα στο παράδειγμά μας είναι irreducible. Αντίθετα η Μαρκοβιανή αλυσίδα του παρακάτω σχήματος δεν είναι irreducible, διότι από την κατάσταση Δ δεν μπορούμε να

μεταβούμε σε καμία από τις καταστάσεις Α, Β ή Γ. (Αυτή η αλυσίδα θα μπορούσε να παριστάνει τη λειτουργία μιας μηχανής, η οποία αν βρίσκεται στην κατάσταση Γ και συμβεί κάποιο ειδικό γεγονός, πχ. κάποιο λάθος, μεταβαίνει στην κατάσταση Δ και αρχίζει να εκτελεί μία διαφορετική εργασία.)



Σχήμα 17: Μη irreducible Μαρκοβιανή αλυσίδα

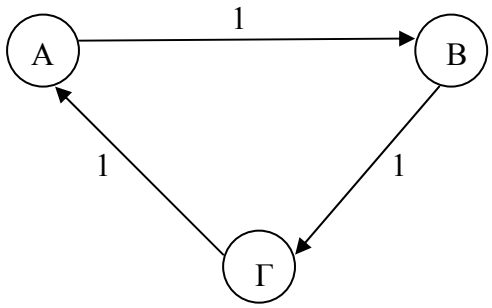
Μία κατάσταση i λέγεται *περιοδική* (*periodic*) αν το σύστημα επιστρέφει σίγουρα σε αυτή ύστερα από συγκεκριμένο αριθμό βημάτων. Πιο τυπικά, μία κατάσταση i λέγεται περιοδική αν υπάρχει ακέραιος $k > 1$ τέτοιος ώστε για κάθε n να μπορούμε να βρούμε έναν ακέραιο j ώστε

$$p_{ii}^{(n)} = \begin{cases} 1 & n = kj \\ 0 & n \neq kj \end{cases}$$

(Δηλαδή το σύστημα επιστρέφει στην κατάσταση i αν και μόνο αν συμπληρώσει αριθμό βημάτων πολλαπλάσιο κάποιου αριθμού k). Αντίθετα μία κατάσταση i λέγεται *απεριοδική* (*aperiodic*) αν υπάρχει κάποιος ακέραιος n τέτοιος ώστε για κάθε $m > n$ να ισχύει

$$p_{ii}^{(m)} > 0 .$$

Στην αλυσίδα Markov του παραδείγματος με τα μουσικά κομμάτια όλες οι καταστάσεις είναι απεριοδικές. Πράγματι, αν το σύστημα βρίσκεται στην κατάσταση Α, μπορεί να ξαναβρεθεί σε αυτή είτε μετά από 1 βήμα (διαδρομή Α→Α) είτε μετά από 2 βήματα (διαδρομή Α→Β→Α ή Α→Γ→Α) είτε μετά από 3 βήματα (διαδρομή Α→Β→Γ→Α ή Α→Γ→Β→Α) κ.ο.κ. Αντίθετα η κατάσταση Α στη Μαρκοβιανή αλυσίδα του παρακάτω σχήματος (μία τέτοια αλυσίδα μπορεί να παριστάνει μία μηχανή που κατά τη λειτουργία της εκτελεί 3 εργασίες Α, Β, Γ με την ίδα πάντα σειρά) είναι περιοδική, γιατί το σύστημα επιστρέφει σε αυτή αν και μόνο αν έχει παρέλθει αριθμός βημάτων που είναι πολλαπλάσιο του 3 (το ίδιο ισχύει βέβαια και για τις καταστάσεις Β και Γ).



Σχήμα 18: Περιοδική Μαρκοβιανή αλυσίδα

Αποδεικνύεται ότι αν μία αλυσίδα Markov είναι irreducible και έχει μία απεριοδική κατάσταση, τότε και όλες οι υπόλοιπες καταστάσεις είναι απεριοδικές.

Αν μία αλυσίδα Markov είναι irreducible και aperiodic, τότε λέγεται *εργοδική* (*ergodic*). Μία εργοδική αλυσίδα Markov έχει ακριβώς μία invariant distribution π.

3.3 Μαρκοβιανές αλυσίδες συνεχούς χρόνου

Οι Μαρκοβιανές αλυσίδες συνεχούς χρόνου (*Continuous Time Markov Chains – CTMC*) είναι διαδικασίες Markov στις οποίες ο χώρος καταστάσεων είναι διακριτός, όπως και στις Discrete Time Markov Chains, αλλά οι χρονικές στιγμές παρατήρησης δεν παίρνουν επίσης διακριτές τιμές, όπως στις Discrete Time Markov Chains, αλλά αντίθετα παίρνουν συνεχείς τιμές. Και στην περίπτωση των αλυσίδων Markov συνεχούς χρόνου πρέπει βέβαια να ικανοποιείται η Μαρκοβιανή ιδιότητα, η οποία στην περίπτωση αυτή μπορεί να γραφεί με την εξής μορφή:

$$\begin{aligned}\Pr\{X_{t_n+\Delta t} = P' \mid X_{t_n} = P, X_{t_{n-1}} = P_{t_{n-1}}, X_{t_{n-2}} = P_{t_{n-2}}, ..., X_{t_0} = P_{t_0}\} = \\ = \Pr\{X_{t_n+\Delta t} = P' \mid X_{t_n} = P\} = \\ = \Pr\{X_{\Delta t} = P' \mid X_0 = P\}\end{aligned}$$

Υποθέτουμε και εδώ, όπως και στις αλυσίδες Markov διακριτού χρόνου, ότι έχουμε ομοιογένεια ως προς το χρόνο, δηλαδή η πιθανότητα να βρίσκεται το σύστημα σε κάποια κατάσταση κάποια χρονική στιγμή δεν εξαρτάται από τη χρονική στιγμή της παρατήρησης. Ωστόσο εξαρτάται από το μέγεθος του χρονικού διαστήματος Δt. Για να έχουμε γραμμική εξάρτηση ως προς το χρόνο, πρέπει για κάθε ζεύγος καταστάσεων P και P΄ να υπάρχει κάποια παράμετρος λ, τέτοια ώστε για «αρκετά μικρό» Δt να ισχύει η σχέση:

$$\Pr\{X_{\Delta t} = P' \mid X_0 = P\} = \lambda \Delta t + o(\Delta t)$$

όπου ο όρος ο(Δt) περιλαμβάνει τις πιθανότητες να περάσει το σύστημα από ενδιάμεσες καταστάσεις μέχρι να μεταβεί από την P στην P΄ κατά το χρονικό διάστημα Δt. Η ποσότητα λ, που είναι ένας μη αρνητικός πραγματικός αριθμός, αποτελεί συνεπώς το *ρυθμό μετάβασης* (*transition rate*) από την κατάσταση P στην κατάσταση P΄. Μέχρι στιγμής υποθέσαμε ότι η

κατάσταση P΄ είναι διαφορετική από την P. Αν όμως πρόκειται για την ίδια κατάσταση, τότε ουσιαστικά μιλάμε για την πιθανότητα το σύστημα να έχει μείνει στην ίδια κατάσταση ύστερα από κάποιο χρονικό διάστημα Δt. Η πιθανότητα αυτή ξεκινά αρχικά από την τιμή 1, δηλαδή για Δt = 0, και μειώνεται με την πάροδο του χρόνου. Μπορούμε έμμεσα να την υπολογίσουμε αν απλά αφαιρέσουμε από τη μονάδα την πιθανότητα το σύστημα να μεταβεί σε οποιαδήποτε άλλη κατάσταση στο χρονικό διάστημα Δt. Έτσι έχουμε:

$$\Pr\{X_{\Delta t} = P \mid X_0 = P\} = 1 - \sum_{i \in S} \lambda_i \Delta t + o(\Delta t)$$

Σε αντίθεση με τις πιθανότητες μετάβασης, οι οποίες στην περίπτωση των αλυσίδων Markov συνεχούς χρόνου εξαρτώνται από το μήκος του χρονικού διαστήματος, οι ρυθμοί μετάβασης δεν εξαρτώνται από το μήκος του χρονικού διαστήματος. Έτσι η συμπεριφορά μίας αλυσίδας Markov συνεχούς χρόνου μπορεί να περιγραφεί πλήρως αν γνωρίζουμε την αρχική κατάσταση και τους ρυθμούς μετάβασης ανάμεσα στις διάφορες καταστάσεις.

Έτσι, σε αντιστοιχία με τον πίνακα μεταβάσεων που χρησιμοποιούμε στην περίπτωση των αλυσίδων Markov διακριτού χρόνου, μία αλυσίδα Markov συνεχούς χρόνου μπορεί να περιγραφεί με τη βοήθεια ενός τετραγωνικού πίνακα Q, του οποίου κάθε στοιχείο i, j (i ≠ j) ισούται με το ρυθμό μετάβασης από την κατάσταση i στην κατάσταση j. Εξ' ορισμού, τα στοιχεία της διαγωνίου του πίνακα Q, δηλαδή τα στοιχεία q_{ii}, ισούνται με το αντίθετο του αθροίσματος των υπόλοιπων στοιχείων της ίδιας γραμμής, δηλαδή

$$q_{ii} = -\sum_{j \in S, i \neq j} q_{ij}$$

Το παραπάνω άθροισμα εκφράζει ουσιαστικά το ρυθμό με τον οποίο το σύστημα φεύγει από την κατάσταση i. Επιπλέον, σύμφωνα με την παραπάνω ιδιότητα, είναι προφανές ότι τα στοιχεία κάθε γραμμής του πίνακα Q έχουν άθροισμα που ισούται με μηδέν.

Οι διάφοροι ορισμοί που δόθηκαν για τις αλυσίδες Markov διακριτού χρόνου ισχύουν και εδώ. Μία αλυσίδες Markov συνεχούς χρόνου λέγεται irreducible αν είναι δυνατό από οποιαδήποτε κατάσταση το σύστημα να μεταβεί σε οποιαδήποτε άλλη. Επίσης αν η εξίσωση

$$\pi \cdot Q = 0$$

έχει μοναδική λύση τότε η αλυσίδα είναι εργοδική και ο πίνακας π λέγεται, όπως και στην προηγούμενη περίπτωση, invariant ή equilibrium ή stationary probability distribution.

Για να γίνουν περισσότερο κατανοητά τα παραπάνω και για να φανεί σε τι χρησιμεύουν και πως χρησιμοποιούνται στην πράξη οι αλυσίδες Markov συνεχούς χρόνου, θα εξετάσουμε στη συνέχεια ένα παράδειγμα. Το παράδειγμα προέρχεται από το χώρο των συστημάτων αναμονής, που είναι ένα από τα σημαντικότερα πεδία στα οποία βρίσκουν εφαρμογή οι αλυσίδες Markov.

Γενικά ένα *σύστημα αναμονής* είναι ένα σημείο εξυπηρέτησης εφοδιασμένο με έναν χώρο αναμονής. Χρησιμοποιώντας την τυπική ορολογία, οι πελάτες που φτάνουν στο σύστημα υποχρεούνται να περάσουν κάποιο χρόνο, που λέγεται χρόνος εξυπηρέτησης, με έναν εξυπηρετητή. Μπορεί να υπάρχουν περισσότεροι από ένας εξυπηρετητές και οι πελάτες μπορεί να χρειαστεί να περιμένουν για έναν διαθέσιμο εξυπηρετητή. Οι πελάτες εγκαταλείπουν την ουρά αφού μείνουν για την απαιτούμενη διάρκεια του χρόνου εξυπηρέτησης. Συχνά θεωρούμε ότι υπάρχει περιορισμός στον αριθμό των πελατών που είναι

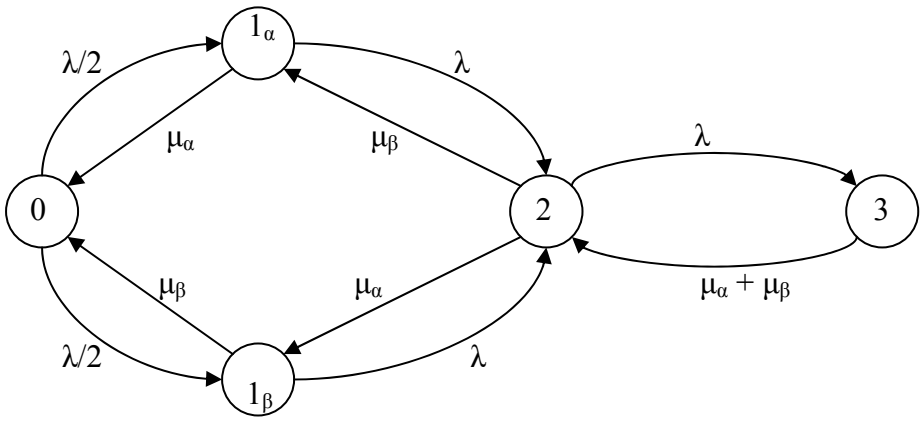
δυνατό να υπάρχουν στο σύστημα οποιαδήποτε στιγμή. Στην περίπτωση αυτή, αν στο σύστημα υπάρχει ο μέγιστος αριθμός πελατών και συμβεί μία άφιξη, θεωρούμε ότι ο πελάτης αυτό χάνεται. Τυπικά θεωρούμε ως παραμέτρους του συστήματος το ρυθμό αφίξεων πελατών και το ρυθμό αναχωρήσεων, δηλαδή την ταχύτητα με την οποία έρχονται πελάτες στο σύστημα και την ταχύτητα με την οποία κάθε εξυπηρετητής εξυπηρετεί τους πελάτες, και προσπαθούμε να υπολογίσουμε διάφορες ποσότητες που έχουν σχέση με την απόδοση του συστήματος, όπως για παράδειγμα το μέσο μήκος της ουράς αναμονής, το μέσο χρόνο εξυπηρέτησης, το ρυθμό απωλειών κλπ.

Ας θεωρήσουμε ένα σύστημα αναμονής M/M/2/3, δηλαδή ένα σύστημα αναμονής στο οποίο υπάρχουν δύο εξυπηρετητές, ο μέγιστος αριθμός πελατών στο σύστημα οποιαδήποτε στιγμή είναι 3 –συμπεριλαμβανομένου αυτού που εξυπηρετείται– και οι αφίξεις και οι αναχωρήσεις πελατών είναι τυχαίες μεταβλητές που ακολουθούν την εκθετική κατανομή (διαδικασίες Poisson). Δηλαδή

$$P\{A(t+\tau)-A(t)=n\}=e^{-\lambda\tau}\frac{(\lambda\tau)^n}{n!}$$

όπου A(t) είναι ο αριθμός των γεγονότων (πχ. αφίξεων) στο χρονικό διάστημα t. (Υπενθυμίζουμε ότι στις διαδικασίες Poisson ισχύει η ιδιότητα της «έλλειψης μνήμης» συνεπώς η χρήση αλυσίδων Markov σε αυτές τις περιπτώσεις είναι ιδανική). Έστω ότι ο ρυθμός αφίξεων των πελατών είναι λ, ότι ο ρυθμός εξυπηρέτησης για τον πρώτο εξυπηρετητή είναι μ_α, για το δεύτερο μ_β και ότι όταν έρθει ένας νέος πελάτης μπορεί να πάει με ίση πιθανότητα σε οποιονδήποτε από τους δύο διαθέσιμους εξυπηρετητές.

Η συμπεριφορά του παραπάνω συστήματος μπορεί να περιγραφεί σχηματικά όπως φαίνεται παρακάτω, θεωρώντας ότι η κατάσταση του συστήματος αντιπροσωπεύεται από τον αριθμό των πελατών που βρίσκονται μέσα σε αυτό κάθε στιγμή (στην περίπτωση που στο σύστημα υπάρχει 1 πελάτης έχουμε 2 δυνατές καταστάσεις: 1_α και 1_β, ανάλογα με το αν αυτός βρίσκεται στον πρώτο ή το δεύτερο εξυπηρετητή αντίστοιχα).



Σχήμα 19: Μαρκοβιανή αλυσίδα συνεχούς χρόνου

Αρχικά θέλουμε να υπολογίσουμε την πιθανότητα να βρίσκεται κάποια στιγμή το σύστημα σε καθεμιά από τις καταστάσεις 0, 1_α, 1_β, 2 ή 3. Για το σύστημα αυτό μπορούμε, όπως φαίνεται και από το σχήμα, να γράψουμε τις ακόλουθες καταστάσεις ισορροπίας:

$$\frac{\lambda}{2}P_0+\frac{\lambda}{2}P_0=\mu_\alpha P_{1\alpha}+\mu_\beta P_{1\beta} \qquad (1)$$

$$\lambda P_{1a} + \mu_\alpha P_{1a} = \frac{\lambda}{2} P_0 + \mu_\beta P_2 \qquad (2)$$

$$\lambda P_{1\beta} + \mu_\beta P_{1\beta} = \frac{\lambda}{2} P_0 + \mu_\alpha P_2 \qquad (3)$$

$$\lambda P_2 + \mu_\alpha P_2 + \mu_\beta P_2 = \lambda P_{1a} + \lambda P_{1\beta} + (\mu_\alpha + \mu_\beta) P_3 \qquad (4)$$

$$(\mu_\alpha + \mu_\beta) P_3 = \lambda P_2 \qquad (5)$$

(Σε κάθε μία από τις παραπάνω εξισώσεις το πρώτο μέλος εκφράζει το ρυθμό εξόδου από μία κατάσταση και το δεύτερο μέλος το ρυθμό εισόδου σε αυτή.)
Επιπλέον ισχύει η εξίσωση

$$P_0 + P_{1\alpha} + P_{1\beta} + P_2 + P_3 = 1 \qquad (6)$$

διότι κάθε στιγμή το σύστημα θα βρίσκεται αναγκαστικά σε μία από τις καταστάσεις 0, 1_α, 1_β, 2 ή 3.

Από τη λύση του συστήματος των εξισώσεων (1) – (6) προκύπτουν οι πιθανότητες P_0, $P_{1\alpha}$, $P_{1\beta}$, P_2 και P_3. Ας υποθέσουμε για απλότητα ότι $\lambda = \mu_\alpha = \mu_\beta = 1$ πελ./min. Τότε έχουμε:

$$(1) \Rightarrow P_0 = P_{1a} + P_{1\beta} \qquad (1')$$

$$(2) \Rightarrow 2P_{1a} = 0,5P_0 + P_2 \qquad (2')$$

$$(3) \Rightarrow 2P_{1\beta} = 0,5P_0 + P_2 \qquad (3')$$

$$(4) \Rightarrow 3P_2 = P_{1a} + P_{1\beta} + 2P_3 \qquad (4')$$

$$(5) \Rightarrow 2P_3 = P_2 \qquad (5')$$

Από τις εξισώσεις (4΄) και (5΄) προκύπτει ότι:

$$2P_2 = P_{1a} + P_{1\beta} \qquad (7)$$

και αντικαθιστώντας από την (7) στην (1΄) παίρνουμε:

$$P_2 = 0,5P_0 \qquad (8)$$

Μπορούμε τώρα εύκολα να εκφράσουμε και τις υπόλοιπες πιθανότητες $P_{1\alpha}$, $P_{1\beta}$ και P_3 συναρτήσει της P_0:

$$P_{1\alpha} = 0,5P_0 \qquad (9)$$

$$P_{1\beta} = 0,5P_0 \qquad (10)$$

$$P_3 = 0,25P_0 \qquad (11)$$

Αντικαθιστώντας τις εξισώσεις (8) – (11) στην εξίσωση (6) βρίσκουμε:

$$P_0 + 0,5P_0 + 0,5P_0 + 0,5P_0 + 0,25P_0 = 1 \Rightarrow P_0 = \frac{4}{11}$$

Συνεπώς:

$$P_{1\alpha} = P_{1\beta} = P_2 = \frac{2}{11} \qquad \text{και} \qquad P_3 = \frac{1}{11}$$

Έχοντας προσδιορίσει τις πιθανότητες των διαφόρων καταστάσεων, μπορούμε να υπολογίσουμε και μερικά άλλα μεγέθη που αφορούν την απόδοση του συστήματος:

- Μέσο μήκος της ουράς αναμονής:

$$\mathrm{E}[n] = \sum_{n=0}^{3} nP_n = 0 \cdot P_0 + 1 \cdot (P_{1\alpha} + P_{1\beta}) + 2 \cdot P_2 + 3 \cdot P_3 = 1$$

- Βαθμός εξυπηρέτησης για κάθε εξυπηρετητή:

$$u_\alpha = 1 - (P_0 + P_{1\beta}) = \frac{5}{11}$$

$$u_\beta = 1 - (P_0 + P_{1\alpha}) = \frac{5}{11}$$

- Ρυθμός απωλειών:

$$\lambda_{loss} = \lambda \cdot P_3 = \frac{1}{11} \text{ πελ./min}$$

- Απόδοση συστήματος:

$$\gamma = \gamma_\alpha + \gamma_\beta = \mu_\alpha \cdot u_\alpha + \mu_\beta \cdot u_\beta = \frac{10}{11}$$

$$\left(\text{ή αλλιώς: } \gamma = \lambda \cdot (1 - P_3) = \frac{10}{11}\right)$$

- Μέσος χρόνος παραμονής στο σύστημα:

$$\mathrm{E}[\mathrm{T}] = \frac{\mathrm{E}[n]}{\gamma} = 1,1 \text{ min.} \qquad \text{(Νόμος του Little)}$$

Μέρος 4ο: Κατηγοριοποίηση κειμένων

4.1 Περιγραφή του προβλήματος

Το πρόβλημα της *κατηγοριοποίησης κειμένων* (*text categorization*) είναι ένα πρόβλημα ταξινόμησης στο οποίο τα αντικείμενα είναι προφανώς κείμενα. Δηλαδή έχουμε κάποιες κατηγορίες κειμένων και θέλουμε να δούμε σε ποια ή ποιες από αυτές ταιριάζει καλύτερα ένα νέο κείμενο. Το ποιες θα είναι αυτές οι κατηγορίες μπορεί να έχει αποφασιστεί εκ των προτέρων ή μπορεί να προηγηθεί μία unsupervised διαδικασία ομαδοποίησης κάποιων αρχικών κειμένων και στη συνέχεια οι ομάδες που θα προκύψουν να αποτελέσουν τις κατηγορίες στις οποίες θα ταξινομηθούν τα υπόλοιπα κείμενα.

Η ομαδοποίηση κειμένων με βάση τη θεματική ενότητα έχει πολύ μεγάλη σημασία σε πλήθος εφαρμογών και ανθρώπινων δραστηριοτήτων καθώς μπορεί να διευκολύνει σε πολύ μεγάλο βαθμό την αναζήτηση και ανεύρεση πληροφορίας. Δεν είναι τυχαίο ότι όλες οι μεγάλες πύλες στο Διαδίκτυο, όπως για παράδειγμα το Yahoo!, χωρίζουν τους συνδέσμους που παρέχουν προς διάφορες ιστοσελίδες σε θεματικές ενότητες, ώστε να διευκολύνουν τους χρήστες και να τους κατευθύνουν ανάλογα με τα ενδιαφέροντα και τις ανάγκες του καθενός. Συνήθως τέτοιου είδους εργασίες γίνονται ακόμα από ομάδες ειδικών. Ωστόσο ο τεράστιος όγκος κειμένων που ήδη υπάρχουν στο Διαδίκτυο και σε ψηφιακές βιβλιοθήκες, ο οποίος μάλιστα αυξάνεται με ταχείς ρυθμούς, καθιστούν αυτό τον τρόπο μη αποδοτικό. Για το λόγο αυτό έχουν γίνει πολλές προσπάθειες να εφαρμοστούν στο πρόβλημα της κατηγοριοποίησης κειμένων τεχνικές ταξινόμησης και ομαδοποίησης, χωρίς ωστόσο να έχει βρεθεί ακόμα κάποια βέλτιστη λύση.

4.2 Μερικές χρησιμοποιημένες τεχνικές

Ένας από τους πιο συχνά χρησιμοποιούμενους αλγόριθμους για το πρόβλημα της κατηγοριοποίησης κειμένων είναι ο k-Nearest Neighbor. Ο τρόπος με τον οποίο λειτουργεί αυτός ο αλγόριθμος έχει ήδη περιγραφεί στην αντίστοιχη ενότητα του 1ου μέρους. Για να ταξινομήσουμε λοιπόν ένα νέο κείμενο, βρίσκουμε τους k κοντινότερους γείτονές του στο training set, δηλαδή τα k κείμενα γνωστής κατηγορίας που μοιάζουν περισσότερο με αυτό και βλέπουμε σε ποια κατηγορία ανήκουν τα περισσότερα από αυτά. Ένα μειονέκτημα αυτού του αλγόριθμου είναι η αποδοτικότητά του, για το λόγο ότι για να βρούμε τους k κοντινότερους γείτονες ενός κειμένου, πρέπει να το συγκρίνουμε με κάθε ένα από τα κείμενα που υπάρχουν στο training set. Επίσης πρέπει να δοθεί πολύ μεγάλη προσοχή σε δύο παράγοντες: α) στον τρόπο με τον οποίο θα προσδιορίζεται η απόσταση δύο κειμένων και β) στην επιλογή κατάλληλης τιμής για την παράμετρο k. Αν το k είναι πολύ μεγάλο, υπάρχει κίνδυνος οι μεγάλες κλάσεις να εξαφανίσουν σταδιακά τις μικρές. Από την άλλη πλευρά, αν το k είναι πολύ μικρό, χάνεται το βασικό πλεονέκτημα αυτής της μεθόδου, δηλαδή το γεγονός ότι λαμβάνει την απόφαση στηριζόμενη στην «ετυμηγορία» πολλών «ειδικών». Συνήθως η κατάλληλη τιμή για το k βρίσκεται με δοκιμάζοντας διάφορες τιμές και

εκτιμώντας πόσο καλή είναι η ταξινόμηση που επιτυγχάνεται σε κάθε περίπτωση. Αυτό όμως δεν είναι πάντα εύκολο ή αποδοτικό να γίνει. Ένας άλλος τρόπος είναι να χρησιμοποιηθεί μία παραλλαγή του k-Nearest Neighbor, η οποία χρησιμοποιεί διαφορετικές τιμές του k για τις διάφορες κατηγορίες και όχι το ίδιο k για όλες. Δηλαδή όσο μεγαλύτερο πλήθος αντικειμένων περιέχει μία κατηγορία τόσο μεγαλύτερο ποσοστό από τους k κοντινότερους γείτονες θα πρέπει να ανήκουν σε αυτή ώστε να αποφασίσουμε υπέρ αυτής.

Μία πολύ σημαντική απόφαση που πρέπει να ληφθεί είναι ο τρόπος με τον οποίο θα αναπαρίστανται τα κείμενα. Συνήθως ο τρόπος αναπαράστασης που επιλέγεται είναι ο εξής: Κάθε κείμενο αναπαριστάται με έναν πίνακα διαστάσεων 1xN, όπου N το πλήθος των διαφορετικών λέξεων που έχουν βρεθεί στο σύνολο των κειμένων που χρησιμοποιούμε. Κάθε στοιχείο αυτού του πίνακα εκφράζει τη συχνότητα με την οποία εμφανίζεται η αντίστοιχη λέξη στο συγκεκριμένο κείμενο (ένας ακόμη απλούστερος τρόπος, αλλά φυσικά φτωχότερος σε πληροφορία, είναι να περιέχει ο πίνακας αυτός μόνο άσσους ή μηδενικά ανάλογα με το αν η αντίστοιχη λέξη εμφανίζεται ή όχι στο συγκεκριμένο κείμενο).

Εφόσον έχουμε αναπαραστήσει κάθε κείμενο με έναν τέτοιο πίνακα d, μπορούμε να υπολογίσουμε την απόσταση δύο κειμένων χρησιμοποιώντας κάποια από τις μετρικές που έχουν περιγραφεί στο 2^{ο} μέρος. Μία άλλη μετρική που επίσης χρησιμοποιείται πολύ συχνά είναι αυτή του συνημιτόνου (cosine measure), σύμφωνα με την οποία η απόσταση δύο κειμένων που περιγράφονται από τους πίνακες d_1 και d_2 είναι η εξής:

$$d = \cos(d_1, d_2) = \frac{d_1 \cdot d_2}{\| d_1 \| \cdot \| d_2 \|}$$

όπου $d_1 \cdot d_2$ το εσωτερικό γινόμενο των δύο πινάκων-διανυσμάτων και $\| d_1 \|, \| d_2 \|$ η διάσταση του καθενός (που εδώ θα ισούται με το N).

Επίσης είναι δυνατό κατά τον υπολογισμό της απόστασης να προσδώσουμε διαφορετική βαρύτητα σε κάποιες λέξεις, δηλαδή να λάβουμε υπόψη μας περισσότερο τις λέξεις εκείνες οι οποίες θεωρούμε ότι περιέχουν περισσότερη πληροφορία σχετικά με το ερώτημα του καθορισμού της κατηγορίας στην οποία ανήκει ένα κείμενο. Ένας τρόπος που χρησιμοποιείται για να μετρηθεί αυτό ακριβώς το πράγμα, δηλαδή σε ποιο βαθμό μία λέξη μπορεί να μας πληροφορήσει για την κατηγορία ενός κειμένου, είναι η λεγόμενη mutual information της λέξης –η έννοια αυτή προέρχεται από το χώρο της information retrieval. Η mutual information μίας λέξης w ως προς κάποια κατηγορία c μπορεί να οριστεί ως εξής:

$$MI(w) = P(c,w) \cdot \log \frac{P(c,w)}{P(c) \cdot P(w)} + P(c,\overline{w}) \cdot \log \frac{P(c,\overline{w})}{P(c) \cdot P(\overline{w})}$$

όπου P(c) είναι η πιθανότητα της κατηγορίας c, P(w) η πιθανότητα παρουσίας της λέξης w, P(c,w) η πιθανότητα ένα κείμενο που περιέχει τη λέξη w να ανήκει στην κατηγορία c και $P(c, \overline{w})$ η πιθανότητα ένα κείμενο που δεν περιέχει τη λέξη w να ανήκει στην κατηγορία c.

Μία ενδιαφέρουσα εναλλακτική προσέγγιση που έχει χρησιμοποιηθεί είναι η εξής: Αντί να προσπαθούμε να ομαδοποιήσουμε άμεσα τα διαθέσιμα κείμενα, μπορούμε να επιχειρήσουμε να ομαδοποιήσουμε τις διάφορες λέξεις που εμφανίζονται σε αυτά τα κείμενα. Για κάθε λέξη σχηματίζουμε μία ομάδα που περιέχει όλα τα κείμενα στα οποία εμφανίζεται αυτή η λέξη. Κατόπιν χρησιμοποιούμε agglomerative hierarchical clustering, δηλαδή συνδυάζουμε τις ομάδες που περιέχουν πολλά κοινά κείμενα. Έτσι καταλήγουμε σε μερικές ομάδες κειμένων, οι οποίες έχουν το χαρακτηριστικό ότι τα κείμενα που περιέχονται στην

ίδια ομάδα περιέχουν παρόμοιες λέξεις. Ένα πλεονέκτημα που προσφέρει αυτή η προσέγγιση είναι ότι επιτρέπει σε κάθε κείμενο να ανήκει σε περισσότερες από μία ομάδας –δηλαδή πρόκειται για disjunctive ή soft clustering. Αυτό είναι πολύ σημαντικό σε περιπτώσεις κειμένων που αναφέρονται σε περισσότερα από ένα θέματα.

Όπως είδαμε, οι διάφορες τεχνικές για την κατηγοριοποίηση κειμένων χρησιμοποιούν λέξεις ως το δομικό στοιχείο για την αναπαράσταση των κειμένων. Συνεπώς όσο μεγαλύτερο είναι το πλήθος των διαφορετικών λέξεων που εμφανίζονται στα κείμενα που εξετάζονται τόσο περισσότερο αυξάνεται το υπολογιστικό κόστος. Σε ένα σύνολο μερικών μόνο κειμένων είναι δυνατό το πλήθος των διαφορετικών λέξεων να είναι της τάξης των 10000 γεγονός που σημαίνει ότι το υπολογιστικό κόστος είναι σημαντικό και συνεπώς θα ήταν βολικό να μειωθεί το πλήθος των λέξεων. Εξάλλου υπάρχουν πολλές λέξεις –κυρίως προθέσεις, σύνδεσμοι κλπ– οι οποίες εμφανίζονται με την ίδια περίπου συχνότητα σε όλα τα κείμενα ανεξαρτήτου θέματος και συνεπώς παρέχουν ελάχιστη ή καθόλου πληροφορία για τη θεματική ενότητα στην οποία ανήκει ένα κείμενο. Για το λόγο αυτό λοιπόν στις περισσότερες περιπτώσεις οι διάφορες τεχνικές, πριν την αναπαράσταση και τη σύγκριση των κειμένων, αφαιρούν πολλές από τις λέξεις. Αυτό μπορεί να γίνει βλέποντας τη συχνότητα με την οποία κάθε λέξη εμφανίζεται στα διάφορα κείμενα. Για παράδειγμα αν δούμε ότι κάποια λέξη εμφανίζεται με υψηλή συχνότητα στην πλειοψηφία των κειμένων μπορούμε να υποθέσουμε ότι πρόκειται για κάποιο άρθρο ή σύνδεσμο και να μην τη λάβουμε υπόψη στην υπόλοιπη διαδικασία. Για τους ίδιους λόγους, εκτός από την αφαίρεση τέτοιου είδους λέξεων, γίνεται μερικές φορές και το εξής: Όλες οι λέξεις που έχουν κοινή ρίζα αντιμετωπίζονται σαν μία. Για παράδειγμα, οι λέξεις «υπολογίζω», «υπολογισμός», «υπολογιστής» κλπ. μπορούν να θεωρηθούν ίδιες και να αντικατασταθούν από μία από αυτές.

4.3 Προσέγγιση του προβλήματος με χρήση αλυσίδων Markov

Όπως έχει ήδη αναφερθεί, η επιλογή ενός αλγορίθμου ομαδοποίησης μπορεί να οδηγήσει σε μια περισσότερο ή λιγότερο ικανοποιητική ομαδοποίηση. Ωστόσο η απόδοση οποιουδήποτε αλγόριθμου ταξινόμησης εξαρτάται προφανώς από την πληροφορία που έχει συλλεγεί στα προηγούμενα στάδια της διαδικασίας. Δηλαδή αν κάποιος έχει επιλέξει σωστά τις μεταβλητές που θα χρησιμοποιήσει για να περιγράψει τα αντικείμενα που διαθέτει, τον τρόπο που θα κςδικοποιήσει τις τιμές τους, τη μετρική που θα χρησιμοποιήσει για την εύρεση της απόστασης ανάμεσα στα αντικείμενα κλπ., τότε μπορεί εύκολα να πειραματιστεί με διάφορους αλγόριθμους ταξινόμησης ώσπου να πάρει τα επιθυμητά αποτελέσματα. Αντίθετα αν δεν έχουν γίνει οι κατάλληλες επιλογές στα προηγούμενα στάδια και κατά συνέπεια η πληροφορία που έχει συλλεχθεί δεν απεικονίζει ικανοποιητικά τις ομοιότητες και τις διαφορές των αντικειμένων, τότε κανείς αλγόριθμος δε θα είναι σε θέση να οδηγήσει σε μία ομαδοποίηση που να πληρεί τις συνθήκες που έχουν τεθεί. Για το λόγο αυτό θα μελετήσουμε στη συνέχεια το πρόβλημα της κατηγοροιοποίησης κειμένων επικεντρώνοντας το ενδιαφέρον στον τρόπο αναπαράστασης των αντικειμένων και στον τρόπο προσδιορισμού της μεταξύ τους απόστασης και όχι τόσο στη δοκιμή διάφορων αλγορίθμων ομαδοποίησης.

Ξεκινάμε λοιπόν από το στάδιο της επιλογής του τρόπου αναπαράστασης των κειμένων. Στην περίπτωση που τα αντικείμενα που μελετάμε είναι κείμενα, οι μεταβλητές που χρησιμοποιούνται για την περιγραφή τους και κατ’ επέκταση για τον ορισμό των μεταξύ τους αποστάσεων είναι οι κατά κανόνα οι λέξεις. Δηλαδή κάθε κείμενο αναπαρίσταται από

έναν πίνακα που περιέχει τη συχνότητα με την οποία εμφανίζεται κάθε λέξη στο κείμενο αυτό και κατόπιν ο προσδιορισμός της απόστασης δύο κειμένων ανάγεται κατά συνέπεια σε προσδιορισμό της απόστασης δύο μονοδιάστατων πινάκων. Είναι αλήθεια ότι οι λέξεις είναι το μικρότερο δομικό στοιχείο ενός κειμένου που έχει νοηματική αξία, δηλαδή που περιέχει πληροφορία σχετικά με το αντικείμενο στο οποίο αναφέρεται το συγκεκριμένο κείμενο. Είναι επίσης αλήθεια ότι όταν κάποιος θέλει να μιλήσει για ένα θέμα θα χρησιμοποιήσει σχεδόν σίγουρα κάποιες συγκεκριμένες λέξεις, οπότε κατ' επέκταση, βλέποντας κανείς τις λέξεις που περιέχονται σε ένα κείμενο μπορεί να μαντέψει –πολύ συχνά με επιτυχία– το θέμα στο οποίο αναφέρεται. Γι' αυτό άλλωστε η αναζήτηση πληροφορίας, για παράδειγμα στο Διαδίκτυο, γίνεται χρησιμοποιώντας λέξεις-κλειδιά (keywords) και γι' αυτό η ομαδοποίηση των κειμένων γίνεται εξετάζοντας σε ποιο βαθμό δύο κείμενα χρησιμοποιούν τις ίδιες λέξεις. Ωστόσο δεν μπορεί να παραβλέψει κανείς το γεγονός ότι, όταν κάποιος μιλάει ή γράφει, δεν έχει σημασία μόνο ποιες λέξεις χρησιμοποιεί και πόσο συχνά αλλά και με ποια σειρά τις χρησιμοποιεί. Άλλωστε, με δεδομένο ένα σύνολο λέξεων, μόνο ελάχιστες από όλες τις δυνατές ακολουθίες λέξεων είναι συντακτικά ορθές και ακόμα λιγότερες νοηματικά ορθές. Μάλιστα είναι δυνατό η αναδιάταξη των λέξεων σε μία φράση, πρόταση ή κείμενο να επιφέρει και αλλαγή του νοήματος. Για παράδειγμα οι φράσεις:

«Αυτή η ευχάριστη ημέρα τελείωσε με μία δυσάρεστη έκπληξη.»
και
«Αυτή η δυσάρεστη ημέρα τελείωσε με μία ευχάριστη έκπληξη.»

χρησιμοποιούν και οι δύο τις ίδιες ακριβώς λέξεις και μάλιστα με την ίδια συχνότητα αλλά το νόημά τους είναι προφανώς πολύ διαφορετικό.

Από την παρατήρηση αυτή προκύπτει το ακόλουθο ερώτημα: Μπορούμε να βρούμε έναν τρόπο αναπαράστασης των κειμένων που να περιλαμβάνει όχι μόνο την πληροφορία για το ποιες λέξεις εμφανίζονται σε καθένα από αυτά και με ποια συχνότητα αλλά και την αλληλουχία των λέξεων; Και αν ναι, μπορεί πράγματι ο τρόπος αυτός να μας βοηθήσει να επιτύχουμε καλύτερη ταξινόμηση των κειμένων;

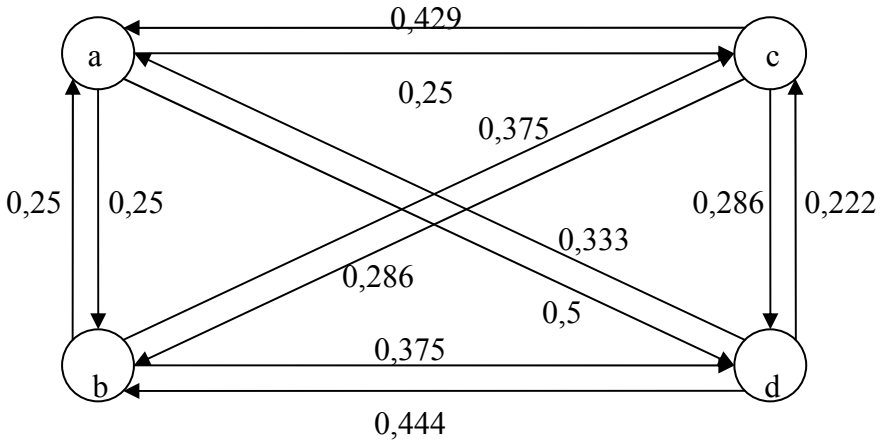
Μία απάντηση που μπορεί να δοθεί στο παραπάνω ερώτημα είναι η χρησιμοποίηση αλυσίδων Markov για την αναπαράσταση των κειμένων. Δηλαδή μπορούμε να θεωρήσουμε ότι το κείμενο είναι ένα σύστημα και οι λέξεις είναι οι δυνατές καταστάσεις οπότε η μετάβαση από λέξη σε λέξη αντιστοιχεί σε μετάβαση του συστήματος από κατάσταση σε κατάσταση. Με τον τρόπο αυτό μπορούμε να αναπαραστήσουμε κάθε κείμενο με μία αλυσίδα Markov (συγκεκριμένα Discrete-Time Markov Chain) και κατ' επέκταση με έναν –διδιάστατο πλέον και όχι μονοδιάστατο όπως συνήθως πίνακα– που είναι ο πίνακας μεταβάσεων αυτής της αλυσίδας Markov. Ο πίνακας αυτός, εφόσον είναι ο πίνακας μεταβάσεων της αλυσίδας Markov που αντιστοιχεί στο κείμενο, θα είναι ένας τετραγωνικός πίνακας διαστάσεων N x N, όπου N το πλήθος των διαφορετικών λέξεων που έχουμε συναντήσει, και κάθε γραμμή του θα εκφράζει ουσιαστικά την πιθανότητα η αντίστοιχη λέξη να ακολουθείται από κάθε άλλη λέξη από το σύνολο των λέξεων που έχουμε θεωρήσει. (Για λόγους που θα εξηγηθούν στη συνέχεια, θεωρούμε και μία μετάβαση από την τελευταία λέξη του κειμένου στην πρώτη.)

Ας υποθέσουμε, για παράδειγμα, ότι το σύνολο {a, b, c, d} είναι ένα σύνολο λέξεων και ότι έχουμε τα ακόλουθα δύο κείμενα:

i) a b d a c b a d b c d a d b c a d b c a c b d c a b d c d a d b

ii) b a c d a d a c d b a c d b a b a b c d c d b c b a d a b c b c

Σύμφωνα με τα παραπάνω, το κείμενο (i) θα μπορούσε να αναπαρασταθεί με την ακόλουθη Μαρκοβιανή αλυσίδα:



Σχήμα 20: Μαρκοβιανή αλυσίδα για την αναπαράσταση του κειμένου (i)

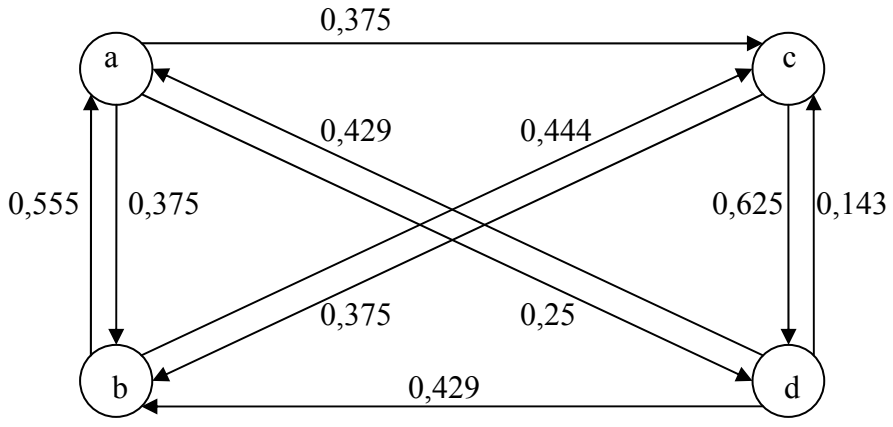
Ο πίνακας που περιέχει τη συχνότητα εμφάνισης κάθε λέξης είναι:

$$\pi_1 = [0,25 \quad 0,25 \quad 0,219 \quad 0,281]$$

Ο πίνακας μεταβάσεων είναι:

$$\tau_1 = \begin{bmatrix} 0 & 0,25 & 0,25 & 0,5 \\ 0,25 & 0 & 0,375 & 0,375 \\ 0,429 & 0,286 & 0 & 0,286 \\ 0,333 & 0,444 & 0,222 & 0 \end{bmatrix}$$

Αντίστοιχα για το κείμενο (ii) έχουμε:



Σχήμα 21: Μαρκοβιανή αλυσίδα για την αναπαράσταση του κειμένου (ii)

$$\pi_2 = [0,25 \quad 0,281 \quad 0,25 \quad 0,219]$$

$$\tau_2 = \begin{bmatrix} 0 & 0,375 & 0,375 & 0,25 \\ 0,555 & 0 & 0,444 & 0 \\ 0 & 0,375 & 0 & 0,625 \\ 0,429 & 0,429 & 0,143 & 0 \end{bmatrix}$$

το σημείο αυτό αξίζει να σημειωθούν ορισμένες παρατηρήσεις:

Παρατήρηση 1: Είναι ενδιαφέρον να παρατηρήσουμε ότι ο πίνακας π, που περιέχει τη συχνότητα με την οποία κάθε λέξη εμφανίζεται στο συγκεκριμένο κείμενο, αποτελεί ουσιαστικά την invariant probability distribution της αντίστοιχης Μαρκοβιανής αλυσίδας. Πράγματι:

$$\pi_1 \cdot \tau_1 = \begin{bmatrix} 0,25 & 0,25 & 0,219 & 0,281 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0,25 & 0,25 & 0,5 \\ 0,25 & 0 & 0,375 & 0,375 \\ 0,429 & 0,286 & 0 & 0,286 \\ 0,333 & 0,444 & 0,222 & 0 \end{bmatrix} = \begin{bmatrix} 0,25 & 0,25 & 0,219 & 0,281 \end{bmatrix} = \pi_1$$

και

$$\pi_2 \cdot \tau_2 = \begin{bmatrix} 0,25 & 0,281 & 0,25 & 0,219 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0,375 & 0,375 & 0,25 \\ 0,555 & 0 & 0,444 & 0 \\ 0 & 0,375 & 0 & 0,625 \\ 0,429 & 0,429 & 0,143 & 0 \end{bmatrix} = \begin{bmatrix} 0,25 & 0,281 & 0,25 & 0,219 \end{bmatrix} = \pi_2$$

Αυτό βέβαια ήταν αναμενόμενο, διότι όπως αναφέρθηκε κατά τη μελέτη των ιδιοτήτων των αλυσίδων Markov η invariant probability distribution εκφράζει το ποσοστό του χρόνου που το σύστημα περνά σε κάθε κατάσταση, οπότε στην προκειμένη περίπτωση εκφράζει το πόσο συχνά εμφανίζεται κάθε λέξη στο κείμενο. Συνεπώς η αναπαράσταση του κειμένου με μία αλυσίδα Markov περικλείει και την πληροφορία για τη συχνότητα εμφάνισης των λέξεων και επομένως αρκεί από μόνη της για την αναπαράσταση των κειμένων. Προφανώς στη συγκεκριμένη περίπτωση είναι πολύ πιο εύκολο υπολογιστικά να βρίσκουμε τη συχνότητα των λέξεων στο κείμενο απλά διαβάζοντάς το αντί να βρίσκουμε την invariant probability distribution της αντίστοιχης αλυσίδας Markov είτε λύνοντας την εξίσωση $\pi \cdot \tau = \pi$ είτε βρίσκοντας που συγκλίνουν οι δυνάμεις του πίνακα τ. Για αυτό το λόγο στις δοκιμές που θα περιγράψουμε στη συνέχεια υπολογίζουμε τη συχνότητα των λέξεων απ' ευθείας από την επεξεργασία του κειμένου. Ωστόσο είναι πολύ σημαντικό να επισημάνουμε ότι η πληροφορία αυτή εμπεριέχεται στη Μαρκοβιανή αλυσίδα, διότι σε κάποια άλλη περίπτωση θα ήταν ίσως ο μοναδικός τρόπος να υπολογιστεί.

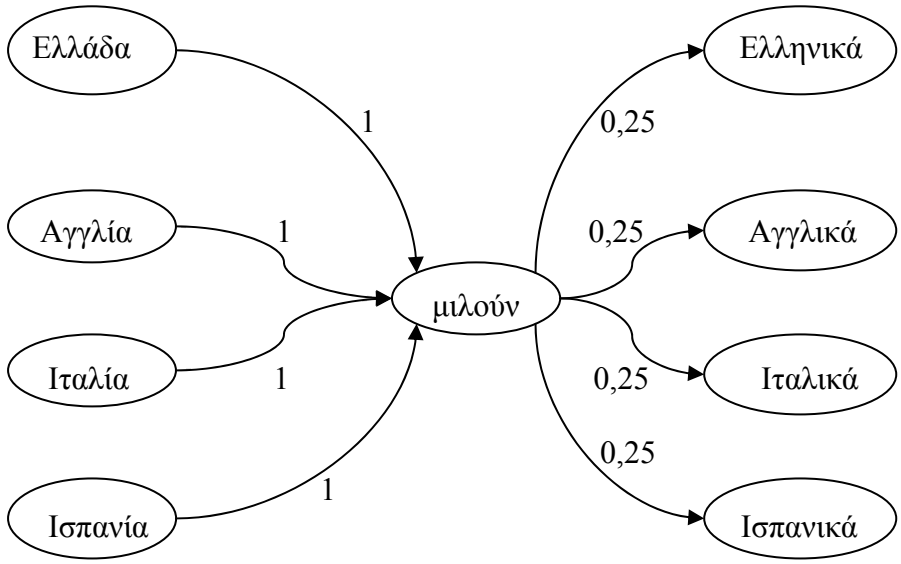
Παρατήρηση 2: Έχοντας υπόψη την επισήμανση που μόλις έγινε, μπορεί να εξηγηθεί ο λόγος για τον οποίο χρειάζεται να θεωρήσουμε μία μετάβαση από την τελευταία λέξη ενός κειμένου στην πρώτη. Για να έχει μία αλυσίδα Markov μία ακριβώς invariant probability distribution πρέπει, όπως αναφέρθηκε στην ενότητα 3.2, να είναι εργοδική. Από αυτό συνεπάγεται ότι πρέπει αυτή η αλυσίδα Markov να είναι irreducible, που σημαίνει ότι πρέπει να υπάρχει δυνατότητα από οποιαδήποτε κατάσταση να μεταβούμε, κάνοντας έναν πεπερασμένο αριθμό βημάτων, σε οποιαδήποτε άλλη. Είναι ωστόσο δυνατό να τελειώνει ένα κείμενο με μία λέξη η οποία δεν εμφανίζεται σε κανένα άλλο σημείο του κειμένου. Αν συμβεί αυτό, τότε θα υπάρχει στην αλυσίδα μία κατάσταση που αντιστοιχεί στη λέξη αυτή και από την οποία δε θα υπάρχει μετάβαση σε καμία άλλη λέξη, δηλαδή η αλυσίδα δε θα είναι πλέον irreducible. Για να αποφύγουμε αυτό το ενδεχόμενο, θεωρούμε μία μετάβαση

από την τελευταία λέξη του κειμένου στην πρώτη και έτσι εξασφαλίζουμε ότι σε κάθε περίπτωση η αντίστοιχη Μαρκοβιανή αλυσίδα θα είναι irreducible.

Παρατήρηση 3: Υπάρχει ένα μειονέκτημα στη χρήση αλυσίδων Markov για την αναπαράσταση κειμένων. Όπως αναφέρθηκε στην ενότητα 3.2, ένα βασικό χαρακτηριστικό των αλυσίδων Markov –και μάλιστα απαραίτητη προϋπόθεση για να χαρακτηριστεί μία αλυσίδα ως αλυσίδα Markov– είναι η «έλλειψη μνήμης». Όταν έχουμε μία Μαρκοβιανή αλυσίδα και βρισκόμαστε σε μία κατάσταση, η πιθανότητα να μεταβούμε σε κάποια άλλη κατάσταση δεν εξαρτάται από το ποιες ήταν οι προηγούμενες καταστάσεις. Αυτό ωστόσο δεν ισχύει στην περίπτωση των κειμένων –και γενικά του λόγου είτε γραπτού είτε προφορικού. Δηλαδή αν σε κάποιο σημείο ενός κειμένου εμφανίζεται κάποια λέξη, τότε η πιθανότητα να ακολουθούν κάποιες άλλες λέξεις δεν εξαρτάται μόνο από τη λέξη αυτή αλλά και από τις λέξεις που είχαν προηγηθεί –άλλοτε σε μεγαλύτερο και άλλοτε σε μικρότερο βαθμό. Ας υποθέσουμε για παράδειγμα τις ακόλουθες φράσεις:

«Στην Ελλάδα μιλούν Ελληνικά. Στην Αγγλία μιλούν Αγγλικά. Στην Ιταλία μιλούν Ιταλικά. Στην Ισπανία μιλούν Ισπανικά.»

Η αντίστοιχη Μαρκοβιανή αλυσίδα έχει την παρακάτω μορφή (για απλότητα έχουμε παραλείψει την κατάσταση που αντιστοιχεί στη λέξη «Στην» και τις μεταβάσεις από και προς αυτήν):



Σχήμα 22: Παράδειγμα Μαρκοβιανής αλυσίδας

Στο υποθετικό κείμενο του παραδείγματος η λέξη που βρίσκεται πριν από τη λέξη «μιλούν» καθορίζει κάθε φορά εκείνη που θα ακολουθήσει. Ωστόσο αυτό δεν είναι δυνατό να απεικονιστεί στην αντίστοιχη Μαρκοβιανή αλυσίδα. Μία επιπλέον επίπτωση που απορρέει από το γεγονός αυτό είναι το εξής: Όπως είναι γνωστό, οι δυνάμεις του πίνακα μεταβάσεων μιας Μαρκοβιανής αλυσίδας δίνουν τις πιθανότητες για μεταβάσεις αντίστοιχου αριθμού βημάτων. Στο παραπάνω παράδειγμα αν βρούμε το τετράγωνο του πίνακα μεταβάσεων, θα παρατηρήσουμε –όπως άλλωστε φαίνεται καθαρά και από το σχήμα– ότι αν βρισκόμαστε πχ. στη λέξη «Ελλάδα» η μεθεπόμενη λέξη μπορεί να είναι μία από τις «Ελληνικά», «Αγγλικά», «Ιταλικά» ή «Ισπανικά» με πιθανότητα 0,25 σε κάθε περίπτωση. Ωστόσο από το κείμενο

φαίνεται ότι αν βρισκόμαστε στη λέξη «Ελλάδα» τότε η μεθεπόμενη λέξη είναι σίγουρα η λέξη «Ελληνικά» και δεν υπάρχει ποτέ μετάβαση δύο βημάτων από τη λέξη «Ελλάδα» προς τη λέξη «Αγγλικά», «Ιταλικά» ή «Ισπανικά».

Το γεγονός αυτό σημαίνει ότι αναπαριστώντας ένα κείμενο με μία αλυσίδα Markov είναι πιθανό σε κάποιες περιπτώσεις να υπάρξει μικρή απώλεια πληροφορίας. Ωστόσο είναι δυνατό να βρεθούν τεχνικές, ώστε να ξεπεραστεί αυτό το πρόβλημα. Ένας τέτοιος τρόπος θα ήταν να θεωρήσουμε καταστάσεις που αποτελούνται όχι από μία λέξη αλλά από δύο ή περισσότερες. Αν για παράδειγμα στην προηγούμενη περίπτωση θεωρήσουμε καταστάσεις των 2 λέξεων, τότε θα είχαμε ότι από την κατάσταση {Στην Ελλάδα} πηγαίνουμε πάντα στην κατάσταση {μιλούν Ελληνικά}, οπότε το πρόβλημα έχει λυθεί. Θα μπορούσε μάλιστα να αποτελέσει αντικείμενο μελέτης η εύρεση του βέλτιστου αριθμού λέξεων ανά κατάσταση (βέβαια όσο αυξάνεται το πλήθος των λέξεων ανά κατάσταση αυξάνεται και το μέγεθος του χώρου καταστάσεων και κατά συνέπεια το υπολογιστικό κόστος). Πάντως, όπως θα φανεί παρακάτω από τα αποτελέσματα που προέκυψαν από κάποιες δοκιμές που κάναμε, η μέθοδος αυτή αποδίδει πολύ καλά και μάλιστα καλύτερα από τις υπόλοιπες μεθόδους που δοκιμάστηκαν.

Έχοντας ολοκληρώσει τις παρατηρήσεις αναφορικά με τον τρόπο αναπαράστασης των κειμένων, το επόμενο στάδιο είναι η μελέτη του τρόπου προσδιορισμού της απόστασης ανάμεσα σε δύο κείμενα. Στη συνέχεια θα μελετήσουμε 4 περιπτώσεις που θα μπορούσαν να χρησιμοποιηθούν:

1ος τρόπος: Ο πρώτος, και απλούστερος τρόπος, που θα μπορούσε να χρησιμοποιήσει κάποιος, για να υπολογίσει την απόσταση ανάμεσα σε δύο κείμενα είναι να εφαρμόσει την κλασική Ευκλείδεια απόσταση στους πίνακες π_1 και π_2, που περιέχουν τη συχνότητα εμφάνισης των λέξεων. Δηλαδή:

$$d_{1,2} = \sqrt{\sum_{i=1}^{n} (\pi_{1i} - \pi_{2i})^2} \qquad \text{(distance 1)}$$

όπου n το πλήθος των διαφορετικών λέξεων στα κείμενα που μελετούνται και π_{1i}, π_{2i} η συχνότητα εμφάνισης της λέξης i σε κάθε ένα από τα δύο κείμενα.

Ας υπολογίσουμε, για παράδειγμα, την απόσταση ανάμεσα στα κείμενα (i) και (ii) που είχαμε χρησιμοποιήσει προηγουμένως, όταν αναφερόμασταν στον τρόπο αναπαράστασης των κειμένων. Οι πίνακες π_1 και π_2 για τα κείμενα αυτά είναι:

$\pi_1 = [0,25 \quad 0,25 \quad 0,219 \quad 0,281]$

$\pi_2 = [0,25 \quad 0,281 \quad 0,25 \quad 0,219]$

Συνεπώς η μεταξύ τους απόσταση με τον παραπάνω τρόπο είναι:

$$d_{1,2} = \sqrt{(0,25-0,25)^2 + (0,25-0,281)^2 + (0,219-0,25)^2 + (0,281-0,219)^2} = 0,076$$

2ος τρόπος: Ο δεύτερος τρόπος είναι και πάλι εφαρμογή της Ευκλείδειας απόστασης αλλά συνυπολογίζοντας και τις διαφορές όχι μόνο στη συχνότητα εμφάνισης των λέξεων αλλά και στην αλληλουχία των λέξεων, δηλαδή χρησιμοποιώντας και τους πίνακες μεταβάσεων τ_1 και τ_2 των αντίστοιχων αλυσίδων Markov. Δηλαδή η απόσταση των δύο κειμένων για κάθε λέξη να θεωρείται ως το γινόμενο της διαφοράς της συχνότητας εμφάνισης αυτής της λέξης σε καθένα από τα δύο κείμενα επί την Ευκλείδεια απόσταση των γραμμών των πινάκων τ_1 και τ_2 που αντιστοιχούν σε αυτή τη λέξη:

$$d_{1,2} = \sqrt{\sum_{i=1}^{n} w_i (\pi_{1i} - \pi_{2i})^2} \qquad \text{(distance 2)}$$

όπου τα w_i υπολογίζονται από της σχέση

$$w_i = \sum_{j=1}^{n} (\tau_{1ij} - \tau_{2ij})^2$$

Για τα κείμενα του παραδείγματος έχουμε:

$$\tau_1 = \begin{bmatrix} 0 & 0,25 & 0,25 & 0,5 \\ 0,25 & 0 & 0,375 & 0,375 \\ 0,429 & 0,286 & 0 & 0,286 \\ 0,333 & 0,444 & 0,222 & 0 \end{bmatrix} \quad \text{και} \quad \tau_2 = \begin{bmatrix} 0 & 0,375 & 0,375 & 0,25 \\ 0,555 & 0 & 0,444 & 0 \\ 0 & 0,375 & 0 & 0,625 \\ 0,429 & 0,429 & 0,143 & 0 \end{bmatrix}$$

οπότε

$$w_1 = (0-0)^2 + (0,25-0,375)^2 + (0,25-0,375)^2 + (0,5-0,25)^2 = 0,094$$
$$w_2 = (0,25-0,555)^2 + (0-0)^2 + (0,375-0,444)^2 + (0,375-0)^2 = 0,238$$
$$w_3 = (0,429-0)^2 + (0,286-0,375)^2 + (0-0)^2 + (0,286-0,625)^2 = 0,307$$
$$w_4 = (0,333-0,429)^2 + (0,444-0,429)^2 + (0,222-0,143)^2 + (0-0)^2 = 0,016$$

και

$$d_{1,2} = \sqrt{0,094 \cdot (0,25-0,25)^2 + 0,238 \cdot (0,25-0,281)^2 + 0,307 \cdot (0,219-0,25)^2 + 0,016 \cdot (0,281-0,219)^2}$$
$$= 0,024$$

3ος τρόπος: Ο τρίτος τρόπος υπολογισμού της απόστασης, σε αντίθεση με τους δύο προηγούμενους, δε βασίζεται στην Ευκλείδεια απόσταση αλλά στη Mahalanobis. Όπως είδαμε στην ενότητα 2.4, η Mahalanobis distance υπερέχει σε σχέση με την Ευκλείδεια για το λόγο ότι λαμβάνει υπόψη το πόσο μεγάλες ή μικρές είναι οι διακυμάνσεις που παρουσιάζουν οι τιμές κάθε μεταβλητής, καθώς επίσης και το αν υπάρχουν συσχετίσεις ανάμεσα σε κάποιες μεταβλητές. Στην περίπτωση που τα αντικείμενα που πρέπει να ομαδοποιηθούν είναι κείμενα, είναι ενδιαφέρον να ληφθούν υπόψη αυτοί οι παράγοντες. Για παράδειγμα αν μία λέξη εμφανίζεται με την ίδια περίπου συχνότητα σε κάθε κείμενο, είναι λογικό να υποθέσουμε ότι πρόκειται για κάποια κοινή λέξη, δηλαδή κάποιο άρθρο, σύνδεσμο κλπ., η οποία δεν παρέχει κάποια χρήσιμη πληροφορία αναφορικά με το νόημα του κειμένου και

συνεπώς είναι προτιμότερο να της δώσουμε λιγότερη έμφαση. Αντίθετα αν μία λέξη εμφανίζεται πολύ συχνά σε κάποια κείμενα και πολύ σπάνια ή καθόλου σε κάποια άλλα, είναι πιθανότερο να πρόκειται για λέξη-κλειδί και επομένως να πρέπει να αντιμετωπιστεί με μεγαλύτερη έμφαση. Επίσης κάποιες λέξεις μπορεί να είναι συσχετισμένες, διότι εμφανίζονται μαζί σε φράσεις, όπως για παράδειγμα «Δίκτυα Υπολογιστών», «Λειτουργικά Συστήματα», «Βάσεις Δεδομένων», «Αρχαία Ελληνική Φιλοσοφία» κλπ., οπότε θα πρέπει να αντιμετωπιστούν με μεγαλύτερη σημασία διότι τέτοιου είδους φράσεις είναι ενδεικτικές για τη θεματική ενότητα στην οποία ανήκει ένα κείμενο.

Για να εφαρμόσουμε τη Mahalanobis distance, πρέπει προηγουμένως να κατασκευάσουμε τον covariance matrix των λέξεων. Ένας τρόπος είναι να κατασκευάσουμε για κάθε λέξη έναν μονοδιάστατο πίνακα που να περιέχει τις συχνότητες εμφάνισης αυτής της λέξης σε καθένα από τα κείμενα που έχουν εξεταστεί και κατόπιν να χρησιμοποιήσουμε αυτούς τους πίνακες για τον υπολογισμό του covariance matrix. Δηλαδή

$$C_{ij} = \sum_{k=1}^{N} (\rho_{ik} - \mu_i) \cdot (\rho_{jk} - \mu_j)$$

όπου C_{ij} η τιμή του covariance matrix για τις λέξεις i και j, ρ_{ik} η συχνότητα εμφάνισης της λέξης i στο κείμενο k, μ_i ο μέσος όρος των συχνοτήτων εμφάνισης της λέξης i και N το πλήθος των κειμένων.

Τότε η απόσταση των δύο κειμένων θα δίνεται από τη σχέση:

$$d_{1,2} = \sqrt{(\pi_1 - \pi_2) \cdot C \cdot (\pi_1 - \pi_2)^{\mathrm{T}}} \qquad \text{(distance 3)}$$

Πρέπει να επισημάνουμε ότι στη Mahalanobis distance χρησιμοποιείται ο αντίστροφος του covariance matrix, διότι συνήθως η επιδίωξη είναι να δοθεί μικρότερη έμφαση στις μεταβλητής που οι τιμές τους παρουσιάζουν μεγάλες διακυμάνσεις ή που είναι συσχετισμένες μεταξύ τους, ενώ αντίθετα στη συγκεκριμένη περίπτωση θέλουμε να δώσουμε μεγαλύτερη έμφαση σε αυτές τις περιπτώσεις και γι' αυτό έχουμε χρησιμοποιήσει στην παραπάνω σχέση τον ίδιο τον covariance matrix και όχι τον αντίστροφό του.

Σύμφωνα με τον τρόπο αυτό, η απόσταση των δύο κειμένων του παραδείγματος θα μπορούσε να υπολογιστεί με την ακόλουθη διαδικασία:
Κατασκευάζουμε τον πίνακα ρ που περιέχει τις συχνότητες εμφάνισης κάθε λέξης σε καθένα από τα κείμενα, καθώς και τον αντίστοιχο πίνακα μ με τους μέσους όρους (οι γραμμές αντιστοιχούν στις λέξεις και οι στήλες στα κείμενα):

$$\rho = \begin{bmatrix} 0,25 & 0,25 \\ 0,25 & 0,281 \\ 0,219 & 0,25 \\ 0,281 & 0,219 \end{bmatrix} \qquad \mu = \begin{bmatrix} 0,25 \\ 0,266 \\ 0,235 \\ 0,25 \end{bmatrix}$$

Κατόπιν υπολογίζουμε τον covariance matrix σύμφωνα με την παραπάνω σχέση:

$$C = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 4,81\cdot 10^{-4} & 4,81\cdot 10^{-4} & -9,61\cdot 10^{-4} \\ 0 & 4,81\cdot 10^{-4} & 4,81\cdot 10^{-4} & -9,61\cdot 10^{-4} \\ 0 & -9,61\cdot 10^{-4} & -9,61\cdot 10^{-4} & 19,22\cdot 10^{-4} \end{bmatrix}$$

Επίσης έχουμε:

$$\pi_1 - \pi_2 = \begin{bmatrix} 0 & -0,031 & -0,031 & 0,062 \end{bmatrix}$$

Συνεπώς η απόσταση των κειμένων είναι:

$$d_{1,2} = \sqrt{\begin{bmatrix} 0 & -0,031 & -0,031 & 0,062 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 4,81\cdot 10^{-4} & 4,81\cdot 10^{-4} & -9,61\cdot 10^{-4} \\ 0 & 4,81\cdot 10^{-4} & 4,81\cdot 10^{-4} & -9,61\cdot 10^{-4} \\ 0 & -9,61\cdot 10^{-4} & -9,61\cdot 10^{-4} & 19,22\cdot 10^{-4} \end{bmatrix} \cdot \begin{bmatrix} 0 \\ -0,031 \\ -0,031 \\ 0,062 \end{bmatrix}}$$

$$= \sqrt{0,166\cdot 10^{-4}} = 0,004$$

4ος τρόπος: Ο τέταρτος τρόπος υπολογισμού της απόστασης ανάμεσα σε δύο κείμενα βασίζεται επίσης στη Mahalanobis distance, δηλαδή γίνεται και πάλι μέσω της σχέσης

$$d_{1,2} = \sqrt{(\pi_1 - \pi_2)\cdot C \cdot (\pi_1 - \pi_2)^{\mathrm{T}}} \qquad \text{(distance 4)}$$

αλλά διαφέρει από τον προηγούμενο στο ότι για τον υπολογισμό του covariance matrix δεν χρησιμοποιείται ο πίνακας ρ που περιγράψαμε προηγουμένως αλλά ο πίνακας μεταβάσεων. Για το λόγο αυτό χρειάζεται να έχουμε έναν πίνακα μεταβάσεων, δηλαδή μια Μαρκοβιανή αλυσίδα, που να μην αντιπροσωπεύει ένα συγκεκριμένο κείμενο αλλά το σύνολο των κειμένων. Έχοντας στη διάθεσή μας τη Μαρκοβιανή αλυσίδα που περιγράφει κάθε κείμενο, είναι εύκολο να χρησιμοποιήσουμε μια συνάρτηση μέσου όρου ώστε να υπολογίσουμε τη Μαρκοβιανή αλυσίδα που αντιστοιχεί σε όλα τα κείμενα συνολικά. (Σύμφωνα με όσα είχαμε αναφέρει σχετικά με το «centroid» μίας ομάδας αντικειμένων στην ενότητα 2.7 αυτή η Μαρκοβιανή αλυσίδα μπορούμε να πούμε ότι αντιστοιχεί σε ένα υποθετικό κείμενο που είναι το «κέντρο βάρους» όλων των κειμένων που έχουν εξεταστεί.) Συγκεκριμένα η κατασκευή αυτής της αλυσίδας μπορεί να γίνει ως εξής: Έστω ότι έχουμε επεξεργαστεί N κείμενα και ο πίνακας μεταβάσεων που αντιστοιχεί σε αυτό το σύνολο κειμένων είναι ο $\tau_{o\lambda}$. Αν κάποια στιγμή θέλουμε να προσθέσουμε ένα νέο κείμενο με πίνακα μεταβάσεων τ, τότε ο νέος πίνακας $\tau_{o\lambda}$ θα είναι:

- Αν N = 0, τότε $\tau'_{o\lambda} = \tau$

Αν N > 0, τότε $\tau'_{o\lambda(ij)} =$

$$\begin{cases} \tau_{o\lambda(ij)}, & \text{αν η λέξη i δεν εμφανίζεται στο καινούριο κείμενο} \\ \dfrac{\mathrm{N}\cdot\tau_{o\lambda(ij)} + \tau_{ij}}{\mathrm{N}+1}, & \text{αν η λέξη i εμφανίζεται στο καινούριο κείμενο} \end{cases}$$

Ακολουθώντας την παραπάνω διαδικασία μπορούμε να υπολογίσουμε την απόσταση μεταξύ των δύο κειμένων του παραδείγματος ως εξής:

$$\tau_{o\lambda} = \begin{bmatrix} 0 & 0,313 & 0,313 & 0,375 \\ 0,403 & 0 & 0,41 & 0,188 \\ 0,215 & 0,331 & 0 & 0,456 \\ 0,381 & 0,437 & 0,183 & 0 \end{bmatrix} \qquad \mu = \begin{bmatrix} 0,25 \\ 0,25 \\ 0,25 \\ 0,25 \end{bmatrix}$$

$$C = \begin{bmatrix} 0,086 & -0,052 & 0,024 & -0,056 \\ -0,052 & 0,115 & -0,078 & -0,022 \\ 0,024 & -0,078 & 0,113 & -0,024 \\ -0,056 & -0,022 & -0,024 & 0,119 \end{bmatrix}$$

$$d_{1,2} = \sqrt{\begin{bmatrix} 0 & -0,031 & -0,031 & 0,062 \end{bmatrix} \cdot \begin{bmatrix} 0,086 & -0,052 & 0,024 & -0,056 \\ -0,052 & 0,115 & -0,078 & -0,022 \\ 0,024 & -0,078 & 0,113 & -0,024 \\ -0,056 & -0,022 & -0,024 & 0,119 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ -0,031 \\ -0,031 \\ 0,062 \end{bmatrix}} = 0,027$$

4.4 Υλοποίηση εφαρμογής - Αποτελέσματα

Στην ενότητα αυτή περιγράφουμε μία εφαρμογή που υλοποιήσαμε, για να δοκιμάσουμε στην πράξη τις τεχνικές που περιγράφτηκαν παραπάνω για την αναπαράσταση κειμένων και την εύρεση της μεταξύ τους απόστασης.

Η πληροφορία που εξάγεται από την επεξεργασία κάθε κειμένου αποθηκεύεται σε μία βάση δεδομένων. Για το σκοπό αυτό επιλέξαμε τη MySQL (έκδοση 4.0), διότι αν και μπορεί να μην υποστηρίζει κάποια χαρακτηριστικά που συναντά κανείς σε άλλα Συστήματα Διαχείρησης Βάσεων Δεδομένων (πχ. nested queries, triggers κλπ.) είναι αρκετά γρήγορη (άλλωστε στη συγκεκριμένη εφαρμογή αυτό που μας ενδιαφέρει είναι η δυνατότητα για γρήγορη ανάσυρση ή αποθήκευση πληροφορίας στη βάση και όχι κάποιες πολύπλοκες λειτουργίες διαχείρησης της βάσης) και επιπλέον διατίθεται δωρεάν στο Internet. Η βάση δεδομένων που χρησιμοποιήσαμε περιλαμβάνει 13 πίνακες. Στην επόμενη σελίδα φαίνονται τα ονόματα των πινάκων, τα πεδία που περιλαμβάνει ο καθένας και ο τύπος κάθε πεδίου και ακολουθεί μία περιγραφή του ρόλου που έχει κάθε πίνακας.

DOC	
DOC_FILE	varchar(30)
DOC_CAT	varchar(30)
DOC_DATE	varchar(30

DW	
DW_DOC	varchar(30)
DW_WORD	varchar(30)
DW_PROB	float

DT	
DT_DOC	varchar(30)
DT_FROM	varchar(30)
DT_TO	varchar(30)
DT_PROB	float

CW	
CW_CAT	varchar(30)
CW_WORD	varchar(30)
CW_PROB	float
CW_CHECKED	int

CT	
CT_CAT	varchar(30)
CT_FROM	varchar(30)
CT_TO	varchar(30)
CT_PROB	float
CT_CHECKED	int

TABLE1	
TB1_WORD	varchar(30)
TB1_PROB	float

TABLE2	
TB2_WORD	varchar(30)
TB2_PROB	float

TABLE3	
TB3_WORD	varchar(30)
TB3_PROB	float

TABLE4	
TB4_FROM	varchar(30)
TB4_TO	varchar(30)
TB4_PROB	float

TABLE5	
TB5_FROM	varchar(30)
TB5_TO	varchar(30)
TB5_PROB	float

TABLE6	
TB6_FROM	varchar(30)
TB6_TO	varchar(30)
TB6_PROB	float

COV1	
CV1_FROM	varchar(30)
CV1_TO	varchar(30)
CV1_PROB	float

COV2	
CV2_FROM	varchar(30)
CV2_TO	varchar(30)
CV2_PROB	float

Σχήμα 23: Οι πίνακες της βάσης δεδομένων

Ο πίνακας doc χρησιμεύει για να κρατούνται κάποιες πληροφορίες σχετικά με κάθε κείμενο που επεξεργάζεται. Συγκεκριμένα στον πίνακα αυτό αποθηκεύεται το όνομα του αρχείου στο οποίο υπήρχε το κείμενο (αυτό θεωρείται ως το όνομα του κειμένου), η κατηγορία στην οποία ανήκει το κείμενο, αν πρόκειται για κείμενο που χρησιμοποιείται για training, ή η λέξη «unknown», αν πρόκειται για κείμενο άγνωστης θεματικής ενότητας, και τέλος η ημερομηνία κατά την οποία έγινε η επεξεργασία του κείμενου. Φυσικά ο πίνακας αυτός θα μπορούσε να εμπλουτιστεί και με άλλα πεδία, όπως για παράδειγμα το όνομα του συγγραφέα του κειμένου, χρονολογία έκδοσης, γλώσσα κλπ., αν κάποια από αυτές τις πληροφορίες ήταν χρήσιμη για κάποιο λόγο.

Στον πίνακα dw αποθηκεύεται ο πίνακας π που αντιστοιχεί σε κάθε κείμενο, δηλαδή ο πίνακας που περιέχει τις συχνότητες εμφάνισης κάθε λέξης στο κείμενο αυτό. Στον πίνακα dt αποθηκεύεται ο πίνακας μεταβάσεων τ κάθε κειμένου. Αντίστοιχη είναι η χρησιμότητα των πινάκων cw και ct, με τη διαφορά ότι η πληροφορία που περιέχεται σε αυτούς δεν αναφέρεται σε κάθε κείμενο χωριστά αλλά κάθε κατηγορία κειμένων. Όπως δηλαδή στην προηγούμενη ενότητα είπαμε ότι διατηρούμε έναν πίνακα $\tau_{ολ}$ ο οποίος μπορούμε να πούμε ότι αντιστοιχεί στο κέντρο βάρους του συνόλου των κειμένων, έτσι και για κάθε κατηγορία διατηρούμε έναν πίνακα π και έναν πίνακα τ που αντιπροσωπεύουν όλα τα κείμενα που ανήκουν σε αυτή την κατηγορία. Αυτό, όπως θα δούμε στη συνέχεια, είναι χρήσιμο στην περίπτωση που θέλουμε να χρησιμοποιήσουμε την εφαρμογή για ταξινόμηση κειμένων, έχοντας κάνει βέβαια προηγουμένως ένα training.

Οι πίνακες cov1 και cov2 χρησιμοποιούνται για την αποθήκευση των covariance matrices που χρησιμοποιούμε στις μεθόδους 3 και 4 αντίστοιχα.

Τέλος οι πίνακες table1 έως table6 δεν αποθηκεύουν κάποια συγκεκριμένη πληροφορία αλλά απλώς χρησιμεύουν για την εκτέλεση των πράξεων και την αποθήκευση ενδιάμεσων αποτελεσμάτων κατά τη διαδικασία υπολογισμού της απόστασης των κειμένων. Στο παράρτημα που ακολουθεί υπάρχουν οι απαιτούμενες εντολές SQL για τη δημιουργία της βάσης δεδομένων, καθώς επίσης και ο κώδικας της εφαρμογής.

Ο κώδικας της εφαρμογής έχει υλοποιηθεί στη γλώσσα προγραμματισμού Java (έκδοση 1.4). Η επιλογή αυτή έγινε διότι η Java είναι μία γλώσσα που προσφέρει πάρα πολλά και σημαντικά πλεονεκτήματα. Κατ' αρχήν είναι αντικειμενοστραφής, γεγονός που την καθιστά εύκολη στη χρήση και κυρίως αυξάνει την αναγνωσιμότητα των προγραμμάτων και τη δυνατότητα για διόρθωση, συντήρηση και επαναχρησιμοποίησή τους, έννοιες-κλειδιά στο χώρο της Τεχνολογίας Λογισμικού. Επίσης έχει το σημαντικό πλεονέκτημα της φορητότητας, καθώς είναι ανεξάρτητη τόσο από το υλικό όσο και από το λειτουργικό σύστημα και τέλος είναι ιδανική για Δικτυακές εφαρμογές. Στη συγκεκριμένη περίπτωση μάλιστα αυτό είναι αρκετά σημαντικό, καθώς η εφαρμογή αυτή θα μπορούσε να επεκταθεί ώστε να λειτουργεί δικτυακά ή να συνεργαζέται με τέτοιου είδους προγράμματα, όπως spiders –προγράμματα που η λειτουργία τους είναι να διατρέχουν ιστοσελίδες στο Διαδίκτυο και να συγκεντρώνουν πληροφορία– ή Mobile Agents κλπ.

Η εφαρμογή αποτελείται από 4 προγράμματα (κλάσεις) που είναι τα εξής: ProcessDocument, Covariance1, Covariance2 και CalculateDistance. Στη συνέχεια περιγράφουμε συνοπτικά τη λειτουργία καθενός από αυτά.

Η κλάση ProcessDocument χρησιμοποιείται για την επεξεργασία κειμένων. Κάθε φορά που ο χρήστης εκτελεί το πρόγραμμα καλείται αρχικά να προσδιορίσει αν επιθυμεί να κάνει training του συστήματος, οπότε πρέπει να επιλέξει μία από τις προβλεπόμενες κατηγορίες, ή αν τα κείμενα που θα δοθούν για επεξεργασία είναι άγνωστης θεματικής ενότητας. Στη συνέχεια καλείται να επιλέξει τα κείμενα που επιθυμεί να επεξεργαστεί. Για το σκοπό αυτό εμφανίζεται στην οθόνη ένα παράθυρο μέσα από το οποίο ο χρήστης μπορεί να

διατρέχει το σύστημα αρχείων του υπολογιστή, ώστε να εντοπίσει και να επιλέξει τα αρχεία που περιέχουν τα επιθυμητά κείμενα. Μάλιστα ο χρήστης μπορεί να επιλέξει για επεξεργασία περισσότερα από ένα κείμενα κάθε φορά, αρκεί όλα να βρίσκονται στον ίδιο φάκελο. Κατόπιν το σύστημα ελέγχει αν τα αρχεία που επέλεξε ο χρήστης είναι πράγματι αρχεία κειμένου (δηλαδή της μορφής <όνομα_αρχείου>.txt) και επίσης –με τη βοήθεια του πίνακα doc της βάσης δεδομένων– αν κάποιο ή κάποια από αυτά έχουν επαναχρησιμοποιηθεί στο παρελθόν. Αν υπάρχουν κείμενα που δεν πληρούν τις δύο παραπάνω προϋποθέσεις, το πρόγραμμα τα αγνοεί, ενημερώνοντας κατάλληλα το χρήστη. Έπειτα ξεκινά η διαδικασία της επεξεργασίας των κειμένων, η οποία αποτελείται από τρία στάδια (κάθε στάδιο υλοποιείται από μία χωριστή μέθοδο της κλάσης).

Κατά το πρώτο στάδιο γίνεται ένας «καθαρισμός» του κειμένου. Συγκεκριμένα θεωρούμε δύο ομάδες χαρακτήρων: τους «έγκυρους» χαρακτήρες, στους οποίους ανήκουν όλοι οι λατινικοί χαρακτήρες από a έως z και τα αντίστοιχα κεφαλαία, και στους «άκυρους» χαρακτήρες που είναι οτιδήποτε άλλο (πχ: 2, @, &, #, α κλπ). Κάθε κείμενο διαβάζεται χαρακτήρα προς χαρακτήρα ξεκινώντας από την αρχή. Κάθε φορά που συναντάται ένας έγκυρος χαρακτήρας παραμένει όπως έχει (αν πρόκειται για κεφαλαίο γράμμα μετατρέπεται στο αντίστοιχο μικρό, ώστε λέξεις όπως this, This, THIS, ThiS, κλπ να μη θεωρούνται διαφορετικές). Αντίθετα αν πρόκειται για μη έγκυρο χαρακτήρα τότε αυτός παραλείπεται και στη θέση του τοποθετείται ένα κενό διάστημα (αν ο προηγούμενος χαρακτήρας ήταν επίσης άκυρος, τότε δεν τοποθετούμε και πάλι κενό, ώστε να μην έχουμε στο «καθαρισμένο» κείμενο διαδοχικά κενά). Το κείμενο που προκύπτει από αυτή τη διαδικασία φιλτραρίσματος τοποθετείται σε ένα συγκεκριμένο αρχείο, που δημιουργείται από το πρόγραμμα, ώστε να είναι διαθέσιμο για τα επόμενα στάδια της διαδικασίας. Προφανώς για οικονομία χώρου υπάρχει μόνο ένα τέτοιο βοηθητικό αρχείο και κάθε φορά γράφεται σε αυτό το τρέχον κείμενο. Μάλιστα είναι εύκολο το αρχείο αυτό να διαγράφεται αυτόματα μετά την ολοκλήρωση της διαδικασίας αλλά από την άλλη πλευρά είναι καλό να παραμένει, ώστε σε περίπτωση που το επιθυμεί ο χρήστης να μπορεί να επιβεβαιώσει ότι το συγκεκριμένο στάδιο ολοκληρώθηκε σωστά. Επίσης είναι ενδιαφέρον να σημειώσουμε ότι η διαδικασία αυτή επιδέχεται παραμετροποίηση, δηλαδή θα μπορούσε κάποιος να αποφασίσει να τροποποιήσει τον κώδικα ώστε να μην αγνοεί για παράδειγμα τους χαρακτήρες του Ελληνικού αλφαβήτου ή κάποιους ειδικούς χαρκτήρες που θα καθορίσει ο ίδιος κλπ.

Στο δεύτερο στάδιο διαβάζεται το «καθαρισμένο» κείμενο που έχει προκύψει και αποθηκεύονται στη μνήμη οι πίνακες π και τ που περιέχουν τη συχνότητα εμφάνισης των λέξεων και τις πιθανότητες μεταβάσεων αντίστοιχα για το κάθε κείμενο. Υπενθυμίζουμε ότι θεωρούμε και μία επιπλέον μετάβαση από την τελευταία λέξη του κειμένου στην πρώτη για τους λόγους που εξηγήθηκαν στην προηγούμενη ενότητα.

Το τρίτο και τελευταίο στάδιο είναι αυτό που αναλαμβάνει την αποθήκευση των πληροφοριών που συλλέχθηκαν για το κείμενο στους αντίστοιχους πίνακες της βάσης δεδομένων. Συγκεκριμένα εισάγεται πληροφορία στους πίνακες doc, dw και dt, ενημερώνονται τα στοιχεία των πινάκων cw και ct που αναφέρονται στο «κέντρο βάρους» του συνόλου όλως των κειμένων και σε περίπτωση που το συγκεκριμένο κείμενο ανήκει σε μία κατηγορία –δηλαδή πρόκειται για διαδικασία εκπαίδευσης του συστήματος– ενημερώνονται και τα στοιχεία των πινάκων cw και ct που αναφέρονται στο «κέντρο βάρους» της αντίστοιχης κατηγορίας.

Οι κλάσεις Covariance1 και Covariance2 χρησιμοποιούνται για την κατασκευή των covariance matrices που απαιτούνται για τον υπολογισμό των αποστάσεων με χρήση των μεθόδων 3 και 4, όπως περιγράφεται στην προηγούμενη ενότητα. Αυτό που αξίζει να σημειωθεί στο σημείο αυτό είναι το εξής: Για τον υπολογισμό του covariance matrix στην τελευταία περίπτωση χρησιμοποιείται η πρώτη δύναμη του πίνακα μεταβάσεων. Είναι πιθανό οι συσχετίσεις μεταξύ των μεταβλητών να αρχίσουν να αποκαλύπτονται όχι από την πρώτη

κιόλας μετάβαση αλλά ύστερα από κάποιο μικρό ή μεγάλο αριθμό βημάτων, γεγονός που σημαίνει ότι σε μια τέτοια περίπτωση θα έπρεπε να χρησιμοποιηθεί όχι η πρώτη δύναμη αυτού του πίνακα αλλά κάποια υψηλότερη. Στη συγκεκριμένη περίπτωση προτιμήσαμε να χρησιμοποιήσουμε την πρώτη δύναμη (δηλαδή τον ίδιο τον πίνακα μεταβάσεων) για τον εξής λόγο: Λόγω της φύσης της γλώσσας, σε οποιαδήποτε λέξη και αν βρισκόμαστε είναι σχεδόν σίγουρο ότι το πολύ μετά από δύο-τρεις λέξεις θα ακολουθεί κάποια πολύ κοινή λέξη, όπως άρθρο, σύνδεσμος κλπ, και συνεπώς οι δυνάμεις του πίνακα μεταβάσεων θα συγκλίνουν πολύ γρήγορα, δηλαδή θα βλέπουμε ότι όλες ή σχεδόν όλες οι λέξεις είναι συσχετισμένες αν θεωρήσουμε μεταβάσεις μεγάλου αριθμού βημάτων. Ωστόσο σε ένα άλλο πρόβλημα μπορεί η κατάσταση να ήταν διαφορετική. Τελικά αυτό που χρειάζεται να επισημανθεί είναι ότι και αυτό το σημείο θα ήταν ίσως καλό να αντιμετωπιστεί ως παράμετρος της μεθόδου και να γίνονται δοκιμές, ώστε να βρεθεί η καταλληλότερη τιμή ανάλογα με το είδος των αντικειμένων που μελετούνται κάθε φορά.

Η κλάση CalculateDistance είναι αυτή που υπολογίζει την απόσταση των κειμένων. Κατά την εκκίνηση του προγράμματος ο χρήστης καλείται να επιλέξει έναν από τους 4 τρόπους υπολογισμού απόστασης που περιγράφτηκαν στην προηγούμενη ενότητα και κατόπιν το είδος της εργασίας που επιθυμεί. Συγκεκριμένα υπάρχουν δύο δυνατότητες: Ο χρήστης να ζητήσει να υπολογιστεί α) η απόσταση είτε μεταξύ δύο συγκεκριμένων κειμένων είτε μεταξύ ενός συγκεκριμένου κειμένου και όλων των υπόλοιπων είτε μεταξύ όλων των κειμένων ή β) η κατηγορία στην οποία ανήκει είτε ένα συγκεκριμένο κείμενο είτε όλα τα άγνωστα κείμενα. Δηλαδή η εφαρμογή υποστηρίζει ένα σενάριο υπολογισμού αποστάσεων και ένα σενάριο ταξινόμησης. Ουσιαστικά πρόκειται για την ίδια διαδικασία, διότι η ταξινόμηση ενός κειμένου γίνεται βρίσκοντας απλά την απόστασή του από το «κέντρο βάρους» κάθε κατηγορίας. Αυτός ακριβώς είναι και ο λόγος για τον οποίο είπαμε προηγουμένως ότι για κάθε κατηγορία διατηρούμε στη βάση δεδομένων ένα πίνακα π και έναν πίνακα τ που αντιστοιχεί στο σύνολο των κειμένων αυτής της κατηγορίας. Βέβαια ένας εναλλακτικός τρόπος θα ήταν με χρήση του αλγορίθμου k-Nearest Neighbor, που έχουμε περιγράψει σε προηγούμενες ενότητες, δηλαδή να συγκρίνουμε το υπό εξέταση κείμενο με κάθε ένα από τα γνωστά, ώστε να βρούμε τα k κείμενα που μοιάζουν περισσότερο με αυτό και κατόπιν με βάση αυτά να αποφασίσουμε για την κατηγορία του συγκεκριμένου κειμένου. Ωστόσο η μέθοδος αυτή θα απαιτούσε πολύ μεγαλύτερο αριθμό συγκρίσεων και συνεπώς θα ήταν λιγότερο αποδοτική.

Για την εκτέλεση των δοκιμών κατασκευάσαμε ένα σύνολο κειμένων, που αποτελείται από 4 ομάδες κειμένων ως εξής:

α) art1, art2, art3, art4: αποσπάσματα από ένα βιβλίο με θέμα την Αρχαία Ελληνική τέχνη
β) hist1, hist2, hist3, hist4: αποσπάσματα από ένα βιβλίο με θέμα την ιστορία της Κίνας
γ) med1, med2, med3, med4: αποσπάσματα από ένα βιβλίο με ιατρικά θέματα
δ) phil1, phil2, phil3, phil4: αποσπάσματα από ένα βιβλίο φιλοσοφίας

Προφανώς αναμένουμε ότι τα κείμενα που ανήκουν στην ίδια τετράδα θα απέχουν μεταξύ τους λιγότερο από ό,τι τα κείμενα που ανήκουν σε διαφορετικές τετράδες.

Συγκεκριμένα χρησιμοποιήσαμε την εφαρμογή για να κάνουμε δύο ειδών δοκιμές. Στην πρώτη περίπτωση εισάγαμε στο σύστημα όλα τα κείμενα και υπολογίσαμε όλες τις μεταξύ τους αποστάσεις, χρησιμοποιώντας τους 4 τρόπους υπολογισμού της απόστασης που έχουμε περιγράψει. Τα αποτελέσματα που προέκυψαν σε κάθε περίπτωση φαίνονται στους πίνακες των επόμενων σελίδων.

	art1	art2	art3	art4	hist1	hist2	hist3	hist4	med1	med2	med3	med4	phil1	phil2	phil3	phil4
art1	0	52	56	39	101	91	96	94	108	100	102	96	122	132	136	124
art2	52	0	50	39	96	88	92	90	102	91	95	90	118	126	130	119
art3	56	50	0	42	97	85	91	88	102	90	94	89	119	126	130	120
art4	39	39	42	0	99	89	96	92	106	94	97	91	122	130	134	123
hist1	101	96	97	99	0	53	54	39	78	71	75	77	98	97	113	98
hist2	91	88	85	89	53	0	53	38	79	76	78	77	98	102	112	98
hist3	96	92	91	96	54	53	0	42	79	75	75	77	100	101	116	100
hist4	94	90	88	92	39	38	42	0	78	72	74	75	99	101	113	99
med1	108	102	102	106	78	79	79	78	0	59	59	52	94	90	102	90
med2	100	91	90	94	71	76	75	72	59	0	56	48	101	99	111	99
med3	102	95	94	97	75	78	75	74	59	56	0	54	93	93	103	92
med4	96	90	89	91	77	77	77	75	52	48	54	0	106	105	115	103
phil1	122	118	119	122	98	98	100	99	94	101	93	106	0	57	58	40
phil2	132	126	126	130	97	102	101	101	90	99	93	105	57	0	56	39
phil3	136	130	130	134	113	112	116	113	102	111	103	115	58	56	0	48
phil4	124	119	120	123	98	98	100	99	90	99	92	103	40	39	48	0

Πίνακας 1: Υπολογισμός αποστάσεων με χρήση της distance 1

	art1	art2	art3	art4	hist1	hist2	hist3	hist4	med1	med2	med3	med4	phil1	phil2	phil3	phil4
art1	0	40	39	29	65	63	65	64	73	71	66	72	70	81	78	73
art2	40	0	41	30	70	67	64	65	71	70	63	68	70	79	74	70
art3	39	41	0	33	64	61	63	61	73	69	65	71	73	81	78	74
art4	29	30	33	0	66	64	63	63	66	68	61	65	70	82	75	72
hist1	65	70	64	66	0	41	40	31	63	62	61	62	68	71	71	66
hist2	63	67	61	64	41	0	39	30	62	64	61	61	67	73	73	67
hist3	65	64	63	63	40	39	0	32	64	63	59	63	67	72	71	68
hist4	64	65	61	63	31	30	32	0	62	62	59	61	67	72	71	67
med1	73	71	73	66	63	62	64	62	0	46	49	37	68	68	72	64
med2	71	70	69	68	62	64	63	62	46	0	49	37	73	75	81	71
med3	66	63	65	61	61	61	59	59	49	49	0	43	63	69	72	67
med4	72	68	71	65	62	61	63	61	37	37	43	0	71	76	78	70
phil1	70	70	73	70	68	67	67	67	68	73	63	71	0	42	41	32
phil2	81	79	81	82	71	73	72	72	68	75	69	76	42	0	39	28
phil3	78	47	7	75	71	73	71	71	72	81	72	78	41	39	0	31
phil4	73	70	74	72	66	67	68	67	64	71	67	70	32	28	31	0

Πίνακας 2: Υπολογισμός αποστάσεων με χρήση της distance 2

	art1	art2	art3	art4	hist1	hist2	hist3	hist4	med1	med2	med3	med4	phil1	phil2	phil3	phil4
art1	0	5	9	2	26	23	27	25	29	26	28	21	38	44	44	39
art2	5	0	5	1	23	20	24	22	26	22	25	18	35	40	39	36
art3	9	5	0	5	21	17	20	19	22	19	20	14	30	35	34	31
art4	2	1	5	0	27	24	27	25	31	25	28	22	38	43	42	39
hist1	26	23	21	27	0	0	1	0	8	3	6	1	16	22	23	19
hist2	23	20	17	24	0	0	3	2	11	6	9	1	20	24	26	21
hist3	27	24	20	27	1	3	0	0	10	4	7	0	19	23	26	20
hist4	25	22	19	25	0	2	0	0	10	5	7	0	18	24	25	20
med1	29	26	22	31	8	11	10	10	0	2	1	8	11	15	16	13
med2	26	22	19	25	3	6	4	5	2	0	3	3	15	20	20	17
med3	28	25	20	28	6	9	7	7	1	3	0	5	14	18	19	15
med4	21	18	14	22	1	1	0	0	8	3	5	0	21	25	25	22
phil1	38	35	30	38	16	20	19	18	11	15	14	21	0	5	4	1
phil2	44	40	35	43	22	24	23	24	15	20	18	25	5	0	0	2
phil3	44	39	34	42	23	26	26	25	16	20	19	25	4	0	0	0
phil4	39	36	31	39	19	21	20	20	13	17	15	22	1	2	0	0

Πίνακας 3: Υπολογισμός αποστάσεων με χρήση της distance 3

	art1	art2	art3	art4	hist1	hist2	hist3	hist4	med1	med2	med3	med4	phil1	phil2	phil3	phil4
art1	0	77	74	52	174	189	191	188	163	153	172	169	176	184	182	179
art2	77	0	78	56	178	187	188	187	159	162	169	174	168	176	174	168
art3	74	78	0	54	177	185	186	187	155	167	163	168	164	171	173	166
art4	52	56	54	0	173	186	187	187	165	148	166	168	168	176	174	170
hist1	174	178	177	173	0	73	71	55	138	133	145	142	153	160	156	157
hist2	189	187	185	186	73	0	77	48	151	148	161	158	171	172	168	171
hist3	191	188	186	187	71	77	0	62	153	147	160	158	168	174	172	172
hist4	188	187	187	187	55	48	62	0	149	145	157	154	165	171	169	168
med1	163	159	155	165	138	151	153	149	0	69	89	59	145	146	146	146
med2	153	162	167	148	133	148	147	145	69	0	82	60	145	148	145	143
med3	172	169	163	166	145	161	160	157	89	82	0	55	145	143	143	145
med4	169	174	168	168	142	158	158	154	59	60	55	0	155	152	154	152
phil1	176	168	164	168	153	171	168	165	145	145	145	155	0	55	50	40
phil2	184	176	171	176	160	172	174	171	146	148	143	152	55	0	47	36
phil3	182	174	173	174	156	168	172	169	146	145	143	154	50	47	0	53
phil4	179	168	166	170	157	171	172	168	146	143	145	152	40	36	53	0

Πίνακας 4: Υπολογισμός αποστάσεων με χρήση της distance 4

Στη δεύτερη περίπτωση χρησιμοποιήσαμε τα δύο πρώτα κείμενα κάθε τετράδας για να κάνουμε training και κατόπιν δοκιμάσαμε να ταξινομήσουμε τα υπόλοιπα κείμενα. Τα αποτελέσματα που πήραμε για κάθε τύπο απόστασης φαίνονται στους παρακάτω πίνακες.

	art	hist	med	phil
art3	46	87	91	119
art4	29	90	95	123
hist3	91	46	71	97
hist4	88	28	69	96
med3	95	72	50	88
med4	89	72	40	102
phil3	130	109	102	49
phil4	119	94	90	27

Πίνακας 5: Ταξινόμηση των κειμένων με χρήση της distance 1

	art	hist	med	phil
art3	33	59	65	71
art4	20	61	61	70
hist3	62	33	59	65
hist4	62	20	57	66
med3	61	58	42	60
med4	66	58	25	69
phil3	73	72	72	32
phil4	69	64	62	18

Πίνακας 6: Ταξινόμηση των κειμένων με χρήση της distance 2

	art	hist	med	phil
art3	5	19	21	32
art4	1	25	28	40
hist3	14	0	7	21
hist4	13	1	7	21
med3	15	2	2	16
med4	8	8	8	23
phil3	26	3	2	1
phil4	24	3	1	1

Πίνακας 7: Ταξινόμηση των κειμένων με χρήση της distance 3

	art	hist	med	phil
art3	81	180	155	166
art4	54	178	155	171
hist3	189	75	149	170
hist4	187	44	146	167
med3	175	151	77	143
med4	179	158	55	153
phil3	169	159	141	64
phil4	163	161	138	33

Πίνακας 8: Ταξινόμηση των κειμένων με χρήση της distance 4

Παράρτημα

Α) Εντολές SQL για τη δημιουργία της βάσης δεδομένων

```
create table DOC
(
  DOC_FILE                varchar(30)              not null,
  DOC_CAT                 varchar(30)              not null,
  DOC_DATE                varchar(30)              not null,
  primary key (DOC_FILE)
);

create table DW
(
  DW_DOC                  varchar(30)              not null,
  DW_WORD                 varchar(30)              not null,
  DW_PROB                 float                    not null,
  primary key (DW_DOC, DW_WORD)
);

create table DT
(
  DT_DOC                  varchar(30)              not null,
  DT_FROM                 varchar(30)              not null,
  DT_TO                   varchar(30)              not null,
  DT_PROB                 float                    not null,
  primary key (DT_DOC, DT_FROM, DT_TO)
);

create table CW
(
  CW_CAT                  varchar(30)              not null,
  CW_WORD                 varchar(30)              not null,
  CW_PROB                 float                    not null,
  CW_CHECKED              int                      not null,
  primary key (CW_CAT, CW_WORD)
);

create table CT
(
  CT_CAT                  varchar(30)              not null,
  CT_FROM                 varchar(30)              not null,
  CT_TO                   varchar(30)              not null,
  CT_PROB                 float                    not null,
  CT_CHECKED              int                      not null,
```

```
  primary key (CT_CAT, CT_FROM, CT_TO)
);

create table TABLE1
(
  TB1_WORD              varchar(30)             not null,
  TB1_PROB              float                   not null,
  primary key (TB1_WORD)
);

create table TABLE2
(
  TB2_WORD              varchar(30)             not null,
  TB2_PROB              float                   not null,
  primary key (TB2_WORD)
);

create table TABLE3
(
  TB3_WORD              varchar(30)             not null,
  TB3_PROB              float                   not null,
  primary key (TB3_WORD)
);

create table TABLE4
(
  TB4_FROM              varchar(30)             not null,
  TB4_TO                varchar(30)             not null,
  TB4_PROB              float                   not null,
  primary key (TB4_FROM, TB4_TO)
);

create table TABLE5
(
  TB5_FROM              varchar(30)             not null,
  TB5_TO                varchar(30)             not null,
  TB5_PROB              float                   not null,
  primary key (TB5_FROM, TB5_TO)
);

create table TABLE6
(
  TB6_FROM              varchar(30)             not null,
  TB6_TO                varchar(30)             not null,
  TB6_PROB              float                   not null,
  primary key (TB6_FROM, TB6_TO)
);

create table COV1
(
```

```
  CV1_FROM              varchar(30)              not null,
  CV1_TO                varchar(30)              not null,
  CV1_PROB              float                    not null,
  primary key (CV1_FROM, CV1_TO)
);

create table COV2
(
  CV2_FROM              varchar(30)              not null,
  CV2_TO                varchar(30)              not null,
  CV2_PROB              float                    not null,
  primary key (CV2_FROM, CV2_TO)
);
```

Β) Κώδικας της εφαρμογής σε Java

```
package org.dskoutas.tc;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashSet;
import java.util.Hashtable;

import javax.swing.JFileChooser;
import javax.swing.JOptionPane;

public class ProcessDocument
{

  static String cat;
  static String fileName;
  static String filePath;
```

```
static String filtered;
static Connection con;
static int numOfFiles;
static Hashtable hashTable;
static Hashtable stateOccurrences;
static int numOfTransitions;

public static void main(String[] args)
{
 // select category
 String[] catList =
   { "art", "history", "medicine", "philosophy", "unknown" };
 cat =
  (String) JOptionPane.showInputDialog(
    null,
    "Please select a category",
    "Category",
    JOptionPane.QUESTION_MESSAGE,
    null,
    catList,
    catList[0]);

 if (cat == null)
  System.exit(1);

 // select texts
 JFileChooser fc = new JFileChooser();
 fc.setFileSelectionMode(JFileChooser.FILES_ONLY);
 fc.setMultiSelectionEnabled(true);
 fc.showOpenDialog(null);
 File[] files = fc.getSelectedFiles();
 numOfFiles = files.length;

 if (numOfFiles == 0)
  System.exit(1);

 // this array is used to help skip files that have been used before
 // or are not of the right type (*.txt files)
 int[] skipBit = new int[numOfFiles];

 // connect to database
 try
 {
  // connect to database
  String mm_driver = "org.gjt.mm.mysql.Driver";
  String url =
    "jdbc:mysql://127.0.0.1:3306/tcdb?useUnicode=true&characterEncoding=ISO-8859-7";

  Class.forName(mm_driver);
```

```
con = DriverManager.getConnection(url);

PreparedStatement stmt =
  con.prepareStatement("SELECT * FROM doc WHERE doc_file = ?");
ResultSet results;

String msg = "\nYou have selected the following files:";
String fileExtension;
for (int i = 0; i < numOfFiles; i++)
{
  fileName = files[i].getName();
  filePath = files[i].getAbsolutePath();
  fileExtension = fileName.substring(fileName.lastIndexOf("."));
  stmt.setString(1, fileName);
  results = stmt.executeQuery();
  msg += ("\n" + (i + 1) + ") " + fileName);
  if (results.next())
  {
    msg
      += (" (warning: will be skipped, it was used on "
        + results.getString(3)
        + ")");
    skipBit[i] = 1;
  }
  else
  {
    if (!fileExtension.equalsIgnoreCase(".txt"))
    {
      msg += (" (warning: will be skipped, not a '*.txt' file)");
      skipBit[i] = 1;
    }
    else
    {
      skipBit[i] = 0;
    }
  }
}
System.out.println(msg);

int tmp1 =
  JOptionPane.showConfirmDialog(
    null,
    msg + "\n\nDo you wish to proceed?",
    "Verify choices",
    JOptionPane.YES_NO_OPTION);
if (tmp1 != 0)
  System.exit(1);

// start process of each file
System.out.println("\n\n* * *  Operation started  * * *");
```

```
  for (int i = 0; i < numOfFiles; i++)
  {
   fileName = files[i].getName();
   filePath = files[i].getAbsolutePath();
   if (skipBit[i] == 1)
   {
    System.out.print("\n" + (i + 1) + ") " + fileName + ": skipped");
   }
   else
   {
    System.out.print("\n" + (i + 1) + ") " + fileName + ": started...");

    // filter file
    System.out.print("\n     Filtering file...");
    filterFile();

    if (numOfTransitions < 1)
     continue;

    // count transitions
    System.out.print("\n     Counting words and transitions...");
    countWordsAndTransitions();

    // update database
    System.out.print("\n     Updating database...");
    updateDB();

    System.out.print("\n     " + fileName + ": finished!");
   }
  }

  System.out.println("\n\n* * *  Operation completed succesfully  * * *");
}
catch (SQLException sqle)
{
  System.out.println("Error: " + sqle.getMessage());
}
catch (ClassNotFoundException e)
{
  e.printStackTrace();
}
finally
{
  try
  {
   con.close();
  }
  catch (SQLException e)
  {
```

```
      e.printStackTrace();
    }
   }
   System.exit(0);
}

private static void filterFile()
{
  try
  {
    filtered = "filteredFile.txt";

    FileInputStream fis = new FileInputStream(filePath);
    InputStreamReader isr = new InputStreamReader(fis, "UTF-8");
    BufferedReader in = new BufferedReader(isr);

    FileOutputStream fos = new FileOutputStream(filtered);
    OutputStreamWriter osw = new OutputStreamWriter(fos, "UTF-8");
    BufferedWriter out = new BufferedWriter(osw);

    boolean spaceFlag = true;
    int digit;
    numOfTransitions = -1;
    while ((digit = in.read()) > -1)
    {
      if ((digit >= 65) && (digit <= 90))
      {
        if (spaceFlag)
        {
          numOfTransitions++;
          spaceFlag = false;
        }
        out.write(Character.toLowerCase((char) digit));
      }
      else if ((digit >= 97) && (digit <= 122))
      {
        if (spaceFlag)
        {
          numOfTransitions++;
          spaceFlag = false;
        }
        out.write((char) digit);
      }
      else
      {
        if (spaceFlag == false)
        {
          out.write(" ");
          spaceFlag = true;
        }
```

```
      }
    }

    out.flush();

    in.close();
    isr.close();
    fis.close();

    out.close();
    osw.close();
    fos.close();
  }
  catch (IOException ioe)
  {
    System.out.println("Error: " + ioe.getMessage());
  }
}

private static void countWordsAndTransitions()
{
  // open file
  FileInputStream fis;
  try
  {
    fis = new FileInputStream(filtered);
    InputStreamReader isr = new InputStreamReader(fis, "UTF-8");
    BufferedReader in = new BufferedReader(isr);

    // initializations
    int digit;
    String state1 = "";
    String state2 = "";
    String firstWord = "";
    hashTable = new Hashtable(1000);
    stateOccurrences = new Hashtable(500);
    Object oldValue;

    // read 1st state
    digit = in.read();
    while (Character.isLetter((char) digit))
    {
      state1 += String.valueOf((char) digit);
      firstWord = state1;
      digit = in.read();
    }

    // read 2nd state
    digit = in.read();
    while (Character.isLetter((char) digit))
```

```
{
 state2 += String.valueOf((char) digit);
 digit = in.read();
}

// store 1st transition
stateOccurrences.put(state1, String.valueOf(1));
oldValue = stateOccurrences.put(state2, String.valueOf(1));
if (oldValue != null)
{
 stateOccurrences.put(
   state2,
   String.valueOf(1 + Integer.parseInt(String.valueOf(oldValue))));
}
hashTable.put(state1 + " " + state2, String.valueOf(1));

// store the rest of the transitions
for (int i = 0; i < (numOfTransitions - 1); i++)
{
 state1 = state2;
 state2 = "";
 digit = in.read();
 while (Character.isLetter((char) digit))
 {
  state2 += String.valueOf((char) digit);
  digit = in.read();
 }
 oldValue = stateOccurrences.put(state2, String.valueOf(1));
 if (oldValue != null)
 {
  stateOccurrences.put(
    state2,
    String.valueOf(1 + Integer.parseInt(String.valueOf(oldValue))));
 }
 oldValue = hashTable.put(state1 + " " + state2, String.valueOf(1));
 if (oldValue != null)
 {
  hashTable.put(
    state1 + " " + state2,
    String.valueOf(1 + Integer.parseInt(String.valueOf(oldValue))));
 }
}

// transition for End Of File
state1 = state2;
state2 = firstWord;
oldValue = hashTable.put(state1 + " " + state2, String.valueOf(1));
if (oldValue != null)
{
 hashTable.put(
```

```
        state1 + " " + state2,
        String.valueOf(1 + Integer.parseInt(String.valueOf(oldValue))));
    }
   }
   catch (FileNotFoundException e)
   {
    e.printStackTrace();
   }
   catch (IOException e)
   {
    e.printStackTrace();
   }
}

private static void updateDB()
{
  try
  {
    PreparedStatement stmt;
    ResultSet results;

    // find how many documents exist in the same category
    int w = 0;
    if (!cat.equalsIgnoreCase("unknown"))
    {
      stmt =
        con.prepareStatement("SELECT count(*) FROM doc WHERE doc_cat = ?");
      stmt.setString(1, cat);
      results = stmt.executeQuery();
      results.next();
      w = results.getInt(1);
    }

    // find how many documents exist totally in the database
    stmt = con.prepareStatement("SELECT count(*) FROM doc");
    results = stmt.executeQuery();
    results.next();
    int wUniv = results.getInt(1);

    // update tables dw and cw
    PreparedStatement unCheck;
    if (!cat.equalsIgnoreCase("unknown"))
    {
      unCheck =
        con.prepareStatement(
          "UPDATE cw SET cw_checked = 0 WHERE cw_cat = 'universal' "
            + "OR cw_cat = '"
            + cat
            + "'");
    }
```

```
else
{
 unCheck =
   con.prepareStatement(
     "UPDATE cw SET cw_checked = 0 WHERE cw_cat = 'universal'");
}
unCheck.executeUpdate();

int numOfWords = numOfTransitions + 1;

PreparedStatement insDoc =
 con.prepareStatement("INSERT INTO dw VALUES (?, ?, ?)");
insDoc.setString(1, fileName);
PreparedStatement insCat =
 con.prepareStatement("INSERT INTO cw VALUES (?, ?, ?, '1')");
PreparedStatement updCat =
 con.prepareStatement(
   "UPDATE cw SET cw_prob = ?, cw_checked = '1' "
    + "WHERE cw_cat = ? AND cw_word = ?");
PreparedStatement updRest =
 con.prepareStatement(
   "UPDATE cw SET cw_prob = ? * cw_prob, cw_checked = '1' "
    + "WHERE cw_cat = ? AND cw_checked = '0'");
stmt =
 con.prepareStatement(
   "SELECT cw_prob FROM cw WHERE cw_cat = ? AND cw_word = ?");

Enumeration enum = stateOccurrences.keys();
String key;
float value;
while (enum.hasMoreElements())
{
 key = (String) enum.nextElement();
 insDoc.setString(2, key);
 value =
  Float.parseFloat((String) stateOccurrences.get(key)) / numOfWords;
 insDoc.setFloat(3, value);
 insDoc.executeUpdate();

 stmt.setString(1, "universal");
 stmt.setString(2, key);
 results = stmt.executeQuery();
 if (results.next())
 {
  updCat.setString(2, "universal");
  updCat.setString(3, key);
  updCat.setFloat(
    1,
    ((wUniv * results.getFloat(1)) + value) / (wUniv + 1));
  updCat.executeUpdate();
```

```
  }
  else
  {
   insCat.setString(1, "universal");
   insCat.setString(2, key);
   insCat.setFloat(3, value / (wUniv + 1));
   insCat.executeUpdate();
  }

  if (!cat.equalsIgnoreCase("unknown"))
  {
   stmt.setString(1, cat);
   stmt.setString(2, key);
   results = stmt.executeQuery();
   if (results.next())
   {
    updCat.setString(2, cat);
    updCat.setString(3, key);
    updCat.setFloat(1, ((w * results.getFloat(1)) + value) / (w + 1));
    updCat.executeUpdate();
   }
   else
   {
    insCat.setString(1, cat);
    insCat.setString(2, key);
    insCat.setFloat(3, value / (w + 1));
    insCat.executeUpdate();
   }
  }
}

updRest.setFloat(1, ((float) wUniv) / (wUniv + 1));
updRest.setString(2, "universal");
updRest.executeUpdate();
if (!cat.equalsIgnoreCase("unknown"))
{
 updRest.setFloat(1, ((float) w) / (w + 1));
 updRest.setString(2, cat);
 updRest.executeUpdate();
}

// update tables dt and ct
if (!cat.equalsIgnoreCase("unknown"))
{
 unCheck =
   con.prepareStatement(
     "UPDATE ct SET ct_checked = 0 WHERE (ct_cat = 'universal' "
       + "OR ct_cat = '"
       + cat
       + "'"
```

```
      + ") AND ct_from = ?");
}
else
{
 unCheck =
   con.prepareStatement(
     "UPDATE ct SET ct_checked = 0 WHERE ct_cat = 'universal' "
      + "AND ct_from = ?");
}

enum = stateOccurrences.keys();
while (enum.hasMoreElements())
{
 key = (String) enum.nextElement();
 unCheck.setString(1, key);
 unCheck.executeUpdate();
}

stmt =
 con.prepareStatement(
   "SELECT distinct(ct_from) FROM ct WHERE ct_cat = ?");
stmt.setString(1, "universal");
results = stmt.executeQuery();
HashSet univSet = new HashSet();
HashSet catSet = new HashSet();
while (results.next())
{
 univSet.add(results.getString(1));
}
if (!cat.equalsIgnoreCase("unknown"))
{
 stmt.setString(1, cat);
 results = stmt.executeQuery();
 while (results.next())
 {
  catSet.add(results.getString(1));
 }
}

insDoc = con.prepareStatement("INSERT INTO dt VALUES (?, ?, ?, ?)");
insDoc.setString(1, fileName);
insCat = con.prepareStatement("INSERT INTO ct VALUES (?, ?, ?, ?, '1')");
updCat =
 con.prepareStatement(
   "UPDATE ct SET ct_prob = ?, ct_checked = '1' "
    + "WHERE ct_cat = ? AND ct_from = ? AND ct_to = ?");
updRest =
 con.prepareStatement(
   "UPDATE ct SET ct_prob = ? * ct_prob, ct_checked = '1' "
    + "WHERE ct_cat = ? AND ct_checked = '0'");
```

```
stmt =
  con.prepareStatement(
    "SELECT ct_prob FROM ct WHERE ct_cat = ? AND ct_from = ? AND "
      + "ct_to = ?");

enum = hashTable.keys();
String from, to;
boolean found;
while (enum.hasMoreElements())
{
  key = (String) enum.nextElement();
  from = key.substring(0, key.indexOf(" "));
  to = key.substring(key.indexOf(" ") + 1, key.length());
  value =
    Float.parseFloat((String) hashTable.get(key))
      / Float.parseFloat((String) stateOccurrences.get(from));

  insDoc.setString(2, from);
  insDoc.setString(3, to);
  value =
    Float.parseFloat((String) hashTable.get(key))
      / Float.parseFloat((String) stateOccurrences.get(from));
  insDoc.setFloat(4, value);
  insDoc.executeUpdate();

  if (univSet.contains(from))
  {
    stmt.setString(1, "universal");
    stmt.setString(2, from);
    stmt.setString(3, to);
    results = stmt.executeQuery();
    if (results.next())
    {
      updCat.setString(2, "universal");
      updCat.setString(3, from);
      updCat.setString(4, to);
      updCat.setFloat(
        1,
       ((wUniv * results.getFloat(1)) + value) / (wUniv + 1));
      updCat.executeUpdate();
    }
    else
    {
      insCat.setString(1, "universal");
      insCat.setString(2, from);
      insCat.setString(3, to);
      insCat.setFloat(4, value / (wUniv + 1));
      insCat.executeUpdate();
    }
  }
```

```
  else
  {
   insCat.setString(1, "universal");
   insCat.setString(2, from);
   insCat.setString(3, to);
   insCat.setFloat(4, value);
   insCat.executeUpdate();
  }

  if (!cat.equalsIgnoreCase("unknown"))
  {
   if (catSet.contains(from))
   {
    stmt.setString(1, cat);
    stmt.setString(2, from);
    stmt.setString(3, to);
    results = stmt.executeQuery();
    if (results.next())
    {
     updCat.setString(2, cat);
     updCat.setString(3, from);
     updCat.setString(4, to);
     updCat.setFloat(1, ((w * results.getFloat(1)) + value) / (w + 1));
     updCat.executeUpdate();
    }
    else
    {
     insCat.setString(1, cat);
     insCat.setString(2, from);
     insCat.setString(3, to);
     insCat.setFloat(4, value / (w + 1));
     insCat.executeUpdate();
    }
   }
   else
   {
    insCat.setString(1, cat);
    insCat.setString(2, from);
    insCat.setString(3, to);
    insCat.setFloat(4, value);
    insCat.executeUpdate();
   }
  }
}

updRest.setFloat(1, ((float) wUniv) / (wUniv + 1));
updRest.setString(2, "universal");
updRest.executeUpdate();
if (!cat.equalsIgnoreCase("unknown"))
{
```

```
        updRest.setFloat(1, ((float) w) / (w + 1));
        updRest.setString(2, cat);
        updRest.executeUpdate();
      }

      // update table doc
      stmt = con.prepareStatement("INSERT INTO doc VALUES(?, ?, ?)");
      stmt.setString(1, fileName);
      stmt.setString(2, cat);
      stmt.setString(3, String.valueOf(new Date()));
      stmt.executeUpdate();

      stmt.close();
      insDoc.close();
      insCat.close();
      updCat.close();
      updRest.close();
    }
    catch (SQLException e)
    {
      e.printStackTrace();
    }
  }
}
```

```
package org.dskoutas.tc;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.LinkedList;

public class Covariance1
{

  static Connection con;
  static PreparedStatement stmt;
  static ResultSet results;

  public static void main(String[] args)
  {
    try
    {
      // connect to database
      String mm_driver = "org.gjt.mm.mysql.Driver";
```

```
String url =
  "jdbc:mysql://127.0.0.1:3306/tcdb?useUnicode=true&characterEncoding=ISO-8859-7";

Class.forName(mm_driver);

con = DriverManager.getConnection(url);

// find the number of words
PreparedStatement stmt =
  con.prepareStatement("SELECT COUNT(DISTINCT(dw_word)) FROM dw");
ResultSet results = stmt.executeQuery();
results.next();
int numOfWords = results.getInt(1);
System.out.println("Number of words = " + numOfWords);

// find the number of texts
stmt = con.prepareStatement("SELECT COUNT(doc_file) FROM doc");
results = stmt.executeQuery();
results.next();
int numOfTexts = results.getInt(1);
System.out.println("Number of texts = " + numOfTexts);

// clear table1, table2, table4, table5 and cov1
stmt = con.prepareStatement("DELETE FROM table1");
stmt.executeUpdate();
stmt = con.prepareStatement("DELETE FROM table2");
stmt.executeUpdate();
stmt = con.prepareStatement("DELETE FROM table4");
stmt.executeUpdate();
stmt = con.prepareStatement("DELETE FROM table5");
stmt.executeUpdate();
stmt = con.prepareStatement("DELETE FROM table6");
stmt.executeUpdate();
stmt = con.prepareStatement("DELETE FROM cov1");
stmt.executeUpdate();

// fill table4
stmt =
  con.prepareStatement(
    "INSERT INTO table1 SELECT dw_word, sum(dw_prob) / "
      + numOfTexts
      + " FROM dw GROUP BY dw_word");
stmt.executeUpdate();
stmt =
  con.prepareStatement(
    "INSERT INTO table2(tb2_word) SELECT doc_file FROM doc");
stmt.executeUpdate();

stmt = con.prepareStatement("SELECT tb2_word FROM table2");
results = stmt.executeQuery();
```

```
    stmt =
     con.prepareStatement(
      "INSERT INTO table4 SELECT tb1_word, tb2_word, - tb1_prob "
       + "FROM table1, table2 WHERE tb2_word = ? ");

    while (results.next())
    {
     stmt.setString(1, results.getString(1));
     stmt.executeUpdate();
    }
    stmt =
     con.prepareStatement(
      "UPDATE table4, dw SET tb4_prob = tb4_prob + dw_prob "
       + "WHERE tb4_from = dw_word AND tb4_to = dw_doc ");

    // fill table5
    stmt = con.prepareStatement("INSERT INTO table5 SELECT * FROM table4");
    stmt.executeUpdate();

    // construct covariance matrix
    stmt = con.prepareStatement("SELECT tb1_word FROM table1");
    results = stmt.executeQuery();
    LinkedList wordList = new LinkedList();
    while (results.next())
    {
     wordList.add(results.getString(1));
    }
    stmt =
     con.prepareStatement(
      "INSERT INTO cov1 SELECT tb4_from, tb5_from, sum(tb4_prob * tb5_prob) "
       + "FROM table4, table5 WHERE tb4_from = ? AND tb5_from = ? AND tb4_to =
tb5_to "
       + "GROUP BY tb4_from");
    for (int i = 0; i < wordList.size(); i++)
    {
     stmt.setString(1, (String) wordList.get(i));
     System.out.println(i);
     for (int j = i; j < wordList.size(); j++)
     {
      stmt.setString(2, (String) wordList.get(j));
      stmt.executeUpdate();
     }
    }

    stmt =
     con.prepareStatement(
      "INSERT INTO table6 SELECT cv1_to, cv1_from, cv1_prob FROM cov1 "
       + "WHERE cv1_from <> cv1_to");
    stmt.executeUpdate();
    stmt = con.prepareStatement("INSERT INTO cov1 SELECT * FROM table6");
```

```
      stmt.executeUpdate();

      con.close();

      System.exit(0);
    }
    catch (SQLException sqle)
    {
      System.out.println("Error: " + sqle.getMessage());
    }
    catch (ClassNotFoundException e)
    {
      e.printStackTrace();
    }
  }
}



package org.dskoutas.tc;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.LinkedList;

public class Covariance2
{

  static Connection con;
  static PreparedStatement stmt;
  static ResultSet results;

  public static void main(String[] args)
  {
    try
    {
      // connect to database
      String mm_driver = "org.gjt.mm.mysql.Driver";
      String url =
        "jdbc:mysql://127.0.0.1:3306/tcdb?useUnicode=true&characterEncoding=ISO-8859-7";

      Class.forName(mm_driver);

      con = DriverManager.getConnection(url);

      // find the number of words
```

```
PreparedStatement stmt =
  con.prepareStatement("SELECT COUNT(DISTINCT(dw_word)) FROM dw");
ResultSet results = stmt.executeQuery();
results.next();
int dim = results.getInt(1);
float mean = (float) (1.0 / dim);
float minusMean = -mean;
System.out.println("Dimension = " + dim);

// clear table1, table2, table4, table5, table6 and cov2
stmt = con.prepareStatement("DELETE FROM table1");
stmt.executeUpdate();
stmt = con.prepareStatement("DELETE FROM table2");
stmt.executeUpdate();
stmt = con.prepareStatement("DELETE FROM table4");
stmt.executeUpdate();
stmt = con.prepareStatement("DELETE FROM table5");
stmt.executeUpdate();
stmt = con.prepareStatement("DELETE FROM table6");
stmt.executeUpdate();
stmt = con.prepareStatement("DELETE FROM cov2");
stmt.executeUpdate();

// fill table4
stmt =
  con.prepareStatement(
    "INSERT INTO table1(tb1_word) SELECT DISTINCT(dw_word) FROM dw");
stmt.executeUpdate();
stmt =
  con.prepareStatement(
    "INSERT INTO table2(tb2_word) SELECT tb1_word FROM table1");
stmt.executeUpdate();

stmt = con.prepareStatement("SELECT tb1_word FROM table1");
results = stmt.executeQuery();
stmt =
  con.prepareStatement(
    "INSERT INTO table4 SELECT tb1_word, tb2_word, "
      + minusMean
      + " FROM table1, table2 WHERE tb1_word = ?");
while (results.next())
{
  stmt.setString(1, results.getString(1));
  stmt.executeUpdate();
}

stmt =
  con.prepareStatement(
    "UPDATE table4, ct SET tb4_prob = tb4_prob + ct_prob "
      + "WHERE tb4_from = ct_from AND tb4_to = ct_to AND ct_cat = 'universal'");
```

```
  stmt.executeUpdate();

  // fill table5
  stmt = con.prepareStatement("INSERT INTO table5 SELECT * FROM table4");
  stmt.executeUpdate();

  // construct covariance matrix
  results.beforeFirst();
  LinkedList wordList = new LinkedList();
  while (results.next())
  {
   wordList.add(results.getString(1));
  }
  stmt =
   con.prepareStatement(
    "INSERT INTO cov2 SELECT tb4_from, tb5_from, "
     + "sum(tb4_prob * tb5_prob) FROM table4, table5 WHERE tb4_from = ? AND "
     + "tb5_from = ? AND tb4_to = tb5_to GROUP BY tb4_from");
  for (int i = 0; i < wordList.size(); i++)
  {
   stmt.setString(1, (String) wordList.get(i));
   System.out.println(i);
   for (int j = i; j < wordList.size(); j++)
   {
    stmt.setString(2, (String) wordList.get(j));
    stmt.executeUpdate();
   }
  }

  stmt =
   con.prepareStatement(
    "INSERT INTO table6 SELECT cv2_to, cv2_from, cv2_prob FROM cov2 "
     + "WHERE cv2_from <> cv2_to");
  stmt.executeUpdate();
  stmt = con.prepareStatement("INSERT INTO cov2 SELECT * FROM table6");
  stmt.executeUpdate();

  con.close();

  System.exit(0);
 }
 catch (SQLException sqle)
 {
  System.out.println("Error: " + sqle.getMessage());
 }
 catch (ClassNotFoundException e)
 {
  e.printStackTrace();
 }
}
```

```
}



package org.dskoutas.tc;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.LinkedList;

import javax.swing.JOptionPane;

public class CalculateDistance
{

  static Connection con;
  static PreparedStatement stmt;
  static ResultSet results;

  public static void main(String[] args)
  {
    try
    {
      // connect to database
      String mm_driver = "org.gjt.mm.mysql.Driver";
      String url =
        "jdbc:mysql://127.0.0.1:3306/tcdb?useUnicode=true&characterEncoding=ISO-8859-7";

      Class.forName(mm_driver);

      con = DriverManager.getConnection(url);

      // choose distance type
      String[] distTypes =
        { " distance 1", " distance 2", " distance 3", " distance 4" };

      String distType =
       (String) JOptionPane.showInputDialog(
         null,
         "Please select a distance type",
         "Choose Distance type",
         JOptionPane.QUESTION_MESSAGE,
         null,
         distTypes,
         distTypes[0]);
```

```
if (distType == null)
{
  con.close();
  System.exit(1);
}

// choose mode
String[] modes =
  {
    "Compare 2 documents",
    "Compare 1 with all",
    "Compare all with all",
    "Classify document",
    "Classify all" };

String mode =
  (String) JOptionPane.showInputDialog(
    null,
    "Please select an operation",
    "Choose Mode",
    JOptionPane.QUESTION_MESSAGE,
    null,
    modes,
    modes[0]);

if (mode == null)
{
  con.close();
  System.exit(1);
}

// retrieve document names from the database
stmt = con.prepareStatement("SELECT doc_file FROM doc");
results = stmt.executeQuery();
LinkedList docList = new LinkedList();
while (results.next())
{
  docList.add(results.getString(1));
}
Object[] docs = docList.toArray();

// retrieve unknown document names from the database
stmt =
  con.prepareStatement(
    "SELECT doc_file FROM doc WHERE doc_cat = 'unknown'");
results = stmt.executeQuery();
LinkedList docList2 = new LinkedList();
while (results.next())
{
  docList2.add(results.getString(1));
```

```
}

if (docs.length < 2)
{
 System.out.println("There are less than 2 documents in the database!");
 con.close();
 System.exit(1);
}

System.out.println("Distance type: " + distType + "\n");

if (mode.equals("Compare all with all"))
{
 for (int i = 0; i < docs.length; i++)
 {
  for (int j = (i + 1); j < docs.length; j++)
  {
   distance(
    (String) docList.get(i),
    (String) docList.get(j),
    distType,
    1);
  }
  System.out.print("\n");
 }
}
else if (mode.equals("Classify all"))
{
 String[] catList = { "art", "history", "medicine", "philosophy" };
 for (int i = 0; i < docList2.size(); i++)
 {
  for (int j = 0; j < catList.length; j++)
  {
   distance((String) docList2.get(i), catList[j], distType, 2);
  }
  System.out.print("\n");
 }
}
else
{
 String doc1 =
  (String) JOptionPane.showInputDialog(
   null,
   "Please select a document",
   "Choose Document",
   JOptionPane.QUESTION_MESSAGE,
   null,
   docs,
   docs[0]);
```

```
    if (doc1 == null)
    {
     con.close();
     System.exit(1);
    }

    if (mode.equals("Compare 1 with all"))
    {
     for (int i = 0; i < docs.length; i++)
     {
      distance(doc1, (String) docList.get(i), distType, 1);
     }
    }
    else if (mode.equals("Compare 2 documents"))
    {
     String doc2 =
      (String) JOptionPane.showInputDialog(
        null,
        "Please select another document",
        "Choose Document",
        JOptionPane.QUESTION_MESSAGE,
        null,
        docs,
        docs[0]);

     if (doc2 == null)
     {
      con.close();
      System.exit(1);
     }

     distance(doc1, doc2, distType, 1);
    }
    else
    {
     String[] catList = { "art", "history", "medicine", "philosophy" };
     for (int i = 0; i < catList.length; i++)
     {
      distance(doc1, catList[i], distType, 2);
     }
    }
   }

   con.close();

   System.exit(0);
  }
  catch (SQLException sqle)
  {
   System.out.println("Error: " + sqle.getMessage());
```

```
  }
  catch (ClassNotFoundException e)
  {
    e.printStackTrace();
  }
}

private static void distance(
  String obj1,
  String obj2,
  String dType,
  int type)
{
  try
  {
    // clear tables
    stmt = con.prepareStatement("DELETE FROM table1");
    stmt.executeUpdate();
    stmt = con.prepareStatement("DELETE FROM table2");
    stmt.executeUpdate();
    stmt = con.prepareStatement("DELETE FROM table3");
    stmt.executeUpdate();
    stmt = con.prepareStatement("DELETE FROM table4");
    stmt.executeUpdate();
    stmt = con.prepareStatement("DELETE FROM table5");
    stmt.executeUpdate();
    stmt = con.prepareStatement("DELETE FROM table6");
    stmt.executeUpdate();

    float distance = 0;

    // prepare vector for 1st document
    stmt =
      con.prepareStatement(
        "INSERT INTO table1 SELECT dw_word, dw_prob "
          + "FROM dw WHERE dw_doc = '"
          + obj1
          + "'");
    stmt.executeUpdate();

    if (dType.equals("distance 2"))
    {
      stmt =
        con.prepareStatement(
          "INSERT INTO table4 SELECT dt_from, dt_to, dt_prob "
            + "FROM dt WHERE dt_doc = '"
            + obj1
            + "'");
      stmt.executeUpdate();
    }
```

```
// prepare vector for 2nd document or category
if (type == 1)
{
 stmt =
  con.prepareStatement(
   "INSERT INTO table2 SELECT dw_word, dw_prob "
    + "FROM dw WHERE dw_doc = '"
    + obj2
    + "'");
 stmt.executeUpdate();

 if (dType.equals("distance 2"))
 {
  stmt =
   con.prepareStatement(
    "INSERT INTO table5 SELECT dt_from, dt_to, dt_prob "
     + "FROM dt WHERE dt_doc = '"
     + obj2
     + "'");
  stmt.executeUpdate();
 }
}
else
{
 stmt =
  con.prepareStatement(
   "INSERT INTO table2 SELECT cw_word, cw_prob "
    + "FROM cw WHERE cw_cat = '"
    + obj2
    + "'");
 stmt.executeUpdate();

 if (dType.equals("distance 2"))
 {
  stmt =
   con.prepareStatement(
    "INSERT INTO table5 SELECT ct_from, ct_to, ct_prob "
     + "FROM ct WHERE ct_cat = '"
     + obj2
     + "'");
  stmt.executeUpdate();
 }
}

// calculate distance
stmt =
 con.prepareStatement(
  "INSERT INTO table3 SELECT tb1_word, tb1_prob - tb2_prob "
   + "FROM table1, table2 WHERE tb1_word = tb2_word");
```

```
stmt.executeUpdate();

stmt =
  con.prepareStatement(
    "DELETE table1, table2 FROM table1, table2 WHERE tb1_word = tb2_word");
stmt.executeUpdate();

stmt = con.prepareStatement("INSERT INTO table3 SELECT * FROM table1");
stmt.executeUpdate();

stmt = con.prepareStatement("INSERT INTO table3 SELECT * FROM table2");
stmt.executeUpdate();

if (dType.equals("distance 1"))
{
  stmt =
    con.prepareStatement(
      "UPDATE table3 SET tb3_prob = tb3_prob * tb3_prob");
  stmt.executeUpdate();
  stmt = con.prepareStatement("SELECT sum(tb3_prob) FROM table3");
}

if (dType.equals("distance 2"))
{
  stmt =
    con.prepareStatement(
      "INSERT INTO table6 SELECT tb4_from, tb4_to,  abs(tb4_prob - tb5_prob) "
        + "FROM table4, table5 WHERE tb4_from = tb5_from AND tb4_to = tb5_to");
  stmt.executeUpdate();

  stmt =
    con.prepareStatement(
      "DELETE table4, table5 FROM table4, table5 WHERE tb4_from = tb5_from "
        + "AND tb4_to = tb5_to");
  stmt.executeUpdate();

  stmt = con.prepareStatement("INSERT INTO table6 SELECT * FROM table4");
  stmt.executeUpdate();

  stmt = con.prepareStatement("INSERT INTO table6 SELECT * FROM table5");
  stmt.executeUpdate();

  stmt = con.prepareStatement("DELETE FROM table1");
  stmt.executeUpdate();

  stmt =
    con.prepareStatement(
      "INSERT INTO table1 SELECT "
        + "tb6_from, sum(tb6_prob * tb6_prob) FROM table6 GROUP BY tb6_from");
  stmt.executeUpdate();
```

```
  stmt = con.prepareStatement("DELETE FROM table2");
  stmt.executeUpdate();

  stmt =
    con.prepareStatement(
      "UPDATE table3 SET tb3_prob = tb3_prob * tb3_prob");
  stmt.executeUpdate();

  stmt =
    con.prepareStatement(
      "INSERT INTO table2 SELECT "
        + "tb1_word, tb1_prob * tb3_prob FROM table1, table3 WHERE "
        + "tb1_word = tb3_word");
  stmt.executeUpdate();

  stmt = con.prepareStatement("SELECT sum(tb2_prob) FROM table2");
}

if (dType.equals("distance 3"))
{
  stmt =
    con.prepareStatement(
      "INSERT INTO table4 SELECT 'a', cv1_to, "
        + "sum(tb3_prob * cv1_prob) FROM table3, cov1 WHERE tb3_word = cv1_from "
        + "GROUP BY cv1_to");
  stmt.executeUpdate();

  stmt =
    con.prepareStatement(
      "INSERT INTO table5 SELECT 'a', 'a', "
        + "sum(tb4_prob * tb3_prob) FROM table4, table3 WHERE tb4_to = tb3_word "
        + "GROUP BY tb4_from");
  stmt.executeUpdate();

  stmt = con.prepareStatement("SELECT tb5_prob FROM table5");
}

if (dType.equals("distance 4"))
{
  stmt =
    con.prepareStatement(
      "INSERT INTO table4 SELECT 'a', cv2_to, "
        + "sum(tb3_prob * cv2_prob) FROM table3, cov2 WHERE tb3_word = cv2_from "
        + "GROUP BY cv2_to");
  stmt.executeUpdate();

  stmt =
    con.prepareStatement(
      "INSERT INTO table5 SELECT 'a', 'a', "
```

```
            + "sum(tb4_prob * tb3_prob) FROM table4, table3 WHERE tb4_to = tb3_word "
            + "GROUP BY tb4_from");
        stmt.executeUpdate();

        stmt = con.prepareStatement("SELECT tb5_prob FROM table5");
      }

      results = stmt.executeQuery();
      results.next();
      distance = (float) Math.sqrt(results.getFloat(1));
      distance *= 1000;
      System.out.println(
        "Distance between " + obj1 + " and " + obj2 + " = " + (int) distance);
    }
    catch (SQLException e)
    {
      e.printStackTrace();
    }
  }
}
```

Βιβλιογραφία

Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, Ramasamy Uthurusamy. *Advance in Knowledge Discovery and Data Mining*, AAAI Press/The MIT Press

K.S. Fu. *Digital Pattern Recognition*, Springer-Verlag

Richard O. Duda, Peter E. Hart. *Pattern Classification (2nd edition)*, Wiley Interscience

Keinosuke Fukunga. *Introduction to Statistical Pattern Recognition*, Academic Press

David Hand, Heikki Mannila, Padhraic Smyth. *Principles of Data Mining*, The MIT Press

Bhavani Thuraisingham. Data Mining: *Technology, Techniques, Tools and Trends*, CRC Press

Christopher Westphal, Teresa Biaxton. *Data Mining Solutions: Methods and Tools for Solving Real-World Problems*, Wiley Intelligence

Simon Haykin. *Neural Networks: A Comprehensive Foundation*, Prentice Hall

Elaine Rich, Kevin Knight. *Artificial Intelligence (2nd Edition)*, McGraw-Hill

David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison_wisley Publishing Company

Leonard Kleinrock, Richard Gail. *Gueueing Systems Problems and Solutions*, Willey-Interscience

Leonard Kleinrock. *Gueueing Systems Volume I: Theory*, Willey-Interscience

Erol Gelenbe, Guy Pujolle. *Introduction to Gueueing Networks (2nd edition)*,John Wiley& Sons

R. Lyman Ott. An *Introduction to Statistical Methods and Data Analysis (4th edition)*, Dexbury Press

Samardasa Weerahandi. Exact *Statistical Methods for Data Analysis*, Springer

Ajit C. Tamhane, Dorothy D. Dunlop. *Statistic and Data Analysis for Elementary to Intermediate*, Prentice Hall

John A. Rice. *Mathematical Statistics and Data Analysis (2nd edition)*, Dexbury Press

Hwei P. Hsa. Probability, *Random Variables and Random Processes*, McGraw- Hill

Richard M. Feldman, Ciriaco Valdez- Flores. *Applicd Probability and Stochastic Processes*, PWS Publishing Company

William Feller. *And Introduction to Probability Theory and its Applications (3rd edition)*, John Willey& Sons

Siegmund Brandt. Data Analysis: Statistical and Computational Methods for Scientists and Engineers (3rd edition), Springer

Sheldon M. Ross. *Stochastic Processes (2nd edition)*, John Wiley& Sons

Athanasios Papoulis. *Probabilitu, Random Variables and Stochastic Processes (3rd edition)*, McGraw-Hill

D. Revuz. *Markov Chains (Revised Edition)*, North-Holland

IEEE International Conference on Data Mining: http://www.cs.uvm.edu/~icdm/

The Data Mine: http://www.the-data-mine.com/

ACM Special Interest Group on Knowledge Discovery in Data and Data Mining: http://www.acm.org/sigkdd/

SIAM International Conference on Data Mining: http://www.siam.org/meetings/sdm02/

An Introduction to Data Mining by Kurt Thearling: http://www.thearling.com/dmintro/dmintro.htm

Data Clustering and Its Applications: http://members.tripod.com/asim_saeed/paper.htm

Data clustering: http://www.cp.eng.chula.ac.th/faculty/pjw/talk/clustering.htm

Data clustering: http://erin.mit.jyu.fi/projects/nda/usrman/node110.html

Distance Metrics Overview: http://www.predictivepatterns.com/docs/WebSiteDocs/Clustering/Clustering_Parameters/Distance_Metrics_Overview.htm

Distance Metrics: http://www.cee.hw.ac.uk/hipr/html/metric.html

Mahalanobis Metric: http://www.engr.sjsu.edu/~knapp/HCIRODPR/PR_Mahal/M_metric.htm

Markov Chains: http://www.utdallas.edu/~jjue/cs6352/markov/

Discrete Time Markov Chains: http://www.control.auc.dk/~henrik/undervisning/trafik2/riska/www.cs.wm.edu/~riska/main/node14.html

Review of K-Nearest Neighbor Text Categorization Method: http://www.usenix.org/events/sec02/full_papers/liao/liao_html/node4.html

Text Classification and Clustering: http://www-i6.informatik.rwth-aachen.de/web/Research/Classification_frame.html#Seitenbeginn

Introduction to Text Classification: www.andrew.cmu.edu/course/20-760/notes/ intro-textclassification.ppt