Ιονιο Πανεπιστημιο

Σχολη Επιστημων της Πληροφοριας και Πληροφορικης

Τμήμα Πληροφορικής



Πτυχιακή Εργασία

Διαχείρηση Πολυμεσικής Πληροφορίας

Τουλάκης Αλέξανδρος

Επιβλέπων: Ιωάννης Καρύδης

28 Ιανουαρίου 2020

Επιβλέπων Καθηγητής

Ιωάννης Καρύδης, *Επίκουρος Καθηγητής,*
Ιόνιο Πανεπιστήμιο

Τριμελής Επιτροπή

Ιωάννης Καρύδης, *Επίκουρος Καθηγητής,*
Ιόνιο Πανεπιστήμιο

Κάτια Ληδία Κερμανίδου, *Επίκουρη Καθηγήτρια,*
Ιόνιο Πανεπιστήμιο

Μάρκος Αυλωνίτης, *Επίκουρος Καθηγητής,*
Ιόνιο Πανεπιστήμιο

Αφιερώνω την Πτυχιακή αυτήν στην οικογένια μου και τους φίλους μου για την αγάπη και την υποστηριξή τους.

Περίληψη

Η τρέχουσα πτυχιακή, ασχολείται με την διαχείριση πολυμεσικής πληροφορίας σε έγγραφα, μουσικά κομμάτια και εικόνες. Αρχικά έγινε υλοποίηση του αλγόριθμου LDA(Latent Dirichlet Allocation) με σκοπό να γίνει εξαγωγή της θεματολογίας ελληνικών κειμένων από ροές πληροφορίας(Rss Feeds). Αναλυτικότερα, οι ροές αυτές περιείχαν κείμενο, μουσική και ηχητικό περιεχόμενο ωστόσο έγινε μόνο χρήση του κειμένου. Για την επαλήθευση της ορθότητας του υλοποιημένου LDA πραγματοποιήθηκε σύγκριση του με ένα προϋπάρχον πλαίσιο(LDA Gensim) με αποδεδειγμένη λειτουργικότητα και απόδοση. Εν συνεχεία για τις εικόνες και τα μουσικά κομμάτια εφαρμόστηκε μια παραλλαγή της μεθόδου subset προκειμένου να εντοπιστούν παρόμοια μουσικά κομμάτια και εικόνες. Η παραλλαγή αυτή χρησιμοποιεί ως πρότυπο έναν πόρο κ ετικετών για την εύρεση ενός άλλου πόρου με τον μέγιστο αριθμό κοινών ετικετών. Για την εκτέλεση του πειράματος για την μουσική και τις εικόνες, Χρησιμοποιήθηκαν οι ετικέτες οι οποίες εξήχθησαν από το Api της υπηρεσίας τουLast-Fm, ενώ για τις εικόνες από το API της υπηρεσίας του Google Vision. Ως τιμές αναφοράς Ground truth για τα ηχητικά κομμάτια αξιοποιήθηκε η μέθοδος Track. getsimilar(Last-fm). Για τις εικόνες χρησιμοποιήθηκαν οι ετικέτες από την πλατφόρμα του LabelMe στις οποίες πραγματοποιήθηκε επιπλέον επεξεργασία για μεγαλύτερη πληρότητα. Μετέπειτα έγινε εκτέλεση της Subset μεθόδου. Η LabelMe παρέχει ένα σύνολο εικόνων και ετικετών που αποδόθηκαν από ανθρώπους. Συμπερασματικά στο πείραμα με την μουσική και τις εικόνες σημειώθηκε υψηλό f1-score: 0.81 και 0.72 αντίστοιχα. Τα αποτελέσματα αυτά υποδηλώνουν πως η μέθοδος που προτάθηκε, αποτέλεσε ικανοποιητικό μέτρο για την εύρεση παρόμοιων πολυμεσικών πόρων τα οποία μπορούν να συγκριθούν σε αποτελεσματικότητα με αυτά που θεωρούνται σωστά από τους ανθρώπους. Τέλος στο πείραμα με τα έγγραφα το υλοποιημένο LDA σημείωσε f1-score :0.519125 ενώ το Gensim σημείωσε f1-score: 0.560125 . Τα αποτελέσματα αυτά υποδηλώνουν πως από άποψη αποδοτικότητας είναι αρκετά κοντά μεταξύ τους. Ωστόσο το χαμηλό f1-score κρίνεται

πως οφείλεται σε μία σειρά παραγόντων όπως: στη δυσκολία εύρεσης μιας λίστας με συχνές λέξεις(Stopwords) οι οποίες ανταποκρίνονται στην θεματολογία των κειμένων). Εναλλακτικά η βελτίωση των αποτελεσμάτων θα μπορούσε να γίνει με την υλοποίηση της δυνατότητας τροποποίησης των υπερπαραμέτρων(hyperparameters) α, β. Οι βελτιώσεις που προτάθηκαν δεν υλοποιήθηκαν/ελέγχθηκαν διεξοδικά καθώς υπερβαίνουν το εύρος του ενδιαφέροντος της τρέχουσας εργασίας.

Abstract

This dissertation focuses on the management of multimedia goods such as documents, tracks and images. Initially the LDA(Latent Dirichlet Allocation) was implemented and Rss Feeds were used to extract topics from Greek texts. More Specifically, these feeds contained text, music and images but only the text was used. The implemented LDA was compared with an already existing one(LDA Gensim) that works well. For the images and the musical tracks a variation of the Subset method was used to identify similar musical tracks and images. The variation of the subset uses the k tags of a resource to find a digital good with the maximum number of shared tags. To perform the experiment on music and images, The labels extracted from the last-fm API were used and for the images the Google-Vision API. The Ground-truth that was used for the audio tracks was the Track.getsimilar(Last-fm). For the images "LabelMe" platform was used to retrieve the tags, which were further processed for greater completeness. Subsequently, the subset method was executed. In regard to the "LabelMe", it is a platform that provides a set of labels and images that were annotated by humans. For the images and musical tracks experiments they scored an f1-score of 0.81 and 0.72 respectively. These results suggest that the proposed method can be considered a good measure to find similar multimedia resources that can be compared in efficacy to what is rightly believed by humans. Additionally, it is a simple and quick method. Finally in the experiment with the documents the implemented LDA scored an f1-score of 0.31 while Gensim scored an f1-score of 0.32. These results suggest that in terms of efficiency they are close. However, the low f1-score is judged to be due to a number of factors, such as: the difficulty to create a stopword list that included a great number of words that are domain specific. Alternatively, the results could be improved by implementing a feature that updates the hyperparameters a, b which are the Dirichlet priors. As well as, the experimentation with a number of parameters such as the number of

iterations that were specified to ran the LDA. The above suggestions have not been thoroughly implemented / tested as they go beyond the scope of this dissertation.

Ευχαριστίες

Στο σημείο αυτό θα ήθελα να ευχαριστήσω όλους οι οποίοι με τον δικό τους τρόπο συνέβαλλαν στην ολοκλήρωσης της πτυχιακής αυτής εργασίας. Πρώτα από όλα, στον επιβλέποντα Κύριο Ιωάννη Καρύδη Οικονόμου που ανέλαβε διπλωματική εργασία μαζί μου. Ακόμα θα ήθελα να τον ευχαριστήσω για την συνεχή υποστήριξη, τις χρήσιμες συμβουλές, ενθάρρυνση καθώς και την άριστη συνεργασία. Ακόμα θα ήθελα να ευχαριστήσω τους φίλους και την οικογένεια μου για τη συμπαράσταση τους.

Περιεχόμενα

Α΄ Εισαγωγή 1
Α΄.1 Βιβλιογραφική έρευνα . . . 2
Α΄.2 Προτεινόμενη Λύση . . . 7

Β΄ Βασικές έννοιες 8
Β΄.1 Precision . . . 8
Β΄.2 Recall . . . 8
Β΄.3 F1-Score . . . 9
Β΄.4 Train-Test split . . . 9
Β΄.5 K-fold cross validation . . . 10
Β΄.6 Subset Similary . . . 10
Β΄.7 XML . . . 10
Β΄.8 JSON . . . 11
Β΄.9 RSS Feeds . . . 11
Β΄.10 StopWords . . . 13
Β΄.11 Stemming . . . 13
Β΄.12 Http . . . 14
Β΄.12.1 Http Requests . . . 14
Β΄.12.2 Get . . . 14
Β΄.12.3 Head . . . 14
Β΄.12.4 Post . . . 14
Β΄.13 Http Headers . . . 15

Γ΄ Πείραμα με κείμενα 16
Γ΄.1 Συνοπτική περιγραφή της μεθοδολογίας . . . 16

Γ′.2 Συλλογή και Προεπεξεργασία των δεδομένων 16
Γ′.2.1 Πίνακας Items 17
Γ′.2.2 Προβλήματα Κωδικοποίησης 17
Γ′.2.3 Ψευδοκώδικας Επίλυσης Προβλήματος Κωδικοποίησης 18
Γ′.2.4 Περιγραφή του ψευδοκώδικα Επίλυσης Κωδικοποίησης 19
Γ′.2.5 Πίνακας Feeds 19
Γ′.2.6 Ψευδοκώδικας RSS-Feeds 21
Γ′.2.7 Αναλυτική Περιγραφή Ψευδοκώδικα RSS 22
Γ′.3 Προεπεξεργασία κειμένου 22
Γ′.3.1 Μετατροπή πεζών σε κεφαλαίους χαρακτήρες 23
Γ′.3.2 Ψευδοκώδικας συνάρτησης μετατροπής μιας λέξης σε κεφαλαία γράμματα 24
Γ′.3.3 Αναλυτική Περιγραφή Μεθόδου Μετατροπής λέξεων από μικρά σε κεφαλαία 24
Γ′.3.4 Αφαίρεση Stopwords 24
Γ′.3.5 Ψευδοκώδικας StopWord Manager 25
Γ′.3.6 Stemming-Manager 26
Γ′.3.7 Περιγραφή διαδικτυακής πλατφόρμας stemming για την Ελληνική Γλώσσα 26
Γ′.3.8 Προβλήματα διαδιακτυακής εφαρμογής Greek-Stemmer 27
Γ′.3.9 Ψευδοκώδικας Stemmer 28
Γ′.3.10 Μέθοδος StemmEnglish 28
Γ′.3.11 Περιγραφή μεθόδου Stemm-English 28
Γ′.3.12 Μέθοδος Stemming-Greek 29
Γ′.3.13 Περιγραφή Μεθόδου Stemming-Greek 29
Γ′.4 Αλγόριθμος Latent Dirichlet Allocation 30
Γ′.4.1 Statistical Inference 30
Γ′.4.2 Collapsed Gibs Sampling 31
Γ′.4.3 Παράμετροι Εκτέλεσης Πειράματος 33
Γ′.4.4 Αξιολόγηση LDA 33
Γ′.4.5 Αποτελέσματα 34

Δ′ Πείραμα με Μουσικά Κομμάτια και Εικόνες 37
Δ′.1 Σύντομη Περιγραφή 37

Δ΄.2 Δεδομένα 38
Δ΄.2.1 Προβλήματα με τα δεδομένα 38
Δ΄.2.2 Last fm API 38
Δ΄.2.3 Μέθοδος Track.getSimilar 39
Δ΄.2.4 Χτίσιμο URL Track.getSimilar 39
Δ΄.2.5 Ψευδοκώδικας last - fm API Track.getsimilar 41
Δ΄.2.6 Περιγραφή Ψευδοκώδικα Track.getSimilar 42
Δ΄.2.7 μέθοδος Track.getInfo 42
Δ΄.2.8 Χτίσιμο URL Track.getInfo 43
Δ΄.2.9 Ψευδοκώδικας της συνάρτηση Track.getInfo 43
Δ΄.2.10 Περιγραφή ψευδοκώδικα Track.getinfo 43
Δ΄.3 Αρχιτεκτονική Συστήματος Πειράματος Μουσικής - Αποτελέσματα . . . 44
Δ΄.4 Σύντομη περιγραφή 44
Δ΄.5 Δεδομένα 45
Δ΄.5.1 Επίλυση προβλήματος με τα δεδομένα 46
Δ΄.6 Αρχιτεκτονική Συστήματος Πειράματος Εικόνων 46
Δ΄.6.1 Αποτελέσματα πειραμάτων 48

Ε΄ Συμπεράσματα 50

Κατάλογος Σχημάτων

Β′.1 Παράδειγμα XML κώδικα 11
Β′.2 Παράδειγμα json κώδικα 12
Β′.3 Παράδειγμα Rss κώδικα 12
Β′.4 Παράδειγμα Http Headers eclass ionio 15

Γ′.1 Το Dictionary στο οποίο όταν στο κείμενο βρεθεί χαρακτήρας ο οποίος βρίκσεται στα αριστερά του πίνακα αντιστοιχίζεται στον αντίστοιχο χαρακτήρα του δεξιά 23
Γ′.2 Greek Stemmer Διαδικτυακή εφαρμογή 27
Γ′.3 Συμμετρική κατανομή Dirichlet Αριστερά για α=4, δεξιά α=2, Το σκούρο χρώμα υποδηλώνει μεγαλύτερη πιθανότητα. 31
Γ′.4 Τύπος υπολιγισμού ζ, ϑ, φ 31
Γ′.5 Τύπος πιθανότητας λέξης να ανήκει στο θέμα j 32
Γ′.6 Μαθηματικοί τύποι για το φ και ϑ 32
Γ′.7 Αξιολόγηση των δύο υλοποιήσεων LDA 34
Γ′.8 Συγκρίσεις υλοποιήσεων LDA μεταξύ τους 36

Δ′.1 json Labrosa βασικές πληροφορίες 39
Δ′.2 Σχήμα βάσης αποθήκευσης δεδομένων με πληθικότητα πολλά προς πολλά 44
Δ′.3 ΔΡΔ διάγραμμα συμπλήρωσης/διόρθωσης ετικετών 47
Δ′.4 Βάση δεδομένων για τις εικόνες 48
Δ′.5 ΔΡΔ για επισημείωση εικόνων 49

Κατάλογος Πινάκων

Γ΄.1 Σημαντικότερες στήλες για τον πίνακα items 17
΄.2 Πίνακες με κωδικοποιήσεις που χρησιμοποιήθηκαν 20
΄.3 Αποτελέσματα των δύο υλοποιήσεωνLDA 35

Δ΄.1 πίνακας με τις παραμέτρους που παίρνει η συνάρτηση track.getSimilar . 40
Δ΄.2 Αποτελέσματα F1-scores για τα πειράματα της μουσικής και των εικόνων 48

Κεφάλαιο Α΄

Εισαγωγή

Καθημερινά γινόμαστε δέκτες πληθώρας πληροφορίας πολυμεσικού περιεχομένου όπως έγγραφα, μουσική, εικόνες που λόγο του όγκου, της πολυμορφίας καθώς και της πολυπλοκότητα τους είναι δύσκολο να διαχειριστούν. Ως απόρροια δημιουργείται η επιτακτική ανάγκη της εισαγωγής μεθόδων οι οποίες είναι ικανές να οργανώσουν τους πόρους σε θέματα καθώς και τον υπολογισμό ομοιότητας μεταξύ τους. Αναλυτικότερα η ομοιότητα μεταξύ δύο κειμένων υπολογίζεται με στατιστικά μέτρα, όπως η ομοιότητα συνημίτονου(Cosine Similarity), Tanimoto, Dice, Pearson , ενώ όσο αφορά την ανεύρεση θεματολογίας γίνεται χρήση αλγορίθμων μοντέλων θεμάτων(Topic Models) οι οποίοι είναι υπεύθυνοι για την ανεύρεση της θεματολογίας σε μεγάλες συλλογές κειμένων λ.χ LDA, LSA. Για τις εικόνες χρησιμοποιούνται τεχνικές οι οποίες εξάγουν χαμηλού επιπέδου οπτικά χαρακτηρίστηκα όπως το σχήμα[34] το texture[30][32] και το gradient[33]. Αυτά τα οπτικά χαρακτηριστικά αποτελούν έναν αποτελεσματικό τρόπο για να γίνει αναπαράσταση πληροφορίας σε μία εικόνα. Οι εικόνες συγκρίνονται με βάση τα χαρακτηρίστηκα χρησιμοποιώντας συναρτήσεις απόστασης όπως η ευκλείδεια και η κατανομή χ τετράγωνο(chi square). Επιπρόσθετα χρησιμοποιούνται χαρακτηρίστηκα υψηλού επιπέδου(ετικέτες) τα οποία ανατίθενται από ανθρώπινες οντότητες. Αντίστοιχα με τις εικόνες στα ηχητικά κομμάτια γίνεται χρήση χαρακτηριστικών που εξάγονται από τα μουσικά κομμάτια όπως MFCC coefficients[35] καθώς και χρήση ετικετών που αποδόθηκαν από ανθρώπους. Στόχος της πτυχιακής εργασίας είναι η υλοποίηση του αλγορίθμου Latent Dirichlet Allocation προκειμένου να γίνει εξαγωγή της θεματολογίας σε Ελληνικά κείμενα καθώς και την χρήση της παραλλαγής της μεθόδου subset που είναι μια παραλαγή της subset με σκοπό την ανεύρεση παρόμοιων

μουσικών κομματιών και εικόνων.

Α΄.1 Βιβλιογραφική έρευνα

Αρχικά σύμφωνα με την εργασία [30] αναφέρεται πως η φυσική επεξεργασία της γλώσσας(Natural Language Processing) η αλλιώς NLP, είναι ένας από τους πιο απαιτητικούς κλάδους στην επιστήμη των υπολογιστών. Η οποία χρησιμοποιείται για εφαρμογές όπως διαχείριση πληροφορίας (information management), σημασιολογική εξόρυξη (Semantic Mining) και επίσης δίνεται η δυνατότητα στους υπολογιστές να "καταλάβουν", να εξάγουν γνώση από το κείμενο. Οι τεχνικές οι οποίες είναι βασισμένες στη μοντελοποίηση θεμάτων "Topic Modeling", έχουν εφαρμοστεί εκτεταμένα στην φυσική επεξεργασία της γλώσσας για την ανακάλυψη της θεματολογίας και την σημασιολογική εξόρυξη των εγγράφων. Επίσης οι συγγραφείς αναφέρουν, πως σημαντική πρόοδος σημειώθηκε στον τομέα της ανάκτησης της πληροφορίας «Information Retrieval ». Ιστορικά Η βασική μεθοδολογία η οποία προτάθηκε από τους ερευνητές της ανάκτησης της πληροφορίας για συλλογές κειμένων είναι η αντιστοίχιση ενός εγγράφου σε ένα διάνυσμα πραγματικών αριθμών, όπου κάθε ένα από το οποίο αναπαριστά αναλογία μετρητών των λέξεων. Στο tf-idf[31], γίνεται επιλογή ένα σύνολο όρων η λέξεων και για κάθε έγγραφο στην συλλογή υπολογίζεται ένας μετρητής από τον αριθμό εμφανίσεων κάθε λέξης. Μετά από την κατάλληλη κανονικοποίηση αυτός ο μετρητής συχνότητας όρων συγκρίνεται με την αντίστροφη συχνότητα εγγράφου, η οποία υπολογίζεται από τον αριθμό των εμφανίσεων μιας λέξης μέσα στην συλλογή κειμένου. Το τελικό αποτέλεσμα είναι ένας πίνακας X του οποίου οι στήλες περιέχουν τις τιμές tf-idf για κάθε έγγραφο στην συλλογή. Η προσέγγιση του tf-idf έχει κάποια θελκτικά χαρακτηρίστηκα και πιο συγκεκριμένα, στην αναγνώριση συνόλου λέξεων τα οποία είναι διαχωρίστηκα για τα έγγραφα στην συλλογή, η προσέγγιση αυτή ακόμα παρέχει μικρή μείωση στο μήκος της περιγραφής του κειμένου και αποκαλύπτει ελάχιστα πράγματα για την στατιστική δομή των εγγράφων. Για την αντιμετώπιση των παρακάτω ελλείψεων οι ερευνητές πρότειναν εναλλακτικές μεθόδους μείωσης διαστάσεων όπως η μέθοδος του LSI η οποία προέρχεται από τα αρχικά Latent Semantic Indexing[32]. Το LSI χρησιμοποιεί Singular Value Decompositionτο διάνυσμα Ξ για την αναγνώριση μιας γραμμική υποπεριοχής(Linear Subspace) στον χώρο των tf-idf χαρακτηριστικών τα οποία εμφανίζουν την μεγαλύτερη διακύμανση στο μοντέλο. Η προσέγγιση αυτή προσφέρει σημαντική συμπίεση σε μεγάλες συλλογές εγγράφων. Επιπρόσθετα, σύμ-

φωνα με το [32] τα χαρακτηρίστηκα του LSI τα οποία είναι γραμμικοί συνδυασμοί των tf-idf χαρακτηριστικών μπορούν να συλλάβουν γλωσσικά χαρακτηρίστηκα όπως συνωνυμία και πολυσημία λέξεων. Το πρόβλημα όμως που παρουσιάζεται με την τεχνική του LSI εξηγείται αναλυτικά από την εργασία[32]. Στην συνέχεια ο Hoffman(1999) δημιούργησε το PLSI(Probabilistic Latent Semantic Indexing) το οποίο είναι γνωστό και ως aspect μοντέλο, ως μια εναλλακτική μέθοδος από το LSI. Η μέθοδος του pLSI παίρνει δείγματα για κάθε λέξη από ένα μοντέλο δειγμάτων (mixture model) και τα mixture συστατικά μπορούν να θεωρηθούν ως multinomial τυχαίες μεταβλητές οι οποίες μπορούν να αναπαραστήσουν θέματα. Έτσι κάθε λέξη ανήκει σε ένα θέμα το οποίο για όλες τις λέξεις δεν είναι απαραίτητα το ίδιο. Κάθε έγγραφο αναπαρίσταται από μια κατανομή πιθανότητας σε έναν σταθερό αριθμό θεμάτων. Αυτή η κατανομή πιθανότητας των θεμάτων είναι η σύντομη περιγραφή του εγγράφου. Παρόλο που η δουλεία του Hoffman είναι σημαντική για τα πιθανότητά μοντέλα κειμένου(probabilistic text modelling), το πρόβλημα που σημειώνεται είναι πως δεν παρέχουν κάποιο πιθανοτικό μοντέλο για το επίπεδο των εγγράφων[32]. Επίσης αναφέρεται από τον ίδιο ότι στο pLSI κάθε έγγραφο αναπαρίσταται από μια κατανομή πιθανότητας και δεν υπάρχει κάποιο πιθανοτικό γενερατιε μοντέλο που αξιοποιεί την κατανομή αυτή πιθανότητας για να δημιουργήσει ένα καινούργιο έγγραφο. Ως απόρροια αυτό δημιουργεί τα εξής προβλήματα: Το πρώτο από αυτά είναι το Overfitting και δεν είναι ξεκάθαρο για το πως θα γίνει ανάθεση πιθανοτήτων σε έγγραφα τα οποία βρίσκονται εκτός από το σύνολο της εκπαίδευσης (training set). Στην εργασία [33] Το LDA η αλλιώς Latent Dirichlet Allocation είναι ένας αλγόριθμος μη επιβλεπόμενης μάθησης, είναι μια στατιστική προσέγγιση για την μοντελοποίηση μιας συλλογής κειμένου, βρίσκοντας τα θέματα στις συλλογές αυτές. Στο LDA οι λέξεις έχουν ισχυρή σημασιολογική πληροφορία για το έγγραφο. Επίσης τα έγγραφα τα οποία σημειώνουν παρόμοια θεματολογία θα χρησιμοποιούν κοινές λέξεις. Ο αλγόριθμος αυτός είναι μη επιβλεπόμενος καθώς δεν είναι γνωστό εξαρχής η θεματολογία των κειμένων. Επίσης σύμφωνα με τους [34] προτού γίνει εκτέλεση ενός από τους αλγορίθμους μη επιβλεπόμενης μάθησης η χρήση τεχνικών προεπεξεργασίας(Preprocessing) μπορεί να έχει σημαντική επίδραση στα αποτελέσματα. Επίσης τεχνικές προεπεξεργασίας όπως μετατροπή λέξεων από κεφαλαίες σε μικρές, μετατροπή μιας λέξης στην αρχική της ρίζα και αφαίρεση κοινών λέξεων. Στην συνέχεια γίνεται αναφορά πως στην προεπεξεργασία κειμένων σκοπός τον τεχνικών αυτών είναι η μείωση της πολυπλοκότητας των δεδομένων, ενώ παράλληλα το χάσιμο της πληροφορίας να είναι σχετικά χαμηλό [35]. συνέχεια οι συγγραφείς,

αναφέρουν μερικές από τις τεχνικές οι οποίες χρησιμοποιούνται στην προεπεξεργασία Οι τεχνικές που αναφέρθηκαν είναι : Punctuation στην οποία γίνεται αφαίρεση ειδικών συμβόλων από τα κείμενα όπως (!, ;, κ.τ.λ.), html ετικέτες. Παρόλα αυτά αναφέρουν ότι αυτό το είδος προεπεξεργασίας μπορεί σε κάποιες εφαρμογές να θεωρηθεί λάθος η αφαίρεση τους(π.χ. hashtags που εμφανίζονται σε δεδομένα κάποιου μέσου κοινωνικής δικτύωση όπως Twitter, Facebook). Μια ακόμα τεχνική είναι αυτή των αριθμών στην οποία γίνεται αφαίρεση αριθμών. Στο σημείο αυτό να σημειωθεί πως υπάρχουν περιπτώσεις που η αφαίρεση τους μπορεί να αφαίρεσει σημαντική πληροφορίας λ.χ. σε κείμενα τα οποία ασχολούνται με τον νόμο. Μετά μια άλλη τεχνική που αναφέρθηκε είναι η μετατροπή ολοκλήρου του κειμένου σε κεφαλαία η μικρά γράμματα. Η λογική πίσω από την ενέργεια αυτή γίνεται προκειμένου οι αλγόριθμοι να μην θεωρήσουν λέξεις που είναι ίδιες ως διαφορετικές λ.χ. η λέξη Γρηγορος, γρήγορος έχουν την ίδια σημασία. Μια άλλη τεχνική ο προεπεξεργασίας είναι η τεχνική του Stemming, στην οποία μια λέξη μετατρέπεται στην βασική της ρίζα(Porter 1980). Επιπρόσθετα το Stemming χρησιμοποιείται ως μια τεχνική με την οποία μικραίνει το μέγεθος του λεξιλογίου. Stopword Removal, στην μέθοδο αυτή γίνεται αφαίρεση των λέξεων οι οποίες είναι απίθανο να προσφέρουν πληροφορία. Επίσης δεν υπάρχει κάποιο standard για το πόσο μεγάλο είναι το κείμενο άλλα οι περισσότερες λίστες ο αριθμός τους πηγάζει από 100 και 1000 όρους [36]. Το N-gram inclusion από τις προτάσεις παίρνονται παράθυρα λέξεων μεγέθους N και θεωρούνται ως μια ενιαία πρόταση. Η μέθοδος αυτή χρησιμοποιείται κυρίως για λέξεις οι οποίες είναι ανάλογες με το κείμενο μέσα στο οποίο βρίσκονται. Στην βιβλιογραφία ένας συνηθισμένος αριθμός για το N είναι ο 1, 2, 3. Στην συνέχεια αναφέρεται πως μπορεί να εφαρμοστεί και φιλτράρισμα των N-grams στην εργασία όμως αυτήν δεν έγινε κάτι τέτοιο. Το τελευταίο μέτρο προεπεξεργασίας που αναφέραναν είναι η τεχνική του Infrequently Used Term στην οποία γίνεται αφαίρεση λέξεων οποίες δεν εμφανίζονται συχνά στο έγγραφο. Η λογική πίσω από αυτήν την διαδικασία είναι ότι οι λέξεις αυτές εφόσον δεν εμφανίζονται σε μεγάλη συχνότητα δεν συνεισφέρουν στο μοντέλο και επομένως μπορούν να διαγραφούν. Επιπρόσθετα ένα ακόμα όφελος είναι ότι μπορεί το μέγεθος του λεξιλογίου να μειωθεί αισθητά. Ως απόρροια αυτό μπορεί να επιταχύνει την ανάλυση του κειμένου.Ένας γενικός κανόνας είναι η αφαίρεση όρων οι οποίοι εμφανίζονται σε λιγότερο από 0.5-1 τις εκατό των εγγράφων. Ακόμα από την βιβλιογραφία συνέλεξαν έρευνες [36] και τις κατέταξαν με βάση αναφορές. Για κάθε μια από αυτές έγινε αναφορά στις ποιες τεχνικές προεπεξεργασίας εκτελέστηκαν. Τα πειράματα εκτελέστηκαν σε αλγορίθμους μη επιβλεπόμενης

μάθησης, η εργασία που σημείωσε τα περισσότερα citation είναι αυτή των [37] οι οποίοι εκτέλεσαν της εξής μορφές προεπεξεργασίας: Πρώτα έγινε αφαίρεση των ειδικών χαρακτήρων, στην συνέχεια εφαρμόστηκε Στεμμινγ, μετατροπή χαρακτήρων σε μικρούς. Ένα άλλο σημαντικό ζήτημα είναι πως μπορεί να γίνει αξιολόγηση των μοντέλων θεμάτων. Στην εργασία [33] οι συγγραφείς ορίζουν τα εξής μέτρα τα οποία μπορούν να χρησιμοποιηθούν για την αξιολόγηση των μοντέλων θεμάτων(topic models). Το πρώτο μέτρο που όρισαν είναι το μέτρο του Perplexity το οποίο θεωρείται μια από της στάνταρτ μεθόδους αξιολόγησης στατιστικών μοντέλων φυσικής γλώσσας. Το μέτρο αυτό υποδηλώνει την αβεβαιότητα πρόβλεψης μιας λέξης. Μικρότερες τιμές είναι πιο επιθυμητές, το οποίο υποδηλώνει καλύτερη ικανότητα γενικοποίησης. Perplexity με τιμή που ισοδυναμεί με το μέγεθος του λεξιλογίου υποδηλώνει πως η απόδοση του μοντέλου είναι τυχαία. Η μέθοδος αυτή όμως αρκετές τονίζεται πως δεν είναι ικανοποιητική καθώς τα αποτελέσματα που προκύπτουν πολλές φορές δεν συμβαδίζουν με αυτά που θα συμφωνούσε κάποιος άνθρωπος. Μια δεύτερη μέθοδος είναι αυτή του Analysis of inferred parameters, στην μέθοδο αυτή γίνεται μελέτη των κατανομών πιθανότητας για τα έγγραφα που προέκυψαν. Η αξιολόγηση γίνεται από κάποιον άνθρωπο. Στην εργασία [37] [40] Τεχνικές θεμάτων(topic modeling), οι οποίες είναι βασισμένες στο LDA έχουν εφαρμοστεί αντίστοιχα σε μια ευρεία γκάμα εφαρμογών όπως φυσική επεξεργασία της γλώσσας, εξόρυξη κειμένου(text mining), ανάλυση μέσων κοινωνικής δικτύωσης (Social Media Analytics), ανάκτηση πληροφορίας, στις πολιτικές επιστήμες, στην αρχιτεκτονική λογισμικών, γεωγραφία κ.τ.λ. Για παράδειγμα η εφαρμογή τεχνικών μοντελοποίησης θεμάτων στα Social Media Analytics μπορεί να διευκολύνει την κατανόηση των συζητήσεων που διεξάγουν μεταξύ τους οι άνθρωποι που ανήκουν στα κοινωνικά δίκτυα. Όσο αφορά τις φυσικές επιστήμες στην εργασία [41] οι ερευνητές χρησιμοποιήσαν ένα μη επιβλεπόμενο μοντέλο θεμάτων το οποίο είναι βασισμένο στο LDA για την μοντελοποίηση αντίθετων απόψεων του οποίου σκοπός είναι η εύρεση απόψεων από διαφορετικές οπτικές γωνίες καθώς και η εύρεση της απόστασης της κάθε μίας από μία με συγκεκριμένα κριτήρια. Το μοντέλο αυτό ονομάστηκε Cross-Perspective Topic(CPT). Στην συνέχεια διεξάχθηκαν πειράματα τόσο με ποιοτικά όσο και με ποσοτικά μέτρα σε δύο σύνολα δεδομένων το πρώτο από αυτά αναφέρονται σε δηλώσεις γερουσιαστών των Η.Π.Α που υποδηλώνουν τις πολιτικές θέσεις των γερουσιαστών. Όσο αφορά το δεύτερο σύνολο των δεδομένων αυτό εξήχθη από τρία μέσα μαζικής ενημέρωσης. Το πρώτο βρίσκεται στις Ηνωμένες πολιτείες της Αμερικής(New York Times), Κίνα(Xinhua News) και Ινδία(Hindu). Για την επαλήθευση της απόδο-

σης του μοντέλου αυτού έγινε χρήση του LDA και του corrLDA ως baseline. Μια άλλη ομάδα ερευνητών χρησιμοποιήσαν το topic modeling στον τομέα της μηχανικής λογισμικού η μεθοδολογία που ακολούθησαν είναι η εξής: πρώτον έγινε εκτέλεση του αλγόριθμου LDA, προκειμένου να γίνει εξαγωγή των θεμάτων από τον πηγαίο κώδικα, ακόμα έγινε οπτικοποίηση της ομοιότητας μεταξύ του κώδικα των εφαρμογών και η απόκτηση για κάθε έγγραφο πηγαίου κώδικα τις κατανομές πιθανότητας των θεμάτων τους. Η μέθοδος που προτείναν χρησιμοποιήθηκε σε 1555 προγράμματα ανοικτού κώδικα, με προέλευση την Apache και το Source Forge το οποίο περιέχει 19 εκατομμύρια γραμμές κώδικα. Στην εργασία [42] Οι συγγραφείς χρησιμοποίησαν μια μέθοδο η οποία είναι βασισμένη στο LDA για την αυτόματη κατηγοριοποίηση λογισμικών. Η μέθοδος αυτή της αποδόθηκε το όνομα LACT. Για την αξιολόγηση του LACT έγινε χρήση 43 λογισμικών ανοικτού κώδικα σε διαφορετικές γλώσσες προγραμματισμού. Το αποτέλεσμα που προέκυψε είναι ότι το LACTμπορεί να κατηγοριοποιήσει λογισμικό με βάση την γλώσσα προγραμματισμού που χρησιμοποιήθηκε για την συγγραφή του κώδικα. Στην εργασία [43][44] έγινε μια πρόταση για μια έκδοση η οποία βασίζεται στο LDA με σκοπό τον εντοπισμό λογικών σφαλμάτων στον κώδικα (Bug Localization). Η ιδέα τους εφαρμόστηκε σε λογικά σφάλματα (Bugs) στο Mozila και το Eclipse και τα αποτελέσματα τους έδειξαν πώς η προσέγγιση LDA που πρότειναν σημείωσε καλύτερα από την αντίστοιχη προσέγγιση με χρήση του LDA για την αξιολόγηση και την ανάλυση λογικών σφαλμάτων(bugs) τα οποία εντοπίζονται μέσα στον πηγαίο κώδικα. Στην εργασία [45], οι συγγραφείς διεξήγαγαν μια έρευνα στο Topic Modelingμε εφαρμογή στην Μηχανική λογισμικού (Software Engineering) προκειμένου να προσδιορίσουν σε τι βαθμό έχει γίνει εφαρμογή μοντέλα θεμάτων(topic model) σε εφαρμογές λογισμικού. Στην έρευνα αυτή έγινε επικέντρωση σε άρθρα τα οποία είχαν γραφεί από 14 Δεκεμβρίου του 1999 έως το Δεκέμβριο του 2014 και έλεγξαν διεξοδικά 167 άρθρα τα οποία χρησιμοποιούν Topic Modeling σε κλάδο του Software Engineering . Αυτοί βρήκαν πως η συντριπτική πλειοψηφία των ερευνών επικεντρώθηκαν σε έναν περιορισμένο αριθμό εργασιών στην μηχανική λογισμικού(Software Engineering) π.Χ. διόρθωση κώδικα. Ακόμα στις περισσότερες μελέτες έγινε χρήση απλών μοντέλων θεμάτων(topic model). Στην εργασία [46], οι συγγραφείς επικεντρώθηκαν, σε μία έρευνα που έγινε σε μοντέλα θεμάτων με χρήση soft clusteringτεχνικών σε συλλογές κειμένων. Επιπρόσθετα, οι συγγραφείς παρουσίασαν κάποιες εφαρμογές μοντέλων θεμάτων για την μοντελοποίηση συλλογών κειμένων. Ακόμα το LDA μπορεί να εφαρμοστεί και σε εφαρμογές εικόνας και μουσικής. Πιο συγκεκριμένα όσο αφορά τους

τομείς αυτούς χρήσιμες θεωρούνται οι ετικέτες με τις οποίες μπορούν να συνδέονται οι πόροι αυτοί. Σύμφωνα με το [44] οι ετικέτες μπορούν να προσφέρουν μεγάλο όγκο σε πληροφορία λόγο των μέσων κοινωνικής δικτύωσης τα οποία επέτρεψαν στους χρήστες να κάνουν επισημείωση των δεδομένων τους. Οι συγγραφείς χρησιμοποίησαν για τις εικόνες από το κοινωνικό δίκτυο του Flickr όπου για κάθε εικόνα είχαν πρόσβαση και στις αντίστοιχες ετικέτες. Μετά χρησιμοποίησαν τις ετικέτες, δόθηκαν ως είσοδο στο api της Google η οποία επέστρεψε μετρητές αποτελεσμάτων οι οποίοι χρησιμοποιήθηκαν για τον υπολογισμό ομοιοτήτων. Τα μέτρα που πρότειναν είναι το Jaccard Index, Overlap, Dice, PMI. Επίσης Στην εργασία [45] παρουσίασαν μια σειρά τεχνικών ομοιότητας για μικρά κομμάτια κειμένου. Επίσης προτείνουν ένα υβριδικό μοντέλο υπολογισμού ομοιότητας όπου ένα από τα κομμάτια αυτά χρησιμοποιεί μεταδεδομένα. Τέλος στην εργασία [46], έγινε υλοποίηση επίσης ενός υβριδικού συστήματος το οποίο είναι βασισμένο στο μέτρου ομοιότητας του Subset Similarity. Οι αλλαγές που πρότειναν βρίσκονται στο κομμάτι της Συμμετρικής ομοίοτητας(Symetric Similarity), εντοπισμός συνωνυμίας(Synonymy Detection) και προσσεγιστικό ταίριασμα για την επαλήθευση της μεθόδους αυτής χρησιμοποιήθηκαν ως Baseline οι αλγόριθμοι: Exact, Substring, Subset. Επίσης η μέθοδος σημείωσε καλύτερα αποτελέσματα από τους αλγορίθμους Baseline.

Α΄.2 Προτεινόμενη Λύση

Η λύση που προτείνεται για τα κείμενα είναι η ανάπτυξη του αλγόριθμου LDA ο οποίος είναι σχετικά εύκολος στην υλοποίηση, ενώ ταυτόχρονα επιδιορθώνει το πρόβλημα της πολυσημίας του LSA. Επιπλέον ένα άλλο πλεονέκτημα αποτελεί η ευκολία στην ερμηνεία των θεμάτων που προέκυψαν από την εκτέλεση του αλγορίθμου LDA, καθώς κάθε θέμα περιέχει τις λέξεις που την περιγράφουν και οι οποίες είναι ταξινομημένες σε φθίνουσα σειρά. Η ταξινόμηση γίνεται ως προς την πιθανότητα της λέξης να ανήκει στο θέμα. Επιπρόσθετα, για τις εικόνες και τα μουσικά κομμάτια χρησιμοποιήθηκαν ετικέτες που αποδόθηκαν από ανθρώπους. Η μέθοδος που αξιοποιεί τα υψηλά επιπέδου αυτά χαρακτηριστικά(ετικέτες) αποτελεί παραλλαγή της subset και αναλαμβάνει να βρει για κάθε πόρο έναν άλλο πόρο με τον οποίο μοιράζεται τις περισσότερες κοινές ετικέτες.

Κεφάλαιο Β΄

Βασικές έννοιες

Στην συνέχεια παρατίθονται απαραίτητοι ορισμοί για την κατανόηση της πτυχιακής.

Β΄.1 Precision

Στην ανάκτηση της πληροφορίας η έννοια του Πρεσισιον αφορά τον λόγο των σχετικών εγγράφων που επιστράφηκαν, προς τον συνολικό αριθμό των εγγράφων που επιστράφηκαν. Στο μέτρο αυτό έμφαση δίνεται στα σωστά αποτελέσματα που επιστράφηκαν(TP) ενώ τα λανθασμένα συμβολίζονται ως (FP). τύπος του Precision δίνεται από την παρακάτω σχέση:

$$\frac{TP}{(TP + FP)} \tag{Β΄.1}$$

Β΄.2 Recall

Η έννοια του Recall, είναι ο λόγος των σχετικών εγγράφων που επιστράφηκαν, ενώ στον παρονομαστή έχουμε τον αριθμό των κειμένων ο οποίος έπρεπε να επιστραφεί. Δίνεται έμφαση στα σωστά αποτελέσματα που επιστράφηκαν από όλα τα σωστά αποτελέσματα για παράδειγμα σε περίπτωση που από τα 100 συνολικά έγγραφα επιστραφούν τα 50 τα οποία όλα είναι TP και κανένα FP. Ενώ στα υπόλοιπα 50 που δεν επιστράφηκαν τα 40 είναι σωστά τότε έχουμε Recall = 50 / 90 = 0.555, ο τύπος του Recall δίνεται απο την παρακάτω σχέση:

$$Recall = \frac{TP}{(TP + FN)} \tag{Β΄.2}$$

B΄.3 F1-Score

Το F1-score είναι μια άλλη μετρική ακρίβειας που υπολογίζεται από τον αρμονικό μέσο όρο του precision και του recall και αναπαρίσταται ως εξής:

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{B΄.3}$$

Όπου το P είναι το Precision και το R είναι το Recall που ορίστηκαν παραπάνω. Το F1-score παίρνει τιμές οι οποίες κυμαίνονται μέσα στο εύρος μηδέν και ένα. Η τιμή ένα σημαίνει ότι τα επιστρεφόμενα αποτελέσματα είναι σωστά και παράλληλα επιστράφηκε όλος ο αριθμός των σωστών. Σε αντίθεση με το μηδέν με το οποίο υποδηλώνει το σύστημα μας δεν επιστρέφει ούτε ένα σωστό αποτελέσματα.

Επίσης υπάρχει και μια γενικότερη μορφή του f1-score το οποίο δίνει το δικαίωμα να εισαχθεί ένας συντελεστής βαρύτητας στο Πρεσισιον η στο Recall ο λόγος που αυτό μπορεί να φανεί χρήσιμο είναι ότι σε κάποια συστήματα η μια μετρική μπορεί να είναι σημαντικότερη από μια άλλη και για αυτό θα θέλαμε να δώσουμε μεγαλύτερη βαρύτητα σε κάποια από αυτές τις μετρικές.

$$F_b = \frac{(b^2 + 1) \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{B΄.4}$$

B΄.4 Train-Test split

Είναι μια τεχνική που χρησιμοποιείται εκτενέστερα στην μηχανική μάθηση, με την οποία γίνεται διαχωρισμός των δεδομένων. Αναλυτικά λαμβάνουμε τυχαία τα δεδομένα μας και τα χωρίζουμε σε δύο κατηγορίες. Τα πρώτα θα χρησιμοποιηθούν για την εκπαίδευση κάποιου μοντέλου μηχανικής μάθησης και αποτελούν το μεγαλύτερο μέρος των δεδομένων. Απο την άλλη το δεύτερο για να δοκιμάσουμε τα αποτελέσματα και να δούμε το κατά πόσο οι προβλεπόμενες τιμές ανταποκρίνονται στα σωστά αποτελέσματα. Επιπλέον αξιοσημείωτο είναι πως για την αξιολόγηση των αποτελεσμάτων χρησιμοποιήθηκαν τα μέτρα που προαναφέρθηκαν.

Β΄.5 K-fold cross validation

Είναι μια μέθοδος που χρησιμοποιείται για την εκτίμηση της απόδοσης ενός μοντέλου στο οποίο αφού έχει γίνει ανακάτεμα των δεδομένων τα χωρίζουμε σε ν ισομερής ομάδες. Για κάθε ομάδα παίρνουμε το ένα από αυτά και το χρησιμοποιούμε για το validation-set και τα υπόλοιπα για εκπαίδευση. η διαδικασία αυτή θα επαναληφθεί έτσι ώστε κάθε ομάδα να χρησιμοποιηθεί για train set. Τέλος για κάθε εκτέλεση παίρνουμε το αποτέλεσμα της μετρικής της αξιολόγησης και υπολογίζεται ο μέσος όρος του αποτελέσματος.

Β΄.6 Subset Similary

Για έναν πόρο με κ ετικέτες γίνεται υπολογισμός του αριθμού των κοινών ετικετών με τους εναπομείναντες πόρους που είναι κατοχυρωμένοι. Η ομοιότητα στην περίπτωση αυτή καθορίζεται από τον αριθμό από των κοινών ετικετών και όσο περισσότερες κοινές ετικέτες έχουν τότε τόσο πιο όμοιοι θεωρούνται οι πόροι μεταξύ τους.

Β΄.7 XML

Τα αρχικά της προέρχονται από το (eXtensible Markup Language). Είναι μια γλώσσα σήμανσης, δηλαδή χρησιμοποιεί ετικέτες για να περιγράψει την σημασιολογία τμημάτων των κειμένων. Με την χρήση ετικετών, μπορεί να γίνει κωδικοποίηση εγγράφων ώστε να είναι κατανοητά απο τον άνθρωπο αλλα και απο την μηχανή. Η XML χρησιμοποιείται για την αναπαράσταση αυθαίρετων δομών δεδομένων.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

Σχήμα Β΄.1: Παράδειγμα XML κώδικα

Β΄.8 JSON

Είναι ένα ελαφρύ μορφότυπο που χρησιμοποιείται για την ανταλλαγή δεδομένων, μεταξύ εφαρμογών, διαδικτυακά. Η Json είναι πιο απλή από την γλώσσα xml κάτι το οποίο την καθιστά πιο γρήγορη για την ανταλλαγή δεδομένων μεταξύ διαδικτυακών εφαρμογών καθώς μεταδίδεται λιγότερη πληροφορία. Κάθε Json αρχείο ξεκινάει με αριστερή και κλείνει με δεξιά αγκύλη, δηλαδή με το [και το]. Έτσι οτιδήποτε περιβάλλεται μέσα στις αγκύλες θεωρείτε ως αντικείμενο. Υπάρχουν δύο διαφορετικές δομές που μπορούν να χρησιμοποιηθούν στα Json αρχεία. Στην πρώτη δομή στο δεξί μέρος υπάρχει μια αλφαριθμητική σταθερά η οποία μπορεί να ακολουθηθεί από: αριθμητική σταθερά, λογικές σταθερές(True, False), Null καθώς και αντικείμενο. Στην δεύτερη δομή στο δεξί μέρος υπάρχει αλφαριθμητική σταθερά και στο αριστερό μέρος υπάρχει πίνακας ο οποίος υποστηρίζει τους τύπους δεδομένων που προαναφέρθηκαν. Τέλος υποστηρίζονται καιUnicode ονόματα συμβολοσειρών.

Β΄.9 RSS Feeds

Το RSS(Rich Site Summary) Χρησιμοποιείται για την ανταλλαγή περιεχομένου πληροφοριακού χαρακτήρα διαμέσου του παγκόσμιου ιστού. Η RSS στηρίζεται στην γλώσσα XML. Αξιοσημείωτο είναι το γεγονός πως ο χρήστης μπορεί να εγγραφεί σε RSS-Feeds από πολλαπλούς ιστοτόπους και έτσι να μπορεί να λαμβάνει συγκεντρωτική ενημέρωση σε ένα μέρος(λ.χ εφαρμογή περιηγητή)

```
name:        "Alex"
last-name:   "Τουλάκης"
Nums:
    0:       10
    1:       20
    2:       30
    3:       40
Courses:
    C:       0
    C2:      1
```

Σχήμα Β΄.2: Παράδειγμα json κώδικα

```
−<rss version="2.0">
  −<channel>
      <title>RSS Τίτλος</title>
      <description>Αυτό είναι ένα παράδειγμα RSS κειμένου </description>
      <link>http://www.example.com/main.html</link>
      <lastBuildDate>Mon, 06 Sep 2010 00:01:00 +0000 </lastBuildDate>
      <pubDate>Sun, 06 Sep 2009 16:20:00 +0000</pubDate>
      <ttl>1800</ttl>
    −<item>
        <title>Παράδειγμα καταχώρισης</title>
      −<description>
          Ακολουθεί ένα κείμενο που περιέχει μια ενδιαφέρουσα περιγραφή.
        </description>
        <link>http://www.example.com/blog/post/1</link>
        <guid isPermaLink="false">7bd204c6-1655-4c27-aeee-53f933c5395f</guid>
        <pubDate>Sun, 06 Sep 2009 16:20:00 +0000</pubDate>
      </item>
    </channel>
</rss>
```

Σχήμα Β΄.3: Παράδειγμα Rss κώδικα

B΄.10 StopWords

Μία αρκετά συνηθισμένη τεχνική σε εφαρμογές φυσικής επεξεργασίας της γλώσσας είναι η δημιουργία του stop set. Το stop set είναι λίστα η οποία αποτελείται από λέξεις με λειτουργικό χαρακτήρα εμφανιζόμενες σε όλα τα κείμενα, αλλά χωρίς κανένα εννοιολογικό περιεχόμενο. Οι λέξεις που υπάρχουν στο stop λίστα αφαιρούνται από το λεξιλόγιο. Η ιδέα, στην οποία στηρίζεται αυτή η διαδικασία, είναι ότι υπάρχουν λέξεις σε όλα τα κείμενα που δεν σχετίζονται με το περιεχόμενο του κειμένου ούτε βοηθούν στο διαχωρισμό των κειμένων μεταξύ τους. Η διαδικασία αυτή συμβάλλει σημαντικά στη μείωση του λεξιλογίου, γεγονός που καθιστά την αναζήτηση και ανάκτηση των κειμένων πολύ πιο γρήγορη και αποτελεσματική[29]. Για την εξαγωγή των stop words από τα κείμενα στην εργασία αυτή, Στο πείραμα με τα κείμενα έγινε χρήση ενός Stop Word Manager ο οποίος χτίζει ένα μεγάλο set από λέξεις. Ο StopWord Manager: δημιουργεί την stopword λίστα διαβάζοντας από αρχεία τα οποία είναι τοποθετημένα σε φάκελο, με αυτόν τον τρόπο δίνεται η δυνατότητα επέκτασης της λίστας με τις λέξεις σταματήματος. Η δυνατότητα επέκτασης της λίστας με αυτόν τον τρόπο έγινε για την Ελληνική γλώσσα καθώς δεν υπήρχε κάποιο διαθέσιμο προγραμματιστικό πλαίσιο (Framework) που να υποστηρίζει την Νεοελληνική γλώσσα. Για την Αγγλική γλώσσα χρησιμοποιήθηκαν τα stopwords που παρέχονται από την βιβλιοθήκη της NLTK[4] χωρίς να υπάρξει ανάγκη αποθήκευση του σε αρχείο.

B΄.11 Stemming

Στη διαδικασία του Stemming, οι λέξεις οι οποίες έχουν κοινή ρίζα χάνουν την κατάληξη τους και κατατάσσονται στην ίδια ρίζα. Ως αποτέλεσμα το μέγεθος του κειμένου μικραίνει, επίσης βοηθάει και στην διαδικασία στην οποία γίνεται αναζήτηση για την ανάκτηση κειμένων. Για παράδειγμα σε περίπτωση που δοθεί ως είσοδο η λέξη «Πατάτα» και υπάρχει ένα κείμενο που περιέχει τον όρο «Πατάτες» αυτό δεν θα επιστραφεί, αντιθέτως με την μέθοδο του stemming αυτές οι δύο λέξεις θα τοποθετηθούν στην ρίζα «Πατατ». Μία άλλη εφαρμογή του Stemming είναι σε προβλήματα NLP όπου ίδιες λέξεις γράφονται με διαφορετικό τρόπο π.χ λόγω της γραμματικής τους και θεωρείται επιθυμητή η μετατροπή στην ίδια ρίζα. Ένα πρόβλημα που μπορεί να εμφανιστεί με την χρήση τους είναι το γεγονός πως μπορεί λέξεις να ομαδοποιηθούν στην ίδια ρίζα ενώ κάτι τέτοιο δεν θα έπρεπε να ισχύει για παράδειγμα η λέξη «ΓΡΗΓΟΡΟΣ» και

«ΓΡΗΓΟΡΗΣ» θα επιστρέψουν την λέξη «ΓΡΗΓΟΡ».

Β΄.12 Http

Πρωτόκολλο επικοινωνίας που χρησιμοποιείται για την αμφίδρομη δεδομένων ανάμεσα σε έναν διακομιστή(server) και έναν πελάτη (client).

Β΄.12.1 Http Requests

Τύποι ενεργειών που αιτούνται να εκτελεστούν απο τον διακομιστή. Οι κυριότερες εξ΄ αυτών είναι:

Β΄.12.2 Get

Για τα Http αιτήματα GET: Γίνεται αίτηση ενός πόρου. Αιτήματα που κάνουν χρήση του Http Get επιστρέφουν μόνο δεδομένα [28]. Σε αυτό το είδος του αιτήματος η αποστολή παραμέτρων στον διακομιστή γίνεται με την χρήση των Query Strings. Τα Query Strings ξεκινούν με το σύμβολο του αγγλικού ερωτηματικού(?) στην συνέχεια το όνομα της παραμέτρου ακολουθημένη από το ίσον και την τιμή της παραμέτρου για την προσθήκη περισσότερων παραμέτρων στο URL γίνεται χρήση του συμβόλου &. Παράδειγμα μιας διεύθυνσης με Query Strings είναι:

Β΄.12.3 Head

Αυτό το είδος αιτήματος είναι παρόμοιο με αυτό του GET αλλά χωρίς το σώμα επιστροφής του πακέτου π.χ σε περίπτωση που στείλουμε ένα αίτημα στον διακομιστή σε μια διεύθυνση που περιέχει κάποια εικόνα δεν θα επιστραφεί στον χρήστη η εικόνα. Μια ενδεικτική χρησιμότητα είναι ο υπολογισμός στατιστικών π.χ από ένα σύνολο URLS τα οποία περιέχουν πόρους να υπολογιστεί ο μέσος όρος των μεγεθών των αρχείων αποφεύγοντας έτσι το κατέβασμα των πόρων αυτών τοπικά.

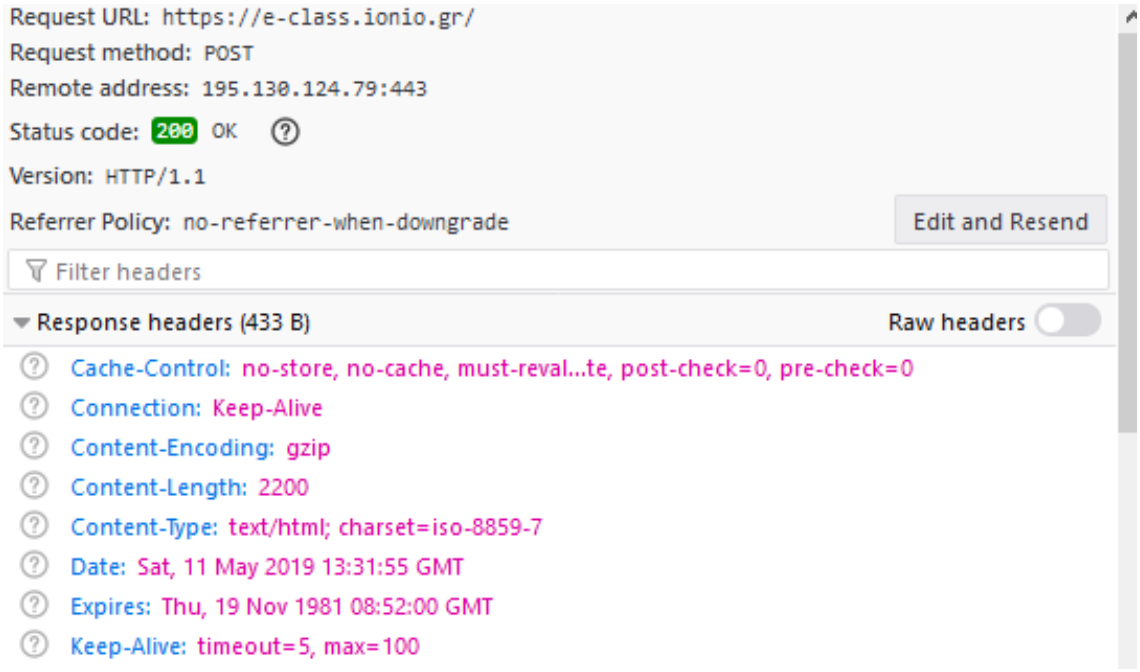
Β΄.12.4 Post

Σε αυτό το είδος του αιτήματος γίνεται εισαγωγή δεδομένων στον server δηλαδή αλλάζουμε την κατάσταση του. Για παράδειγμα στην δημιουργία ενός λογαριασμού ενός

χρήστη σε μια ιστοσελίδα. Αξιοσημείωτο είναι ότι οι πληροφορίες αυτές ενσωματόνονται μέσα στο Http header προσφέροντας έτσι μεγαλύτερη ασφάλεια.

Β΄.13 Http Headers

Τα HTTP headers επιτρέπουν σε μια εφαρμογή πελάτη και εξυπηρετητή να περνούν επιπρόσθετες πληροφορίες με ένα αίτημα η με μια απάντηση. Μία http επικεφαλίδα περιλαμβάνει ονόματα τα οποία είναι case insensitive και ακολουθούνται με άνω και κάτω τέλια και μετά την τιμή του.



Σχήμα Β΄.4: Παράδειγμα Http Headers eclass ionio

Κεφάλαιο Γ΄

Πείραμα με κείμενα

Γ΄.1 Συνοπτική περιγραφή της μεθοδολογίας

Αρχικά έγινε συλλογή και προ επεξεργασία των δεδομένων, στην συνέχεια υλοποιήθηκε ο αλγόριθμος LDA και έγινε σύγκριση του με μια υλοποίηση για την οποία γνωρίζουμε πώς δουλεύε και είναι η βιβλιοθήκη Gensim [1]. Το μέτρο που χρησιμοποιήθηκε για την σύγκριση των δύο μεθόδων LDA είναι το f1-score το οποίο βρίσκεται αρκετά κοντά και για τις δύο υλοποιήσεις. Παρακάτω παρατίθεται πίνακας με τα αποτελέσματα και για τις δύο μεθόδους. Το πείραμα εκτελέστηκε με εικοσιπέντε seeds και σε κάθε εκτέλεση χρησιμοποιήθηκε το μέτρο k-fold cross validation για κ=10. Χρησιμοποιήθηκε το προαναφερθέν νούμερο καθώς είναι αυτό που χρησιμοποιείται περισσότερο [30].

Γ΄.2 Συλλογή και Προεπεξεργασία των δεδομένων

Για την εκτέλεση του αλγορίθμου του LDA από τις πρώτες ενέργειες είναι η δημιουργία της συλλογής των εγγράφων και στην συνέχεια η προεπεξεργασία τους. Παρακάτω παρατίθενται οι πίνακες των βάσεων δεδομένων που χρειαστήκανε για την εκτέλεση του πειράματος με χρήση της μεθόδου Latent Dirichlet Allocation.

Γ΄.2.1 Πίνακας Items

Ο πίνακας αυτός περιέχει το κείμενο το οποίο εξηχθη από τα RSS-feeds και αποτέλεσε την συλλογή κειμένων που χρησιμοποιήθηκε για την εκπαίδευση του αλγορίθμου του Latent Dirichlet Allocation. Το σχήμα της βάσης που χρησιμοποιήθηκε παρατίθεται στο σχήμα Γ΄.1. Παρόλα αυτά οι σημαντικότερες στήλες οι οποίες αξίζει να αναφερθούν αναφέρονται στον πίνακα Γ΄.1.

Στήλη	Περιγραφή
ItemId	Αναγνωριστικό του κειμένου μας το οποίο δημιουργήθηκε μοναδικά από την ιστοσελίδα που προέρχεται το κείμενο και τον τίτλο του.
itemFeed	Κλειδί στον πίνακα feeds των RSS και υποδηλώνει από που έγινε εξαγωγή των δεδομένων
ItemCategory	θεματολογία του κειμένου
ItemDescription	Το κείμενο που θα δοθεί στον αλγόριθμο του LDA
ItemTitle	Ο τίτλος του κειμένου.

Πίνακας Γ΄.1: Σημαντικότερες στήλες για τον πίνακα items

Γ΄.2.2 Προβλήματα Κωδικοποίησης

Μετά το στάδιο της συλλογής των κειμένων διαπιστώθηκε πως ένας μεγάλος όγκος από τα κείμενα δεν μπορεί να διαβαστεί. Στην συνέχεια, έγινε παραδοχή πως το πρόβλημα βρίσκεται στην κωδικοποίηση και έγινε ανάπτυξη script το οποίο αναλαμβάνει να επιλύσει το ζήτημα αυτό. Επιπρόσθετα μερικά από τα κείμενα περιβαλλόντουσαν από html ετικέτες. Η μεθοδολογία που ακολουθήθηκε είναι αυτής της «ωμής βίας». Η τεχνική της «ωμής βίας» δοκιμάζει πολλούς συνδυασμούς κωδικοποίησης(encoding), ενώ για την αποκωδικοποίηση εφαρμόζεται ο αποκωδικοποιήτης(decoder) σε utf-8. Οι κωδικοποιήσεις που χρησιμοποιήθηκαν παρουσιάζονται στον πίνακα ?.2. Μετέπειτα κάθε αποτέλεσμα εξετάστηκε για να διαπιστωθεί ποιά κωδικοποίηση πρέπει να δοκιμαστεί στο κείμενο προκειμένου να είναι κατανοητό. Οι επιλογές των κωδικοποιητών(encoders)) έγιναν με στόχο να υποστηρίζουν Ελληνική γλώσσα. Από τα πειράματα που διεξήχθησαν διαπιστώθηκε πως τα καλύτερα αποτελέσματα επιστράφηκαν με

την χρήση του codec cp1252 γνωστό ως windows-1252. Παρόλα αυτά, τα ροβλήματα με την κωδικοποίηση δεν επιλύθηκαν αφού σε ορισμένες λέξεις έλλειπαν χαρακτήρες. Μεταγενέστερα, γινε χρήση της ιστοσελίδας app_encodings[3] σε μία προσπάθεια να βρεθεί μια διαδικτυακή υπηρεσία που μπορεί να επιλύσει το πρόβλημα. Αυτή η διαδικτυακή υπηρεσία έδωσε την δυνατότητα εκτέλεσης του ίδιου πειράματος με τις ίδιες κωδικοποιήσεις και έτσι δοκιμάστηκε η κωδικοποίηση σε cp1252 και αποκωδικοποίηση σε utf-8 η οποία είχε πετύχει τα καλύτερα αποτελέσματα. Το ενδιαφέρον είναι ότι στο αποτέλεσμα δεν υπήρχε πρόβλημα κωδικοποίησης, σε αντίθεση με την ίδια υλοποίηση στην Python. Άρα ο τρόπος με τον οποίο πραγματοποιήθηκε η κωδικοποίηση- αποκωδικοποίηση δεν λαμβάνει υπόψη προβλήματα με το mojibake[2]. Άρα ο τρόπος με τον οποίο υλοποιήθηκε η κωδικοποίηση- αποδικοποίηση δεν λαμβάνει υπόψη προβλήματα με το mojibake[2]. Για την επίλυση του προβλήματος μια λύση αποτελεί η δημιουργία ενός αυτοματοποημένου προγράμματος το οποίο κάνει αιτήματα στην ιστοσελίδα και λαμβάνει τα αποτελέσματα. Ωστόσο η λύση που προαναφέρθηκε αποφεύχθηκε για τους εξής λόγους: Πρώτον, Η επιστροφή των αποτελεσμάτων θα είναι πιο αργή λόγω της συμφόρησης(overhead) π.χ εγκαθίδρυση σύνδεσης με τον διακομιστή και πολλά αιτήματα τύπου post. Ακόμα υπάρχει το πρόβλημα της πιθανής μη διαθεσιμότητας της διαδικτυακής υπηρεσίας. Η λύση που τελικά επιλέχθηκε είναι η επιλογή κάποιας βιβλιοθήκης η οποία μπορεί να επιλύσει τα ζητήματα με το mojibake[2] σε συνδυασμό, με την κωδικοποίηση σε cp1252 και αποκωδικοποίηση σε utf8. Στην συνέχεια, έγινε αφαίρεση των ετικετών από το κείμενο καθώς και μετατροπή των html οντοτήτων στους χαρακτήρες που τις αντιπροσωπεύουν.

Γ΄.2.3 Ψευδοκώδικας Επίλυσης Προβλήματος Κωδικοποίησης

Είσοδος: documents: Τα έγγραφα τα οποία θέλουμε να επιδιορθώσουμε την κωδικοποίηση τους
documents_ids: Τα αναγνωριστικά των εγγράφων Η μέθοδος αυτή επιδιορθώνει ζητήματα με τα κείμενα και γίνεται ενημέρωση του πίνακα items.

```
def Fix_items_enctext(doc_ids, documents)
        Len_docs = len(documents)
        for i in range(0, len_docs)
```

```
New_doc_text = Documents[i]
New_doc_text = html_parse(New_doc_text).get_te()
New_doc_text =  ftfy_fix_encoding(New_doc_text)
New_doc_text = ftfy_fix_text(New_doc_text)
Update_table_items(doc_ids[i], New_doc_text)
```

Γ΄.2.4 Περιγραφή του ψευδοκώδικα Επίλυσης Κωδικοποίησης

Αρχικά βρίσκεται το μέγεθος του πίνακα με τα έγγραφα με την συνάρτηση len. Στην συνέχεια για κάθε έγγραφο δίνεται ως είσοδο στην συνάρτηση html_get_text η οποία αναλαμβάνει να εξάγει τις html ετικέτες σε περίπτωση που υπάρχουν. Στις επόμενες γραμμές γίνονται κλήσεις στις συναρτήσεις της ftfy για την επιδιόρθωση προβλημάτων με την κωδικοποίηση και των html entities. Τέλος τα αποτελέσματα αυτά πρέπει να αποθηκευτούν στον πίνακα με τα items της βάσης δεδομένων και εδώ είναι που χρειάζεται το document_id προκειμένου να ενημερωθεί η κατάλληλη εγγραφή της βάσης.

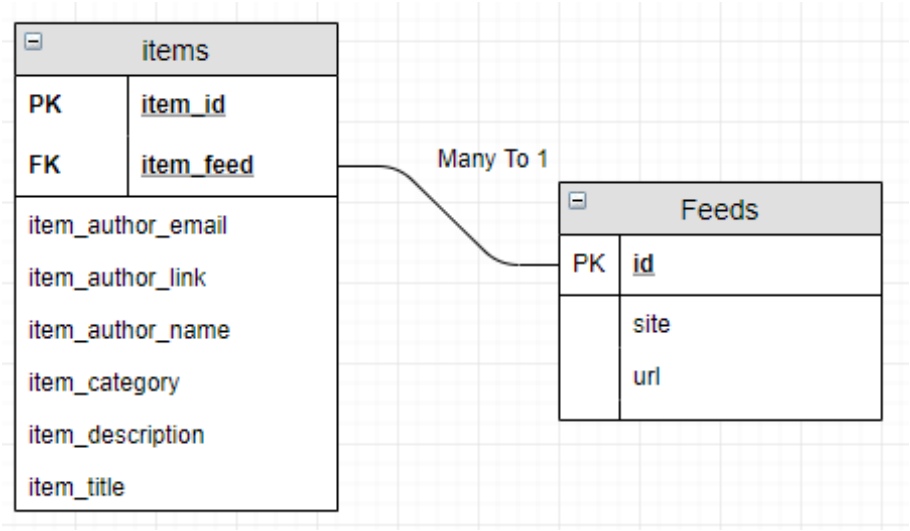
Γ΄.2.5 Πίνακας Feeds

Ο πίνακας των Feeds περιέχει πληροφορίες από διάφορες πηγές δεδομένων (Rss Feeds). Οι οποίες χρησιμοποιήθηκαν ως μέσο προσθήκης περισσότερων εγγράφων στον πίνακα Items τα οποία θα χρησιμοποιηθούν για την εκτέλεση του αλγορίθμου Latent Diriclet Allocation. Για την επίτευξή της λειτουργικότητας αυτής υλοποιήθηκε μια συνάρτηση η οποία λαμβάνοντας ως είσοδο πολλά Urls τα οποία δείχνουν σε Rss-Feeds, αναλαμβάνει να εξάγει τις κατάλληλες πληροφορίες από αυτά. Αξιοσημείωτο είναι ότι κάθε φεεδ αποτελείται απο πολλά άρθρα. Οι πληροφορίες που αποθηκεύονται για κάθε άρθρο είναι ο τίτλος του, το περιεχόμενο του και το μοναδικό αναγνωριστικό του. Εντούτοις η μέθοδος αυτή λαμβάνει υπόψη της και προβληματικές περιπτώσεις. Αρχικά σιγουρεύεται ότι η διεύθυνση Url που δίνεται είναι διαθέσιμη, επίσης όταν ζητήσει το αρχείο με τα Rss-Feeds, γίνονται κατάλληλοι έλεγχοι με head http αιτήματα για να διαπιστωθεί αν ο αιτούμενος πόρος είναι xmlη xml-rss. Μετά βεβαιώνεται ότι το αρχείο αυτό περιέχει όλες τις πληροφορίες που αναφέρθηκαν προηγουμένως σχετικά με το Rss-Feed και σε περίπτωση αδυναμίας συγκέντρωσης έστω και μίας από αυτές τότε το κείμενο απορρίπτεται. Τέλος γίνεται αποθήκευση των αποτελεσμάτων στον πίνακα των

Codec	Aliases	Languages
cp500	EBCDIC-CP-BE, EBCDIC-CP-CH, IBM500	Western Europe
cp737		Greek
cp850	850, IBM850	Western Europe
cp858		Western Europe
cp869		Greek
cp875		Greek
cp1140		Western Europe
cp1252		Western Europe
cp1253		Greek
iso8859_7		Greek
iso8859_15		Greek
mac_greek		Greek
utf_32		Western Europe
mac_roman		Western Europe
mac_roman		Western Europe
utf_32		all languages
utf_32_be		all languages
utf_32_le		all languages
utf_16		all languages
utf_16_be		all languages
utf_16_le		all languages
utf_7		all languages
utf_8		all languages
utf_8_sig		all languages

Table C.2: Πίνακες με κωδικοποιήσεις που χρησιμοποιήθηκαν

items. Στην συνέχεια παρατίθεται ψευδοκώδικας ο οποίος δείχνει την λειτουργικότητα που αναφέρθηκε.



Γ΄.2.6 Ψευδοκώδικας RSS-Feeds

Είσοδος rss_urls : Λίστα με rss_feeds.
Η συνάρτηση ινσερτ ρος επισκέπτεται τα rss_urls, εξάγει τα έγγραφα και για κάθε έγγραφο αντλεί το μοναδικό αναγνωριστικό του, το περιεχόμενο του καθώς και τον τίτλο του.

```
def update_items_rss(rss_urls, rss_ids)
        Rss_len = len(rss_urls)
        for i in range(Rss_len)
                url = rss_urls[i]
                if not is_xml(url)
                        continue
                data = http_get(url).content
                entries = rss_parse(data)
                for each entry in entries
                        id = entry["id"]
                        title = entry["title"]
```

```
            text = entry["summary"]
        insert_items_table(id, rss_ids[i], title, text)
```

Γ΄.2.7 Αναλυτική Περιγραφή Ψευδοκώδικα RSS

Για κάθε διαθέσιμο url στα Rss-urls εκτελούνται οι ακόλουθες λειτουργείες. Αρχικά, γίνεται έλεγχος για να διαπιστωθεί άμα το url είναι xml αρχείο. Σε περίπτωση που το αρχείο δεν είναι XML τότε μεταβαίνουμε στο επόμενο Url. Στην αντίθετη περίπτωση, γίνεται κλήση στην http_get(url).content για την εξαγωγή των περιεχόμενων του rss αυτού αρχείου, στην συνέχεια αυτό δίνεται ως είσοδο στην συνάρτηση rss_parse η οποία αναλαμβάνει να επιστρέψει έναν πίνακα με τις εγγραφές που έχει μέσα το αρχείο αυτό. Τέλος για κάθε μία από αυτά τα αντικείμενα γίνεται η εξαγωγή του id, title του κειμένου και εισάγονται στον πίνακα με τα items.

Γ΄.3 Προεπεξεργασία κειμένου

Αφού έγινε συλλογή και διόρθωση της κωδικοποίησης των δεδομένων πραγματοποιήθηκε η προ-επεξεργασία τουCorpus με τα κείμενα. Η μεθοδολογία που ακολουθήθηκε είναι η εξής: Αρχικά το κάθε έγγραφο χωρίστηκε σε tokens ενώ, κρατήθηκαν τα κείμενα με αριθμό λέξεων μεγαλύτερο από έναν αριθμό. Ενδεικτικά στην πτυχιακή αυτή χρησιμοποιήθηκε αυθαίρετα ως όριο το 150 αφού δόθηκε μεγαλύτερη βάση στην υλοποίηση του LDA. Στην συνέχεια έγινε αφαίρεση των stopwords. Τα stopwords είναι λέξεις οι οποίες εμφανίζονται σε πολλά κείμενα και αφαιρούνται καθώς δεν προσφέρουν κάποια πληροφορία. Όσον αφορά την υλοποίηση χρησιμοποιήθηκαν δύο λίστες για stopwords η πρώτη για την Ελληνική γλώσσα, ενώ η δεύτερη για την Αγγλική γλώσσα. Επίσης έγινε αφαίρεση αριθμών, ημερομηνιών, ιστοσελίδων(Urls). Μετά χρησιμοποιήθηκε ο StemmerManager που αναπτύχθηκε ο οποίος αναλαμβάνει να κάνει stemming τις ελληνικές καθώς και τις Αγγλικές λέξεις. Τέλος αξίζει να αναφερθεί πως ο Stemmer απαιτεί οι λέξεις να είναι αποτυπωμένες με κεφαλαίους χαρακτήρες. Στην συνέχεια γίνεται εξήγηση των συστατικών στοιχείων που χρησιμοποιήθηκαν στο στάδιο της προεπεξεργασίας.

Χωρίς τόνο	Με τόνο
Α	ἁ, ἀ, ἄ, ἀ, ᾶ, ᾷ, ἁ, ἆ, ᾶ, ἅ
Ε	ε, ὲ, ἑ, ἕ, ἕ, ἑ, ἐ
Η	ἡ, ἡ, ἥ, ῆ, ἠ, ἤ, ῇ, ἦ, ἠ, ἦ, ῃ, ἡ, ἢ
Ι	ἰ, ὶ, ῖ, ῖ, ἱ, ἰ, ἴ, ἴ, ἶ, ϊ, ἶ, ἶ
Ο	ο, ὁ, ὸ, ὅ, ὅ, ὁ, ὀ
Υ	υ, ὺ, ὑ, ὐ, ῦ, ὖ, ὔ, ὑ, ὕ, ϋ
Ω	ω, ὡ, ὥ, ῶ, ῷ, ὧ, ᾠ, ὡ, ὢ, ὠ, ὤ, ὦ, ὼ
Ρ	ῥ

Σχήμα Γ΄.1: Το Dictionary στο οποίο όταν στο κείμενο βρεθεί χαρακτήρας ο οποίος βρίκσεται στα αριστερά του πίνακα αντιστοιχίζεται στον αντίστοιχο χαρακτήρα του δεξιά

Γ΄.3.1 Μετατροπή πεζών σε κεφαλαίους χαρακτήρες

Η σημασία της παρακάτω ενέργειας ενδείκνυται στο γεγονός πως εργαλεία όπως ο Greek Stemmer απαιτούν οι χαρακτήρες που θα δωθούν ως είσοδο να είναι όλοι κεφαλαίοι. Επίσης η ενιαία αντιμετώπιση των χαρακτήρων ως κεφαλαία η μικρά γίνεται καθώς μια λέξη μπορεί να γραφεί με διαφορετικούς συνδυασμούς με κεφαλαία και μικρά γράμματα. Αφού ο υπολογιστής δεν έχει την ικανότητα να καταλάβει αν μία λέξη γραμμένη με κεφαλαίους χαρακτήρες(πχ ΜΠΙΣΚΟΤΟ) είναι η ίδια με την αντίστοιχη με μικρά γράμματα.(μπισκότο). Η μετατροπή λέξεων στις αντίστοιχες κεφαλαίες τους γίνεται από την γλώσσα Python μέχρι ένα σημείο. Παρόλα αυτά το πρόβλημα που εμφανίζεται είναι ότι χαρακτήρες με τόνο συνεχίζουν να έχουν τόνο και μετά την μετατροπή κάτι το οποίο δεν είναι επιθυμητό. Για αυτό δημιουργήθηκε μια αντιστοίχιση δηλαδή ένα Dictionary η αλλιώς Map. Στην δομή αυτή ορίζεται ένα κλειδί και μια τιμή για παράδειγμα ως κλειδί ορίζεται ο χαρακτήρας έ και ως τιμή του ο χαρακτήρας Ε.

Γ΄.3.2 Ψευδοκώδικας συνάρτησης μετατροπής μιας λέξης σε κεφαλαία γράμματα

Είσοδος: λέξη την οποία θα θέλαμε να μετατρέψουμε σε κεφαλαία
Η μέθοδος αυτή μετατρέπε το κείμενο σε κεφαλαία γράμματα.

```
def GreekStrCapital(string text)
        New_str = ""
        For each word in text
                Normalized_char = upper(get_char_without_suffixes
                (word))
                New_str += Normalized_char
        return New_str
```

Γ΄.3.3 Αναλυτική Περιγραφή Μεθόδου Μετατροπής λέξεων από μικρά σε κεφαλαία

Αρχικά στην πρώτη γραμμή γίνεται αρχικοποίηση της μεταβλητής new_str στην οποία αποθηκέυεται το τελικό αποτέλεσμα. Στην συνέχεια για κάθε χαρακτήρα που εξάγεται από την λέξη δίνεται ως είσοδος στην συνάρτηση get_char_without_suffixes(word) η οποία όπως μπορούμε να διαπιστώσουμε παίρνει ως είσοδο έναν χαρακτήρα και επιστρέφει τον ίδιο χαρακτήρα χωρίς τόνους. Σε περίπτωση που ο χαρακτήρας αυτός δεν έχει τόνο τότε θα επιστραφεί ο ίδιος ο χαρακτήρας χωρίς να γίνει κάποια αλλαγή πάνω του. Ο κανονικοποιήμενος αυτός χαρακτήρας δίνεται ως παράμετρος στην συνάρτηση upper η οποία αναλαμβάνει να τον μετατρέψει σε κεφαλαίο. Στην επόμενη γραμμή προστίθεται στην μεταβλητή New_str. Τέλος γίνεται επιστροφή του κεφαλαίου κειμένου απαλλαγμένο από τόνους.

Γ΄.3.4 Αφαίρεση Stopwords

Στο πείραμα με τα κείμενα, για την εξαγωγή τον stop words αναπτύχθηκε ένας stop Word Manager ο οποίος χτίζει μια μεγάλη λίστα από λέξεις. Ο Manager αυτός διαβάζει τις λέξεις από αρχεία τα οποία εντοπίζονται σε φάκελο πού ορίζεται απο τον χρήστη. Η λειτουργεία αποδείχτηκε χρήσιμη καθώς, η λίστα με τις Stop Words για το set της

Νεοελληνική γλώσσα δεν είχε παραχθεί από κάποια βιβλιοθήκη υψηλού επιπέδου όπως η NLTK[4]. Ενώ τα Stop Words για την Αγγλική γλώσσα εξήχθησαν από την NLTK.

Γ΄.3.5 Ψευδοκώδικας StopWord Manager

Είναι το τμήμα της προεπεξεργασίας που χρησιμοποιήθηκε για την αφαιρέση των stopwords. Ο Stop Word Manager περιέχει δύο συναρτήσεις μια για την δημιουργία του set με τα stop words και μία άλλη η οποία λαμβάνει ως είσοδο μία λέξη και πραγματοποιεί αναζήτηση της για να δούμε αν υπάρχει μέσα στο set σε περίπτωση που υπάρχει τότε η λέξη αυτή μπορεί να αφαιρεθεί καθώς θεωρείται stopword.

```
def create_stopwords_set()
        sw_set = nltk_stopwords_enligh()
        for each file in get_stopword_files()
                for each line in file
                        for each word in line
                                sw_set.add(word)
```

Η συνάρτηση create_stopwords_set αρχικοποιείται με τα στοπωορδς της Αγγλικής γλώσσας. Στην συνέχεια, διαβάζει από τον φάκελο που ορίσαμε όπου βρίσκονται τα αρχεία των stopwords. Για κάθε ένα από αυτά τα αρχεία που βρέθηκαν, διαβάζεται το περιεχόμενο τους κάθε γραμμή του αρχείου σπάει σε λέξεις οι οποίες αποθηκεύονται στο set. Η χρήση και του set έγινε προκειμένου να αποφευχθούν διπλότυπες λέξεις γλυτώνοντας έτσι χώρο στην μνήμη, επίσης τα sets μπορούν να βρουν πολύ γρήγορα στοιχεία, σε αντίθεση με μία λίστα όπου η αναζήτηση ενός στοιχείου μέσα σε αυτήν γίνεται ελέγχοντας κάθε στοιχείο της μέχρι να βρεθεί το επιθυμητό.

```
def isStop_Word(word):
        return sw_set.exists(word)
```

Είσοδος: Word: λέξη για να ελεγχθεί άμα είναι stopword.
Η συνάρτηση isStop_Word παίρνει μια λέξη ως είσοδο και επιστρέφει το αποτέλεσμα της συνάρτησης exists. Συγκεκριμένα η εξιστ ειναι συνάρτηση μέλος του αντικείμενου set όπου πραγματοποιείται αναζήτηση για μια λέξη και επιστρέφει True σε περίπτωση που έχει βρεθεί και False σε αντίθετη περίπτωση.

Είσοδος:Stopword_path: μονοπάτι που βρίσκονται τα stopwords.
Η συνάρτηση get_stopword_files παίρνει ως είσοδο το μονοπάτι στο οποίο αναζητάει stopwords αρχεία και επιστρέφει τα ονόματά τους σε μία λίστα για την άμεση προσπέλασή τους. Πιο συγκεκριμένα, στην πρώτη γραμμή αρχικοποιείται μια λίστα στην οποία θα αποθηκευτούν τα ονόματα των αρχείων. Στην δεύτερη γραμμή καλείται η συνάρτηση listfiles η οποία παίρνει ως παράμετρο το Stopword_Path και επιστρέφει μια λίστα με όλα τα αρχεία τα οποία υπάρχουν στο path. Στην Τρίτη γραμμή στο for loop γίνεται προσπέλαση κάθε αρχείου και άμα είναι αρχείο κειμένου το πρόσθετει στην λίστα. Τέλος μόλις προσπεραστούν όλα τα αρχεία τότε γίνεται επιστροφή όλων των αρχείων του κειμένου.

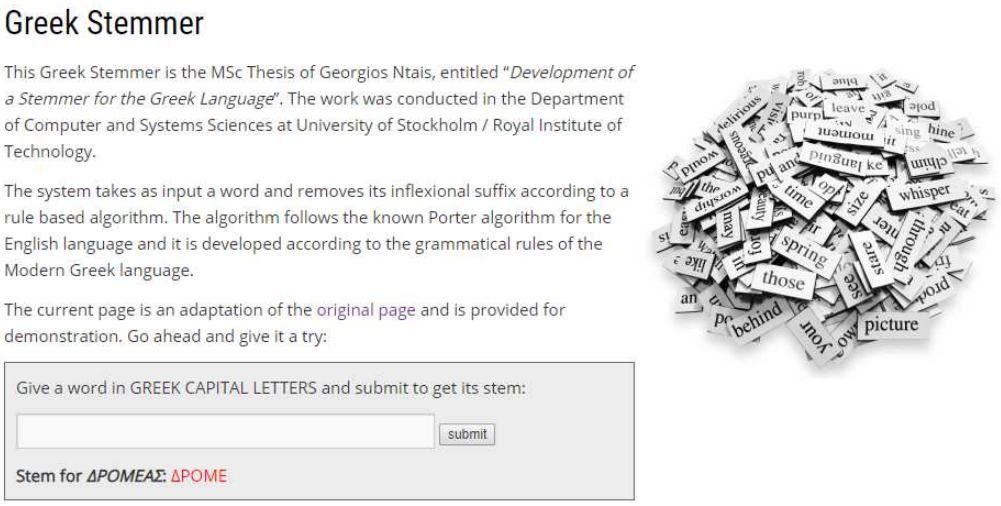
Γ΄.3.6 Stemming-Manager

Όπως και για την διαχείριση stop words έτσι και για το stemming αναπτύχθηκε ο Stemming Manager για την διαχείριση της διαδικασίας του Stemming για την Ελληνική και Αγγλική γλώσσα. Στα πλαίσια της εργασίας αυτής όπως και για τα stopwords χρησιμοποιήθηκε ένας stemmer για την Αγγλική γλώσσα από το NLTK[4]. Ενώ για την Ελληνική γλώσσα χρησιμοποιήθηκε μια διαδικτυακή υπηρεσία η οποία αποτελεί μια υλοποίηση σε php του αλγόριθμου stemmer που βασίζεται στην εργασία του George Ntais ο οποίος τροποποίησε τον αλγόριθμο του Porter προκειμένου να δουλέψει για την Ελληνική γλώσσα [5]. Ό λόγος που χρησιμοποιήθηκε διαδικτυακή εφαρμογή έιναι γιατί δεν βρέθηκε κάποια βιβλιοθήκη στην Python που να δουλέψει με επιτυχία. Επίσης, η συγγραφή κώδικα για την εξαγωγή των αποτελεσμάτων αποδείχτηκε σχετικά εύκολη διαδικασία.

Γ΄.3.7 Περιγραφή διαδικτυακής πλατφόρμας stemming για την Ελληνική Γλώσσα

Ο τρόπος χρήσης της διαδικτυακής εφαρμογής είναι ο εξής:Υπάρχει μια φόρμα η οποία δέχεται μια λέξη στα κεφαλαία και επιστρέφει μια νέα λέξη στα κεφαλαία, όπου έχει εφαρμοστεί η λειτουργία του stemming. Για να γίνει stemm πολλαπλών λέξεων έχει υλοποιηθεί μια διαδικτυακή υπηρεσία η οποία δέχεται Get Http requests και επιστρέφει τα αποτελέσματα της μετατροπής αξιοσημείωτο είναι ότι κάθε λέξη εκτός από το ότι

πρέπει να βρίσκεται στα κεφαλαία πρέπει και να είναι και χωρισμένη με κόμμα.



Σχήμα Γ΄.2: Greek Stemmer Διαδικτυακή εφαρμογή

Για να γίνει χρήση της διαδικτυακής υπηρεσίας πρέπει να δημιουργηθεί url με την εξής μορφή:

http://deixto.com/gr-stemmer.php?clientID=ClientID&**input**=Data

. Όπου clientId είναι ένα αναγνωριστικό αντίστοιχο με το APIKey το οποίο μπορεί να ποκτηθεί με επικοινωνία με τον διαχειριστή της εφαρμογής, το προκαθορισμένο κλειδί είναι το deixto το οποίο επιτρέπει την χρήση της εφαρμογής. Ωστόστο υπάρχει δώδεκα δευτερόλεπτα καθυστέρησης για κάθε αίτημα που γίνεται. Σε περίπτωση απόκτησης ClientId δεν υπάρχει ο περιορισμός αυτός αλλά για πολλά αιτήματα στον διακομιστή εισάγεται ένα μικρότερο είδος καθυστέρησης το οποίο πρέπει να είναι τουλάχιστον 4-5 δευτερόλεπτα. Η εφαρμογή αυτή επιστρέφει στεμμεδ λέξεις οι οποίες είναι κεφαλαίες και χωρισμένες με κόμμα. Για παράδειγμα έστω ότι δοθεί ως είσοδος στην εφαρμογή οι λέξεις ΔΡΟΜΕΑΣ, ΝΙΚΗΤΕΣ, ΓΡΗΓΟΡΟΣ. Ως έξοδος η υπηρεσία επιστρέφει το αποτέλεσμα:ΔΡΟΜΕ, ΝΙΚΗΤ, ΓΡΗΓΟΡ.

Γ΄.3.8 Προβλήματα διαδιακτυακής εφαρμογής Greek-Stemmer

Ένα ζήτημα που έπρεπε να επιδιορθωθεί είναι ότι το σύνολο των λέξεων που δίνεται ως είσοδος δεν πρέπει α ξεπερνάει το κατώφλι των 100 λέξεων. Σε περίπτωση που

ξεπεραστεί αυτό το όριο τότε η εφαρμογή τερματίζει αναπάντεχα την σύνδεση με τον διακομιστή. Έτσι για τον σκοπό αυτόν δημιουργήθηκε κώδικας ο οποίος αναλαμβάνει να σπάσει το κείμενο μας σε κομμάτια των εκατό λέξεων που είναι το όριο και στην συνέχεια καλεί την διαδικτυακή αυτήν υπηρεσία με κάθε ένα κομμάτι την φορά π.χ άμα ένα κείμενο αποτελείται απο εκατόδύο λέξεις και γνωρίζει ότι το όριο είναι εκατό λέξεις τότε θα σπάσει το κείμενο σε μία ομάδα των εκατό και μια ομάδα με δύο λέξεις. Ακόμα ένα σημαντικό ζήτημα το οποίο έπρεπε να αντιμετωπιστεί είναι ότι η είσοδος για τον Stemmer πρέπει να είναι σε κεφαλαία ελληνικά ώστε να είναι λειτουργική.

Γ΄.3.9 Ψευδοκώδικας Stemmer

Παρακάτω παρατίθεται ο κώδικας που χρησιμοποιήθηκε για το Stemming της Ελληνικής και της Αγγλικής γλώσσας με την βοήθεια εξατομικευμένων συναρτήσεων. Στην συνέχεια παρατίθεται ο ψευδοκώδικας του stemmer και γίνεται ανάλυση του κώδικα των συστατικών του στοιχείων.

Γ΄.3.10 Μέθοδος StemmEnglish

Η συνάρτηση αυτήν παίρνει ένα κείμενο του οποίου οι λέξεις είναι χωρισμένες μεταξύ τους με τον χαρακτήρα του διαστήματος και επιστρέφει το αντίστοιχο κείμενο της Αγγλικής γλώσσα στο οποίο έχει εφαρμοστεί το Stemming.

```
def stemmEnglish(text)
        Stemmed_words = list()
        words = split(text)
        for each word in words
                stemmed_text = porterStemmer_stem(word)
                stemmed_words.append(stemmed_text)
        return Stemmed_words
```

Γ΄.3.11 Περιγραφή μεθόδου Stemm-English

Αρχικά αναφορικά με τον ψευδοκώδικα στην πρώτη γραμμή δημιουργείται μια κενή λίστα στην οποία γίνεται εισαγωγή των λέξεων οι οποίες υφιστάνται stemming. Στην

δεύτερη γραμμή στην λίστα words γίνεται εισαγωγή όλων των λέξεων οι οποίες είναι χωρισμένες με το σπαςε μεταξύ τους. Κάθε λέξη που υπάρχει στην λίστα με τις λέξεις δίνεται ως είσοδος στον PorterStemmer και το αποτέλεσμα εισάγεται στην λίστα με τις Stemmed λέξεις αυτή η διαδικασία σταματάει όταν επισκεφτούν όλες οι λέξεις. Τέλος επιστρέφεται η λίστα με τις stemmed λέξεις.

Γ΄.3.12 Μέθοδος Stemming-Greek

Η συνάρτηση αυτήν παίρνει ένα κείμενο του οποίου οι λέξεις είναι χωρισμένες μεταξύ τους με τον χαρακτήρα του διαστήματος και επιστρέφει το κείμενο της Ελληνικής γλώσσας στο οποίο έχει εφαρμοστεί το Stemming.

```
def Stemming_Greek(clientid, text, chunksize)
        bstr, bsize = BucketStrings(text, chunksize)
        for i in range(0, bsize)
        out += requests_get(?http://deixto.com/gr-stemmer.php?
        ClientId=%s&input=%s?, clientid, data)
        return out
```

Γ΄.3.13 Περιγραφή Μεθόδου Stemming-Greek

Στην πρώτη γραμμή καλείται η συνάρτηση BucketStrings η οποία χωρίζει το κείμενο σε κομμάτια λέξεων προκειμένου να αποφευχθεί ο τερματισμός της σύνδεσης με τον διακομιστή, επίσης η συνάρτηση αυτή επιστρέφει και τις πόσες ομάδες(Buckets) έχουν δημιουργηθεί. Στην συνέχεια για κάθε ομάδα από chunks που έχει κατασκευαστεί δημιουργείται ένα Get Http Request στο οποίο έχουν περαστεί οι απαραίτητες παράμετροι.Οι οποίες όπως φένεται από την γραμμή 3 του ψευδοκώδικα είναι το ClientId και οι λέξεις οι οποίες είναι κεφαλαίες και χωρισμένες με κόμμα.Το αποτέλεσμα που επιστρέφεται είναι λέξεις με κεφαλαία γράμματα χωρισμένες σε κόμματα όπου σε κάθε λέξη έχει εφαρμοστεί η λειτουργία του στεμμινγ της Ελληνικής γλώσσας.

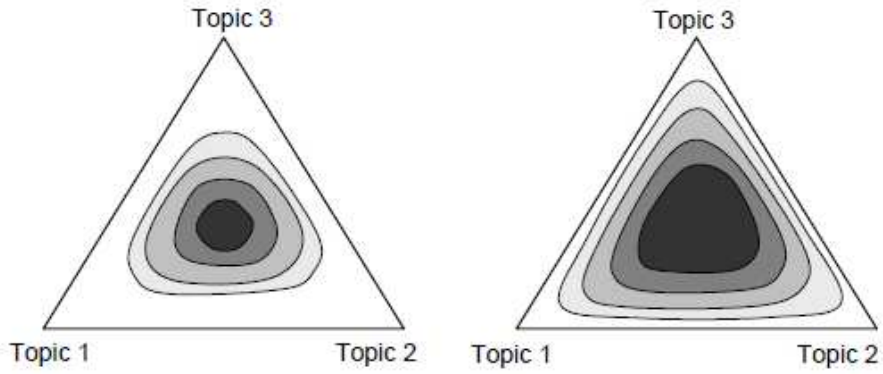
.

Γ΄.4 Αλγόριθμος Latent Dirichlet Allocation

Το LDA είνα topic modeling αλγόριθμος που χρησιμοποιείται για την ανακάλυψη θεμάτων σε ένα Corpus. Το Corpus αποτελείται από μια μεγάλη συλλογή έγγραφων και σε κάθε έγγραφο περιέχονται πολλές λέξεις. Το LDA Είναι αλγόριθμος μη επιβλεπόμενης μάθησης καθώς δεν γνωρίζουμε τις σωστές ετικέτες. Το μοντέλο αυτό είναι bag of words δηλαδή η σειρά εμφάνισης των λέξεων σε ένα έγγραφο δεν λαμβάνεται υπόψη. Επίσης μια λέξη μπορεί να ανήκει σε πολλαπλά θέματα. Το κάθε θέμα είναι μια multinomial κατανομή πιθανότητας όπου σε καθένα υπάρχουν λέξεις που συνοδεύονται από μια πιθανότητα εμφάνισης(κατανομή φ) και για κάθε έγγραφο υπάρχει μια multinomial κατανομή πιθανότητας για κάθε θέμα π.χ ένα έγγραφο μπορεί να έχει 30% θέμα1 70% θέμα2(κατανομή ϑ) . Επίσης στο LDA υπάρχουν και τα hyperparameters α και β τα οποία είναι Dirichlet priors. Η Dirichlet κατανομή παίρνει ως είσοδο πολλές παραμέτρους η κατανομή και αυτή αποτελεί μια βολική επιλογή ως prior καθώς απλοποιεί το πρόβλημα του Statistical Inference[7] . Η παράμετρος α είναι ο αριθμός των φορών στο οποίο ένα θέμα ϑ γίνεται sample σε ένα έγγραφο χωρίς να έχουν παρατηρηθεί λέξεις από το έγγραφο, όσο πιο μεγάλη είναι η τιμή του α τόσο πιο πιθανό είναι για ένα έγγραφο να περιέχει όλα τα θέματα. Αντίστοιχα το β είναι ο αριθμός των φορών που μια λέξη γίνεται sample σε ένα θέμα χωρίς να έχει ληφθεί από κάποια παρατήρηση λέξης του corpus. Οι παράμετροι α, β προσφέρουν smoothing στις κατανομές ϑ και φ, επίσης οι τιμές α, β είναι σταθερές και συμμετρικές κατά την διάρκεια εκτέλεσης του πειράματος δηλαδή ισχύει $\alpha 1 = \alpha 2 = \ldots = \alpha T = \alpha$ και $\beta 1 = \beta 2 = \ldots = BN$ οπού T, N οι διαστάσεις τις Dirichlet κατανομής. Στο σχήμα Γ΄.3 γίνεται απεικόνιση της συμμετρικής Dirichlet κατανομής τριών διαστάσεων σε ένα δυσδιάστατο Simplex. Στο οποίο μπορεί να γίνει αναπαράσταση όλων των πιθανοτήτων για τις τρείς διαστάσεις, επίσης το άθροισμα των πιθανοτήτων αυτών πρέπει να αθροίζει στο ένα. Όσο πιο μεγάλη τιμή της συμμετρικής κατανομής Dirichlet τόσο περισσότερο γίνεται απομάκρυνση από τις κορυφές του Simplex αυτού δημιουργώντας μεγαλύτερο Smoothing.

Γ΄.4.1 Statistical Inference

Στο statistical Inference γίνεται υπολογισμός των αγνώστων παραμέτρων ζ, ϑ, φ που αναφέρθηκαν προηγουμένως. Για το LDA οι παράμετροι αυτοί μπορούν να βρεθούν από τον παρακάτω τύπο:



Σχήμα Γ΄.3: Συμμετρική κατανομή Dirichlet Αριστερά για α=4, δεξιά α=2, Το σκούρο χρώμα υποδηλώνει μεγαλύτερη πιθανότητα.

$$p(\theta, \phi, \mathbf{z}|\mathbf{w}, \alpha, \beta) = \frac{p(\theta, \phi, \mathbf{z}, \mathbf{w}|\alpha, \beta)}{p(\mathbf{w}|\alpha, \beta)}$$

Σχήμα Γ΄.4: Τύπος υπολιγισμού ζ, ϑ, φ

Το πρόβλημα είναι ότι δεν μπορούμε να υπολογίσουμε την πράξη αυτή λόγω του παρονομαστή. Για την αντιμετώπιση του προβλήματος αυτού υπάρχουν τεχνικές Inference οι οποίες μπορούν να εφαρμοστούν στο πρόβλημα όπως είναι το Variational Inference και Collapsed Gibs Sampling[7, 8]. Στην εργασία αυτή για την αντιμετώπιση του προβλήματος αυτού έγινε υλοποίηση της δεύτερης προσέγγισης, καθώς ήταν πιο εύκολη στην υλοποίηση της.

Γ΄.4.2 Collapsed Gibs Sampling

Το Collapsed Gibs Sampling είναι μια τεχνική MCMC(Markov Chain Monte Carlo)[11] η οποία εκτελείται επαναληπτικά και λαμβάνει δείγματα από απλούστερες κατανομές. Τελικά μετά από αρκετές επαναλήψεις οι τιμές που εξάγονται από την κατανομή αυτή προσεγγίσουν την επιθυμητή κατανομή. Στον αλγόριθμο του Collapsed Gibs Sampling γίνεται επαναληπτική δειγματοληψία και υπολογίζεται η υπό συνθήκη πιθανότητα για την τρέχων λέξη να ανατεθεί στο θέμα ϑ με δεδομένο τις παραμέτρους z_i, wi, di και το(.) Το z_i δείχνει όλες τις αναθέσεις για το έγγραφο εκτός από την ι και το(.) τις υπόλοιπες γνωστές παραμέτρους όπως τα hyperpameters α και β.

$$P(z_i = j \mid \mathbf{z}_{-i}, w_i, d_i, \cdot) \propto \frac{C^{WT}_{w_i j} + \beta}{\sum_{w=1}^{W} C^{WT}_{wj} + W\beta} \frac{C^{DT}_{d_i j} + \alpha}{\sum_{t=1}^{T} C^{DT}_{d_i t} + T\alpha}$$

Σχήμα Γ΄.5: Τύπος πιθανότητας λέξης να ανήκει στο θέμα j

Από τον παραπάνω τύπο το C^{WT} είναι μια μεταβλητή μετρητή με μέγεθος WXT και υποδηλώνει τον αριθμό των φορών όπου η λέξη wi ανατέθηκε στο θέμα j. Στον μετρητή αυτό δεν συμπεριλαμβάνεται η τρέχων ανάθεση της λέξης, αντιστοίχως ο μετρητής $C\ DT$ έχει μέγεθος DXT και υποδηλώνει τον αριθμό των φορών που μια λέξη του di ανατέθηκε στο θέμα j. Από την εξίσωση όσο περισσότερες φορές μια λέξη έχει ανατεθεί στο θέμα j και όσες περισσότερες φορές το έγγραφο χρησιμοποιεί το θέμα ϑ τόσο πιο πιθανό είναι η τρέχων λέξη να ανατεθεί στο ϑ. Μια επισημείωση σχετικά με τον μαθηματικό τύπο που προαναφέρθηκε είναι ότι το αποτέλεσμα είναι μια μη κανονικοποιήμενη πιθανότητα και για να πάρουμε την τιμή της πιθανότητας για την λέξη wi πρέπει στο αποτέλεσμα να γίνει διαίρεση με το άθροισμα όλων των πιθανοτήτων που προκύπτουν για όλα τα θέματα. Ο αλγόριθμος του Collapsed Gibs Sampling ξεκινάει με την αρχικοποίηση των μεταβλητών των μετρητών με τυχαίες τιμές. Μετά για κάθε λέξη και θέμα βρίσκεται η πιθανότητα να γίνει ανάθεση στο θέμα αυτό. Στην συνέχεια γίνεται αύξηση των μετρητών CWT και CDT αντίστοιχα ανάλογα με το θέμα στο οποίο έχει ανατεθεί η τρέχων λέξη. Έτσι όταν η διαδικασία αυτή εκτελεστεί για όλα τα έγγραφα τότε έχει εκτελεστεί ένα Collapsed Gibs Sample. Κάτι το οποίο αξίζει να τονιστεί είναι ότι τα πρώτα από αυτά τα δείγματα είναι καλό να απορριφθούν καθώς δεν αντικατοπτρίζουν την κατανομή για τις αναθέσεις θεμάτων σε μια λέξη(Burn in period) [7]. Μετά γίνεται υπολογισμός του φ και ϑ με τους παρακάτω τύπους.

.

$$\phi_i^{\prime(j)} = \frac{C^{WT}_{ij} + \beta}{\sum_{k=1}^{W} C^{WT}_{kj} + W\beta} \qquad \theta_j^{\prime(d)} = \frac{C^{DT}_{dj} + \alpha}{\sum_{k=1}^{T} C^{DT}_{dk} + T\alpha} \tag{4}$$

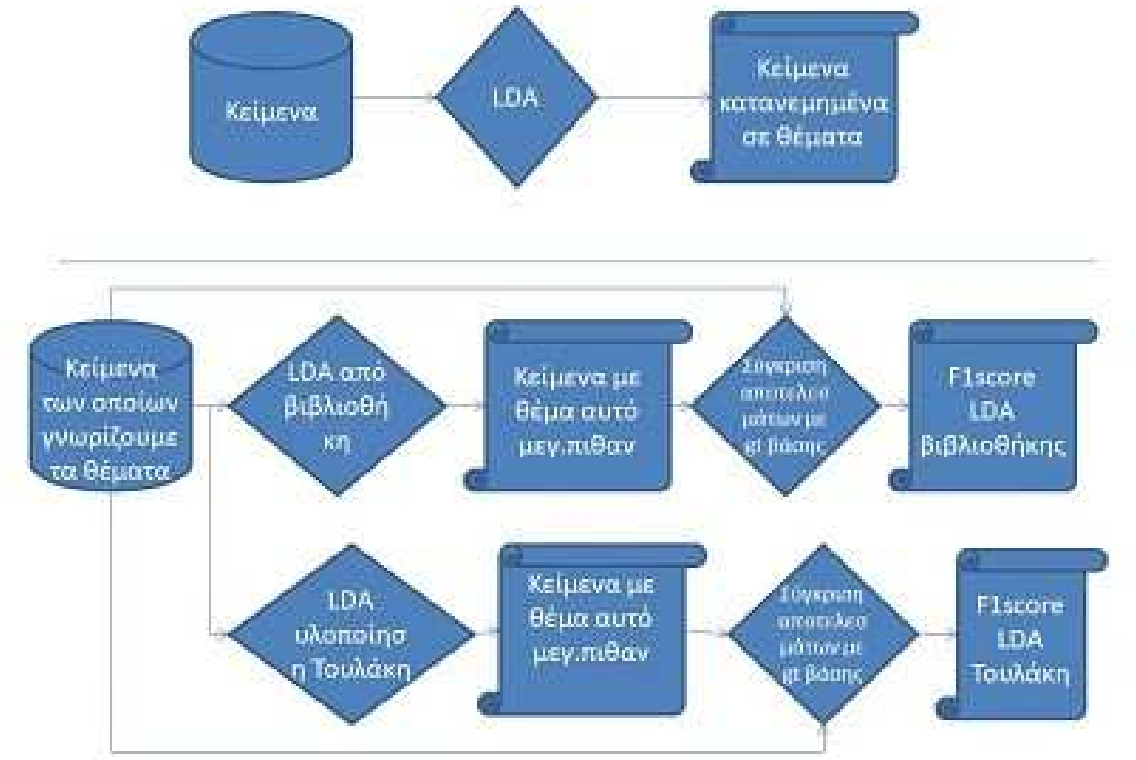
Σχήμα Γ΄.6: Μαθηματικοί τύποι για το φ και ϑ

Γ΄.4.3 Παράμετροι Εκτέλεσης Πειράματος

Αφού γίνει η συγκέντρωση και προεπεξεργασία των κειμένων πραγματοποιείται η εκτέλεση του αλγόριθμου LDA. Έτσι, στην συλλογή κειμένου με τον αλγόριθμο του Gensim[1] ο οποίος θεωρείτε ο ground truth αλγόριθμος δηλαδή αυτός που δουλεύει σωστά και συγκρίνεται με τον ψευδοκώδικα της υλοποίηση προκειμένου να διαπιστωθεί κατά πόσο καλά μπορεί να ανταποκριθεί ο αλγόριθμος αυτός. Για την αξιολόγηση των μοντέλων χρησιμοποιήθηκε το f1-score. Επίσης έγινε χρήση του K-fold Cross Validation για $\kappa = 10$ και εκτελέστηκε για 25 seeds έτσι έγινε υπολογισμός του μέσου όρου των f1-scores. Όσο αφορά τις παραμέτρους οι οποίες δόθηκαν ως είσοδο στους Αλγόριθμους LDA είναι 100 αριθμοί επαναλήψεων. Οι αριθμοί των επαναλήψεων επιλέχθηκαν αυθαίρετα, καθώς δόθηκε μεγαλύτερη έμφαση στην σύγκλιση των αποτελεσμάτων των δύο υλοποιήσεων. Οι τιμές που δόθηκαν για τα α και β που δόθηκαν είναι $a = 50/T$ και $b = 0.1$[10] όπου T ο αριθμός των θεμάτων που χρησιμοποιήθηκαν προέκυψε απο τον πίνακα των. items.

Γ΄.4.4 Αξιολόγηση LDA

Για την αξιολόγηση του LDA έγινε το εξής : ολα τα έγγραφα χωρίζονται σε tokens. Το κάθε token είναι μια λέξη. Στην συνέχεια γίνεται αναζήτηση της λέξης αυτής στην κατανομή φ η οποία για κάθε θέμα περιέχει λέξεις συνοδευόμενες με κάποια πιθανότητα εμφάνισης. Σε περίπτωση που βρεθεί η λέξη αυτή γίνεται κατακράτηση των πιθανοτήτων για το token αυτό με όλα τα θέματα που βρέθηκαν και γίνεται επιλογή του θέματος με την μεγαλύτερη πιθανότητα. Τέλος αφού αυτή η διαδικασία έχει επαναληφθεί για όλες τις λέξεις διαλέγεται το θέμα με την μεγαλύτερη πιθανότητα. Η διαδικασία αυτή επαναλαμβάνεται για όλες τις λέξεις για όλα τα έγγραφα. Για την αξιολόγηση χρησιμοποιήθηκε το μέτρο του f1-score. Η αξιολόγηση πραγματοποιήθηκε και για την υλοποίηση του Γενσιμ που είναι γνωστό πως δουλέυει σωστά, καθώς και για την LDA της πτυχιακής, ενώ στην συνέχεια έγινε σύγκριση των αποτελεσμάτων προκειμένου να διαπιστωθεί κατά πόσο τα αποτελέσματα αυτά είναι κοντά μεταξύ τους. Η διαδικασία αυτή φαίνεται και στο παρακάτω σχήμα.



Σχήμα Γ΄.7: Αξιολόγηση των δύο υλοποιήσεων LDA

Γ΄.4.5 Αποτελέσματα

Τρέχοντας τον αλγόριθμο του LDA και για τις δύο υλοποιήσεις τα αποτελέσματα των f1-score είναι χαμηλά, ωστόσο θετικό στοιχείο αποτελεί πώς τα αποτελέσματα των f1-scores είναι σχετικά κοντά μεταξύ τους το οποίο πρακτικά φανερώνει πως ο αλγόριθμός που υλοποιήθηκε πετυχαίνει αρκετά καλά αποτελέσματα. Από την άλλη όμως αξίζει να αναθερφούν πιθανόν λόγοι για τα χαμηλά f1-scores οι οποίοι είναι: Δεν έγινε υλοποίηση κώδικα ο οποίος αναλαμβάνει να αλλάξει τις τιμές των hyperparameter α όπως περιγράφεται αναλυτικά[9]. Ακόμα ένα πολύ σημαντικό κομμάτι σε όλη αυτήν την διαδικασία της προεπεξεργασίας αποτελεί η λίστα για τις ελληνικές λέξεις stopwords η οποία είναι περιορισμένη και ενδεχόμενως η επέκταση της θα μπορούσε να βελτιώσει τα αποτελέσματα καθώς και το LDA με background κατανομή [8]. Επιπρόσθετα θα μπορούσαν να γίνουν πειραματισμοί με την δοκιμή για διαφορετικές παράμετρους όπως

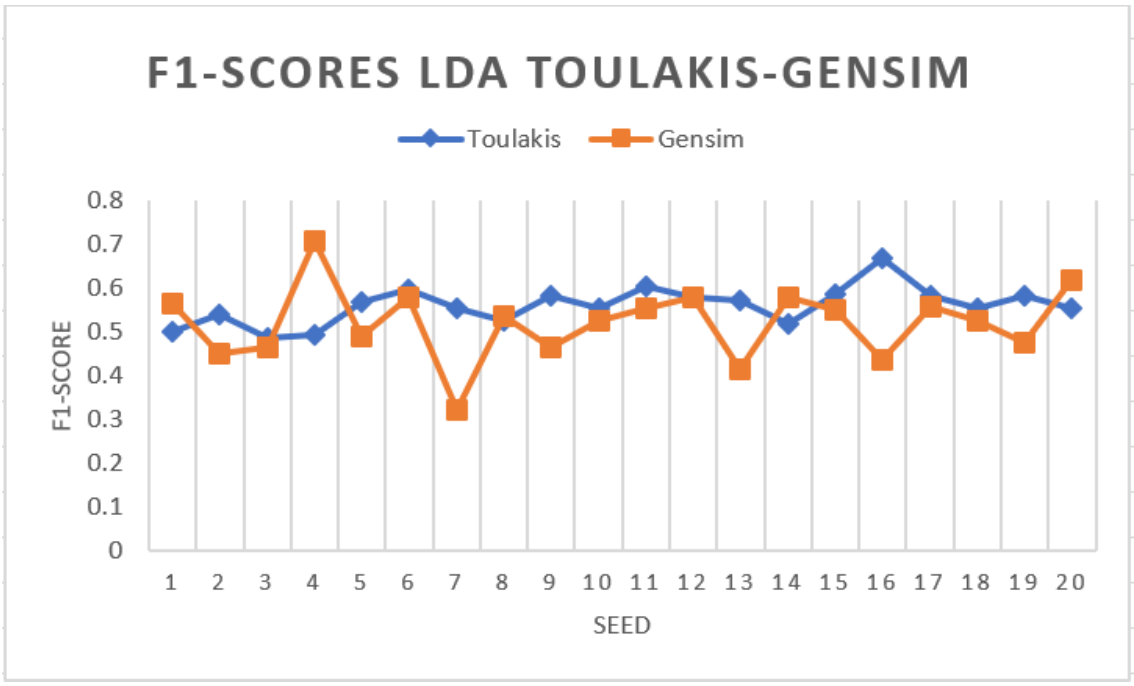
LDA	F1-score
Gensim	0.560125
Toulakis	0.51952

Table C.3: Αποτελέσματα των δύο υλοποιήσεωνLDA

τον ελάχιστο αριθμό των λέξεων που απαιτείται ένα έγγραφο να έχει και η χρήση N grams διαφορετικών μεγεθών. Ακόμα όσο αφορά το LDA θα μπορούσαν να δοκιμαστούν περισσότεροι αριθμοί για το πόσες φορές μπορεί να εκτελεστεί το πείραμα

.

Οι μέσες τιμές για τα f1-score που προέκυψαν προήλθαν απο τα 20 seeds που χρησιμοποιήθαν. Τα οποία παρατίθενται στο επόμενο γράφημα(άξονας χ Seed που χρησιμοποιηθηκε), ενώ η τιμή f1-score παρατίθεται στον άξονα Υ. Απο το παρακάτω γράφημα συμπερένεται πως οι τιμές των F1-score μεταξύ των δύο περιπτώσεων είναι αρκετά κοντά μεταξύ τους με εξαίρεση κάποιες τιμές οι οποίες σημείωσαν απότομες αυξομειώσεις. Τα αποτελέσματα αυτά μπορούν να ωφείλονται σε παράγοντες όπως την διαφορετικότητα ανάμεσα στις δύο υλοποιήσεις, την εμφάνιση στοχαστικότητας στα πειράματα, στην απλοϊκότητα της μεθόδου αξιολόγησης (f1-score) και όχι χρήση μεθόδους αξιολόγης με το χέρι, με μέτρα όπως Perplexity[33].



Σχήμα Γ΄.8: Συγκρίσεις υλοποιήσεων LDA μεταξύ τους

Κεφάλαιο Δ΄

Πείραμα με Μουσικά Κομμάτια και Εικόνες

Σε αυτό το κεφάλαιο γίνεται περιγραφή πειραμάτων καθώς και των αποτελεσμάτων που προέκυψαν για μουσικά κομμάτια και για τις εικόνες στο σημείο αυτό αξίζει να αναφερθεί πως οι ενότητες απο Δ.1 - Δ.3 αναφέρονται στην μεθοδολογία που ακολουθήθηκε για τις εικόνες ενώ ανστίστοιχα οι εναπομειναντές ενότητες αντιστοιχούν στα πειράματα με τις εικόνες.

Δ΄.1 Σύντομη Περιγραφή

Έγινε κατέβασμα πληροφορίας από το Last-fm[15] και από το Labsrosa[18]. Στην συνέχεια έγινε αποθήκευση της πληροφορίας αυτής και έγινε τρέξιμο του πειράματος με χρήση μιάς μεθόδου που είναι παραλλαγή τηςsubset. Αναλυτικότερα για κάθε μουσικό κομμάτι βρέθηκε αυτό με τις περισσότερες κοινές ετικέτες και συγκρίθηκε με τα αποτελέσματα της συνάρτηση Track.getsimilar(), η οποιά θεωρήθηκε ως η βάση αλήθειας(γρουνδ τρυτη), καθώς τα αποτελέσματα της προέρχονται απο τα ακουστικά δεδομένα χρηστών. Για την αξιολόγηση των αποτελεσμάτων χρησιμοποιήθηκε το μέτρο του f1-score το οποίο σημειώθηκε να είναι 0.81. Ο λόγος που διεξήχθει το πείραμα ήταν για να διαπιστωθεί σε τι βαθμό η προτεινόμενη μέθοδος μπορεί να βρεί ένα παρόμοιο τραγούδι το οποίο βρίσκεται μέσα στα αποτελέσματα, αφού η συνάρτηση αυτή προέρχεται απο ανθρώπινα δεδομένα.

Δ΄.2 Δεδομένα

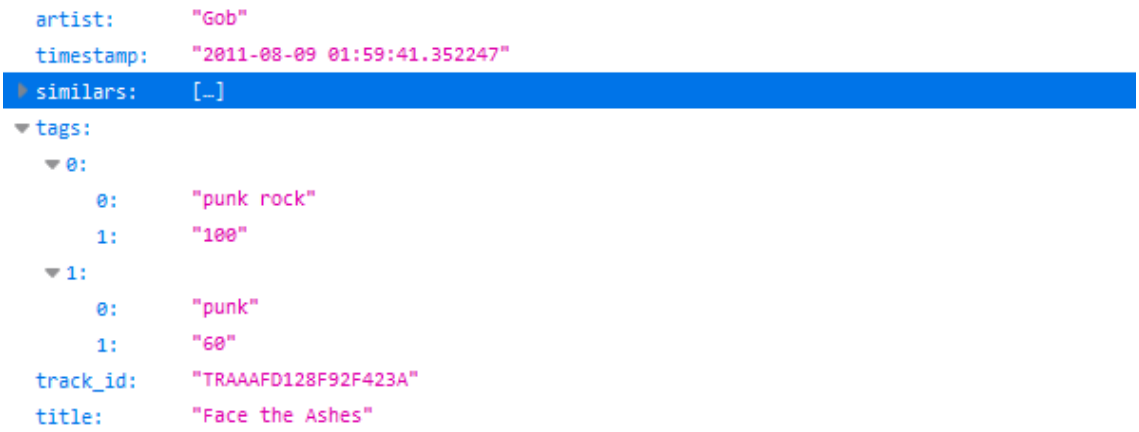
Όπως προαναφέρθηκε και στην σύντομη περιγραφή του πειράματος, τα δεδομένα προήλθαν από το Last-fm και το Labsora. Αναλυτικότερα έγινε κατέβασμα από την ιστοσελίδα του Labrosa ένος συνόλου δεδομένων το οποίο είχε προκατασκευαστεί από το Last-fm. Το σύνολο αυτών των δεδομένων ήταν χωρισμένο σε train και test set . Στο πείραμα αυτό δεν έγινε εκπαίδευση κάποιου αλγόριθμου μηχανικής μάθησης και για την απλοποίηση του κώδικα έγινε τοποθέτηση των αρχείων μέσα σε ένα κοινό φάκελο. Παρακάτω παρατίθεται η δομή ενός αρχείου json που κατέβηκε από το Labrosa. Από το αρχείο παρατηρείται ότι για ένα μουσικό κομμάτι περιέχεται το όνομα του μουσικού κομματιού και του καλλιτέχνη. Επίσης υπάρχει και πίνακας με τα Similar Tracks τα αποτελέσματα αυτά φαίνονται στην επόμενη εικόνα. Επίσης διακρίνεται ότι υπάρχει και πίνακας με ετικέτες. Ακόμα υπάρχει και αναγνωριστικό για το μουσικό άσμα το οποίο είναι το track_id. Ο πίνακας με τα παρόμοια μουσικά κομμάτια έχει τα κορυφαία μουσικά κομμάτια. Για κάθε θέση του πίνακα αυτού, υπάρχει ένας εμφωλευμένος πίνακας δύο θέσεων. Στην πρώτη θέση βρίσκεται το track_id, ενώ στην δεύτερη θέση το ποσοστό match που σημειώθηκε. Ακόμα το όνομα κάθε αρχείου ταυτίζεται με το track_id είναι χρήσιμο καθώς με βάση το track_id υπάρχει η δυνατότητα εύρεσης του Json αρχείου του μουσικού κομματιού και εξαγωγής

Δ΄.2.1 Προβλήματα με τα δεδομένα

Το πρόβλημα ηταν πως μέσα σε κάποια αρχεία υπήρχε αναφορά σε αρχεία(Track ids) του πίνακα με παρόμοια μουσικά κομμάτια τα οποία δεν ήταν διαθέσιμα. Ένα ακόμα ζήτημα αποτέλεσε ότι το track_id δεν δημιουργήθηκε από το last - fm άρα δεν μπορούσε να δωθεί ως είσοδος στο API για την εύρεσηεξαγωγή πληροφοριών. Για την αντιμετώπιση του προβλήματος έγινε η χρήση της συνάρτησης Track.getsimilar του API του last fm η οποία επιστρέφει για ένα μουσικό κομμάτι παραπλήσια άσματα τα οποία συνοδεύονται απο ένα ποσοστό ταιρίασματος.

Δ΄.2.2 Last fm API

: Σε αυτήν την παράγραφο γίνεται μια λεπτομερής περιγραφή των συναρτήσεων του ΑΠΙ του last fm που χρησιμοποιήθηκαν στο πείραμα για τα μουσικά κομμάτια οι οποίες είναι η track.getSimilar και η track.getInfo. Έπειτα παρατίθεται ο ψευδοκώδικας που



Σχήμα Δ΄.1: json Labrosa βασικές πληροφορίες

τραβάει τις απαραίτητες πληροφορίες.

Δ΄.2.3 Μέθοδος Track.getSimilar

Συνάρτηση API: Track.getSimilar Σύμφωνα με την περιγραφή από το documentation του API του last fm, η μέθοδος Track.getSimilar«Επιστρέφει τα παρόμοια τραγούδια ενός άσματος και τα αποτελέσματα αυτά βασίζονται στο ιστορικό των τραγουδιών που ακούστηκαν» Οι παράμετροι πού δέχεται φένεται στον πίνακα Δ΄.1. Από τον πίνακα αυτό αξίζει να γίνει μία επεξήγηση σχετικά με το MusicBrainzId(mbid). Σε περίπτωση που δοθεί αυτή η παράμετρος τότε μπορεί να γίνει παράληψη του Track και του Artist Με βάση το id αυτό γίνεται εξαγωγή της πληροφορίας απο την βάση του Musicbrainz[19]. Οι παράμετροι που χρησιμοποιήθηκαν είναι το Track, Artist, Limit και το API_KEY.

Δ΄.2.4 Χτίσιμο URL Track.getSimilar

: Στην συνέχεια για να μπορούμε να πάρουμε τα αποτελέσματα από το API δεδομένων πρέπει να δημιουργηθεί ένα URL το οποίο δίνει ως έξοδο τις απαραίτητες πληροφορίες. Το API μπορεί να στείλει τις πληροφορίες αυτές είτε μέσω xml είτε μέσω json. Για το

Όνομα παραμέτρου	Περιγραφή	Προκαθορισμένη Τιμή	Υποχρεωτική
Track	Όνομα τραγουδιού	Δεν Παίρνει	ΝΑΙ
Artist	Όνομα Καλλιτέχνη	Δεν Παίρνει	ΝΑΙ
mbid	Το MusicBrainz id του μουσικού κομματιού.	Δεν παίρνει	ΟΧΙ
Autocorrect	Παίρνει τιμές 0 ή 1 σε περίπτωση που κάποιο όνομα καλλιτέχνη δοθεί λανθασμένα αναλαμβάνει να το επιδιορθώσει	Παίρνει τιμή 0. Δηλαδή δεν προσπαθεί να κάνει autocorrect	
Limit	Ο μέγιστος αριθμός των αποτελεσμάτων που να επιστρέψει. Προκαθορισμένη τιμή 10	Ο μέγιστος αριθμός είναι 10	ΟΧΙ
API_KEY	Κλειδί για να μπορούμε να χρησιμοποιήσουμε το API	Δεν παίρνει ΝΑΙ	

Πίνακας Δ΄.1: πίνακας με τις παραμέτρους που παίρνει η συνάρτηση track.getSimilar

πείραμα αυτό προτιμήθηκε το format του json γιατί μεταφέρει λιγότερη πληροφορία. Μετά χρησιμοποιείται ως βάση το URL της μορφής:

```
https://ws.audioscrobbler.com/2.0?method=track.getsimilar
```

στο url αυτό πρέπει να περάσουμε μέσω query strings τις απαραίτητες παραμέτρους, για παράδειγμα έστω ότι θέλουμε να μας επιστραφούν τα κορυφαία 5 τραγούδια για τον τραγουδιστή Cher με όνομα άσματος believe το URL αυτό είναι.

```
http://ws.audioscrobbler.com/2.0/?method=track.getsimilar
&artist=cher&track=believe
```

Η πληροφορία για τα παρόμοια μουσικά κομμάτια βρίσκεται στον πίνακα track ο οποίος περιέχει πίνακα με αντικείμενα. Τα οποία περιέχουν την παρακάτω εικόνα με τις απαραίτητες πληροφορίες οι οποίες είναι το όνομα του αλβυμ και το αντικείμενο του artist. Το οποίο περιέχει την πληροφορία για το όνομα του καλλιτέχνη
.

Δ΄.2.5 Ψευδοκώδικας last - fm API Track.getsimilar

Απο τις παραμέτρους που παίρνει ως είσοδο γίνεται χρήση των παραμέτρων artist, track, topn,limit, api_key. Η συνάρτηση αυτή επιστρέφει μια λίστα όπου κάθε στοιχείο της λίστας αυτής είναι της μορφής (sim_artist_name, sim_track_name, match_score). Παρακάτω παρατίθεται ο ψευδοκώδικας που υλοποιεί το κομμάτι της εξαγωγής των παρόμοιων μουσικών κομματιών απο την πλατφόρμα του Last Fm.

```
def track_get_similar(artist, track, topn, api_key, limit)
        Similars = list()
        Similar_json=http.get(http://ws.audioscrobbler.com
        /2.0/?method=track.getsimilar&artist=s &track=s &limit=d
        &api_key=s & format=json, artist, track, topn, api_key)
        .content
        Similar_json = Json.parse(Similar_json)
        Similar_tracks = Similar_json["similartracks"]["track"]
        if isempty(Similar_tracks)
                return list()
```

```
for i in range(0, topn):
        Sim_album_name = Similar_tracks[i]["name"]
        Sim_album_match = sim_track[i]["match"]
        sim_artist_name = sim_track[i]["artist"]["name"]
        add_tolist(Similars,
         list(Sim_album_name,
         Sim_album_match, sim_artist_name))
return Similars
```

Δ΄.2.6 Περιγραφή Ψευδοκώδικα Track.getSimilar

Αρχικά γίνεται αρχικοποίηση της μεταβλητής Similars για την αποθήκευση των αποτελεσμάτων. Στην συνέχεια πραγματοποιείται ανάκτηση του json από το Api του Last-FM χτίζοντας το κατάλληλο URL και το json αυτό αρχείο γίνεται Parse προκειμένου να μπορέσουμε προγραμματιστικά να αποκτήσουμε πρόσβαση στις επιθυμητές πληροφορίες. Από το json αυτό οι πληροφορίες που ψάχνουμε βρίσκονται στον πίνακα με όνομα track. Ο πίνακας αυτός εντοπίζεται εντοπίζεται μέσα στο αντικείμενο similartracks. Αφού πρώτα γίνει προσπέλαση του πίνακα similartracks(γραμμή 4), γίνεται έλεγχος για να διαπιστωθεί άμα ο πίνακας αυτός περιέχει δεδομένα. Ο έλεγχος αυτός γίνεται στην γραμμή 5. Η διαδικασία με την οποία «τραβάμε» τα απαραίτητα δεδομένα γίνεται στο for loop και επαναλαμβάνεται min(limit, sim_track_len) φορές Όπου sim_track_len είναι το μέγεθος του πίνακα των similar tracks που επιστράφηκε από το api και το limit είναι τα πόσα αποτελέσματα ζητήσαμε από το απι να επιστρέψει. Αφού έχουμε πρόσβαση τώρα τις απαραίτητες πληροφορίες τις εισάγουμε σε μια λίστα (γραμμές 7, 8, 9 , 10, 11). Τέλος επιστρέφουμε την λίστα αυτήν με τα αποτελέσματα.

Δ΄.2.7 μέθοδος Track.getInfo

Η συνάρτηση track.getInfo του API του last fm λαμβάνει ώς είσοδο τον καλλιτέχνη και το μουσικό κομμάτι και επιστρέφει μεταδεδομένα για το άσμα αυτό. Σύμφωνα με την περιγραφή από το δοσυμεντατιον του απι του last fm, η μέθοδος Track.getInfo() «Επιστρέφει πληροφορίες για το μουσικό κομμάτι που δόθηκε ως είσοδο»

Δ΄.2.8 Χτίσιμο URL Track.getInfo

Όπως και για την συνάρτηση του Track.getSimilar έτσι και εδώ πρέπει στο URL να περάσουμε τις απαραίτητες παραμέτρους μέσω query strings στις παραμέτρους. Οι παράμετροι που χρησιμοποιήθηκαν είναι το api_key, Track και Artist. Για παράδειγμα το Url που δίναμε στο api του last fm για να πάρουμε τα μεταδεδομένα για το μουσικό κομμάτι believe της τραγουδίστριας Cher

.

```
http://ws.audioscrobbler.com/2.0/?method=track.getIno
&api_key=bc884699cfc25509817acde7be2f3878
&artist=cher&track=believe&format=json
```

Από το αρχείο που θα επιστραφεί η μοναδική πληροφορία που χρειάζονται είναι οι ετικέτες ενός άσματος. Οι οποίες είναι αποθηκευμένες σε έναν πίνακα που ονομάζεται tag ο οποίος με την σειρά βρίσκεται μέσα στο αντικείμενο toptags και το οποίο περιέχεται στο αντικείμενο track.

Δ΄.2.9 Ψευδοκώδικας της συνάρτηση Track.getInfo

: Η συνάρτηση αυτή παίρνει ως είσοδο: Το όνομα του καλλιτέχνη, το όνομα του τραγουδιού αυτού και το api_key, ενώ επιστρέφει τις ετικέτες που βρέθηκαν.

```
def get_track_tags(artist, track, api_key)
        tags= list()
        json_file = parse_json_byUrl(artist, track, api_key)
        top_tags = json_file["toptags"]["tag"]
        for each tag in top_tags
                Add_to_list(tags, tag)
        return artist, track, tags
```

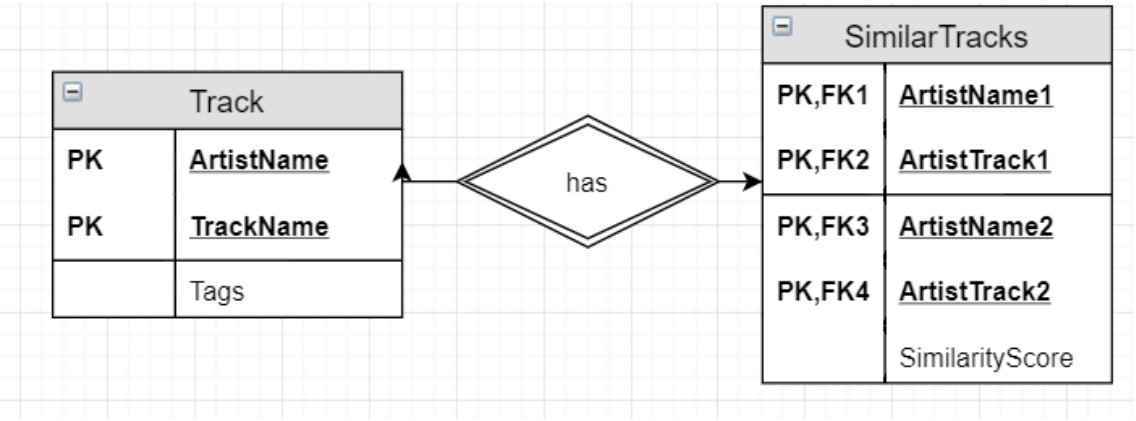
Δ΄.2.10 Περιγραφή ψευδοκώδικα Track.getinfo

Αφού αρχικοποιηθεί η λίστα με τα αποτελέσματα, στην συνέχεια στην δεύτερη γραμμή του κώδικα η συνάρτηση δημιουργεί το επιθυμητό url και επιστρέφει το επιθυμητό json αρχείο και στην Τρίτη γραμμή αποκτάται πρόσβαση στην λίστα με τις ετικέτες. Στο for Loop κάθε ετικέτα που βρέθηκε προστίθεται στην λίστα με τις ετικέτες. Τέλος

γίνεται επιστροφή όλων των ετικετών που βρέθηκαν στο json αρχείο.

Δ΄.3 Αρχιτεκτονική Συστήματος Πειράματος Μουσικής - Αποτελέσματα

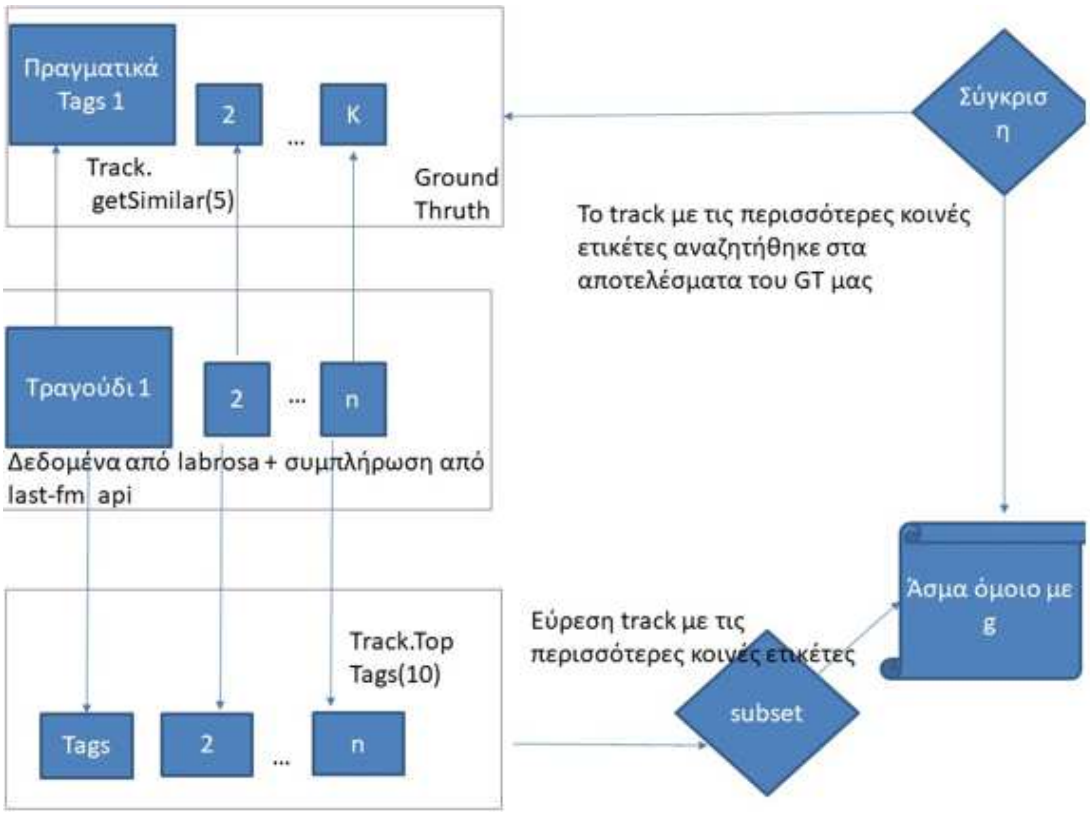
Αρχικά στο πρώτο βήμα έγινε συλλογή των δεδομένων. Στην συνέχεια για κάθε μουσικό κομμάτι της βάσης γίνεται έλεγχος για να να διαπιστωθεί πόσες κοινές ετικέτες έχει με τα υπόλοιπα άσματα. Οι ετικέτες προέρχονται απο την συνάρτηση Track.getinfo και απο τις οποίες κρατάμε τις κορυφαίες. Μετά επιλέγεται το άσμα με τις περισσότερες κοινές ετικέτες και αναζητήται στα αποτελέσματα του ground truth. Σε περίπτωση που βρεθεί ανήκει στην κλάση 1 ενώ, στην αντίθετη περίπτωση παίρνει την τιμή 0. Για την επαλήθευση των αποτελεσμάτων έγινε χρήση του μέτρου του f1-score = 0.81 το οποίο υποδηλώνει ότι οι ετικέτες μπορούν να αποτελέσουν ένα σημαντικό τρόπο για την ανεύρεση ομοιότητας μεταξύ μουσικών κομματιών.



Σχήμα Δ΄.2: Σχήμα βάσης αποθήκευσης δεδομένων με πληθικότητα πολλά προς πολλά

Δ΄.4 Σύντομη περιγραφή

Στο πείραμα με τις εικόνες ακολουθήθηκε παρόμοια μεθοδολογία με αυτή που χρησιμοποιήθηκε για τα μουσικά άσματα. Αυτήν την φορά όμως δεν υπήρχε κάποιο διαθέσιμο μέτρο με το οποίο μπορεί να υπολογιστεί η ομοιότητα χρησιμοποιήθηκε η συνάρτηση



Track.getSimilar της υπηρεσία του Last fm. Για αυτόν τον λόγο ένα άσμα θεωρήθηκε κοινό με ένα άλλο έγινε βάση της ομοιότητας μεταξύ των ετικετών. Στην συνέχεια έγινε χρήση του GoogleVision api[26], εκτέλεσης του πειράματος με χρήση της μεθόδου που είναι παραλλαγή της subset και τέλος για την αξιολόγηση του μοντέλου έγινε χρήση του f1-score το οποίο σημειώθηκε να είναι 0.71.

Δ΄.5 Δεδομένα

Αρχικά για το dataset των εικόνων χρησιμοποιήθηκε το Dataset που εντοπίζεται εδώ [21] στο οποίο περιέχονται εικόνες οι οποίες είχαν εξαχθεί αλγοριθμικά και οι ετικέτες αυτές επιβεβαιωθήκαν από ανθρώπους. Παρόλα αυτά τρέχοντας το πείραμα της παραλλαγής τηςsubset το f1 score ήταν αρκετά χαμηλό 0.15. Μετά από μια διερεύνηση διαπιστώθηκε πως το πρόβλημα βρίσκονταν στο dataset, καθώς οι ετικέτες που δόθηκαν στις εικόνες δεν ανταποκρίνονται στην πραγματική εικόνα. Για αυτόν τον λόγο έγινε χρήση δεδομένων από διαφορετική πηγή. Τα δεδομένα που χρησιμοποιήθηκαν

προέρχονται από την ?labelme? η οποία είναι μια διαδικτυακή πλατφόρμα για το annotation εικόνων. Η πλατφόρμα αυτή παρέχει ελεύθερα πρόσβαση στα δεδομένα αυτά και πιο συγκεκριμένα για κάθε εικόνα υπάρχει ένα αρχείο το οποίο περιέχει τις ετικέτες που δόθηκαν από τους ανθρώπους[20][21]. Έτσι εκτελώντας ξανά το πείραμα της ανάλαφρης παραλλαγής το f1 score που σημειώθηκε ήταν 0.30. Αυτήν την φορά κοιτάζοντας τις ετικέτες διαπιστώθηκε πως ήταν αντιπροσωπευτικές, απλά αρκετές από αυτές θα μπορούσαν να συμπληρωθούν η και κάποιες από αυτές να διαγραφούν.

Δ΄.5.1 Επίλυση προβλήματος με τα δεδομένα

Για την καταπολέμηση του προ αναφέροντος προβλήματος δημιουργήθηκε πρόγραμμα που θα βοηθήσει στο χειροκίνητο αννοτατιον. Αναλυτικότερα δημιουργήθηκε ένας crawler οπού κάνει scraple από την ιστοσελίδα τα urls για τις εικόνες και τις ετικέτες και για κάθε ένα από αυτά τα λινκς εκτελείται η διαδικασία που απεικονίζεται στην εικόνα που ακολουθεί. Ο λόγος που αποφασίστηκε να γίνει μερικό annotation είναι γιατί διαπιστώθηκε πως πολλές από τις ετικέτες αυτές είναι σωστές. Η διαδικασία αυτή επαναλήφθηκε για 100 εικόνες, καθώς αυτός ο αριθμός θεωρήθηκε αρκετός για την εκτέλεση του πειράματος.

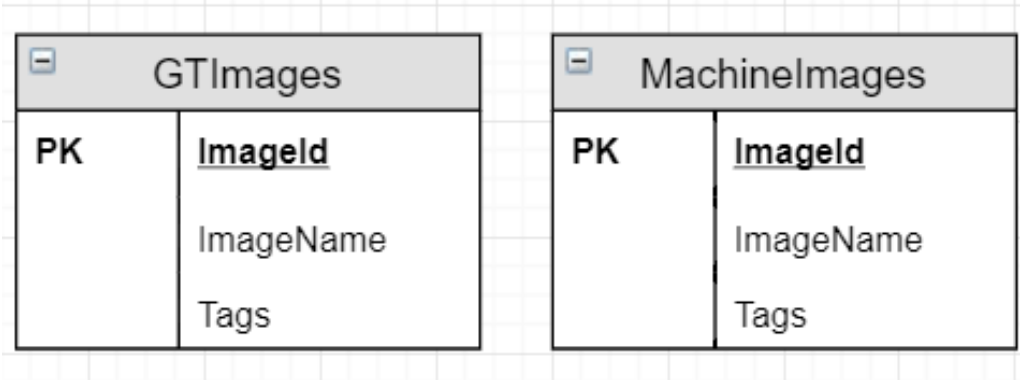
Δ΄.6 Αρχιτεκτονική Συστήματος Πειράματος Εικόνων

Αρχικά στο πρώτο βήμα έγινε συλλογή των ετικετών και των εικόνων. Οι ετικέτες που δημιουργήθηκαν από το μερικό annotation θεωρήθηκαν η ground truth ετικέτες, καθώς επιβεβαιωθήκαν και από ανθρώπους. Στην συνέχεια κάθε μία από αυτές τις εικόνες δίνεται ως είσοδος στο GoogleVision το οποίο εξάγει τις ετικέτες αυτές οι οποίες δεν ξέρουμε κατά πόσο είναι αντιπροσωπευτικές και θέλουμε να δούμε πόσο κοντά μπορούμε να πέσουμε στα αποτελέσματα του ground truth. Για κάθε εικόνα βρίσκουμε αυτήν με την οποία έχει τις περισσότερες κοινές ετικέτες και αναζητούμε άμα αυτή υπάρχει στα αποτελέσματα του ground truth. Σε αυτήν την περίπτωση, μπορούμε να πούμε ότι ανήκει στην κλάση 1 ειδάλλως στην 0. Όσο αφορά τα δεδομένα του ground truth αυτά δημιουργήθηκαν με τον εξής τρόπο: Για κάθε εικόνα αποθηκεύτηκαν οι εικόνες οι οποίες είχαν με αυτή τις περισσότερες 5 κοινές ετικέτες. Τέλος έγινε αξιολόγηση του μοντέλου χρησιμοποιώντας όπως και στο πείραμα της μουσικής το



Σχήμα Δ΄.3: ΔΡΔ διάγραμμα συμπλήρωσης/διόρθωσης ετικετών

f1 score το οποίο σημειώθηκε να είναι 0.71. Από το πείραμα αυτό διαπιστώνουμε πως οι ετικέτες που εξήχθησαν από το GoogleVision Api μπορεί να σημειώσει μια ικανοποιητική απόδοση στην απόδοση σωστών ετικετών.



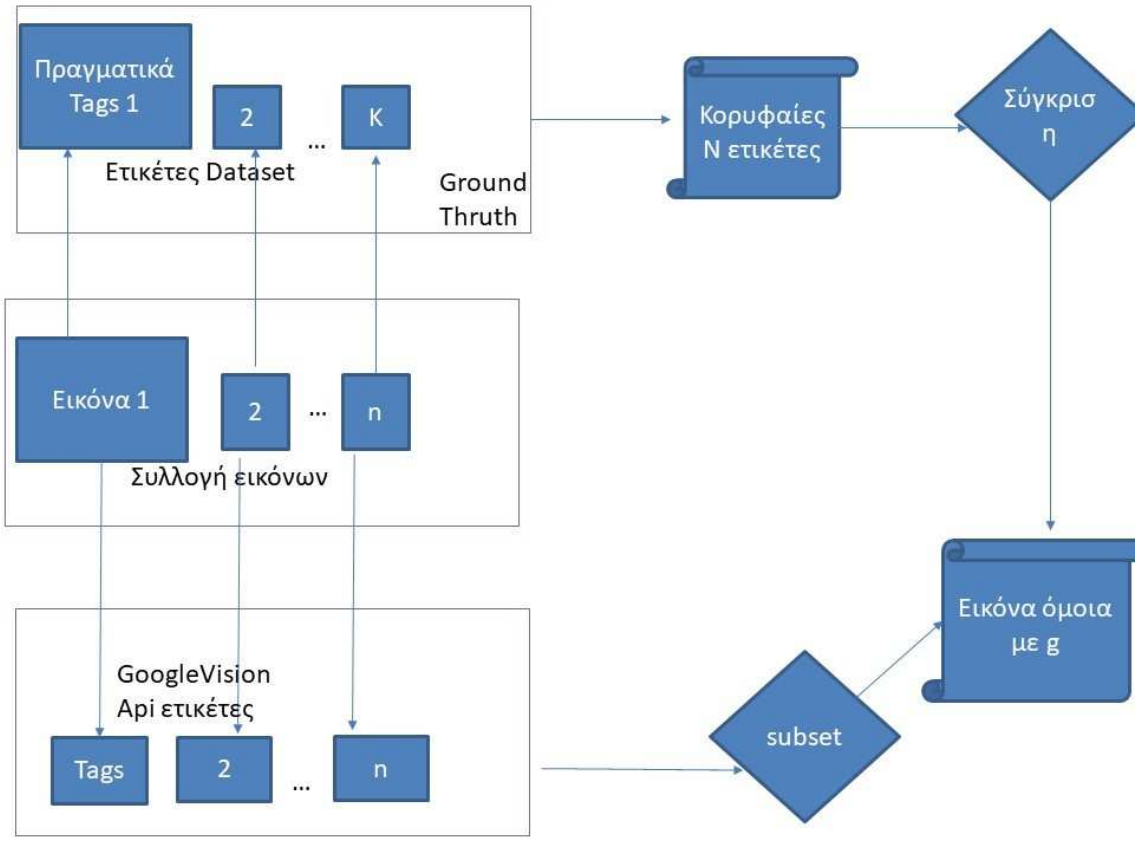
Σχήμα Δ΄.4: Βάση δεδομένων για τις εικόνες

Δ΄.6.1 Αποτελέσματα πειραμάτων

Στην υποενότητα αυτή παρατίθεται πινακάκι με τα αποτελέσματα των πειραμάτων για τα μουσικά κομμάτια και τις εικόνες.

Πείραμα	F1-score
Μουσικής	0.81
Εικόνων	0.71

Πίνακας Δ΄.2: Αποτελέσματα F1-scores για τα πειράματα της μουσικής και των εικόνων



Σχήμα Δ΄.5: ΔΡΔ για επισημείωση εικόνων

Κεφάλαιο Ε΄

Συμπεράσματα

Το κύριο αντικείμενο της πτυχιακής ήταν η διαχείριση πολυμεσικού περιεχομένου από μεγάλες συλλογές κειμένων, μουσικών κομματιών και εικόνων. Αναλυτικότερα για τα κείμενα έγινε υλοποίηση του αλγορίθμου LDA και έγινε σύγκριση του με μια προϋπάρχον ορθή υλοποίηση LDA. Το σύνολο δεδομένων που χρησιμοποιήθηκε ήταν προσημειωμένο για κάθε κείμενο στην θεματολογία στην οποία ανήκει. Στην συνέχεια πραγματοποιήθηκαν πειράματα με εικόνες και μουσικά κομμάτια με την χρήση μεθόδου η οποία είναι παραλλαγή της Subset, η οποία εντοπίζει παρόμοιους πολυμεσικούς πόρους κάνοντας χρήση των ετικετών τους. Το σύνολο των ετικετών για τα μουσικά κομμάτια προήλθε από το api του last-fm. Ενώ παράλληλα για κάθε τραγούδι βρέθηκαν τα παρόμοια μουσικά κομμάτια μαζί με το score ομοιότητας τους. Αυτά θεωρήθηκαν τα σωστά αποτελέσματα και προέκυψαν από το api του last-fm. Αντίστοιχα, στο πείραμα με τις εικόνες έγινε συλλογή εικόνων οι οποίες συνοδευόντουσαν με ετικέτες, επιπρόσθετα έγινε χειρωνακτική συμπλήρωση τους. Οι ετικέτες αυτές θεωρήθηκαν σωστές και σε αυτές έγινε εκτέλεση του πειράματος της παραλλαγής της subset προκειμένου να γίνει υπολογισμός score ομοιότητας. Στην συνέχεια οι εικόνες δόθηκαν ως είσοδο στο api του google vision για την εξαγωγή ετικετών και ξαναέγινε εκτέλεση του πειράματος της παραλλαγής της subset. Η αξιολόγηση όλων των πειραμάτων έγινε με την χρήση του f1-score. Αναλυτικότερα στο πείραμα με τα κείμενα έγινε υπολογισμός του μέσου όρου των f1-scores για πολλαπλά seeds. Για τα πειράματα της μουσικής και των εικόνων δεν πάρθηκαν πολλαπλές μετρήσεις καθώς δεν υπήρχαν παράμετροι που μπορούσαν να πάρουν πολλαπλές τιμές. Αντίστοιχα στα πειράματα της μουσικής Όσο αφορά την υλοποίηση του αλγόριθμου

LDA αυτή προσεγγίζει τα αποτελέσματα της προϋπάρχον ορθής βιβλιοθήκης. Παρόλα αυτά σε μερικά από τα f1-scores για ορισμένα seeds σημειώθηκαν τιμές οι οποίες αποκλίνουν αρκετά μεταξύ τους. Το γεγονός αυτό μπορεί οφείλεται σε παράγοντες όπως η στοχαστικότητα της διαδικασίας, οι διαφορετικές υλοποιήσεις καθώς και η μέθοδος που χρησιμοποιήθηκε για την αξιολόγηση των αποτελεσμάτων. Επίσης σημειώνεται πως οι μέσες τιμές των f1-scores ήταν χαμηλές για την υλοποίηση του LDA το f1-score ισούται με 0.519 ενώ του ορθού πλαίσιού 0.560. Συμπερασματικά, παρόλο που το LDA που υλοποιήθηκε πλησιάζει τα αποτελέσματα ενός προϋπαρχον ορθού πλαίσιού κρίνονται αναγκαίες οι παρακάτω βελτιώσεις. Αρχικά, η προσεκτικότερη μελέτη του συνόλου δεδομένων μπορεί να οδηγήσει σε διαφορετικούς πειραματισμούς στην προεπεξεργασίας του κειμένου οι οποίοι με την σειρά τους να οδηγήσουν σε καλύτερα αποτελέσματα (π.χ. από τα κείμενα να διαπιστωθεί πως η ύπαρξη αριθμών είναι περιττή, άρα θα μπορούσε να γίνει αφαίρεση τους). Επιπρόσθετα, μπορούν να δοκιμαστούν διάφορες βελτιώσεις/προσθήκες που είναι διαθέσιμες στην βιβλιογραφία οι οποίες ενδεχόμενος θα οδηγήσουν στην βελτίωση των αποτελεσμάτων. Ακόμα μια σημαντική βελτίωση αποτελεί η αύξηση της ταχύτητας εκτέλεσης του αλγόριθμου. Το παραπάνω αποτέλεσμα μπορεί να επιτευχθεί με την επιστράτευση τεχνικών πολυνημάτωσης (multithreading). Άλλη αλλαγή που κρίνεται αναγκαία είναι η χρήση διαφορετικών μέτρων αξιολόγησης των αποτελεσμάτων που είναι πιο ταιριαστές σε προβλήματα τύπου topic modeling ένα από τα μέτρα αυτά είναι αυτό του Perplexity. Τέλος, μπορεί να γίνει ανάπτυξη μιας διαδικτυακής ιστοσελίδας η οποία με ένα απλοϊκό api να επιτρέπει σε ιστοσελίδες να δίνουν ως είσοδο ένα κείμενο και αυτόματα ο αλγόριθμος να επιστρέφει την θεματολογία των κειμένων αυτών. Όσο αφορά τα πειράματα με την μουσική διαπιστώθηκε πως η χρήση των ετικετών για την έρευση παρόμοιων μουσικών κομματιών αποτέλεσε ένα ικανοποιητικό μέτρο για την έρευση παρόμοιων πολυμεσικών κομματιών. Συμπερασματικά, η παραλλαγή που έγινε είναι απλή στην κατανοήση, δεν χρειάζεται να γίνει εκπαίδευση κάποιου μοντέλου μηχανικής μάθησης και το υψηλό f1-score(0.81) υποδηλώνει πώς η χρήση της μεθόδου αυτής μπορεί να πλησιάσει αποτελέσματα των ανθρώπων. Παρόλα αυτά η μέθοδος αυτή εμφανίζει τους εξής περιορισμούς: πρώτον ο πόρος πρέπει να αναπαρίσταται με πολλές ετικέτες. Δεύτερον η συλλογή των ετικετών πρέπει να είναι αντιπροσωπευτική δηλαδή να μην περιέχει λανθασμένες ετικέτες το οποίο δεν είναι πάντα εύκολο. Ακόμα δεν προβαίνει στην ανάλυση κάποιου πολυμεσικού πόρου για την εξαγωγή χαρακτηριστικών χαμηλού επιπέδου(π.χ. συχνότητα μουσικού κομματιού).Όσο αφορά μελλοντικές βελτιώσεις που μπορούν να γίνουν είναι ο πειραματισμός με

διαφορετικές μεθόδους όπως αυτές της μηχανικής μάθησης, των τεχνητών νευρονικών δικτύων με σκοπό την ανεύρεση μεθόδων με καλύτερη απόδοση. Μία ακόμα προσθήκη είναι η εφαρμογή τεχνικών προεπεξεργασίας στις ετικέτες. Τέλος όσο αφορά το πείραμα με τις εικόνες παρατηρήθηκαν παρόμοια ευρήματα. Αναλυτικότερα, από το πείραμα αυτό διαπιστώθηκε πως οι ετικέτες που προέκυψαν απο τις εικόνες σημείωσαν καλό f1-score0.71 κάτι το οποίο υποδηλώνει ότι οι ετικέτες του google vision api προσσεγίζουν αυτές που προέρχονται απο ανθρώπους σε ικανοποιητικό βαθμό. Μελλοντικές Βελτιώσεις και μειονεκτήματα είναι αυτά που αναφέρθηκαν στο πείραμα με τα μουσικά κομμάτια.

Βιβλιογραφία

[1] https://radimrehurek.com/gensim/index.html

[2] Ftfy: https://ftfy.readthedocs.io/en/latest/{#}mojibake

[3] Decodings-Encodings
https://r12a.github.io/app-encodings/

[4] Nltk: https://www.nltk.org/

[5] Greek Stemmer:
http://deixto.com/greek-stemmer

[6] Python:
https://www.python.org/

[7] M.Steyvers T. Griffiths:
Probabilistic Topic Models .

[8] William M. Darling:
A Theoretical and Practical Implementation Tutorial on Topic Modeling and Gibbs Sampling

[9] ASUNCION ET AL: *On Smoothing and Inference for Topic Models 2009*

[10] Griffiths*?? and Mark Steyvers§: *Finding scientific topics*

[11] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter: *Markov Chain Monte Carlo in Practise* Chapman and Hall/CRC, 1999.

[12] Ioannis Karydis, Spyros Sioutas, Andreas Kanavos and Markos Avlonitis: *On Smoothing and Inference for Topic Models 2009*

[13] Google Vision Api:
https://cloud.google.com/vision/

[14] BeautifulSoup:
https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[15] Last-fm
https://www.last.fm/abou

[16] Labrosa:
https://labrosa.ee.columbia.edu/

[17] MusicBrainz:
https://musicbrainz.org/

[18] Last-Fm Api:
https://www.last.fm/api

[19] OpenImages:
https://storage.googleapis.com/openimages/web/download.html

[20] LabelMe Annotations:
http://people.csail.mit.edu/brussell/research/LabelMe/Annotations/

[21] LabelMe Images :
http://people.csail.mit.edu/brussell/research/LabelMe/Images/

[22] Python:
https://www.python.org/

[23] Renato L. F. Cunha??, Evandro Caldeira? Luciana Fujii: *Determining Song Similarity via Machine Learning Techniques and Tagging Information*

[24] Wang L, Zhang Y, Feng J . *On the Euclidean distance of images* Transactions on Pattern Analysis and Machine Intelligence. 2005;27(8):1334?1339

[25] Ioannis Karydis, Alexandros Nanopoulos: *AUDIO-TO-TAG MAPPING: A NOVEL APPROACH FOR MUSIC SIMILARITY COMPUTATION*

[26] Google Cloud Vision:
https://cloud.google.com/vision/

[27] Stopwords:
https://nlp.stanford.edu/IR-book/html/htmledition/
dropping-common-terms-stop-words-1.html

[28] Get Requests:
https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/GET

[29] David M. Blei, Andrew Y. Ng, Michael I. Jordan: *Latent Dirichlet Allocation*

[30] Salton, G. and McGill, M.J.: *Introduction to Modern Information Retrieva*

[31] Scott Deerwester Susan T. Dumais George W. Furnas Thomas K. Landauer Richard Harshman: *Indexing by latent semantic analysis*

[32] Diane J. Hu: *Latent Dirichlet Allocation for Text, Images, and Music*

[33] Matthew J. Denny? *Text Preprocessing For Unsupervised Learning: Why It Matters, When It Misleads, And What To Do About It*

[34] Sebastiani, Fabrizio *Machine learning in automated text categorization*

[35] K.P Burnham and N.H. Augustin *Model Selection: An Integral Part of Inference*

[36] Slapin Jonathan B. and Sven-Oliver Proksch *A Scaling Model for Estimating TimeSeries Party Positions from Texts*

[37] Hamed Jelodar, Yongli Wang, Chi Yuan *Latent Dirichlet Allocation (LDA) and Topic modeling: models, applications, a survey*

[38] Sun, S., C. Luo, and J. Chen *n A review of natural language processing techniques for opinion mining*

[39] Xu, Z., et al. *Crowdsourcing based social media data analysis of urban emergency events*

[40] Fang, Y., et al *Mining contrastive opinions on political texts using crossperspective topic model*

[41] Tian, K., M. Revelle, and D. Poshyvanyk . *Using latent dirichlet allocation for automatic categorization of software. in Mining Software Repositories*

[42] Lukins, S.K., N.A. Kraft, and L.H. Etzkorn *Bug Localization*

[43] Chen, T.-H., S.W. Thomas, and A.E. Hassan *A survey on the use of topic models when mining software repositories. Empirical Software Engineering, 2016. 21(5): p. 1843-1919.*

[44] Daud, A., et al *Knowledge discovery through directed probabilistic topic models: a survey*

[45] Zheng Xu, Xiangfeng Luo, Yunhuai Liu, Lin Mei and Chuanping Hu *Measuring Semantic Relatedness between Flickr Images: From a Social Tag Based View*

[46] Ioannis Karydis, Spyros Sioutas, Andreas Kanavos, Markos Avlonitis *LeSiM: A Novel Lexical Similarity Measure Technique for Multimedia Information Retrieval.*