ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΑΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ, ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Διερευνητική μελέτη τεχνικών εξόρυξης κειμένου (Text Mining) και ανάπτυξη εκπαιδευτικών παραδειγμάτων

ΣΥΜΜΕΤΕΧΩΝ ΣΠΟΥΔΑΣΤΗΣ

Μιχαηλίδης Δημήτρης(2698)

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ

Τσιμπίρης Αλκιβιάδης (alkisser@gmail.com)

ΣΕΡΡΕΣ

ΦΕΒΡΟΥΑΡΙΟΣ 2020

ΠΕΡΙΛΗΨΗ

Η εξόρυξη κειμένου είναι ο τομέας εκείνος της επιστήμης υπολογιστών, ο οποίος επιχειρεί να επιλύσει το πρόβλημα της υπερχείλισης πληροφοριών που είναι διαθέσιμες στον παγκόσμιο ιστό. Στην ουσία, αυτό που επιχειρεί να κάνει η εξόρυξη κειμένου, είναι να ανακαλύψει καινούρια πληροφορία, χρησιμοποιώντας πληροφορίες οι οποίες υπάρχουν σε διαφορετικές γραπτές πηγές.

Με την παρούσα πτυχιακή επιχειρείται η μελέτη των βασικών τεχνικών του τομέα εκείνου, καθώς και η ανάπτυξη παραδειγμάτων, εκπαιδευτικού σκοπού, των βασικότερων αλγορίθμων εξόρυξης κειμένου.

Το εργαλείο που χρησιμοποιείται για την συγγραφή των παραδειγμάτων αυτών, είναι το MATLAB (MATrix LABoratory), το οποίο είναι ένα περιβάλλον αριθμητικής υπολογιστικής, το οποίο δηλαδή χρησιμοποιείται κατά κύριο λόγο για μαθηματικά προβλήματα. Το MATLAB, για εργασίες με εξόρυξη κειμένου, παρέχει ειδικό toolbox, το text analytics (υποστηρίζεται από την έκδοση R2017b και μετά), το οποίο έχει υλοποιημένους αλγορίθμους για προεπεξεργασία, ανάλυση και μοντελοποίηση δεδομένων σε μορφή απλού κειμένου.

SUMMARY

Text Mining is the area of computer science that seeks to solve the problem of information overflow that is available on the world wide web. In essence, what text mining is trying to do, is discover new information, using information that exists in different written sources.

The present thesis attempts to study the basic techniques of that field, as well as the development of examples, for educational purposes, of the most basic text mining algorithms.

The tool used to write these examples is MATLAB (MATrix LABoratory), which is a numerical computing environment that is used primarily for mathematical problems. MATLAB, for text mining tasks, provides a special toolbox, text analytics (supported by R2017b and later), which has implemented algorithms for pre-processing, analyzing and modeling data in plain text.

Περιεχόμενα

1.ΕΙΣΑΓΩΓΗ

Η Εξόρυξη Κειμένου (text mining) είναι ένα σύνολο τεχνικών οι οποίες χρησιμοποιούν Επεξεργασία Φυσικής Γλώσσας, για να εξάγουν πολύτιμες πληροφορίες από αδόμητο, πολλές φορές, κείμενο.

Η διαδικασία αυτή είναι γνωστή και ως Ανάλυση Κειμένου (text analysis) και αποτελεί στην ουσία την διαδικασία μετατροπής αδόμητου κειμένου, σε χρήσιμη πληροφορία, η οποία είναι επεξεργάσιμη από τον υπολογιστή. Αυτό είναι δυνατό να πραγματοποιηθεί, αναγνωρίζοντας μέσα στο κείμενο παρόμοια θέματα, μοτίβα ακόμη και σχετικές λέξεις κλειδιά.

Η παρούσα πτυχιακή στοχεύει στην μελέτη αυτών των τεχνικών, καθώς και στην παρουσίαση σχετικών παραδειγμάτων στο εργαλείο MATLAB. Για τον λόγο αυτό, η εργασία δομείται με τον ακόλουθο τρόπο:

Στην αρχή, επιχειρείται να γίνει μια περαιτέρω ανάλυση της εξόρυξης κειμένου, εξηγώντας τις διαφορετικές τεχνικές αλλά και εφαρμογές της.

Στη συνέχεια, παρουσιάζονται συνοπτικά τα χαρακτηριστικά του MATLAB και του toolbox το οποίο πρόκειται να χρησιμοποιηθεί για την ανάπτυξη των παραδειγμάτων. Τα παραδείγματα αυτά, γίνεται προσπάθεια να είναι μικρά σε έκταση, καθώς και να είναι ευνόητα στην ανάγνωση, διότι σκοπό έχουν να βοηθήσουν τον αναγνώστη στην κατανόηση της λογικής της κάθε διαδικασίας που αναλύεται.

Στα επόμενα κεφάλαια, ξεκινά η, σε βάθος, παρουσίαση των διαφόρων τεχνικών, διαχωρίζοντάς τες σε:

- *διαχωρισμός κειμένου σε προτάσεις και λέξεις*, εξαγωγή δηλαδή tokens, δεδομένου ενός συνόλου εγγράφων,
- *κανονικοποίηση τμημάτων*, τροποποίηση δηλαδή των tokens έτσι ώστε να είναι εύκολη η ομαδοποίησή τους (για παράδειγμα, οι λέξεις “δάσκαλος” και “δάσκαλοι” πρέπει να θεωρούνται ως ένα token και όχι δύο),
- *αφαίρεση μη επιθυμητών τμημάτων*, αφαίρεση δηλαδή όλων των tokens τα οποία δεν παρέχουν καμία απολύτως πληροφορία για το κείμενο, όπως τα σημεία στίξης ή ειδικοί χαρακτήρες (π.χ. η παύλα “-”).
- *μοντέλο Bag of Words*, διαδικασία με την οποία βρίσκουμε για κάθε μεμονωμένο token, σε πόσα έγγραφα βρίσκεται.

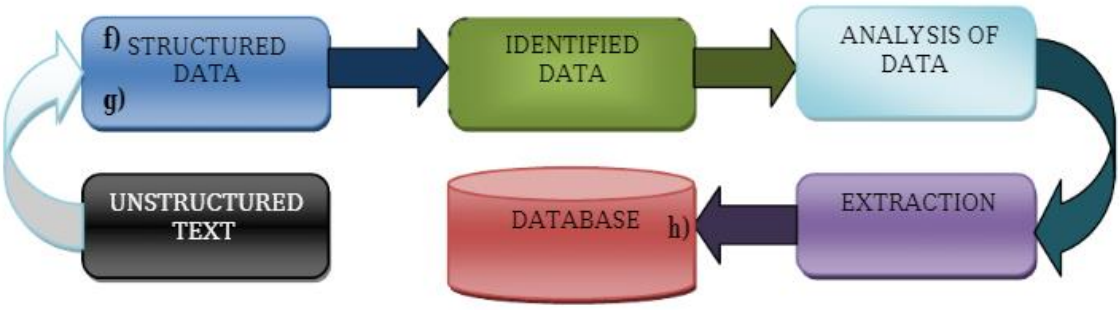
2.ΤΕΧΝΙΚΕΣ ΚΑΙ ΕΦΑΡΜΟΓΕΣ ΤΗΣ ΕΞΟΡΥΞΗΣ ΚΕΙΜΕΝΟΥ

Όπως αναφέρθηκε προηγουμένως, στόχος της εξόρυξης κειμένου είναι η εξαγωγή χρήσιμης πληροφορίας, από κείμενο το οποίο μπορεί να είναι είτε ημι-δομημένο, είτε εντελώς αδόμητο. Για να μπορέσει να το πραγματοποιήσει αυτό, χρησιμοποιεί εργαλεία από διάφορους τομείς όπως η ανάκτηση πληροφορίας (Information Retrieval), εξόρυξη δεδομένων (data mining), μηχανική μάθηση (machine learning), στατιστική κ.τ.λ.

Για την περαιτέρω κατανόηση του τομέα αυτού, μπορούμε να χωρίσουμε την όλη διαδικασία της εξόρυξης κειμένου σε 5 βασικά βήματα:

1. Συλλογή αδόμητης πληροφορίας, από πολλαπλές πηγές (όπως για παράδειγμα απλό κείμενο, ιστοσελίδες, e-mails κ.α.).
2. Προεπεξεργασία δεδομένων, και αφαίρεση μη επιθυμητών δεδομένων. Το τελευταίο μας βοηθά να κρατάμε μόνο τις επιθυμητές πληροφορίες που βρίσκονται στα δεδομένα.
3. Μετατροπή όλης της αδόμητης πληροφορίας που συλλέξαμε και επεξεργαστήκαμε (μόνο η επιθυμητή πληροφορία) σε δομημένη.
4. Αναλύουμε συσχετίσεις μέσα στα δεδομένα.
5. Αποθηκεύουμε όλη την πληροφορία που εξάγαμε σε μια βάση δεδομένων.

Το παρακάτω σχήμα περιγράφει τα προαναφερθέντα βήματα:



Εικόνα 1: Βήματα διαδικασίας εξόρυξης κειμένου

Πηγή: upgrad.com/blog/what-is-text-mining-techniques-and-applications/

2.1. ΤΕΧΝΙΚΕΣ ΕΞΟΡΥΞΗΣ ΚΕΙΜΕΝΟΥ

Τις περισσότερες φορές, όταν θέλουμε να κάνουμε εξόρυξη κειμένου, έχουμε να κάνουμε με έναν πάρα πολύ μεγάλο όγκο πληροφοριών, από τον οποίο επιθυμούμε, αυτοματοποιημένα, να μπορούμε να εξάγουμε ό,τι χρήσιμη πληροφορία μπορούμε. Για τον λόγο αυτό, μπορούμε να διαχωρίσουμε την διαδικασία αυτή σε διάφορες μεμονωμένες τεχνικές ή υπο-διαδικασίες.

Η πρώτη τεχνική που θα δούμε είναι η *ανάκτηση πληροφορίας* (*Information Retrieval*). Η τεχνική αυτή πρόκειται στην ουσία για ένα στάδιο προετοιμασίας. Αυτό που προσπαθούμε να κάνουμε εδώ είναι να συλλέξουμε υλικό, σε μορφή κειμένου, από διάφορες πηγές, όπως ιστοσελίδες, ή αρχεία στον προσωπικό μας υπολογιστή, για ανάλυση. Τα συστήματα αυτά, μπορούν επίσης να χρησιμοποιούν και αλγορίθμους παρακολούθησης της συμπεριφοράς του χρήστη. Ένα μεγάλο παράδειγμα τέτοιου συστήματος είναι και η μηχανή αναζήτησης της Google.

Η *μείωση διάστασης* (*dimensionality reduction*) είναι μια τεχνική προεπεξεργασίας των δεδομένων. Σύμφωνα με αυτή, για κάθε μεμονωμένη λέξη που έχουμε προσπαθούμε να βρούμε τη ρίζα της, έτσι ώστε να μειώσουμε το συνολικό μέγεθος των δεδομένων που έχουμε συλλέξει.

Η *αναγνώριση ονοματιζόμενων οντοτήτων* (*named entity recognition*) πρόκειται για στατιστική τεχνική η οποία στοιχεία μέσα στο κείμενο όπως ανθρώπους, οργανισμούς, ονόματα περιοχών, ορισμένες συντομογραφίες κ.τ.λ.

Η *συσχέτιση* (*coreference*) είναι ο προσδιορισμός όρων (π.χ. ουσιαστικών) που αναφέρονται στο ίδιο αντικείμενο. Για παράδειγμα, όπως αναφέρθηκε προηγουμένως, ο όρος “δάσκαλος” και “δάσκαλοι” είναι δύο όροι που κατά την ανάλυση του κειμένου, πρόκειται για το ίδιο αντικείμενο, δηλαδή τον εκπαιδευτικό. Συνεπώς θα πρέπει να θεωρούνται ως ένα στοιχείο αντί για δύο.

Η *ανάλυση συναισθημάτων* (*Sentiment analysis*) είναι τεχνική με την οποία εξάγονται διάφορες μορφές πληροφορίας σχετικά με τη συμπεριφορά, όπως το συναίσθημα, η άποψη, η διάθεση ή και η συγκίνηση.

Η *ομαδοποίηση εγγράφων* (*document clustering*) είναι η διαδικασία με την οποία προσδιορίζουμε ομάδες από πανομοιότυπα έγγραφα.

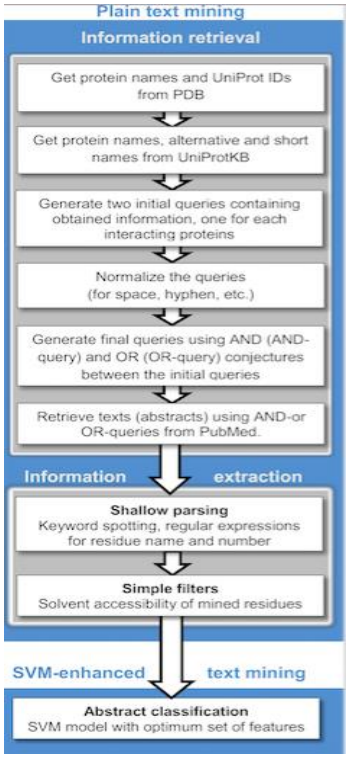
Η *κατηγοριοποίηση* (*categorization*) είναι, τέλος, η διαδικασία επιβλεπόμενης μηχανικής μάθησης, με την οποία ένα κείμενο αντιστοιχείται σε ένα σύνολό προκαθορισμένων θεμάτων, ανάλογα με το περιεχόμενό του.

2.2. ΕΦΑΡΜΟΓΕΣ ΕΞΟΡΥΞΗΣ ΚΕΙΜΕΝΟΥ

Οι τεχνικές και τα εργαλεία εξόρυξης κειμένου, σήμερα, έχουν επηρεάσει αρκετά τη βιομηχανία, βρίσκοντας εφαρμογές από τον ακαδημαϊκό χώρο, την υγειονομική περίθαλψη, μέχρι και τα μέσα κοινωνικής δικτύωσης.

Αρχικά, ας δούμε την εφαρμογή της εξόρυξης κειμένου στα *μέσα κοινωνικής δικτύωσης*. Υπάρχουν πάρα πολλά εργαλεία τα οποία στοχεύουν στην ανάλυση της επίδοσης των μέσων αυτών. Για παράδειγμα, υπάρχουν εργαλεία τα οποία σου επιτρέπουν να δεις τον αριθμό των posts, likes και ακολούθων (followers) της εταιρείας σου, επιτρέποντάς σου να δεις την ανταπόκριση που έχει ο κόσμος για την εταιρεία σου, ή το περιεχόμενο που μπορεί να παρέχεις.

Στο χώρο της *βιοϊατρικής*, η εξόρυξη κειμένου μπορεί να διευκολύνει κλινικές σπουδές ή και την ιατρική ακρίβεια, καθώς μπορούν να ευρετηριάσουν κλινικά συμβάντα σε μεγάλες σειρές δεδομένων από συμπτώματα, παρενέργειες ή και αναφορές από διαγνωστικές εξετάσεις.



Εικόνα 2: παράδειγμα εφαρμογής text mining για τη μελέτη συνδέσεων πρωτεϊνών

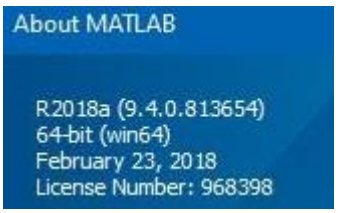
Πηγή: https://en.wikipedia.org/wiki/Text_mining

Πολλές γνωστές εταιρείες που παρέχουν *μηχανές αναζήτησης* (όπως η Google, ή η Yahoo) μπορούν να χρησιμοποιήσουν τις προαναφερθείσες τεχνικές, με σκοπό να βελτιώσουν τον τρόπο αναζήτησης και ευρετηρίασης που χρησιμοποιούν, με σκοπό την καλύτερη ποιότητα των αποτελεσμάτων τους.

Τέλος ακόμα και η ανάλυση συναισθημάτων μπορεί να εφαρμοστεί στο χώρο των *ταινιών*, υπολογίζοντας πόσο καλή είναι μια κριτική για μια συγκεκριμένη ταινία.

3.MATLAB ΚΑΙ TEXT ANALYTICS TOOLBOX

Στην παρούσα πτυχιακή, για κάθε τεχνική εξόρυξης κειμένου που θα αναλυθεί, θα ακολουθεί και ένα μικρό παράδειγμα σε MATLAB. Σε αυτό το κεφάλαιο θα γίνει μία σύντομη επισκόπηση του εργαλείου, καθώς και του toolbox που χρησιμοποιούμε. Η έκδοση η οποία χρησιμοποιήθηκε για την εκτέλεση των παραδειγμάτων είναι η R2018a.



Εικόνα 3: έκδοση MATLAB

Για την εκτέλεση των εφαρμογών που θα υλοποιηθούν, το MATLAB παρέχει ξεχωριστή βιβλιοθήκη, σε διαφορετικό toolbox, το οποίο ονομάζεται text analytics toolbox. Η έκδοση του συγκεκριμένου toolbox που χρησιμοποιείται εδώ, είναι η 1.1.

```
>> help textanalytics
  Text Analytics Toolbox
  Version 1.1 (R2018a) 06-Feb-2018
```

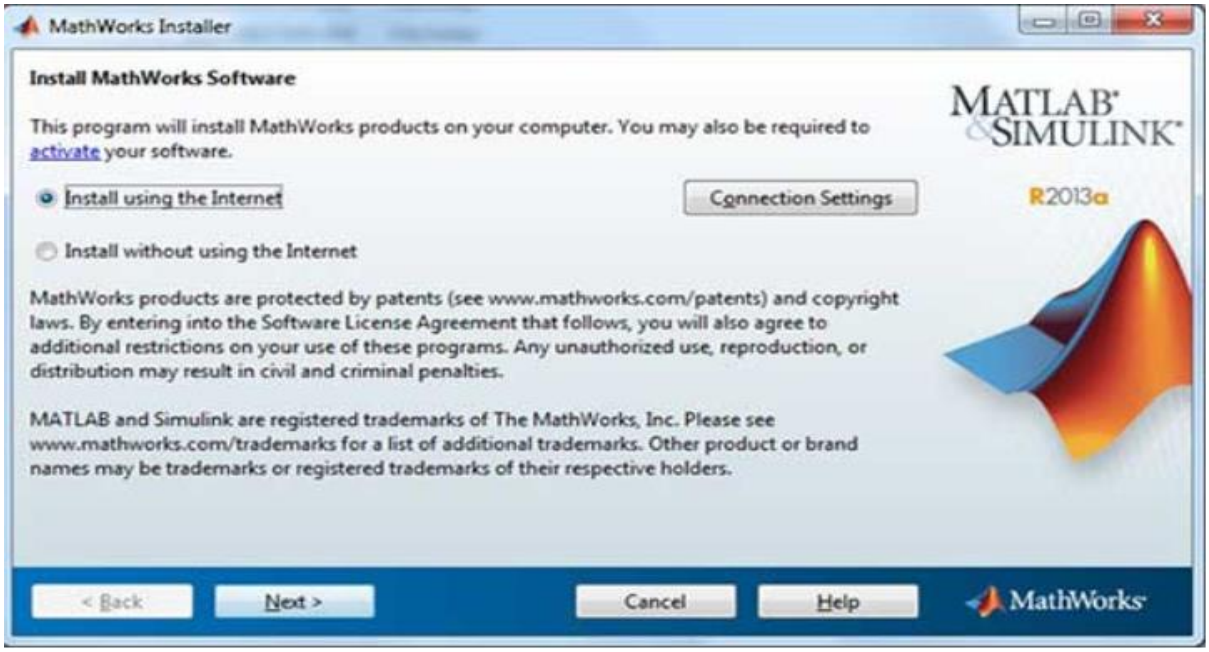
Εικόνα 4: έκδοση text analytics toolbox

3.1. ΣΥΝΤΟΜΗ ΕΙΣΑΓΩΓΗ ΣΤΟ MATLAB

Το MATLAB είναι ένα πρόγραμμα αριθμητικών υπολογισμών βασισμένο στη γραμμική άλγεβρα. Η ονομασία του προέρχεται από τις λέξεις Matrix Laboratory, δηλαδή Εργαστήριο Πινάκων. Το πρόγραμμα αυτό δημιουργήθηκε από την Mathworks και ξεκίνησε ως μια απλή γλώσσα προγραμματισμού πινάκων. Σήμερα, με προσθήκη διαφορετικών toolbox, έχει εξελιχθεί σε ένα ισχυρότατο εργαλείο.

3.1.1. Εγκατάσταση του MATLAB στον υπολογιστή

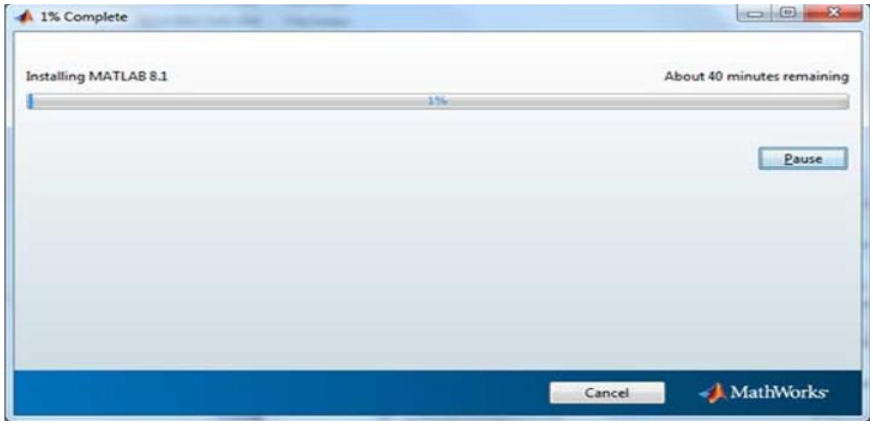
Για να κατεβάσουμε το πρόγραμμα εγκατάστασης πηγαίνουμε στην ιστοσελίδα της Mathworks. Η ιστοσελίδα παρέχει είτε το πλήρες αδειδοτούμενο προϊόν, την δοκιμαστική έκδοση καθώς και μια φοιτητική έκδοση. Για την αγορά άδειας χρήσης του προϊόντος απαιτείται η δημιουργία λογαριασμού στη σελίδα.
Μόλις ολοκληρωθεί η παραπάνω διαδικασία, εκτελούμε το πρόγραμμα εγκατάστασης.



Εικόνα 5: εγκατάσταση του MATLAB - 1

Πηγή: https://www.tutorialspoint.com/matlab/matlab_environment.htm

Στη συνέχεια θα μας ζητηθεί να εισάγουμε το έγγραφο άδειας (licence file) το οποίο αγοράσαμε, και αφού το εισάγουμε, θα οδηγηθούμε στην παρακάτω εικόνα:

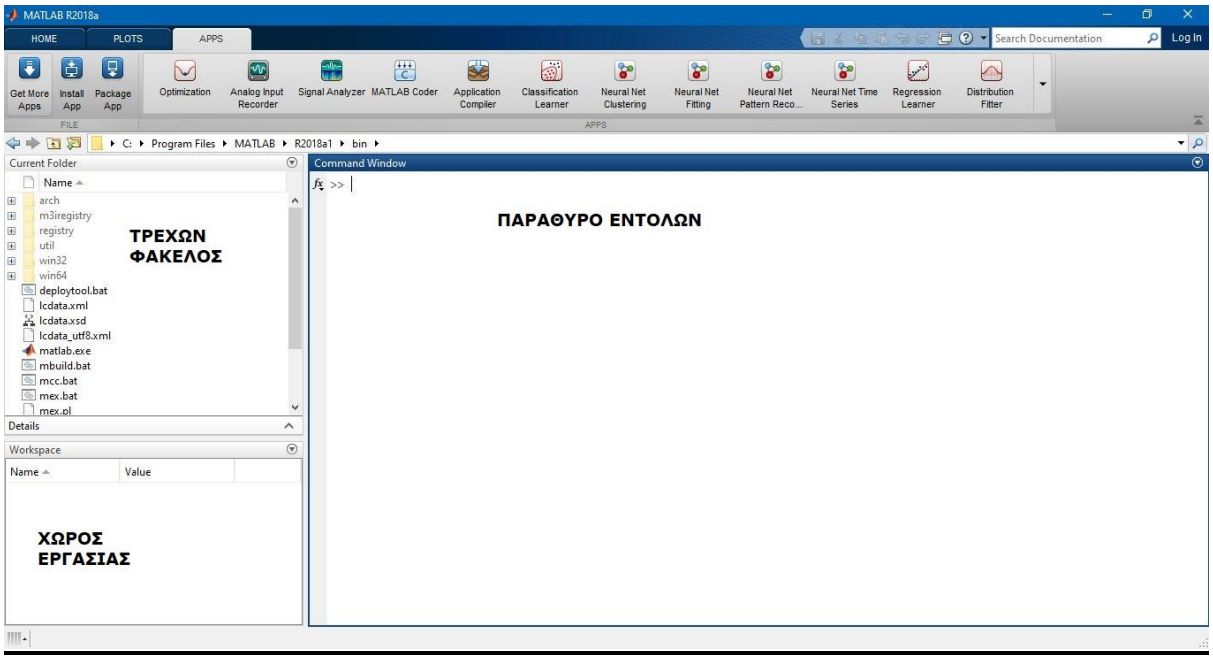


Εικόνα 6: εγκατάσταση του MATLAB - 2

Πηγή: https://www.tutorialspoint.com/matlab/matlab_environment.htm

3.1.2. Το παράθυρο εντολών, ο τρέχων φάκελος και ο χώρος εργασίας

Με την ολοκλήρωση της εγκατάστασης του MATLAB, θα οδηγηθούμε στο κεντρικό παράθυρο της εφαρμογής.



Εικόνα 7: κεντρικό παράθυρο προγράμματος MATLAB

Το παράθυρο εντολών (command window), είναι το παράθυρο στο οποίο πληκτρολογούμε τις εντολές τις οποίες θέλουμε να εκτελέσουμε.

Στο παράθυρο του τρέχοντος φακέλου (current folder), μπορούμε μπορούμε να δούμε τα περιεχόμενα του φακέλου που βλέπει εκείνη τη στιγμή το MATLAB. Η διαδρομή (path) του φακέλου φαίνεται στην μπάρα διεύθυνσης από πάνω. Το παράθυρο αυτό μπορεί να χρησιμεύσει για την καλύτερη οργάνωση των προγραμμάτων μας (scripts).

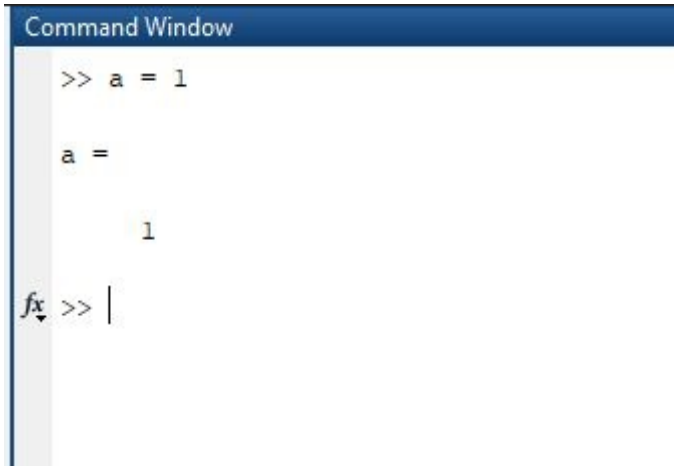
Τέλος, στο χώρο εργασίας (workspace) βρίσκονται όλες οι αποθηκευμένες μεταβλητές και πίνακες που δημιουργήθηκαν από το παράθυρο εντολών.

3.1.3. Ανάθεση μεταβλητών στο MATLAB

Το MATLAB είναι μια γλώσσα προγραμματισμού πινάκων, συνεπώς κάθε μεταβλητή η οποία δημιουργείται εκεί, αντιμετωπίζεται ως πίνακας. Για παράδειγμα, για να δημιουργήσουμε μια απλή μεταβλητή και να της αναθέσουμε την τιμή 1, στο παράθυρο εντολών πρέπει να γράψουμε την εντολή

```
a = 1
```

Το αποτέλεσμα είναι να δούμε στο παράθυρο εντολών το περιεχόμενο της μεταβλητής a.

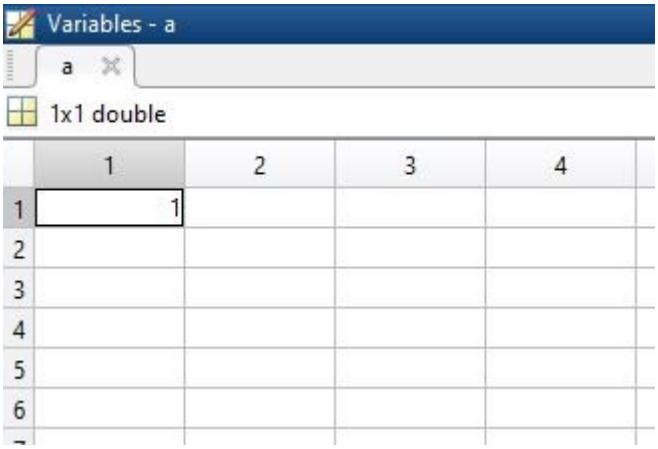


Εικόνα 8: δημιουργία μεταβλητής a

αλλά επιπλέον, μπορούμε να παρατηρήσουμε πως στο χώρο εργασίας εμφανίζεται πλέον η μεταβλητή που μόλις δημιουργήσαμε.

Εικόνα 9: μεταβλητή a στο workspace

Αν κάνουμε διπλό κλικ στη μεταβλητή, θα παρατηρήσουμε πως εμφανίζεται ένα νέο παράθυρο, με τα πλήρη περιεχόμενα της μεταβλητής μας:



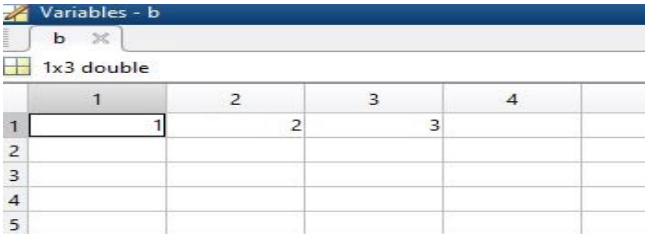
Εικόνα 10: πλήρη περιεχόμενα της μεταβλητής a

Παρατηρούμε δηλαδή πως ακόμα και μια απλή τιμή αποθηκεύεται ως πίνακας 1x1.

Τώρα για να δημιουργήσουμε έναν πίνακα αρκεί στο παράθυρο εντολών να πληκτρολογήσουμε την ακόλουθη εντολή:

```
b = [1 2 3]
```

Το αποτέλεσμα είναι να δημιουργηθεί η μεταβλητή b, η οποία είναι πίνακας 1x3 (1 γραμμή και 3 στήλες). Το παραπάνω μπορούμε να το διακρίνουμε και αν κάνουμε διπλό κλικ στη μεταβλητή στη γραμμή εντολών:



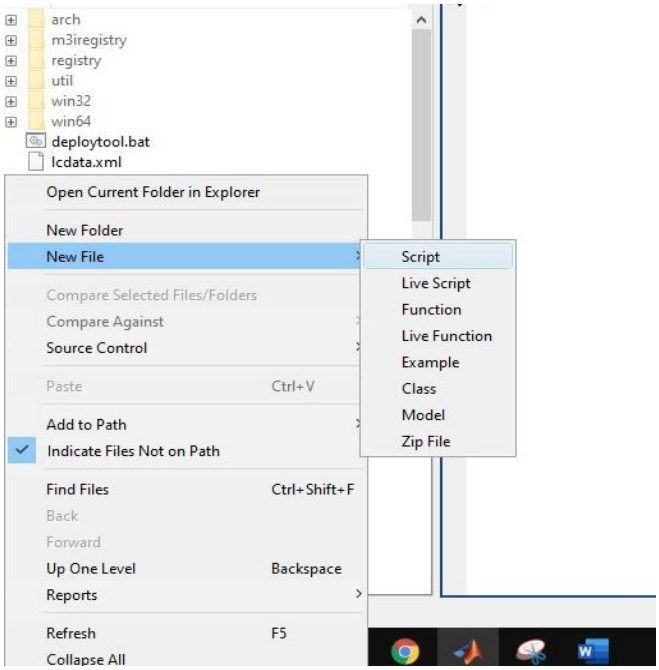
Εικόνα 11: πλήρη περιεχόμενα μεταβλητής b

Με παρόμοιο τρόπο, αν αποθηκεύσουμε σε μεταβλητή μια συμβολοσειρά, αυτό που θα γίνει, είναι να δημιουργηθεί ένας πίνακας 1x1 τύπου string.

3.1.4. Δημιουργία και εκτέλεση αρχείων script

Στο MATLAB μπορούμε να ομαδοποιήσουμε εντολές και να τις εκτελέσουμε όλες μαζί με τη μορφή script. Στα αρχεία script του MATLAB αποθηκεύονται πάντα με την κατάληξη .m.

Εφόσον σιγουρευτούμε πως είμαστε στον σωστό κατάλογο, μπορούμε να κάνουμε δεξί κλικ στο χώρο του τρέχοντος φακέλου, και να επιλέξουμε New File και στη συνέχεια Script.

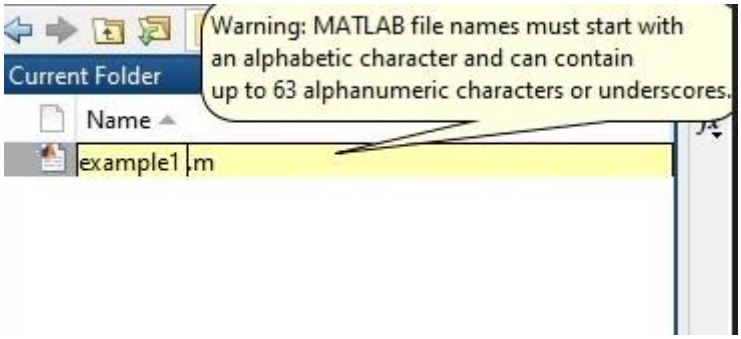


Εικόνα 12: Δημιουργία script

Στον τρέχων φάκελο πλέον φαίνεται το νέο αρχείο που δημιουργήσαμε.

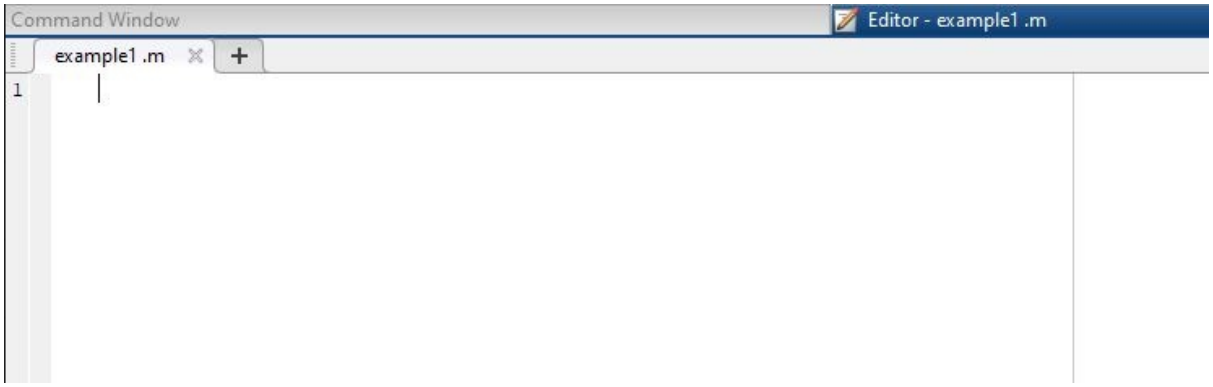
Εικόνα 13: εμφάνιση script στον τρέχων φάκελο

Το αρχείο αυτό τώρα μπορούμε να το μετονομάσουμε. Πρέπει να δοθεί προσοχή στο γεγονός ότι το MATLAB δεν επιτρέπει στα ονόματα των αρχείων του spaces ή άλλους ειδικούς χαρακτήρες. Σε περίπτωση που προσπαθήσουμε να βάλουμε τέτοιο χαρακτήρα, θα εμφανιστεί προειδοποιητικό μήνυμα.



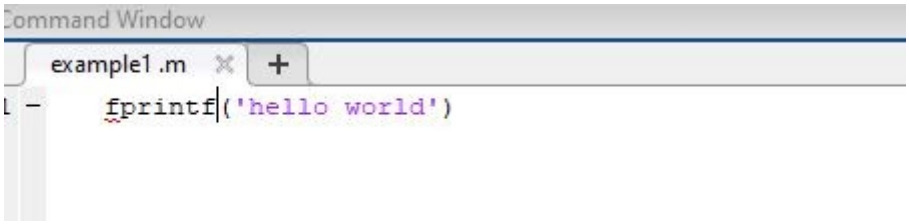
Εικόνα 14: μήνυμα σφάλματος ονόματος

Εάν κάνουμε διπλό κλικ στο αρχείο, θα ανοίξει το παράθυρο του editor στο οποίο μπορούμε να γράψουμε τις εντολές του script.



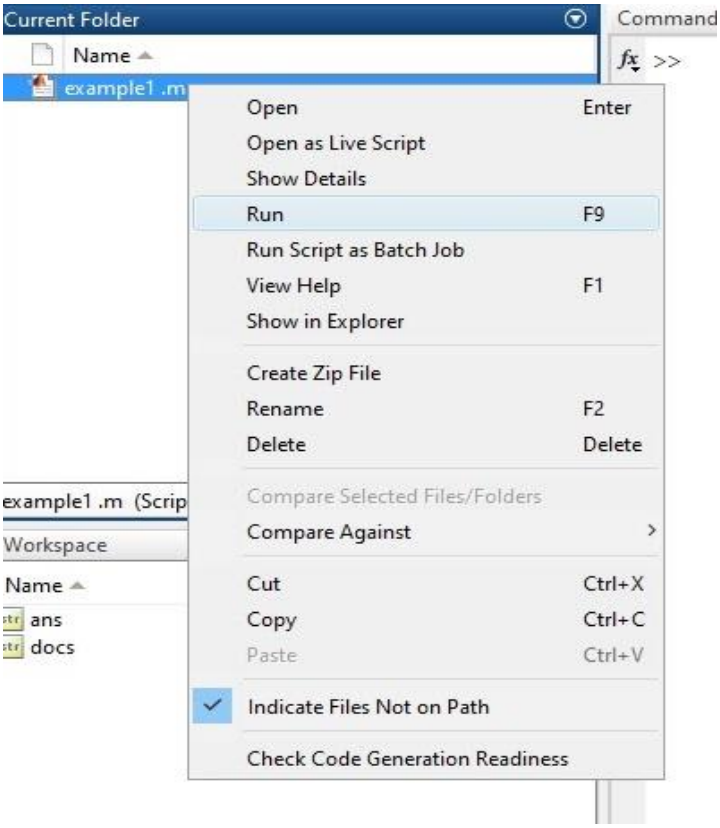
Εικόνα 15: παράθυρο editor

Για τις ανάγκες του παραδείγματος, θα γράψουμε μια απλή εντολή για να εμφανίσουμε το κείμενο “hello world”.



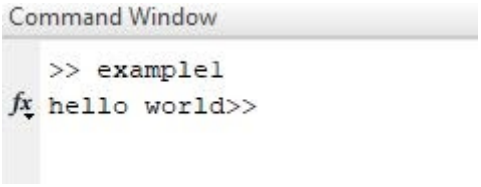
Εικόνα 16: συγγραφή κώδικα στον editor

Για να εκτελέσουμε το script που δημιουργήσαμε, αρκεί να κάνουμε δεξί κλικ στο όνομα του αρχείου που βρίσκεται στον χώρο του τρέχοντος φακέλου, και να πατήσουμε την επιλογή run.



Εικόνα 16: εκτέλεση κώδικα από τον τρέχων φάκελο

Το αποτέλεσμα του script εμφανίζεται στο παράθυρο εντολών.



Εικόνα 17: αποτέλεσμα εκτέλεσης script

3.2. ΤΟ TEXT ANALYTICS TOOLBOX ΤΟΥ MATLAB

Για εκτέλεση εντολών που σχετίζονται με το text mining απαιτείται το text analytics toolbox, το οποίο περιέχει όλες τις συναρτήσεις για χειρισμό τέτοιων λειτουργιών. Για να διαπιστώσουμε ότι όντως έχουμε εγκατεστημένο το toolbox στο παράθυρο εντολών πληκτρολογούμε την εντολή

```
help textanalytics
```

Το αποτέλεσμα της εντολής, σε περίπτωση που έχουμε εγκατεστημένο το toolbox είναι να δούμε μια λίστα με όλες τις διαθέσιμες συναρτήσεις που μας παρέχει.

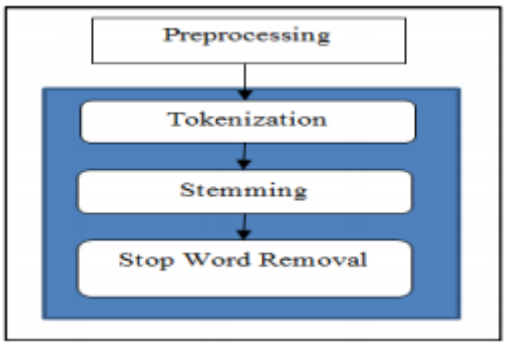
```
>> help textanalytics
  Text Analytics Toolbox
  Version 1.1 (R2018a) 06-Feb-2018

  Input.
    extractFileText     - Extract text from PDF, MSWord, HTML, and plain text files.
    extractHTMLText     - Extract text from HTML.
    readPDFFormData     - Read text from PDF forms.

  Preprocessing.
    eraseTags           - Erase HTML/XML tags from text.
    eraseURLs           - Erase http(s) URLs from text.
    erasePunctuation    - Erase punctuation from text.
    decodeHTMLEntities  - Convert HTML/XML entities into characters.
    tokenizedDocument   - Array of tokenized documents.
    tokenDetails        - Table of token information.
    doc2cell            - Convert document to cell array of strings.
    joinWords           - Convert document to string by joining words.
    context             - Search for word occurrences in context.
    docfun              - Apply function to tokenizedDocument words.
    doclength           - Number of words in document.
    normalizeWords      - Remove inflections using stemming.
    removeLongWords     - Remove long words.
    removeShortWords    - Remove short words.
    removeWords         - Remove words from tokenizedDocument.
    splitSentences      - Split text into sentences.
    stopWords           - List of common words.
    topLevelDomains     - List of common top level internet domain names.
```

Εικόνα 18: εντολή βοήθειας για το text analytics toolbox, περιέχει όλες τις διαθέσιμες συναρτήσεις

4.ΔΙΑΧΩΡΙΣΜΟΣ ΚΕΙΜΕΝΟΥ ΣΕ ΠΡΟΤΑΣΕΙΣ ΚΑΙ ΛΕΞΕΙΣ

Τμηματοποίηση, ή όπως αναγράφεται ο όρος στην αγγλική βιβλιογραφία *tokenization*, είναι ο διαχωρισμός μιας σειράς από προτάσεων σε κομμάτια, τα οποία τα ονομάζουμε τμήματα ή *tokens*. Ως token μπορεί να θεωρηθεί μια λέξη, φράση, ακόμα και ολόκληρη πρόταση.

Στην εξόρυξη κειμένου, η τμηματοποίηση είναι η πρώτη μεγάλη διαδικασία που πρέπει να πραγματοποιηθεί, καθώς κάθε άλλος αλγόριθμος, απαιτεί να διαχειρίζεται μεμονωμένα tokens, και όχι ακατέργαστο κείμενο.



Εικόνα 19: Στάδια προεπεξεργασίας κειμένου

Πηγή: "Advanced Computational Intelligence: An International Journal"

Πολλές φορές, όπως θα δούμε στη συνέχεια, δεν μας είναι όλα τα token χρήσιμα, καθώς μπορεί να μην προσφέρουν καμία απολύτως πληροφορία. Για παράδειγμα, τα σημεία στίξης, όπως θα παρατηρήσουμε και στη συνέχεια του κεφαλαίου, αναγνωρίζονται ως ξεχωριστά tokens, αλλά δεν προσφέρουν πληροφορία. Σε επόμενο κεφάλαιο θα δούμε τρόπους να αφαιρέσουμε τέτοια tokens.

Στο συγκεκριμένο κεφάλαιο θα αναφερθούμε σε δύο βασικούς τρόπους τμηματοποίησης: την τμηματοποίηση *σε προτάσεις* και την τμηματοποίηση *σε λέξεις*.

4.1. ΔΙΑΧΩΡΙΣΜΟΣ ΚΕΙΜΕΝΟΥ ΣΕ ΠΡΟΤΑΣΕΙΣ

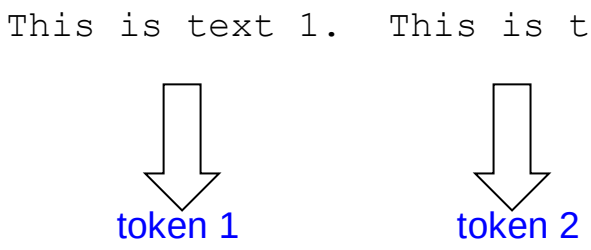
Η πρώτη διαδικασία με την οποία θα ασχοληθούμε είναι αυτή της τμηματοποίησης ενός κειμένου, σε προτάσεις. Αυτό που θέλουμε εδώ να πραγματοποιήσουμε είναι να διαχωρίσουμε ένα κείμενο στις επιμέρους προτάσεις του.

Για παράδειγμα, ας δούμε ένα απλό σύνολο προτάσεων:

```
This is text 1.  This is text 2.
```

Ας παρατηρήσουμε αρχικά, πως η κάθε πρόταση λήγει με ένα σημείο στίξης. Στο παράδειγμα μας είναι η τελεία, αλλά μπορεί να είναι οποιαδήποτε άλλο σημείο στίξης που υποδηλώνει τέλος πρότασης, όπως για παράδειγμα το θαυμαστικό.

Σε αυτήν την περίπτωση, η κάθε πρόταση ξεχωριστά, θεωρείται ως token.



Εικόνα 20: τμηματοποίηση κειμένου σε προτάσεις

Συνεπώς, τα τελικά μας tokens εδώ, θα έχουν ως εξής:

Token 1: This is text 1.

Token 2: This is text 2.

Η μόνη εξαίρεση στον κανόνα, είναι στην περίπτωση συντομογραφιών. Για παράδειγμα, αν έχω την εξής πρόταση:

```
Dr. Smith.
```

Παρόλο που έχει 2 τελείες, η πρώτη αφορά την συντομογραφία Dr. Συνεπώς αυτό το κείμενο θα πρέπει να το θεωρήσουμε ως ένα token και όχι δύο.

4.1.1. Τμηματοποίηση σε προτάσεις στο MATLAB

Στο MATLAB ο διαχωρισμός σε προτάσεις πραγματοποιείται με την βοήθεια της συνάρτησης `splitSentences`. Η συνάρτηση αυτή δέχεται σαν όρισμα μια συμβολοσειρά, και επιστρέφει ένα διάνυσμα με τις διαχωρισμένες προτάσεις. Για να μπορέσουμε να δούμε πληροφορίες στο MATLAB για αυτή την εντολή αρκεί να πληκτρολογήσουμε την εντολή `help splitSentences`

Παρατηρούμε πως το MATLAB θα μας επιστρέψει το ακόλουθο αποτέλεσμα:

```
>> help splitSentences
 splitSentences Split text into sentences
    NEWSTR = splitSentences(STR) splits the sentences from STR and returns
    the vector NEWSTR.
    STR can be a scalar string, character vector or scalar cell array containing
    a character vector.

    Examples
       sent = splitSentences("Sentence one. Sentence two.");
         returns ["Sentence one." ; "Sentence two."]

    See also extractBetween

    Reference page for splitSentences

>>
```

Εικόνα 21: Βοήθεια στο MATLAB για την εντολή splitSentences

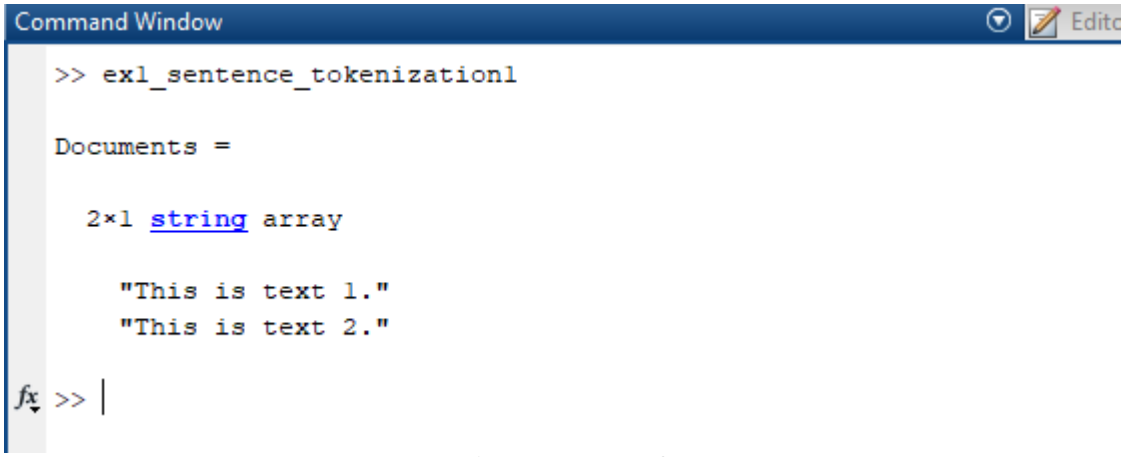
Αυτό που θα προσπαθήσουμε τώρα, είναι να γράψουμε τμήμα κώδικα στο MATLAB το οποίο με χρήση αυτής της συνάρτησης, θα επιχειρήσει να διασπάσει το κείμενο "This is text 1. This is text 2." όπως αναλύσαμε παραπάνω:

```
% example 1 - Single text
textData = "This is text 1.  This is text 2.";
Documents = splitSentences(textData) %split text using
                                      splitSentences
```

Το τμήμα κώδικα αυτό θα το αποθηκεύσουμε στον τρέχων φάκελο ως *ex1_sentence_tokenization1.m.*

Αρκεί να παρατηρήσουμε πως στην μεταβλητή textData, θα καταχωρήσουμε το κείμενό μας, και στη συνέχεια καλούμε την splitSentences για να το διαχωρίσουμε. Το αποτέλεσμα θα είναι ένα διανυσμα, το οποίο θα αποτελείται από τις δύο μεμονωμένες προτάσεις.

Για να μπορέσουμε να δούμε το αποτέλεσμα στο MATLAB αρκεί να πληκτρολογήσουμε το όνομα του .m αρχείου που δημιουργήσαμε χωρίς την κατάληξη:



Εικόνα 22: Εκτέλεση πρώτου παραδείγματος στο MATLAB

Θα παρατηρήσουμε και από το MATLAB πως το Documents είναι στην ουσία ένας 2x1 πίνακας που περιέχει τις δύο προτάσεις που διαχωρίσαμε.

4.1.2: Τμηματοποίηση στην περίπτωση συντομογραφιών

Αναφέρθηκε προηγουμένως, πως στην ουσία αυτό που κάνουμε είναι να κοιτάμε για σημεία στίξης στο κείμενο που υποδηλώνουν τέλος πρότασης. Σε αυτόν τον κανόνα υπάρχει μία εξαίρεση: οι συντομογραφίες. Για παράδειγμα έστω ότι έχουμε το ακόλουθο κείμενο:

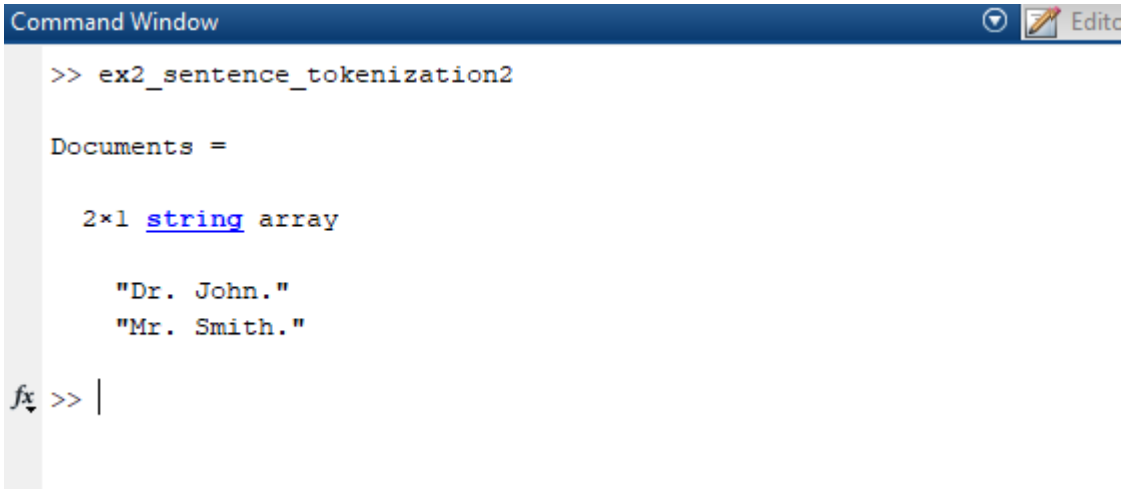
```
Dr. John. Mr. Smith.
```

Το ερώτημα που δημιουργείται εδώ είναι το εξής: πώς θα ξεχωρίσουμε ότι πρόκειται για δύο προτάσεις και όχι τέσσερις, παρόλο που χρησιμοποιείται 4 φορές ο χαρακτήρας *τελεία* (“.”);

Ας προσπαθήσουμε να περάσουμε το κείμενο αυτό από την splitSentences, για να δούμε πώς θα το διαχειριστεί το MATLAB:

```
% example 2 - Single text with Abbreviations
textData = "Dr. John. Mr. Smith.";
Documents = splitSentences(textData) %split different text
                                      using splitSentences
```

Θα αποθηκεύσουμε το script αυτό σε αρχείο με όνομα *ex2_sentence_tokenization2.m*. Για να το εκτελέσουμε θα πληκτρολογήσουμε και πάλι το όνομά του, χωρίς την κατάληξη, όπως πραγματοποιήσαμε προηγουμένως:



Εικόνα 23: Εκτέλεση παραδείγματος 2 στο MATLAB

Μπορούμε να παρατηρήσουμε πως η συνάρτηση splitSentences μπορεί να αναγνωρίσει συντομογραφίες κανονικά, χωρίς κανένα πρόβλημα. Συνεπώς, δεν θα χρειαστεί να ασχοληθούμε με την εξαίρεση αυτή, καθώς έχει ήδη συμπεριληφθεί κατά τον προγραμματισμό της προαναφερθείσας συνάρτησης.

4.2. ΔΙΑΧΩΡΙΣΜΟΣ ΚΕΙΜΕΝΟΥ ΣΕ ΛΕΞΕΙΣ

Αυτό που θα δούμε τώρα είναι το πρώτο και πιο σημαντικό στάδιο της προ-επεξεργασίας κειμένου στην εξόρυξη κειμένου: την κλασική τμηματοποίηση σε λέξεις.
Για να μπορούμε να ασχοληθούμε με τα υπόλοιπα στάδια της επεξεργασίας του κειμένου, και στη συνέχεια στην εξαγωγή πληροφοριών, είναι προφανές ότι δεν μπορούμε να διαχειριστούμε ολόκληρα τμήματα κειμένου, ακόμα και αν είναι προτάσεις ή και απλές φράσεις. Χρειάζεται να πάμε ένα βήμα παραπέρα. Αυτό που θέλουμε να κάνουμε είναι να διαχωρίσουμε οποιοδήποτε κείμενο στις επιμέρους λέξεις του αναγνωρίζοντας κάθε φορά τον χαρακτήρα space (κενό) ή σημεία στίξης. Είναι προφανές ότι τα σημεία στίξης θα πρέπει να θεωρούνται από μόνα τους ως ξεχωριστά token. Δεν τα θέλουμε μαζί με άλλη λέξη διότι πρόκειται για περιττή πληροφορία, καθώς επίσης και μπορεί να παρεμποδίσει περεταίρω διαδικασίες όπως θα δούμε παρακάτω.

Ας δούμε πώς θα πραγματοποιήσουμε την αναφερθείσα διαδικασία για μια απλή πρόταση όπως η παρακάτω:

```
This is an example.
```

Κάθε token σε αυτή την πρόταση διαχωρίζεται από το επόμενό του με την ανίχνευση του χαρακτήρα space. Στην τελευταία λέξη, θα παρατηρήσουμε πως μετά την λέξη example ακολουθεί ο χαρακτήρας “τελεία”, συνεπώς θα θεωρηθούν ως δύο ξεχωριστά tokens.

Το αποτέλεσμα αυτής της διαδικασίας, είναι να έχουμε 5 συνολικά tokens, ως εξής:
Tokens (5): ***"This"***, ***"is"***, ***"an"***, ***"example"***, ***"."***

4.2.1. Τμηματοποίηση σε λέξεις στο MATLAB

Το MATLAB για την διαδικασία της τμηματοποίησης, μας παρέχει μια ξεχωριστή συνάρτηση, την `tokenizedDocument`, η οποία περιλαμβάνεται στο text analytics toolbox. Η συνάρτηση αυτή, δέχεται ένα έγγραφο ως συμβολοσειρά ή έναν πίνακα εγγράφων, και τα διαχωρίζει σε επιμέρους λέξεις. Για να δούμε πληροφορίες στο MATLAB σχετικά με αυτή τη συνάρτηση, αρκεί να πληκτρολογήσουμε στο παράθυρο εντολών, το ακόλουθο:

```
help tokenizedDocument
```

Το αποτέλεσμα που θα επιστρέψει το MATLAB φαίνεται παρακάτω:

```
>> help tokenizedDocument
 tokenizedDocument Array of tokenized documents
    A tokenized document is a document represented as a collection of words (also
    known as tokens) which is used for analysis. tokenizedDocument arrays allows
    you to perform word-level processing tasks such as removing words using
    removeWords and stemming words using normalizeWords.

    DOCUMENTS = tokenizedDocument creates a scalar tokenized document with no
    tokens.

    DOCUMENTS = tokenizedDocument(STR) tokenizes the elements of STR and returns
    an array of tokenized documents.

    DOCUMENTS = tokenizedDocument(STR,Name,Value) specifies additional options
    using one or more name-value pair arguments.
```

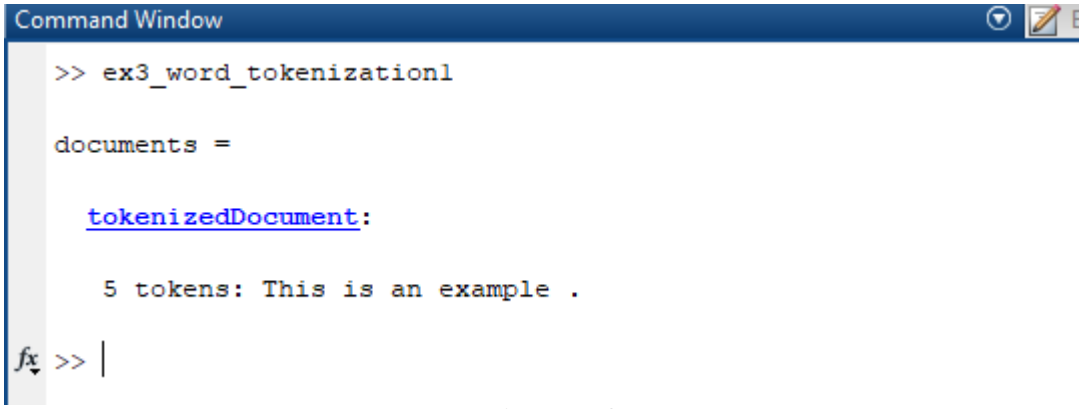
Εικόνα 24: Βοήθεια στο MATLAB για την εντολή splitSentences

Για αρχή, αυτό που θα επιχειρήσουμε είναι να γράψουμε script στο MATLAB το οποίο πραγματοποιεί την τμηματοποίηση της πρότασης που χρησιμοποιήσαμε παραπάνω ("This is an example.").

```
% example 3 - Single text, tokenize words
textData = "This is an example.";
documents = tokenizedDocument(textData) % tokenize words using
                                        tokenizedDocument
```

Θα αποθηκεύσουμε το παραπάνω σε αρχείο με όνομα *ex3_word_tokenization1.m*.

Αν εκτελέσουμε τώρα το παραπάνω πρόγραμμα θα παρατηρήσουμε το εξής αποτέλεσμα:



Εικόνα 25: Εκτέλεση παραδείγματος 3 στο MATLAB

Το MATLAB δηλαδή μας αναφέρει αρχικά το πλήθος των tokens που ανακάλυψε στο έγγραφο, ακολουθούμενο από ένα-ένα τα tokens που περιλαμβάνει. Την μεταβλητή αυτή θα την χρησιμοποιήσουμε για οποιαδήποτε περαιτέρω λειτουργία θα χρειαστεί να κάνουμε πάνω στα tokens του εγγράφου. Με άλλα λόγια, δεν μας ενδιαφέρει πλέον το περιεχόμενο της μεταβλητής textData, παρά μόνο το documents.

4.2.2: Τμηματοποίηση σε πολλαπλά έγγραφα

Όπως παρατηρήσαμε, για όλες τις διαδικασίες που θα χρειαστούμε, στο MATLAB περιλαμβάνεται κάποια έτοιμη συνάρτηση η οποία κάνει την ίδια διαδικασία, αυτοματοποιημένη. Τώρα, στην περίπτωση της τμηματοποίησης σε λέξεις, η tokenizedDocument θα μπορέσει να λειτουργήσει και στην περίπτωση που της δώσουμε πολλαπλά έγγραφα. Τα έγγραφα αυτά δίνονται στη συνάρτηση με τη μορφή ενός διανύσματος, το οποίο περιέχει ως συμβολοσειρές τα έγγραφά μας.

Ας δούμε τώρα πώς συμπεριφέρεται η συνάρτηση, στην περίπτωση που της δώσουμε να τμηματοποιήσει δύο διαφορετικά έγγραφα:

"This is line 1"
"This is line 2"

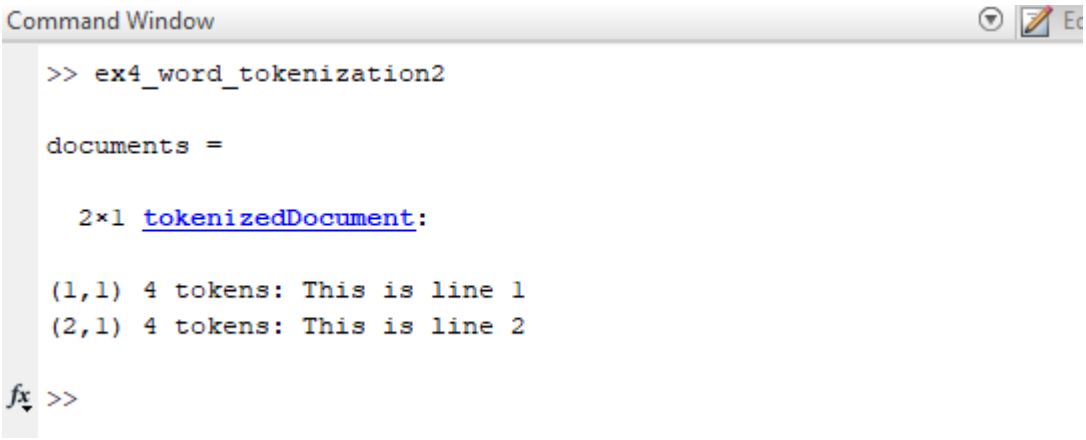
Για τον σκοπό αυτό δημιουργούμε στο MATLAB το ακόλουθο script:

```
% example 4 - multiple sentences, tokenize words
textData = [
            "This is line 1",
            "This is line 2"
           ];
```

```
documents = tokenizedDocument(textData) % tokenize words using
                                          tokenizedDocument
```

Το script αυτό θα το αποθηκεύσουμε σε αρχείο με όνομα *ex4_word_tokenization2.m*.

Αν αυτό το εκτελέσουμε στο MATLAB θα πάρουμε το εξής αποτέλεσμα:



Εικόνα 26: Εκτέλεση παραδείγματος 4 στο MATLAB

Μπορούμε να παρατηρήσουμε δηλαδή, πως το MATLAB πραγματοποιεί τμηματοποίηση *ανά έγγραφο*, δηλαδή κάθε έγγραφο (δηλαδή συμβολοσειρά μέσα στο διάνυσμα) τμηματοποιείται ξεχωριστά, και για κάθε ένα, μας επιστρέφεται ο αριθμός των token του, καθώς και τα token που περιλαμβάνει.

4.2.3: Προβολή λεπτομερειών κάθε τμήματος

Για να πάμε ένα βήμα παρακάτω, εφόσον έχουμε εξάγει τα διαφορετικά tokens που υπάρχουν σε κάθε έγγραφο, μπορεί να θέλουμε να ανακαλύψουμε αν πρόκειται για λέξη, αριθμό ή και σημείο στίξης. Αυτό μπορεί να πραγματοποιηθεί με χρήση της συνάρτησης `tokenDetails`. Η συνάρτηση αυτή δέχεται ώς όρισμα τον πίνακα των token που εξάγαμε προηγουμένως και αναγράφει λεπτομερώς, για κάθε token πληροφορίες όπως σε ποιό έγγραφο βρέθηκε, σε ποιά γραμμή του εγγράφου αυτού, καθώς και τι τύπος είναι (π.χ. λέξη, αριθμός, σημείο στίξης κ.λ.π.).

Για να δούμε πληροφορίες στο MATLAB για την εντολή αυτή πληκτρολογούμε την εντολή

```
help tokenDetails
```

Το αποτέλεσμα που θα δώσει το MATLAB φαίνεται παρακάτω:

```
>> help tokenDetails
--- help for tokenizedDocument/tokenDetails ---

 tokenDetails token details
    TBL = tokenDetails(DOCUMENTS) returns a table of token details for each
    token in each document in the tokenizedDocument array DOCUMENTS.

    Reference page for tokenizedDocument/tokenDetails

>>
```

Εικόνα 27: Βοήθεια στο MATLAB για την εντολή tokenDetails

Για να κατανοήσουμε την χρήση της συνάρτησης θα γράψουμε το ακόλουθο script, για να βρούμε πληροφορίες για τα tokens των δύο εγγράφων που αναλύσαμε προηγουμένως:

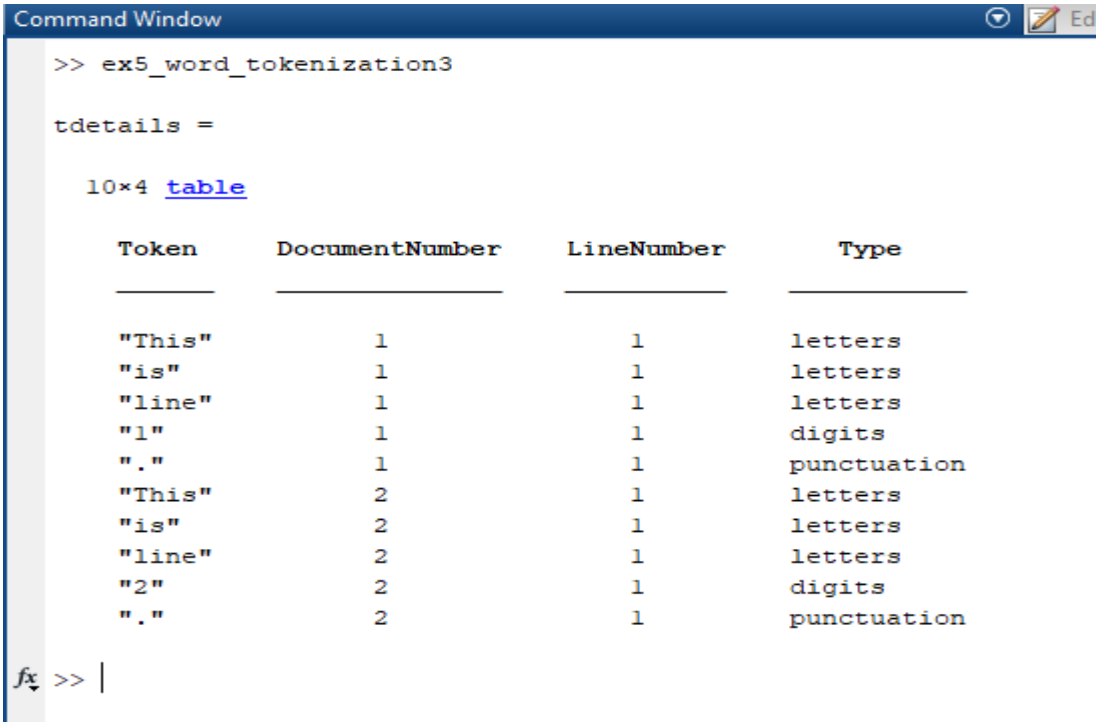
```
% example 5 - multiple sentences, view token details
textData = [
            "This is line 1.",
            "This is line 2."
           ];
documents = tokenizedDocument(textData); % tokenize words
                                    using tokenizedDocument
tdetails = tokenDetails(documents)% view details foreach token
```

Αρκεί δηλαδή, στον κώδικα του παραδείγματος 4 να προσθέσουμε την εντολή

```
tdetails = tokenDetails(documents)
```

Θα αποθηκεύσουμε το script σε αρχείο με όνομα *ex5_word_tokenization3.m*.

Αν εκτελέσουμε το παράδειγμα, θα παρατηρήσουμε επακριβώς τις λεπτομέρειες του κάθε token, στα δύο έγγραφα που έχουμε όπως φαίνεται παρακάτω:



Εικόνα 28: Εκτέλεση παραδείγματος 5 στο MATLAB

5.ΚΑΝΟΝΙΚΟΠΟΙΗΣΗ ΤΜΗΜΑΤΩΝ

Στο στάδιο που βρισκόμαστε, έχουμε ένα σύνολο από tokens για κάθε έγγραφο, αλλά αυτά δεν είναι ακόμα έτοιμα για επεξεργασία. Ο λόγος είναι ότι συνήθως μπορεί να υπάρχουν tokens που περιέχουν παρόμοια έως την ίδια σημασία.

Ένα παράδειγμα, είναι στην περίπτωση των πεζών και κεφαλαίων χαρακτήρων: Πολλές γλώσσες προγραμματισμού κάνουν διάκριση μεταξύ πεζών και κεφαλαίων χαρακτήρων, συνεπώς αν είχαν για παράδειγμα τα tokens “text” και “Text”, θα τα θεωρούσαν ως δύο ξεχωριστά tokens και όχι ένα, παρόλο που ουσιαστικά πρόκειται για την ίδια πρόταση.

Ένα άλλο παράδειγμα, είναι για την περίπτωση διαφορετικών λέξεων με παρόμοιο νόημα. Ας πάρουμε π.χ. τα tokens “reading” και “read”. Μπορούμε να παρατηρήσουμε πως και τα δύο tokens, έχουν ουσιαστικα το ίδιο νόημα: το διάβασμα. Συνεπώς και εδώ, δεν υπάρχει λόγος να τα θεωρήσουμε ως ξεχωριστά tokens, αλλά ως ένα.

Στο κεφάλαιο αυτό θα μελετήσουμε τους διάφορους τρόπους που μπορούμε να χρησιμοποιήσουμε για να κανονικοποιήσουμε ένα σύνολο από tokens.

5.1: STEMMING

Stemming ονομάζεται η διαδικασία με την οποία μεταφέρουμε παράγωγες λέξεις πίσω στη ρίζα τους. Για παράδειγμα, ας πάρουμε τις λέξεις **“likely”**, **“liking”**, **“liked”**, **“likes”**. Μπορούμε ήδη να παρατηρήσουμε πως οι τέσσερις λέξεις αυτές έχουν την ίδια ρίζα: **“like”**.

Η διαδικασία αυτή είναι πολύ χρήσιμη στην εξόρυξη κειμένου, καθώς μας επιτρέπει να μεταφέρουμε τα tokens, στην αρχική τους ρίζα, επιτρέποντας μας, να μην έχουμε παράγωγες λέξεις ως διαφορετικά token.

Η πιο διάσημη τεχνική stemming, είναι αυτή του porter stemming, η οποία είναι απλά μια τεχνική αφαίρεσης καταλήξεων. Ο αλγόριθμος ακολουθεί μια σειρά από βήματα για να παράγει την ρίζα μιας οποιασδήποτε λέξης. Για να κατανοήσουμε το πώς λειτουργεί ο αλγόριθμος, ας δούμε ένα παράδειγμα για την λέξη **“hopefulness”**.

- Βήμα 1α: Εξετάζουμε αν πρόκειται για πληθυντικός αριθμός.
  Στην περίπτωση μας δεν μπορούμε να εξετάσουμε για ενικό ή πληθυντικό.
  Συνεπώς το βήμα αυτό παραλείπεται.
- Βήμα 1β: Εξετάζουμε αν πρόκειται για πληθυντικό αριθμό.
  Για τον ίδιο λόγο με αυτόν του προηγούμενου βήματος, θα το παραλείψουμε και αυτό.
- Βήμα 2: Μετατροπή σε επίθετο.
  Στη δική μας περίπτωση, για να το πετύχουμε αυτό, αφαιρούμε την κατάληξη “ness”, μετατρέποντας τη λέξη σε “hopeful”.
- Βήμα 3: Μετατροπή σε ουσιαστικό.
  Στη δική μας περίπτωση, αφαιρούμε την κατάληξη “ful”, όπου και καταλήγουμε στην ρίζα της αρχικής λέξης, δηλαδή το “hope”.

Στο MATLAB για να εφαρμόσουμε τον αλγόριθμο του stemming στα tokens χρησιμοποιούμε τη συνάρτηση `normalizeWords`. Η συνάρτηση αυτή, όπως και κάθε άλλη που θα δούμε στη συνέχεια, δέχεται ως όρισμα τον πίνακα των tokens, και εφαρμόζει τον αλγόριθμο porter stemming για την κανονικοποίησή τους.
Για να δούμε λεπτομέρειες για την συνάρτηση στο MATLAB πληκτρολογούμε την εντολή:

```
help normalizeWords
```

Το αποτέλεσμα της εντολής φαίνεται παρακάτω:

```
>> help normalizeWords
 normalizeWords Reduce words to common stems using the Porter stemmer
    normalizeWords uses the Porter stemmer to group different forms of English
    words by reducing them to a common stem. This common stem is not necessarily
    a proper English word.

    newDocuments = normalizeWords(DOCUMENTS) stems each word in DOCUMENTS using
    the Porter stemmer.

    newWords = normalizeWords(WORDS) stems each word in WORDS.
```

Εικόνα 29: Βοήθεια στο MATLAB για την εντολή normalizeWords

Για να εξετάσουμε πρακτικά την χρήση της συνάρτησης, θα γράψουμε τον παρακάτω κώδικα:

```
% example 6 - multiple sentences, stem tokens
textData = [
            "A collection of words.",
            "Another collection of words."
           ];
documents = tokenizedDocument(textData); % tokenize words
                                    using tokenizedDocument
stDocuments = normalizeWords(documents)  % stemming using
                                      normalizeWords
```

Η διαδικασία και πάλι είναι απλή. Απλά περνάμε τον πίνακα των tokens στην συνάρτηση normalizeWords η οποία θα αφαιρέσει τις καταλήξεις από κάθε ένα token. Θα αποθηκεύσουμε το script σε αρχείο με όνομα *ex6_stemming.m*.

Αν το εκτελέσουμε στο MATLAB θα έχουμε το ακόλουθο αποτέλεσμα:

```
Command Window                                          Editor - e
  >> ex6_stemming

  stDocuments =

    2×1 tokenizedDocument:

  (1,1) 5 tokens: a collect of word .
  (2,1) 5 tokens: anoth collect of word .

fx >>
```

Εικόνα 30: Εκτέλεση παραδείγματος 6 στο MATLAB

Παρατηρούμε δηλαδή πως για κάθε έγγραφο ξεχωριστά, πραγματοποιήθηκε ο αλγόριθμος stemming σε κάθε token. Αξίζει να παρατηρήσουμε τα εξής:

- Για τη λέξη "words", αφαιρέθηκε ο πληθυντικός και μετατράπηκε σε "word".
- Για την λέξη "collection" αφαιρέθηκε η κατάληξη -ion έτσι ώστε να μετατραπεί σε ρήμα, δηλαδή την λέξη "collect".

5.2. ΜΕΤΑΤΡΟΠΗ ΣΕ ΚΕΦΑΛΑΙΑ ΚΑΙ ΠΕΖΑ

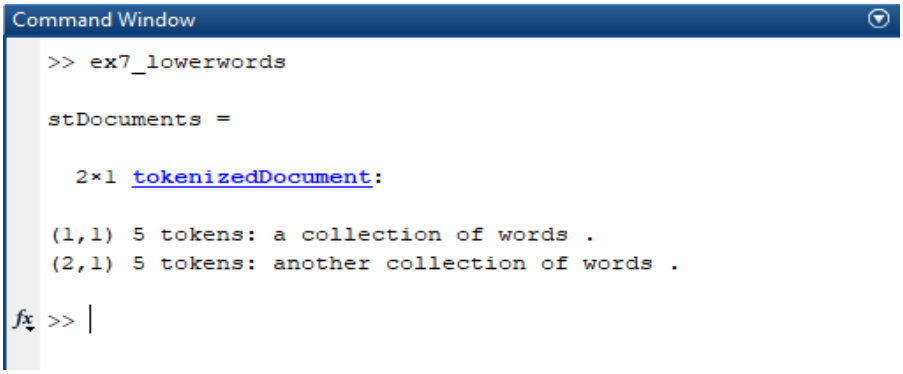
Για την ομαλή λειτουργία του αλγορίθμου κανονικοποίησης που αναφέρθηκε προηγουμένως, είναι πριν εφαρμόσουμε τον αλγόριθμο του stemming, να μετατρέψουμε όλα τα γράμματα των token σε κεφαλαία ή πεζά. Αυτό είναι απαραίτητο επειδή, όπως προαναφέρθηκε, οι γλώσσες προγραμματισμού, όπως και το MATLAB, κάνουν διάκριση κεφαλαίων-πεζών. Εμείς δεν θέλουμε να διαχειρίζονται ως δύο πολλαπλά token αλλά ένα.
Για να το πετύχουμε αυτό, αποφασίζουμε αν θέλουμε να μετατρέψουμε τα tokens σε κεφαλαία, ή πεζά. Για μετατροπή σε κεφαλαία θα χρησιμοποιήσουμε την συνάρτηση `lower` για μετατροπή σε πεζά, ενώ την `upper` για μετατροπή σε κεφαλαία.

Ας δούμε αρχικά ένα παράδειγμα μετατροπής σε πεζά γράμματα.

```
% example 7 - multiple sentences, make words lowercase
textData = [
            "A Collection of Words.",
            "Another Collection of Words."
           ];
documents = tokenizedDocument(textData); % tokenize words
                                          using tokenizedDocument
stDocuments = lower(documents)  % make all words on documents
                                  lowercase
```

Στην ουσία αυτό που κάνουμε, είναι αφού δημιουργήσουμε τον πίνακα των tokens, τον περνάμε στην συνάρτηση lower για να μετατρέψουμε κάθε γράμμα σε πεζό.
Θα αποθηκεύσουμε το script σε αρχείο με όνομα *ex7_lowerwords.m*.

Αν εκτελέσουμε το παράδειγμα στο MATLAB θα παρατηρήσουμε ότι όντως όλα τα tokens μετατρέπονται σε πεζά:



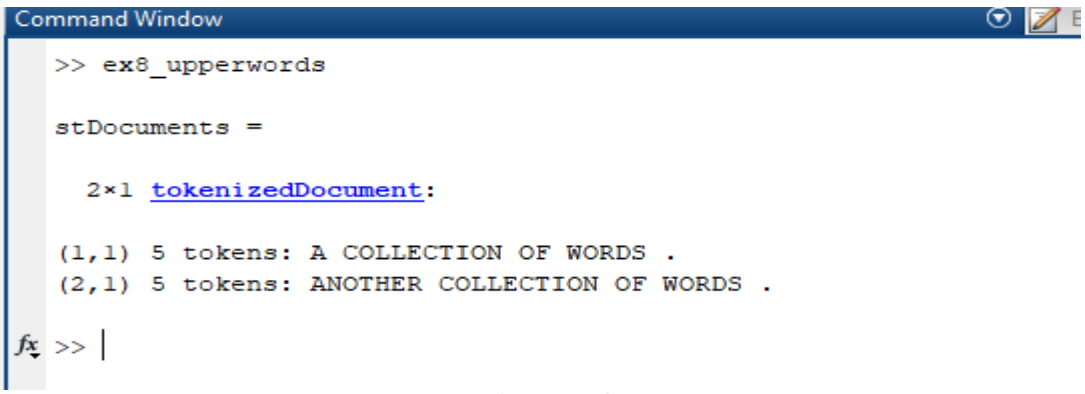
Εικόνα 31: Εκτέλεση παραδείγματος 7 στο MATLAB

Μια λιγότερο συχνή τεχνική, είναι η μετατροπή σε κεφαλαία. Στο προηγούμενο παράδειγμά μας, αν αντικαταστήσουμε την συνάρτηση lower σε upper θα παρατηρήσουμε πως κάθε token γράφεται με κεφαλαία γράμματα.

```
% example 8 - multiple sentences, make words uppercase
textData = [
            "A Collection of Words.",
            "Another Collection of Words."
           ];
documents = tokenizedDocument(textData); % tokenize words
                                          using tokenizedDocument
stDocuments = upper(documents)  % make all words on documents
                                  uppercase
```

Θα αποθηκεύσουμε το script σε αρχείο με όνομα *ex8_upperwords.m*.

Αν εκτελέσουμε το αρχείο θα παρατηρήσουμε πως πλέον όλοι οι πεζοί χαρακτήρες που περιέχονταν στα tokens έχουν μετατραπεί σε κεφαλαίους.



Εικόνα 32: Εκτέλεση παραδείγματος 8 στο MATLAB

5.3. ΑΝΤΙΚΑΤΑΣΤΑΣΗ ΛΕΞΕΩΝ

Στην αγγλική γλώσσα, πολλές φορές δύο λέξεις συνενώνονται ως μία με χρήση αποστρόφου. Για παράδειγμα οι λέξεις “have”, “not” συνενώνονται ως “haven’t”. Πολλές φορές επιθυμούμε εμείς, να μην έχουμε τέτοιες λέξεις στα tokens, οπότε μια καλή πρακτική είναι να τα αντικαταστήσουμε.

Κάποιες από τις λέξεις που πετυχαίνουμε στην αγγλική γλώσσα είναι οι εξής:

- ain’t → μετατρέπεται σε is not
- isn’t → μετατρέπεται σε is not
- aren’t → μετατρέπεται σε are not
- i’m → μετατρέπεται σε i am
- i’ve → μετατρέπεται σε i have

Αυτό που θέλουμε εμείς να κάνουμε είναι να σπάσουμε τις λέξεις αυτές στις επιμέρους αρχικές τους, έτσι ώστε αντί για ένα token να καταλήξουμε σε δύο. Ένας από τους κύριους λόγους που το κάνουμε αυτό, είναι επειδή οι λέξεις με ένα έως τρία γράμματα, αφαιρούνται από τα tokens επειδή δεν προσφέρουν πληροφορία. Προφανώς η σύντομη μορφή, μπορούμε να παρατηρήσουμε πως σχεδόν σε κάθε περίπτωση θα είναι πάνω από τρεις χαρακτήρες. Επομένως δεν πρόκειται να περάσει από το στάδιο της αφαίρεσης tokens (το οποίο θα περιγραφεί σε επόμενο κεφάλαιο).

Για να υλοποιήσουμε σε MATLAB την προαναφερθείσα διαδικασία, θα χρησιμοποιήσουμε την συνάρτηση `replace`.

```
% example 9 - replace words
textData = [
            "I'm a doctor.",
            "They aren't here yet."
           ];
toReplace = ["ain't", "isn't", "aren't", "i'm", "i've"];
replaceWith = ["is not", "is not", "are not", "i am", "i
have"];
% make words lower case before replace
textData = lower(textData);
% replace all occurences from toReplace with replaceWith
```

```
textData = replace(textData, toReplace, replaceWith)
% tokenize words using tokenizedDocument funtion
documents = tokenizedDocument(textData);
```

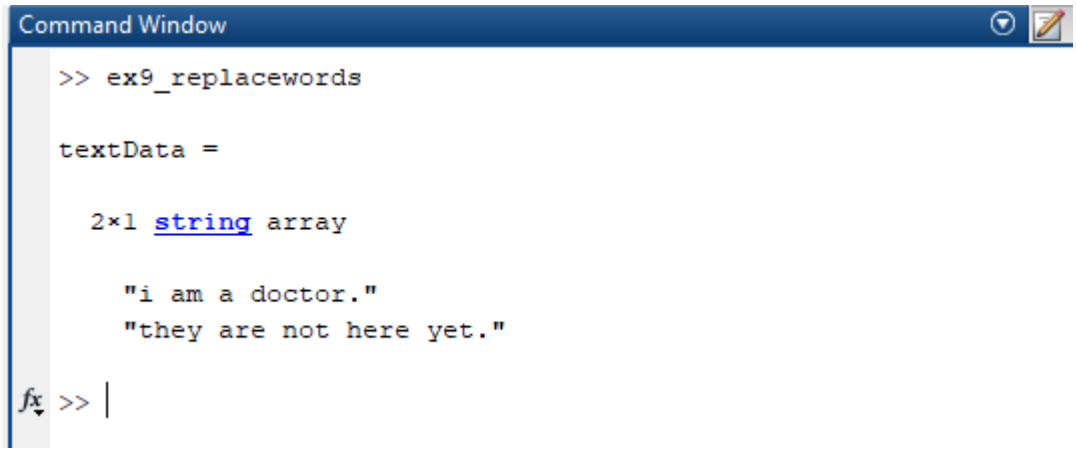
Αυτό που κάνουμε δηλαδή για ευκολία εδώ, είναι να δημιουργήσουμε δύο πίνακες. Ο ένας θα περιέχει τις λέξεις που ψάχνουμε να βρούμε (toReplace), και ο άλλος τις λέξεις με τις οποίες θα αντικατασταθούν (replaceWith).

Η διαδικασία που ακολουθείται έχει ως εξής:

1. Μετατρέπουμε όλα τα γράμματα των εγγράφων σε πεζά, πριν την διαδικασία της αντικατάστασης, έτσι ώστε να μη χρειάζεται να ψάχνουμε και συνδυασμούς κεφαλαίων-πεζών.
2. Με χρήση της replace αντικαθιστούμε κάθε εμφάνιση των λέξεων του πίνακα toReplace στα αντίστοιχα του πίνακα replaceWith, στα περιεχόμενα του textData.
3. Η tokenizedDocument θα δημιουργήσει τα τελικά tokens τα οποία θα περιέχουν σωστά τις λέξεις τους πίνακα replaceWith ως ξεχωριστά tokens.

Θα αποθηκεύσουμε το script σε αρχείο με όνομα *ex9_replacewords.m*.

Με την εκτέλεση του στο MATLAB, τελικά θα δούμε το ακόλουθο αποτέλεσμα:



Εικόνα 33: Εκτέλεση παραδείγματος 9 στο MATLAB

Αν θέλουμε εδώ να δούμε τα περιεχόμενα των token αρκεί να πλήκτρολογήσουμε το όνομα της μεταβλητής: documents

```
>> documents

documents =

  2×1 tokenizedDocument:

(1,1) 5 tokens: i am a doctor .
(2,1) 6 tokens: they are not here yet .

>>
```

Εικόνα 27: Εμφάνιση περιεχομένων του documents

Παρατηρούμε ότι τελικά όντως οι καινούριες λέξεις προσμετρούνται ως ξεχωριστά token.

6.ΑΦΑΙΡΕΣΗ ΜΗ ΕΠΙΘΥΜΗΤΩΝ ΤΜΗΜΑΤΩΝ

Το στάδιο αυτό είναι την τελευταία διαδικασία της προεπεξεργασίας του κειμένου. Πρόκειται για το στάδιο εκείνο στο οποίο επιθυμούμε να αφαιρέσουμε οποιοδήποτε token δεν παρέχει καμία απολύτως πληροφορία.

Ως ανεπιθύμητη πληροφορία θεωρούμε:

- Συγκεκριμένες λέξεις (μικρού μήκους συνήθως) που τις ονομάζουμε stop words (θα εξηγήσουμε τον όρο παρακάτω), καθώς και
- τα σημεία στίξης

6.1: ΑΦΑΙΡΕΣΗ ΤΩΝ STOP WORDS

Με την γενική τους έννοια, stop words θεωρούμε τις λέξεις οι οποίες αφαιρούνται πριν το στάδιο της επεξεργασίας φυσικής γλώσσας. Ανάλογα με τις ανάγκες της κάθε εφαρμογής οποιοδήποτε σύνολο λέξεων μπορεί να θεωρηθεί ως stop words καθώς δεν υπάρχει κάποια κοινώς καθορισμένη λίστα που να τα προσδιορίζει.

Το MATLAB διαθέτει μια προκαθορισμένη λίστα από 190 δικά του stop words ωστόσο είναι δυνατή και η χρήση δικιά μας λίστας σε περίπτωση που δεν θέλουμε να χρησιμοποιήσουμε την προκαθορισμένη.

Για τις ανάγκες της εργασίας, μας αρκεί να χρησιμοποιήσουμε τις προεπιλεγμένες.

```
>> stopWords

ans =

  1×190 string array

  Columns 1 through 12

    "a"    "about"    "above"    "across"    "after"    "all"    "along"    "also"    "am"    "an"    "and"    "any"

  Columns 13 through 23

    "are"    "aren't"    "arent"    "as"    "at"    "be"    "because"    "been"    "before"    "being"    "between"

  Columns 24 through 34

    "both"    "but"    "by"    "can"    "can't"    "cant"    "cannot"    "could"    "couldn't"    "couldnt"    "did"

  Columns 35 through 45

    "didn't"    "didnt"    "do"    "does"    "doesn't"    "doesnt"    "doing"    "done"    "don't"    "dont"    "during"
```

Εικόνα 34: Τμήμα προκαθορισμένων stop words του MATLAB

Όλά τα προκαθορισμένα stop words στην έκδοση που διαθέτουμε βρίσκονται στον πίνακα stopWords, όπως παρατηρήσαμε από την παραπάνω εικόνα. Για να αφαιρέσουμε τις λέξεις αυτές θα χρησιμοποιήσουμε την συνάρτηση `removeWords`, η οποία δέχεται δύο ορίσματα:

- Το πρώτο όρισμα είναι η αρχική λίστα από tokens από την οποία θα αφαιρέσουμε τις λέξεις και
- το δέυτερο όρισμα, είναι η ίδια η λίστα με τα tokens που έχουμε προσδιορίσει ως stop words.

Εμείς ως δεύτερο όρισμα αρκεί να χρησιμοποιήσουμε τον πίνακα stopWords του MATLAB.

```
% example 10 - remove stop words
textData = [
            "This is line 1.",
            "This is line 2."
           ];
 documents = tokenizedDocument(textData); % tokenize words
                                          using tokenizedDocument
documents = lower(documents); % make words lower
documents = removeWords(documents, stopWords) %remove common
                                               stop words
```

Αξίζει να σημειωθεί, πως πριν την διαδικασία αφαίρεσης των stop words είναι απαραίτητη η μετατροπή των tokens σε πεζά (διαδικασία η οποία ήδη περιγράφηκε σε προηγούμενο κεφάλαιο).

Θα αποθηκεύσουμε το script σε αρχείο με όνομα *ex10_remove_stopwords.m*.

Με την εκτέλεση του αρχείου θα παρατηρήσουμε το ακόλουθο αποτέλεσμα:

```
documents =

  2×1 tokenizedDocument:

(1,1) 3 tokens: line 1 .
(2,1) 3 tokens: line 2 .
```

Εικόνα 35: Εκτέλεση παραδείγματος 10 στο MATLAB

Παρατηρούμε, δηλαδή, ότι οι λέξεις “this” και “is” αφαιρούνται από τον πίνακα των tokens, καθώς συμπεριλαμβάνονται στον πίνακα των stop words.

6.2: ΑΦΑΙΡΕΣΗ ΕΙΔΙΚΩΝ ΧΑΡΑΚΤΗΡΩΝ

Παρατηρώντας τα αποτελέσματα του παραδείγματος 10 μπορούμε να διαπιστώσουμε ένα μικρό πρόβλημα: έχουν παραμείνει tokens τα οποία δεν προσφέρουν πληροφορία. Και στα δύο έγγραφα, έχει παραμείνει η τελεία. Η αφαίρεση των ειδικών χαρακτήρων είναι το τελευταίο και πιο εύκολο βήμα στην διαδικασία της προεπεξεργασίας του κειμένου.

Στο MATLAB δεν υπάρχει κάποια έτοιμη συνάρτηση για την αφαίρεση οποιουδήποτε ειδικού χαρακτήρα, ωστόσο μπορούμε να την πραγματοποιήσουμε με μια απλή κλήση της `regexprep`, η οποία χρησιμοποιείται για εύρεση κανονικών εκφράσεων (regular expressions) και αντικατάστασή τους με έναν οποιονδήποτε χαρακτήρα

Αυτό που μας μένει τώρα είναι να κατασκευάσουμε μια κανονική έκφραση η οποία θα αφαιρεί οποιοδήποτε ειδικό χαρακτήρα μπορεί να υπάρξει σε κείμενο. Ας ξεκινήσουμε με ένα παράδειγμα. Έστω ότι προσπαθούμε να αφαιρέσουμε τους ειδικούς χαρακτήρες από τα έγγραφα:

“This is line-1.”

“This is line-2.”

Μια απλοϊκή προσέγγιση είναι να αφαιρέσουμε οποιονδήποτε χαρακτήρα δεν είναι γράμμα ή αριθμός. Η κανονική έκφραση η οποία αναπαριστά την διαδικασία αυτή είναι η ακόλουθη:

```
[^a-zA-Z0-9]
```

Σε περίπτωση που βρεθεί τέτοιος χαρακτήρας τον αντικαθιστούμε με κενό χαρακτήρα, δηλαδή space.

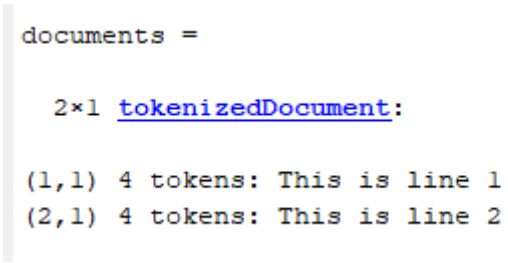
Διαπιστώνουμε επομένως πως η τελική μορφή της εντολής regexprep θα είναι η εξής:

```
regexprep(textData,'[^a-zA-Z0-9]',' ')
```

Μπορούμε επομένως εύκολα να φτιάξουμε τον ολοκληρωμένο κώδικα του παραδείγματος:

```
% example 11 - remove special characters
textData = [
            "This is line-1.",
            "This is line-2."
           ];
% remove any character except letters and numbers
textData = regexprep(textData,'[^a-zA-Z0-9]',' ');
% tokenize words using tokenizedDocument funtion
documents = tokenizedDocument(textData)
```

Θα αποθηκεύσουμε το script σε αρχείο με όνομα *ex11_remove_special_characters.m*.
Με την εκτέλεση του παραδείγματος στο MATLAB μπορούμε να δούμε σε πράξη την εφαρμογή της κανονικής έκφρασης στο αρχικό κείμενο:



Εικόνα 36: Εκτέλεση παραδείγματος 11 στο MATLAB

Αν μας ενδιαφέρει μόνο να αφαιρέσουμε σημεία στίξης από τα έγγραφα, μια δεύτερη υλοποίηση, θα μπορούσε να ακολουθήσει την χρήση της συνάρτησης `erasePunctuation`. Η συνάρτηση αυτή δέχεται τον πίνακα από tokens και αφαιρεί **μόνο** τα σημεία στίξης.

```
% example 12 - erase punctuation
textData = [
            "This is line 1.",
            "This is line 2."
           ];
 % tokenize words using tokenizedDocument funtion
documents = tokenizedDocument(textData);
documents = lower(documents); % make words lower
```

```
% remove punctuation characters from tokens
documents = erasePunctuation(documents)
```

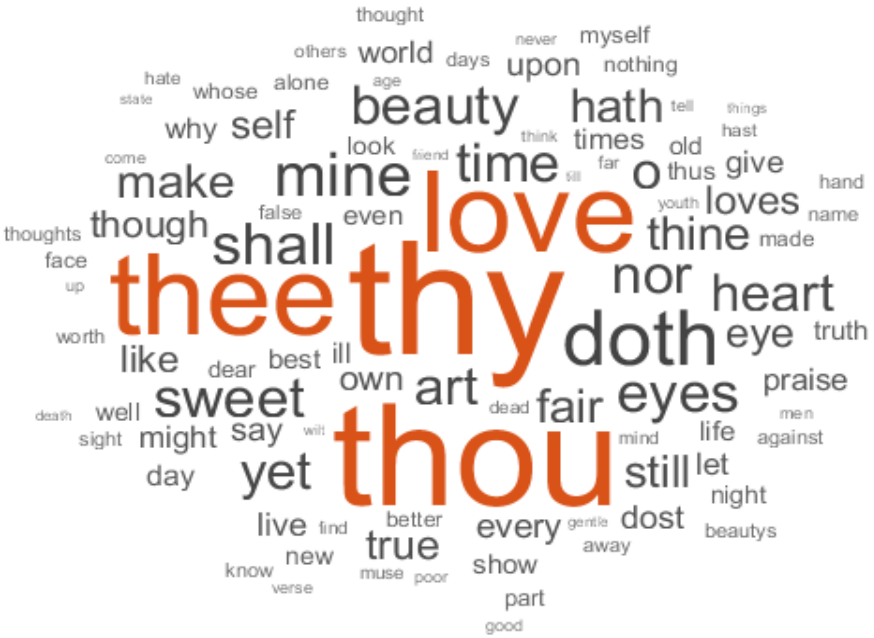
Θα αποθηκεύσουμε το script σε αρχείο με όνομα *ex12_erase_punctuation.m*.
Με την εκτέλεση του στο MATLAB μπορούμε να δούμε το αποτέλεσμα που εμφανίζεται:

```
documents =

  2×1 tokenizedDocument:

(1,1) 4 tokens: this is line 1
(2,1) 4 tokens: this is line 2

>>
```

Εικόνα 37: Εκτέλεση παραδείγματος 12 στο MATLAB

7.ΤΟ ΜΟΝΤΕΛΟ BAG OF WORDS

Το Bag of Words είναι μια απλοποιημένη αναπαράσταση η οποία χρησιμοποιείται τόσο στην επεξεργασία φυσικής γλώσσας, όσο και στην ανάκτηση πληροφορίας. Στην ουσία το κείμενο αναπαρίσταται ως ένας σάκος στον οποίο περιέχονται οι λέξεις του, χωρίς να μας ενδιαφέρει η γραμματική ή η σειρά τους.



Εικόνα 38: Γραφική αναπαράσταση παραδείγματος μοντέλου Bag of Words

Αυτό το μοντέλο τώρα, μπορούμε εμείς να το χρησιμοποιήσουμε έτσι ώστε να βρούμε ποιές λέξεις εμφανίζονται τις περισσότερες φορές σε ένα έγγραφο, ακόμα και πόσο σημαντική είναι μια λέξη για ένα έγγραφο.

7.1. ΤΟ BAG OF WORDS ΣΤΗΝ ΠΡΑΞΗ

Το μοντέλο αυτό είναι γνωστό και ως μετρητής συχνότητας όρων. Για να κατανοήσουμε ευκολότερα τι είναι το bag of words ας χρησιμοποιήσουμε ένα παράδειγμα. Έστω ότι έχουμε ένα έγγραφο το οποίο περιέχει την φράση:

```
John likes to watch movies. Mary likes movies too.
```

Το bag of words θα συνοψίσει όλες τις λέξεις που υπάρχουν στο κείμενο βρίσκοντας επιπλέον το πλήθος εμφάνισής τους:

```
John      {1}
likes     {2}
to        {1}
watch     {1}
movies    {2}
Mary      {1}
too       {1}
```

Στο MATLAB αυτό υλοποιείται εύκολα με την συνάρτηση `bagOfWords`, η οποία, φυσικά, δέχεται ως όρισμα τον πίνακα των tokens που διαθέτουμε. Για να δούμε λεπτομέρειες στο MATLAB για το bag of words αρκεί να πληκτρολογήσουμε την εντολή

```
help bagOfWords
```

Το MATLAB θα επιστρέψει το ακόλουθο αποτέλεσμα:

```
>> help bagOfWords
 bagOfWords Bag-of-words model
    A bag-of-words model (also known as a term-frequency counter) records the
    number of times that words appear in each document of a collection.
    bagOfWords does not split text into words. To create an array of tokenized
    documents, see tokenizedDocument.

    BAG = bagOfWords creates an empty bag-of-words model.

    BAG = bagOfWords(DOCUMENTS) counts the words appearing in DOCUMENTS and
    returns a bag-of-words model.
```

Εικόνα 39: Βοήθεια στο MATLAB για την εντολή bagOfWords

Για να δούμε πρακτικά τον τρόπο χρήσης της συνάρτησης ας δούμε το ακόλουθο παράδειγμα:

```
% example 13 - bag of words
textData = [
            "This is line 1.",
            "This is line 2."
           ];
% tokenize words using tokenizedDocument funtion
documents = tokenizedDocument(textData);
documents = lower(documents); % make words lower
bag = bagOfWords(documents) % create bag of words model
tbl = topkwords(bag,6) % view top 6 words in the bag
```

Μπορούμε εδώ να παρατηρήσουμε πως έχουμε την δυνατότητα να δούμε και συγκεκριμένο πλήθος για τις πιο συχνά χρησιμοποιούμενες λέξεις στο μοντέλο. Εδώ επιλέγουμε να δούμε τις 6 πιο συχνές.
Θα αποθηκεύσουμε το script σε αρχείο με όνομα *ex13_bag_of_words.m*.
Με την εκτέλεσή του στο MATLAB θα έχουμε το εξής αποτέλεσμα:

```
bag =

  bagOfWords with properties:

          Counts: [2×6 double]
      Vocabulary: ["this"    "is"    "line"    "1"    "."    "2"]
        NumWords: 6
    NumDocuments: 2


tbl =

  6×2 table

     Word      Count
    _______    _____

    "this"       2
    "is"         2
    "line"       2
    "."          2
    "1"          1
    "2"          1
```

Εικόνα 40: Εκτέλεση παραδείγματος 13 στο MATLAB

Μπορούμε να παρατηρήσουμε αρχικά ότι η συνάρτηση bagOfWords μας επιστρέφει μόνο το σύνολο των λέξεων που περιέχει το μοντέλο και το πλήθος των εγγράφων που

χρησιμοποίησε για να το σχηματίσει. Εμείς με χρήση της topkwords μπορούμε με ευκολία να ανακαλύψουμε επακριβώς το πλήθος εμφανίσεων των πιο συχνά εμφανιζόμενων λέξεων.

7.2. ΥΠΟΛΟΓΙΣΜΟΣ ΤΟΥ TF-IDF

Το tf-idf είναι ένας επιπλέον τρόπος να υπολογίσουμε πόσο σημαντικός είναι ένας όρος στο έγγραφο. Το ακρωνύμιο tf-idf σημαίνει Term Frequency - Inverse Document Frequency, δηλαδή Συχνότητα Όρων - Αντίστροφη συχνότητα εγγράφων.

Η διαφορά του με την απλή μέτρηση εμφανίσεων που είδαμε στο bag of words, είναι ότι εδώ επιχειρούμε να μετρήσουμε την σχετικότητα ενός όρου, και όχι την συχνότητά του. Για να το πετύχει αυτό εφαρμόζει δύο πράγματα:

- Πρώτα, μετράει την συχνότητα των όρων που εμφανίζονται στο έγγραφο (Συχνότητα όρων).
- Επειδή ορισμένες λέξεις όπως π.χ. στα ελληνικά το "και" ή το "αυτό" εμφανίζονται πολύ συχνά στα έγγραφα, αυτές θα πρέπει να αγνοούνται. (Αντίστροφη-συχνότητα εγγράφων).

Μπορούμε δηλαδή να διαπιστώσουμε πως πιο σημαντικές εδώ είναι οι λέξεις που εμφανίζονται λιγότερο.

Για κάθε όρο σε ένα έγγραφο, δίνεται ένα βάρος, το οποίο προσδιορίζει την σχετικότητά του. Η τιμή του βάρους δίνεται από τον ακόλουθο τύπο:

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

όπου:

$tf_{i,j}$ = ο αριθμός των εμφανίσεων του όρου i στο έγγραφο j

df_i = ο αριθμός των εγγράφων που περιέχουν τον όρο i

N = ο συνολικός αριθμός των εγγράφων.

Στο MATLAB για να βρούμε το tf-idf κάθε όρου, πολύ απλά, χρησιμοποιούμε την έτοιμη συνάρτηση `tfidf`. Η συνάρτηση αυτή δέχεται ως όρισμα ένα μοντέλο bag of words. Συνεπώς παρατηρούμε πως για να χρησιμοποιήσουμε την tfidf, είναι απαραίτητο να έχουμε κατασκευάσει πρώτα ένα μοντέλο bag-of-words για τα δεδομένα μας.

Με το ακόλουθο παράδειγμα μπορούμε να δούμε πως βρίσκουμε το tf-idf κάθε όρου σε ένα έγγραφο.

```
% example 14 - tfidf
textData = [
            "This is line 1.",
            "This is line 2."
           ];
% tokenize words using tokenizedDocument funtion
documents = tokenizedDocument(textData);
documents = lower(documents); % make words lower
bag = bagOfWords(documents)  % create a bag-of-words model
M = tfidf(bag) % compute tf-idf
full(M(1:2, 1:6)) % view full array
```

Θα αποθηκεύσουμε το script σε αρχείο με όνομα *ex14_tfidf.m*.
Εκτελώντας το στο MATLAB μπορούμε να δούμε τα εξής αποτελέσματα:

```
bag =

  bagOfWords with properties:

         Counts: [2×6 double]
     Vocabulary: ["this"    "is"    "line"    "1"    "."    "2"]
       NumWords: 6
   NumDocuments: 2


M =

   (1,4)       0.6931
   (2,6)       0.6931


ans =

         0         0         0    0.6931         0         0
         0         0         0         0         0    0.6931
```

Εικόνα 41: Εκτέλεση παραδείγματος 14 στο MATLAB

Ο πίνακας M είναι αυτός που περιέχει τις τιμές tf-idf. Παρατηρούμε ότι μας δίνει 2 γραμμές:
(1,4) 0.6931
(2,6) 0.6931
Αυτές είναι οι τιμές tf-idf στις αντίστοιχες θέσεις. Π.χ. στην θέση (1,4) του πίνακα έχει τιμή 0.6931. Οι θέσεις που δεν εμφανίζονται έχουν τιμή 0.
Για να δούμε ολόκληρο τον πίνακα χρησιμοποιούμε την εντολή full(M(1:2, 1:6)), δηλαδή προβολή των γραμμών 1-2 και των στηλών 1-6 του πίνακα M.

Παράρτημα

Ευρετήριο κώδικα εργασίας

ex1_sentence_tokenization1.m

```
% example 1 - Single text
textData = "This is text 1.  This is text 2.";
Documents = splitSentences(textData) %split text using
                                      splitSentences
```

ex2_sentence_tokenization2.m

```
% example 2 - Single text with Abbreviations
textData = "Dr. John. Mr. Smith.";
Documents = splitSentences(textData) %split different text
                                      using splitSentences
```

ex3_word_tokenization1.m

```
% example 3 - Single text, tokenize words
textData = "This is an example.";
documents = tokenizedDocument(textData) % tokenize words using
                                          tokenizedDocument
```

ex4_word_tokenization2.m

```
% example 4 - multiple sentences, tokenize words
textData = [
            "This is line 1",
            "This is line 2"
           ];
documents = tokenizedDocument(textData) % tokenize words using
                                          tokenizedDocument
```

ex5_word_tokenization3.m

```
% example 5 - multiple sentences, view token details
textData = [
            "This is line 1.",
            "This is line 2."
           ];
documents = tokenizedDocument(textData); % tokenize words
                                         using tokenizedDocument
tdetails = tokenDetails(documents) % view details for each
                                     token
```

ex6_stemming.m

```
% example 6 - multiple sentences, stem tokens
textData = [
            "A collection of words.",
            "Another collection of words."
           ];
documents = tokenizedDocument(textData); % tokenize words
                                         using tokenizedDocument
stDocuments = normalizeWords(documents)  % stemming using
                                           normalizeWords
```

ex7_lowerwords.m

```
% example 7 - multiple sentences, make words lowercase
textData = [
            "A Collection of Words.",
            "Another Collection of Words."
           ];
documents = tokenizedDocument(textData); % tokenize words
                                         using tokenizedDocument
stDocuments = lower(documents)  % make all words on documents
                                  lowercase
```

ex8_upperwords.m

```
% example 8 - multiple sentences, make words uppercase
textData = [
            "A Collection of Words.",
            "Another Collection of Words."
           ];
documents = tokenizedDocument(textData); % tokenize words
                                          using tokenizedDocument
stDocuments = upper(documents)  % make all words on documents
                                  uppercase
```

ex9_replacewords.m

```
% example 9 - replace words
textData = [
            "I'm a doctor.",
            "They aren't here yet."
           ];

toReplace = ["ain't", "isn't", "aren't", "i'm", "i've"];
replaceWith = ["is not", "is not", "are not", "i am", "i
have"];

% make words lower case before replace
textData = lower(textData);

% replace all occurences from toReplace with replaceWith
textData = replace(textData, toReplace, replaceWith)

% tokenize words using tokenizedDocument funtion
documents = tokenizedDocument(textData);
```

ex10_remove_stopwords.m

```
% example 10 - remove stop words
textData = [
            "This is line 1.",
            "This is line 2."
           ];

documents = tokenizedDocument(textData); % tokenize words
                                         using tokenizedDocument
documents = lower(documents); % make words lower
documents = removeWords(documents, stopWords) %remove common
                                               stop words
```

ex11_remove_special_characters.m

```
% example 11 - remove special characters
textData = [
            "This is line-1.",
            "This is line-2."
           ];

% remove any character except letters and numbers
textData = regexprep(textData,'[^a-zA-Z0-9]',' ');

% tokenize words using tokenizedDocument funtion
documents = tokenizedDocument(textData)
```

ex12_erase_punctuation.m

```
% example 12 - erase punctuation
textData = [
            "This is line 1.",
            "This is line 2."
           ];
```

```
 % tokenize words using tokenizedDocument funtion
documents = tokenizedDocument(textData);
documents = lower(documents); % make words lower
% remove punctuation characters from tokens
documents = erasePunctuation(documents)
```

ex13_bag_of_words.m

```
% example 13 - bag of words
textData = [
            "This is line 1.",
            "This is line 2."
           ];

% tokenize words using tokenizedDocument funtion
documents = tokenizedDocument(textData);
documents = lower(documents); % make words lower
bag = bagOfWords(documents) % create bag of words model
tbl = topkwords(bag,6) % view top 6 words in the bag
```

ex14_tfidf.m

```
% example 14 - tfidf
textData = [
            "This is line 1.",
            "This is line 2."
           ];

% tokenize words using tokenizedDocument funtion
documents = tokenizedDocument(textData);
documents = lower(documents); % make words lower
bag = bagOfWords(documents)  % create a bag-of-words model
M = tfidf(bag) % compute tf-idf
full(M(1:2, 1:6)) % view full array
```

ΣΥΜΠΕΡΑΣΜΑΤΑ

Η πτυχιακή αυτή ασχολήθηκε με τις βασικές τεχνικές εξόρυξης κειμένου, με παραδείγματα υλοποιημένα σε MATLAB. Τα παραδείγματα, έγινε προσπάθεια να είναι μικρά σε μέγεθος και εύκολα κατανοητά έτσι ώστε να μπορούν να χρησιμοποιηθούν για εκπαιδευτικούς σκοπούς.

Στα πρώτα κεφάλαια έγινε μια εισαγωγή στον τομέα της εξόρυξης κειμένου, μελετώντας βασικές τεχνικές και παραδείγματα. Επιπλέον έγινε μια μικρή εισαγωγή στο MATLAB, μελετώντας τα βασικά χαρακτηριστικά του.

Στη συνέχεια ακολούθησε η ανάλυση των βασικότερων τεχνικών εξόρυξης, ξεκινώντας από τις τεχνικές προ-επεξεργασίας κειμένου. Η προ-επεξεργασία κειμένου η οποία περιλαμβάνει τρία τμήματα:

Την τμηματοποίηση του κειμένου σε λέξεις (tokens),

Την κανονικοποίηση των τμημάτων αυτών (χρησιμοποιώντας π.χ. τον αλγόριθμο stemming) καθώς και

Την αφαίρεση stop words και λοιπών σημείων στίξης που δεν παρέχουν κάποιες πληροφορίες.

Η επόμενη διαδικασία που ακολουθεί είναι στην ουσία της ανάλυσης των δεδομένων αυτών προσπαθώντας να ανακαλύψουμε ποιοί όροι μας είναι χρήσιμοι για το έγγραφο. Για τον λόγο αυτό χρησιμοποιούμε το μοντέλο bag of words, το οποίο είναι στην ουσία ένας μετρήτής συχνότητας όρων. Τέλος, για μελέτη της σχετικότητας ενός όρου σε ένα έγγραφο αναλύθηκε ο τρόπος υπολογισμού του tf-idf (συχνότητα όρων-αντίστροφη συχνότητα εγγράφων).

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. THE TEXT MINING HANDBOOK: Advanced Approaches in Analyzing Unstructured Data, περίληψη βιβλίου

2. https://en.wikipedia.org/wiki/Text_mining

3. https://el.wikipedia.org/wiki/MATLAB

4. https://www.mathworks.com/products/text-analytics.html

5. https://www.mathworks.com/help/textanalytics/release-notes.html (release range)

6. https://www.upgrad.com/blog/what-is-text-mining-techniques-and-applications/

7. https://monkeylearn.com/text-mining/

8. https://www.tutorialspoint.com/matlab/index.htm

9. ΣΗΜΑΤΑ ΚΑΙ ΣΥΣΤΗΜΑΤΑ ΜΕ MATLAB, ΑΛΕΞΗΣ ΠΑΛΑΜΙΔΗΣ - ΑΝΑΣΤΑΣΙΑ ΒΕΛΩΝΗ, ΑΘΗΝΑ 2008, (σελ. 13-16)

10. https://www.techopedia.com/definition/13698/tokenization

11. Advanced Computational Intelligence: An International Journal (ACII), Vol.3, No.1, January 2016, (σελ. 37-39)

12. https://gerardnico.com/natural_language/text_normalization

13. https://www.geeksforgeeks.org/introduction-to-stemming/

14. http://people.scs.carleton.ca/~armyunis/projects/KAPI/porter.pdf

15. https://en.wikipedia.org/wiki/Stemming

16. https://en.wikipedia.org/wiki/Stop_words

17. https://en.wikipedia.org/wiki/Bag-of-words_model

18. https://www.mathworks.com/help/textanalytics/

19. https://pathmind.com/wiki/bagofwords-tf-idf