Πανεπιστήμιο Δυτικής Αττικής

Σχολή Μηχανικών

Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

Διπλωματική Εργασία

Πλατφόρμα Ανάλυσης Δεδομένων Εγκλημάτων με Χρήση Τεχνικών Μηχανικής Μάθησης & Εικονικοποίησης βασισμένης σε Περιέκτες

Σιμόνα Μλαντένοβα
711161116

Επιβλέπων: **Βασίλειος Μάμαλης**
Συν-Επιβλέπων: **Απόστολος Αναγνωστόπουλος**

Αιγάλεω - Αθήνα, Οκτώβριος, 2021

Η παρούσα διπλωματική εργασία παρουσιάστηκε από την Σιμόνα Μλαντένοβα (711161116) στις 15/10/2021.

Εγκρίθηκε από την εξεταστική επιτροπή

ΚΑΝΤΖΑΒΕΛΟΥ ΙΩΑΝΝΑ Μέλος επιτροπής	ΚΑΡΚΑΖΗΣ ΠΑΝΑΓΙΩΤΗΣ Μέλος επιτροπής	ΜΑΜΑΛΗΣ ΒΑΣΙΛΕΙΟΣ Επιβλέπων καθηγητής
Ioanna Kantzavelou Digitally signed by Ioanna Kantzavelou Date: 2021.10.20 20:44:03 +03'00'	Panagiotis Karkazis Digitally signed by Panagiotis Karkazis Date: 2021.10.23 15:56:49 +03'00'	Vasileios Mamalis Digitally signed by Vasileios Mamalis Date: 2021.10.19 00:48:50 +03'00'

Σιμόνα Μλαντένοβα
Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών
Πανεπιστήμιο Δυτικής Αττικής

Δήλωση συγγραφέα διπλωματικής εργασίας

Η κάτωθι υπογεγραμμένη Σιμόνα Μλαντένοβα του Νταναΐλ, με αριθμό μητρώου 711161116 φοιτήτρια του Πανεπιστημίου Δυτικής Αττικής της Σχολής Μηχανικών του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών, δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της διπλωματικής εργασίας και κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος. Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Η δηλούσα

Περίληψη

Η ανάλυση των εγκλημάτων είναι μια μεθοδική προσέγγιση για τον προσδιορισμό και την ανάλυση των μοτίβων και των τάσεων στα εγκλήματα. Με την αυξανόμενη προέλευση των μηχανογραφημένων συστημάτων, οι αναλυτές δεδομένων μπορούν να βοηθήσουν τις υπηρεσίες επιβολής του νόμου να επιταχύνουν τη διαδικασία επίλυσης εγκλημάτων. Χρησιμοποιώντας την έννοια της εξόρυξης δεδομένων και της μηχανικής μάθησης, μπορούν να αναλύσουν προηγουμένως άγνωστες, χρήσιμες πληροφορίες από μη δομημένα δεδομένα.

Λόγω της ανόδου του όγκου των ανοιχτών δεδομένων τα τελευταία χρόνια, μπορούν να χρησιμοποιηθούν ψηφιακά δημοσιευμένα άρθρα εγκλημάτων από διάφορες πηγές που αναφέρουν τακτικά εγκλήματα και περιέχουν λεπτομέρειες για το έγκλημα και πληροφορίες για τις κατηγορούμενες οντότητες. Η τελική πλατφόρμα ανάλυσης των άρθρων περιλαμβάνει ένα σύνολο από μικροϋπηρεσίες που συνεργάζονται μεταξύ τους. Αρχικά, μέσω του Scrapy πραγματοποιείται η συλλογή των δεδομένων και η αποθήκευση τους στην αρχική τους μορφή σε μια βάση δεδομένων (MongoDB).

Για να κατηγοριοποιηθεί το κάθε άρθρο που συλλέγεται στην σωστή κατηγορία εγκλήματος, χρησιμοποιούνται μοντέλα αλγορίθμων κατηγοριοποίησης δειγμάτων (όπως Naive Bayes, Support Vector Machines). Για την ανάλυση σημαντικών πληροφοριών από το κείμενο χρησιμοποιούνται αλγόριθμοι επεξεργασίας φυσικής γλώσσας (NLP) σε συνδυασμό με τις αναλύσεις που πραγματοποιούνται από τους αναλυτές του elasticsearch, ενός συνόλου εργαλείων ανοιχτού κώδικα για συλλογή δεδομένων, εμπλουτισμό, αποθήκευση και ανάλυση δεδομένων. Ο σκοπός της ανάλυσης είναι να γίνει η εξαγωγή σημαντικών στοιχείων όπως η τοποθεσία του εγκλήματος, η ηλικία του θύματος και του θύτη, το γένος του θύματος, αν συνελήφθη ο δράστης, καθώς και λέξεις κλειδιά που δηλώνουν τον τρόπο που πραγματοποιήθηκε το έγκλημα.

Η οπτικοποίηση των τελικών δεδομένων πραγματοποιείται χρησιμοποιώντας Plotly's Dash και μας δίνει μια σαφή ιδέα για το τι σημαίνει η πληροφορία δίνοντάς της οπτικό πλαίσιο μέσω χαρτών ή γραφημάτων. Αυτό καθιστά τα τελικά δεδομένα πιο φυσικά για να κατανοήσει το ανθρώπινο μυαλό και επομένως διευκολύνει τον εντοπισμό τάσεων, προτύπων και ακραίων τιμών σε μεγάλα σύνολα δεδομένων. Λόγω της εικονοποίησης βασισμένη σε περιέκτες, τα παραπάνω επίπεδα χτίζουν μια ολοκληρωμένη πλατφόρμα ανάλυσης εγκλημάτων ως μια σειρά από μικροϋπηρεσίες, καθεμία από τις οποίες εκτελείται με τη δική της διαδικασία και είναι ανεξάρτητη.

Λέξεις Κλειδιά: Μηχανική Μάθηση, Εξόρυξη Κειμένου, Κατηγοριοποίηση Κειμένου, Επεξεργασία Φυσική Γλώσσας, Elasticsearch, Οπτικοποίηση Δεδομένων, Ανάλυση Δεδομένων Εγκλημάτων, Εικονοποίηση Βασισμένη σε Περιέκτες, Μικροϋπηρεσίες.

Abstract

Crime analysis is a methodical approach to identifying and analyzing patterns and trends in crime. With the growing origins of computerized systems, data analysts can help law enforcement agencies speed up the crime-solving process. Using the concept of data mining and machine learning, they can analyze previously unknown, useful information from unstructured.

Due to the increase in the volume of open data in recent years, digitally published crime articles from various sources that report regular crimes and contain crime details and information about the accused entities can be used. The ultimate article analysis platform includes a set of microservices that work together. Initially, Scrapy collects the data and stores it in its original form in a database (MongoDB).

To categorize each article collected in the correct crime category, sample categorization algorithm models (such as Naive Bayes, Support Vector Machines) are used. Natural text processing (NLP) algorithms are used to analyze important textual information in conjunction with analyzes performed by elasticsearch analysts, a set of open source tools for data collection, enrichment, storage, and data analysis. The purpose of the analysis is to extract important information such as the location of the crime, the age of the victim and the perpetrator, the gender of the victim, if the perpetrator is arrested, as well as keywords that indicate how the crime was committed.

The visualization of the final data is done using Plotly's Dash and gives us a clear idea of what the information means by giving it a visual framework through maps or graphs. This makes the final data more natural for the human mind to understand and therefore makes it easier to identify trends, patterns and extremes in large data sets. Due to container-based visualization, the above levels build a comprehensive crime analysis platform as a series of microservices, each of which runs its own process and is independent.

Keywords: Machine Learning, Text Mining, Text Categorization, Physical Language Processing, Elasticsearch, Data Visualization, Data Analysis, Container Based Visualization, microservices.

Πίνακας περιεχομένων

Πίνακας σχημάτων

Πίνακας πινάκων

Κεφάλαιο 1

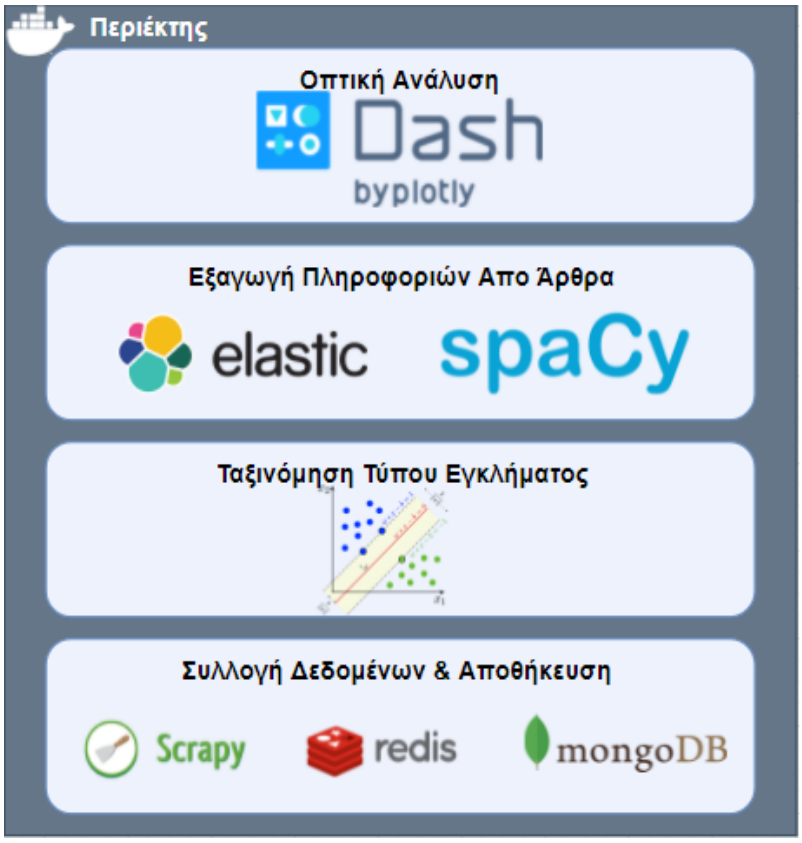
Εισαγωγή

1.1 Συλλογή και Ανάλυση Δεδομένων Εγκλήματος

Η ανάλυση εγκλημάτων είναι μία από τις πιο σημαντικές δραστηριότητες του τομέα της επιβολής του νόμου σε όλο τον κόσμο. Οι υπεύθυνοι οργανισμοί συλλέγουν είτε τοπικά ή παγκόσμια δεδομένα για την πρόληψη μελλοντικών επιθέσεων και για την ανάλυση των δεδομένων ώστε να χρησιμοποιηθούν οι διαθέσιμοι πόροι με τον βέλτιστο τρόπο.

Ο μεγάλος όγκος εγκλημάτων είναι ένα παγκόσμιο πρόβλημα που μπορεί να βλάψει ένα έθνος υπό κοινωνικές και οικονομικές συνθήκες βάζοντας την οικονομική επιβάρυνση στην κυβέρνηση λόγω της ανάγκης επιπλέον αστυνομικής δύναμης, δικαστηρίων κ.λπ. Μια μεγάλη πρόκληση που αντιμετωπίζουν οι περισσότεροι οργανισμοί επιβολής του νόμου είναι η αποτελεσματική και ακριβή ανάλυση των αυξανόμενων όγκων δεδομένων που αφορούν εγκλήματα. Η τεράστια γεωγραφική ποικιλομορφία και η πολυπλοκότητα των μορφών εγκλήματος έχουν κάνει την ανάλυση και καταγραφή τους πιο δύσκολη. Μια προσέγγιση μεταξύ της πληροφορικής και της ποινικής δικαιοσύνης μπορεί να χρησιμοποιηθεί για τη δημιουργία μιας τεχνικής εξόρυξης δεδομένων που μπορεί να βοηθήσει στην ταχύτερη ανάλυση και πρόληψη τους.

Η ουσία της εξόρυξης και ανάλυσης δεδομένων έχει γίνει ένα γρήγορα αναπτυσσόμενο πεδίο για τους αναλυτές εγκλημάτων. Ένας μεγάλος όγκος μη οργανωμένων αρχείων που σχετίζονται με αστυνομικά δεδομένα τονίζει την ανάγκη για την χρήση συστηματικού και αυτόματου τρόπου της συλλογής, ανάλυσης και προσδιορισμού των συσχετισμένων δεδομένων όπως μοτίβα, στατιστικά στοιχεία και χάρτες. Μεγάλος αριθμός στοιχείων εγκλημάτων καταγράφονται συστηματικά από την αστυνομία εδώ και πολλά χρόνια ενώ την περασμένη δεκαετία σημειώθηκε αύξηση των ανοιχτών δεδομένων στο διαδίκτυο καθώς και εφαρμογών ή / και διαδικτυακών εφαρμογών που εμφανίζουν στατιστικά στοιχεία εγκλημάτων στους χάρτες, τόσο από επίσημες πηγές, όπως από την Αστυνομία όσο και απο άλλες πηγές που χρησιμοποιούν τα ίδια επίσημα δεδομένα. Για παράδειγμα, στις στον ιστότοπο data.gov.gr, υπάρχουν εφαρμογές που αναφέρονται από διάφορες πηγές οι οποίες δίνουν στατιστικά στοιχεία και χάρτες σχετικά με το έγκλημα στην Ελλάδα.



Εικόνα 1. Τα τέσσερα Επίπεδα της Πλατφόρμας Ανάλυσης Εγκλημάτων

1.2 Μεθοδολογία

Για την βέλτιστη επεξεργασία των εγκλημάτων, έχει χρησιμοποιηθεί ένα πλαίσιο συλλογής, παρακολούθησης και ανάλυσης των εγκλημάτων. Η υλοποίηση πραγματοποιείται μέσω μικροϋπηρεσιών σε περιέκτη ενώ τα γενικά επίπεδα αρχιτεκτονικής της εφαρμογής μπορούν να κατηγοριοποιηθούν σε τέσσερα επίπεδα επεξεργασίας δεδομένων και ένα επίπεδο που περιλαμβάνει τις μικροϋπηρεσίες αυτές, όπως φαίνεται παρακάτω.

1.2.1 Συλλογή Δεδομένων και Αποθήκευση

Στο πρώτο επίπεδο γίνεται η συλλογή των δεδομένων. Το σύνολο των δεδομένων που χρησιμοποιείται για την υλοποίηση είναι πραγματικό και εξάγεται μέσω python crawlers (Scrapy) σε πραγματικό χρόνο από πολλαπλές σελίδες που περιλαμβάνουν άρθρα εγκλημάτων στην Ελλάδα. Το τελικό σχήμα των δεδομένων θα περιλαμβάνει την χρονολογία του άρθρου, τον συγγραφέα, τον τίτλο, το κείμενο και τις ετικέτες του άρθρου, αν υπάρχουν και αποθηκεύονται στην MongoDB.

1.2.2 Ταξινόμηση τύπου Εγκλήματος

Ένα κομμάτι από την συλλογή των άρθρων χρησιμοποιήθηκε για την ανάπτυξη ενός ταξινομητή που κατηγοριοποιεί τα άρθρα και τα κατατάσσει σε μια συγκεκριμένη κατη-

γορία εγκλήματος. Οι κατηγορίες περιλαμβάνουν: κλοπή/ληστεία, σεξουαλικό έγκλημα, δολοφονία, ναρκωτικά, τρομοκρατική επίθεση και οτιδήποτε δεν συμπεριλαμβάνεται στις 5 αυτές κατηγορίες. Χρησιμοποιούνται αλγόριθμοι μηχανικής μάθησης για την σύγκριση μεταξύ τους και την επιλογή του αλγόριθμου με τα υψηλότερα ποσοστά ακρίβειας.

1.2.3 Εξαγωγή πληροφοριών απο τα άρθρα

Το κάθε άρθρο περνάει από επεξεργασία φυσικής γλώσσας για να εξάγει τα σημαντικότερα στοιχεία από το κείμενο. Τα σημαντικότερα στοιχεία είναι είτε λέξεις ή προτάσεις που θα μας βοηθήσουν ώστε να γίνει μια καλύτερη ανάλυση του εγκλήματος. Πιο συγκεκριμένα, εξάγονται πεδία όπως η τοποθεσία του εγκλήματος, η ηλικία του θύματος και του θύτη, το γένος του θύματος, αν συνελήφθη ο δράστης, καθώς και λέξεις που δηλώνουν τον τρόπο που πραγματοποιήθηκε το έγκλημα. Έχουν αναπτυχθεί αλγόριθμοι ειδικά σχεδιασμένοι για το περιεχόμενο των ιστοσελίδων και για για την ανάλυση της Ελληνικής γλώσσας που εξάγουν τις σχετικές πληροφορίες αναλύοντας το κάθε άρθρο ξεχωριστά.

1.2.4 Οπτική ανάλυση των αποτελεσμάτων

Το τελευταίο βήμα περιλαμβάνει την ανάπτυξη του γραφικού περιβάλλοντος μέσω Plotly's Dash. Το αποτέλεσμα είναι ένας πίνακας που περιλαμβάνει πολλά άρθρα, ανάλογα την κατηγορία και το εύρος ημερομηνιών που έχει επιλεχθεί, με την ανάλυση τους. Έχουν προσθέτει και αναλύσεις μέσω γραφημάτων που εφαρμόζουν οπτικές κατηγοριοποιήσεις καθώς και χάρτης με γεωγραφικές τοποθεσίες όπου έχουν πραγματοποιηθεί τα εγκλήματα.

1.2.5 Εικονοποίηση βασισμένη σε Περιέκτες

Λόγω της εικονοποίησης βασισμένης σε περιέκτες, τα παραπάνω 4 επίπεδα χτίζουν μια πλατφόρμα ανάλυσης εγκλημάτων βασισμένης σε μια σειρά από μικροϋπηρεσίες. Η κάθε μικροϋπηρεσία είναι ανεξάρτητη από την άλλη και συνεργάζονται όλες μεταξύ τους, στο ίδιο δίκτυο. Έχει πραγματοποιηθεί και η δυνατότητα κατανεμημένης λήψης άρθρων με την βοήθεια του redis που αναλαμβάνει την εξάλειψη διπλοεγγραφών καθώς ελέγχει την ουρά μηνυμάτων.

Κεφάλαιο 2

Θεωρητικό Υπόβαθρο

2.1 Εξαγωγή Δεδομένων απο το Διαδίκτυο

Ο όρος αυτός αναφέρεται συνήθως σε αυτοματοποιημένες διαδικασίες που εφαρμόζονται χρησιμοποιώντας μια διαδικασία που αποκαλείται bot ή web crawler. Είναι μια μορφή αντιγραφής στην οποία συλλέγονται και αντιγράφονται συγκεκριμένα δεδομένα από τον Ιστό, συνήθως σε μια κεντρική τοπική βάση δεδομένων ή υπολογιστικό φύλλο, για μετέπειτα ανάκτηση ή ανάλυση. Λόγω του γεγονότος ότι ένας τεράστιος αριθμός ετερογενών δεδομένων δημιουργείται συνεχώς στο διαδίκτυο, η εξαγωγή δεδομένων από το διαδίκτυο (web scraping) αναγνωρίζεται ευρέως ως μια αποτελεσματική και ισχυρή τεχνική συλλογής δεδομένων.

Οι τρέχουσες τεχνικές εξαγωγής των δεδομένων από το διαδίκτυο είναι ικανές να μετατρέψουν ολόκληρους ιστότοπους σε καλά οργανωμένα σύνολο δεδομένων. Η διαδικασία απόκτησης δεδομένων από το Διαδίκτυο μπορεί να χωριστεί σε δύο διαδοχικά βήματα. Το πρώτο είναι η απόκτηση δεδομένων από το διαδίκτυο και στη συνέχεια η εξαγωγή των επιθυμητών πληροφοριών από τα ληφθέντα δεδομένα. Συγκεκριμένα, η διαδικασία ξεκινά με τη σύνθεση ενός HTTP αιτήματος για απόκτηση πόρων από έναν στοχευμένο δικτυακό τόπο. Αυτό το αίτημα μπορεί να μορφοποιηθεί είτε σε διεύθυνση URL που περιέχει ένα ερώτημα GET ή ένα κομμάτι HTTP μηνύματος που περιέχει ένα ερώτημα POST. Μόλις το αίτημα παραληφθεί με επιτυχία και υποβληθεί σε επεξεργασία απο τον ιστότοπο τα δεδομένα θα ανακτηθούν και θα σταλούν πίσω. Τα δεδομένα μπορεί να είναι σε πολλές μορφές, όπως ιστοσελίδες που είναι κατασκευασμένες από HTML, ροές δεδομένων σε XML ή JSON μορφή ή δεδομένα πολυμέσων όπως εικόνες, ήχος, ή αρχεία βίντεο. Μετά την λήψη των δεδομένων η διαδικασία συνεχίζει να αναλύει, αναδιαμορφώνει και οργανώνει τα δεδομένα. Υπάρχουν δύο βασικές ενότητες ενός εξαγωγέα δεδομένων - η πρώτη αφορά την σύνθεση ενός αιτήματος HTTP και η δεύτερη την ανάλυση και εξαγωγή πληροφοριών από ακατέργαστο κώδικα HTML.

2.1.1 Scrapy

Από τους διάφορους τύπους διαδικασιών εξαγωγής δεδομένων από το διαδίκτυο, μερικοί έχουν δημιουργηθεί για να αναγνωρίζουν αυτόματα τη δομή δεδομένων μιας σελίδας, όπως το Scrapy.

Το Scrapy, γραμμένο σε Python, επιταχύνει τη διαδικασία κατασκευής και κλιμάκωσης μεγάλων έργων και είναι έως και 20 φορές ταχύτερο από άλλες παρόμοιες διαδικασίες ενώ χρησιμοποιεί λιγότερους υπολογιστικούς πόρους. Το Scrapy αποτελείται από διαδικασίες εργάτες που αποκαλούνται ”spiders”, όπου στέλνουν αιτήματα στον υπεύθυνο για τον έλεγχο της ροής δεδομένων. Αυτά τα αιτήματα είναι υπεύθυνα για το εκκίνημα της εξαγωγής των δεδομένων. Ο υπεύθυνος για τον έλεγχο της ροής δεδομένων στέλνει τα αιτήματα στον χρονορογραμματιστή, ο οποίος είναι υπεύθυνος για τη συλλογή και αποστολή αιτημάτων που υποβάλλονται. Οι αιτήσεις αποστέλλονται από τον χρονοπρογραμματιστή στον υπεύθυνο για τον έλεγχο της ροής και στη συνέχεια, το πρόγραμμα λήψης κατεβάζει την ζητούμενη ιστοσελίδα, δημιουργεί μια απάντηση και την στέλνει πίσω στη υπεύθυνο συλλογής δεδομένων. Ο υπεύθυνος ροής δεδομένων στέλνει την απόκριση από το πρόγραμμα λήψης στο αντίστοιχο εργάτη που δημιούργησε το αίτημα. Ο εργάτης εξάγει τα στοιχεία που χρειάζεται και στέλνει τα αιτήματα στον υπεύθυνο ροής δεδομένων ο οποίος στέλνει τα εξαγόμενα δεδομένα για περαιτέρω επεξεργασία ή αποθήκευση. Αυτά τα βήματα επαναλαμβάνονται έως ότου δεν διατίθενται περαιτέρω αιτήματα στο χρονοπρογραμματιστη.

2.2 Εξόρυξη Δεδομένων

Η ανάπτυξη της πληροφορικής έχει δημιουργήσει μεγάλο αριθμό βάσεων δεδομένων και τεράστια αριθμό δεδομένων σε διάφορες περιοχές του διαδικτύου. Η έρευνα σε βάσεις δεδομένων και σε τεχνολογίες ανάλυσης πληροφοριών οδήγησε σε μια προσέγγιση αποθήκευσης και περαιτέρω χειρισμού αυτών των δεδομένων για την λήψη αποφάσεων. Η εξόρυξη δεδομένων είναι μια διαδικασία εξαγωγής της χρήσιμης πληροφορίας και μοτίβων από έναν μεγάλο όγκο δεδομένων. Διάφοροι αλγόριθμοι και τεχνικές όπως ταξινόμηση, ομαδοποίηση, παλινδρόμηση, τεχνητή Νοημοσύνη, νευρωνικά δίκτυα, κανόνες σύνδεσης, δέντρα αποφάσεων, πλησιέστερος γείτονας χρησιμοποιούνται για την ανακάλυψη μοτίβων και σημαντικών πληροφοριών από βάσεις δεδομένων.

Η ταξινόμηση είναι η πιο συχνά εφαρμοζόμενη τεχνική εξόρυξης δεδομένων, η οποία χρησιμοποιεί ένα σύνολο προ-ταξινομημένων παραδειγμάτων για την ανάπτυξη ενός μοντέλου που μπορεί να ταξινομήσει ένα σύνολο δεδομένων. Η διαδικασία ταξινόμησης δεδομένων περιλαμβάνει την εκμάθηση και έπειτα την ταξινόμηση. Εάν η ακρίβεια είναι αποδεκτή τότε οι κανόνες μπορούν να εφαρμοστούν στις νέες συστάδες δεδομένων.

Η ομαδοποίηση μπορεί να θεωρηθεί ως αναγνώριση παρόμοιων κατηγοριών αντικειμένων. Χρησιμοποιώντας τεχνικές ομαδοποίησης μπορούμε να εντοπίσουμε περαιτέρω πυκνές και αραιές περιοχές στον χώρο των αντικειμένων και να ανακαλύψουμε το συνολικό μοτίβο κατανομής και συσχετίσεις μεταξύ των χαρακτηριστικών δεδομένων. Η τεχνική αυτή όμως, πολύ συχνά θεωρείται δαπανηρή.

Μια άλλη τεχνική είναι η ανάλυση παλινδρόμησης η οποία μπορεί να χρησιμοποιηθεί για τη μοντελοποίηση της σχέσης μεταξύ μιας ή περισσότερων ανεξάρτητων μεταβλητών και εξαρτημένων μεταβλητών. Στην εξόρυξη δεδομένων οι ανεξάρτητες μεταβλητές είναι γνωστά χαρακτηριστικά και οι μεταβλητές απόκρισης είναι αυτές που θέλουμε να προβλέψουμε. Δυστυχώς, πολλά προβλήματα στον πραγματικό κόσμο δεν είναι απλώς πρόβλεψη. Για παράδειγμα, ο όγκος των πωλήσεων, οι τιμές των μετοχών και τα ποσοστά αποτυχίας προϊόντος είναι πολύ δύσκολο να προβλεφθούν, επειδή μπορεί να εξαρτώνται

από πολύπλοκες αλληλεπιδράσεις πολλαπλών μεταβλητών πρόβλεψης. Επομένως, πιο περίπλοκες τεχνικές (π.χ. λογιστική παλινδρόμηση, δέντρα αποφάσεων ή νευρωνικά δίχτυα) μπορεί να είναι απαραίτητες για την πρόβλεψη μελλοντικών τιμών.

Τα νευρωνικά δίκτυα είναι ένα σύνολο συνδεδεμένων μονάδων εισόδου/εξόδου και κάθε σύνδεση έχει κάποιο βάρος. Κατά τη διάρκεια της μαθησιακής φάσης, το δίκτυο μαθαίνει προσαρμόζοντας τα βάρη έτσι ώστε να είναι σε θέση να προβλέψει τις σωστές ετικέτες τάξης των συστάδων εισόδου. Τα νευρωνικά δίκτυα έχουν την αξιοσημείωτη ικανότητα να αντλούν πληροφορία από περίπλοκα ή ανακριβή δεδομένα και μπορούν να χρησιμοποιηθούν για την εξαγωγή μοτίβων και τον εντοπισμό τάσεων που είναι πολύ περίπλοκες για να παρατηρηθούν είτε από ανθρώπους είτε από άλλες τεχνικές υπολογιστών.

Τέλος, μια ακόμη τεχνική είναι η συσχέτιση η οποία είναι συνήθως η εύρεση κοινών στοιχείων μεταξύ μεγάλων συνόλων δεδομένων.

2.2.1 Εξόρυξη Κειμένου

Η εξόρυξη κειμένου διαφέρει από την εξόρυξη δεδομένων. Αυτό που διακρίνει την εξόρυξη κειμένου από την εξόρυξη δεδομένων είναι ότι ασχολείται με δομημένα δεδομένα. Το κύριο πρόβλημα της είναι ότι τα αποτελέσματα αναζήτησης δεν σχετίζονται με τις απαιτήσεις του χρήστη. Μια λύση είναι να χρησιμοποιηθεί η εξόρυξη κειμένου για να εξάγουμε τις σχετικές πληροφορίες, οι οποίες δεν αναφέρονται ρητά. Αποδείχθηκε ότι οι αποθήκες δεδομένων είναι επιτυχής όσον αφορά την αποθήκευση αριθμητικών πληροφοριών, αλλά ανεπιτυχής όσον αφορά πληροφορίες κειμένου λόγω του υπερβολικά μεγάλου όγκου της.

Η εξόρυξη κειμένου είναι ένα πεδίο που συνδυάζει την εξόρυξη δεδομένων με την ανάκτηση πληροφορίας και την επεξεργασία φυσικής γλώσσας. Μπορεί να εκφραστεί μέσα από δύο συγκεκριμένες φάσεις, αυτή της εκκαθάρισης κειμένου και εκείνη της εξαγωγής γνώσης. Στην πρώτη φάση τα αρχεία κειμένου μετατρέπονται σε μια ειδική ενδιάμεση (ημιδομημένη) μορφή την οποία χρησιμοποιεί η δεύτερη φάση για καταλήξει στην ανακάλυψη χρήσιμης γνώσης. Αυτή η ενδιάμεση μορφή μπορεί να είναι βασισμένη είτε σε έγγραφα (όπου εξάγονται οι σχέσεις μεταξύ ενός συνόλου εγγράφων π.χ. για κατηγοριοποίηση ή συσταδοποίηση των δεδομένων) είτε σε έννοιες (όπου εξάγονται οι σχέσεις μεταξύ διαφορετικών αντικειμένων ή εννοιών π.χ. για μοντελοποίηση προβλέψεων ή αναζήτηση συσχετίσεων στα δεδομένα).

Τεχνολογίες που χρησιμοποιούνται για την εξόρυξη κειμένου περιλαμβάνουν την εξαγωγή πληροφοριών, την σύνοψη κειμένου, την κατηγοριοποίηση κειμένου και την ομαδοποίηση κειμένου. Η εξαγωγή πληροφοριών είναι το αρχικό βήμα ανάλυσης του μη δομημένου κειμένου, αναγνωρίζοντας βασικές φράσεις και σχέσεις μέσα στο κείμενο. Αυτό είναι πολύ χρήσιμο όταν υπάρχουν μεγάλοι όγκοι δεδομένων. Η σύνοψη κειμένου είναι η μείωση του μήκους και της λεπτομέρειας του εγγράφου, διατηρώντας παράλληλα τα πιο σημαντικά στοιχεία του. Η κατηγοριοποίηση εκχωρεί αυτόματα μία ή περισσότερες κατηγορίες στο έγγραφο. Η κατηγοριοποίηση είναι μέθοδος εποπτευόμενης μάθησης επειδή βασίζεται σε παραδείγματα που έχουν κατηγοριοποιηθεί προηγουμένως ως είσοδο για την ταξινόμηση νέων εγγράφων. Τέλος, η μέθοδος ομαδοποίησης μπορεί να χρησιμοποιηθεί για την εύρεση ομάδων εγγράφων με παρόμοιο περιεχόμενο.

Η πολυπλοκότητα της φυσικής γλώσσας είναι το κύριο πρόβλημα στην εξόρυξη κει-

μένου (όπως για παράδειγμα ο σαρκασμός, οι λέξεις με πολλαπλές σημασίες, ή τα κοινωνικοπολιτικά συμφραζόμενα). Μια λέξη μπορεί να έχει πολλαπλές σημασίες και πολλές λέξεις μπορούν να έχουν το ίδιο νόημα. Όταν το κείμενο μπορεί να γίνει κατανοητό με δύο ή περισσότερους πιθανούς τρόπους, τότε δημιουργεί μια ασάφεια . Αυτή η ασάφεια οδηγεί σε μη ορθή εξαγωγή πληροφορίας. Η ασάφεια δεν μπορεί να εξαλειφθεί πλήρως από την φυσική γλώσσα και ενώ γίνονται προσπάθειες για την επίλυση αυτού του προβλήματος, οι έρευνες βρίσκονται ακόμη σε πρώιμα στάδια.

Η διαδικασία εξόρυξης κειμένου περιγράφεται παρακάτω μέσα από έναν αριθμό βημάτων.

Βήμα Εξόρυξης Κειμένου	**Περιγραφή Βήματος**
1. Συλλογή Δεδομένων	Τα δεδομένα συλλέγονται απο την πηγή. Τα δεδομένα μπορεί να είναι HTML σελίδες, documents, σχόλια χρηστών στο διαδίκτυο, άρθρα ή οποιαδήποτε άλλη μορφή πληροφορίας. Συνήθως εξάγεται με κάποιον crawler. Στην συγκεκριμένη περίπτωση τα δεδομένα είναι άρθρα και συλλέγονται με Scrapy.
2. Προεπεξεργασία και Φιλτράρισμα Κειμένου	Μετατροπή ενός εγγράφου σε ένα σύνολο χαρακτηριστικών που αναπαριστούν στοιχεία κειμένου, φιλτράροντας τα από τα στοιχεία που κρίνονται ακατάλληλα. Για αυτό το βήμα χρησιμοποιείται συχνά η επεξεργασίας φυσικής γλώσσας ή πιο απλές τεχνικές όπως η εξαγωγή κειμένου με βάση λέξεις-κλειδιά. Στην παρούσα εργασία χρησιμοποιείται η δεύτερη τακτική για να εξάγει μόνο τις σχετικές πληροφορίες απο τις HTML σελίδες.
3. Επιλογή Χαρακτηριστικών	Επιλογή των πιο σχετικών χαρακτηριστικών από την συλλογή για την δημιουργία του μοντέλου στο επόμενο βήμα χωρίς τα περιττά ή τετριμμένα χαρακτηριστικά να επηρεάζουν την διαδικασία.
4. Εξόρυξη Κειμένου	Δημιουργία και εφαρμογή του μοντέλου εξόρυξης γνώσης. Στην παρούσα περίπτωση η κατηγοριοποίηση κειμένου άρθρων εγκλήματος σε κατηγορίες.

Συνέχεια στην επόμενη σελίδα

Πίνακας 1 – η συνέχεια της προηγούμενης σελίδας

Βήμα Εξόρυξης Κειμένου	Περιγραφή Βήματος
5. Αξιολόγηση Αποτελεσμάτων	Ανάλυση επίδοσης και αποτελεσματικότητας της εξόρυξης μέσω διάφορων μετρικών.

Πίνακας 1. Βήματα Εξόρυξης Κειμένου

2.2.2 Αποθήκευση Δεδομένων - MongoDB

Κατά την τελευταία δεκαετία, σημειώθηκε τεράστια αύξηση στον όγκο δεδομένων που αποθηκεύουν οι βάσεις δεδομένων. Αυτό κατέστησε τα μονολιθικά συστήματα βάσεων δεδομένων αδύνατα να ανταποκριθούν στις σημερινές απαιτήσεις. Οι σχεσιακές βάσεις δεδομένων συχνά αντικαθίστανται από άλλες εναλλακτικές λύσεις, όπως βάσεις δεδομένων NoSQL, για λόγους επεκτασιμότητας και ανομοιογένειας. Πρωταρχικός στόχος των συστημάτων βάσεων δεδομένων NoSQL είναι η ομοιόμορφη διανομή δεδομένων.

Τα NoSQL μοντέλα δεδομένων χωρίζονται σε 3 διαφορετικές κατηγορίες. Η πρώτη αφορά το κλειδί-δείκτης (key-value) μοντέλο. Σε αυτό το μοντέλο χρησιμοποιείται ένας πίνακας κατακερματισμού όπου υπάρχει ένα μοναδικό κλειδί και ένας δείκτης για ένα συγκεκριμένο στοιχείο δεδομένων. Είναι το πιο απλό και εύκολο προς υλοποίηση αλλά είναι αναποτελεσματικό όταν ενδιαφερόμαστε για την αναζήτηση ή ενημέρωση μέρους μιας τιμής. Ένα παράδειγμα είναι το Redis.

Η δεύτερη κατηγορία κατηγοριοποιεί τις στήλες σε ευρύτερες κατηγορίες (column family). Αυτές δημιουργήθηκαν για την αποθήκευση και την επεξεργασία πολύ μεγάλων όγκων δεδομένων που διανέμονται σε πολλά μηχανήματα. Παράδειγμα αυτής της κατηγορίας είναι η HBase και η Cassandra.

Μια ακόμη κατηγορία είναι οι βάσεις δεδομένων βασισμένες σε έγγραφα (document databases). Είναι παρόμοιες με την πρώτη κατηγορία (κλειδί-δείκτης) με την διαφορά ότι επιτρέπουν ένθετες τιμές που σχετίζονται με το κάθε κλειδί. Τα ημι-δομημένα έγγραφα αποθηκεύονται σε μορφές όπως JSON και είναι πιο γρήγορες και αποτελεσματικές. Σε αυτή την κατηγορία ανήκει η MongoDB.

Τέλος, υπάρχουν οι βάσεις δεδομένων που βασίζονται στην θεωρία γραφημάτων (graph database) για να αποθηκεύουν, χαρτογραφούν και να διερευνούν σχέσεις μεταξύ των δεδομένων. Ένα παράδειγμα είναι η Neo4j.

Η MongoDB, μια NoSQL βάση δεδομένων, είναι μια ευέλικτη βάση δεδομένων που δημιουργήθηκε για επεκτασιμότητα, απόδοση και υψηλή διαθεσιμότητα. Η MongoDB και οι στρατηγικές ανθεκτικότητας της, έχουν σχεδιαστεί για υψηλή ανάγνωση και εγγραφή καθώς και εύκολη δυνατότητα κλιμάκωσης. Είναι εύκολη στην χρήση και υποστηρίζει μη δομημένα δεδομένα και δεν απαιτεί δαπανηρές και χρονοβόρες μετακινήσεις όταν αλ-

λάζουν οι απαιτήσεις εφαρμογής. Τα έγγραφα της MongoDB είναι κωδικοποιημένα σε μορφή JSON που ονομάζεται BSON και είναι πολύ γρήγορη και ελαφριά μορφή αποθήκευσης δεδομένων. Τα άλλα χαρακτηριστικά της περιλαμβάνουν αυτόματο sharding, αντιγραφή και εύκολη αποθήκευση.

2.3 Κατηγοριοποίηση Άρθρων

Με την αυξανόμενη διαθεσιμότητα ηλεκτρονικών εγγράφων και την ταχεία ανάπτυξη του Παγκόσμιου Ιστού, έγινε αναγκαία η αυτόματη κατηγοριοποίηση κειμένων. Η επεξεργασία φυσικής γλώσσας, η εξόρυξη δεδομένων και οι τεχνικές μηχανικής μάθησης συνεργάζονται για την αυτόματη ταξινόμηση και ανακάλυψη μοτίβων από το ηλεκτρονικά έγγραφα. Οι αλγόριθμοι μηχανικής μάθησης μπορούν να χωριστούν σε αλγόριθμοι με επίβλεψη, χωρίς επίβλεψη και αλγόριθμοι βασισμένοι στην ενισχυτική μάθηση.

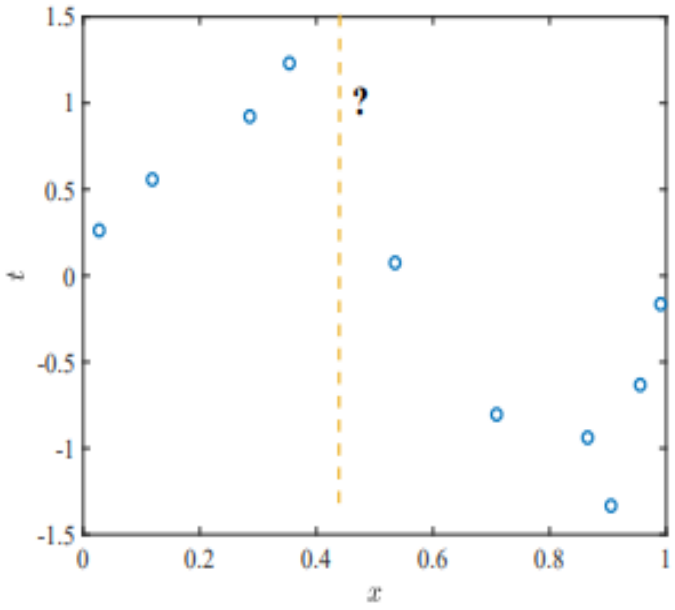
2.3.1 Κατηγορίες Αλγορίθμων Κατηγοριοποίησης

Οι αλγόριθμοι με επίβλεψη περιλαμβάνουν ένα σύνολο δεδομένων προετοιμασίας με βάση το οποίο ο αλγόριθμος μαθαίνει να κατηγοριοποιεί τα δεδομένα που θα του δοθούν αργότερα. Τα δεδομένα προετοιμασίας έχουν ταξινομηθεί και σε κάθε κείμενο έχει δοθεί η αντίστοιχη κατηγορία στην οποία ανήκει. Η εποπτευόμενη μάθηση μπορεί να διαχωριστεί περαιτέρω σε ταξινόμηση και παλινδρόμηση. Οι μέθοδοι ταξινόμησης χρησιμοποιούνται για να προβλέψουν κατηγορηματικές τιμές και οι μέθοδοι παλινδρόμησης χρησιμοποιούνται για την πρόβλεψη αριθμητικών ιδιοτήτων. Οι πιο γνωστές μέθοδοι παλινδρόμησης είναι η πολυωνυμική παλινδρόμηση και η γραμμική παλινδρόμηση. Οι πιο γνωστές μέθοδοι ταξινόμησης είναι οι πλησιέστεροι γείτονες, τα δέντρα απόφασης και η λογιστική παλινδρόμηση. Ως απεικόνιση, στην εικόνα 4(a), οι είσοδοι είναι σημεία στο δισδιάστατο επίπεδο, οι έξοδοι είναι οι ετικέτες που εκχωρούνται σε κάθε είσοδο (κύκλοι ή σταυροί) και ο στόχος είναι η εκμάθηση ενός δυαδικού ταξινομητή.

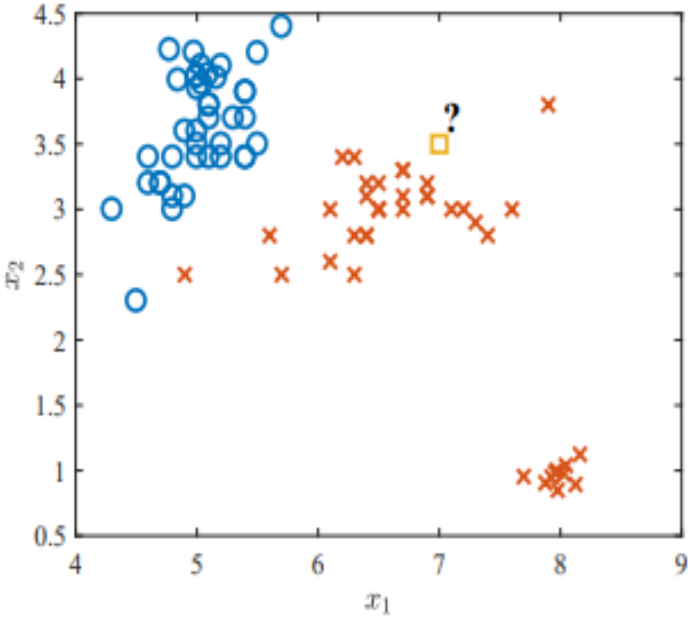
Μπορούμε να διακρίνουμε δύο κατηγορίες προβλημάτων εποπτευόμενης μάθησης ανάλογα με το εάν τα αποτελέσματα είναι συνεχείς ή διακριτές μεταβλητές. Στην πρώτη περίπτωση, έχουμε ένα πρόβλημα παλινδρόμησης (regression), ενώ στην τελευταία έχουμε μια πρόβλημα ταξινόμησης (classification).

Όπως απεικονίζεται στην εικόνα 2, σε ένα πρόβλημα παλινδρόμησης, μας δίνεται ένα σύνολο εκπαίδευσης Δ των σημείων εκπαίδευσης N (χ_n, τ_n) με $n = 1, ..., N$ όπου οι μεταβλητές χ_n είναι οι είσοδοι, επίσης γνωστές ως συντεταγμένες ή επεξηγηματικές μεταβλητές. ενώ οι μεταβλητές τ_n είναι οι έξοδοι, επίσης γνωστές ως εξαρτημένες μεταβλητές, ετικέτες ή αποκρίσεις. Στην παλινδρόμηση, τα αποτελέσματα είναι συνεχείς μεταβλητές. Το πρόβλημα είναι να προβλεφθεί η έξοδος τ για μια νέα είσοδος χ.

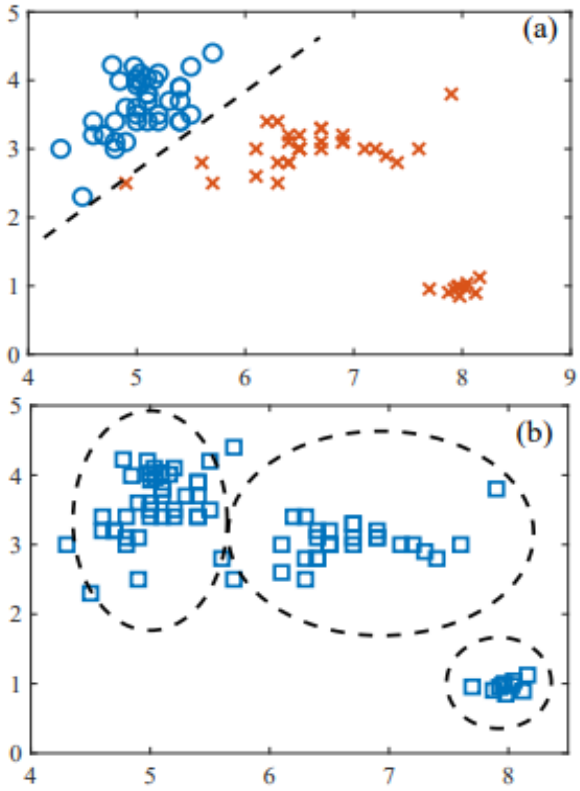
Όπως απεικονίζεται στην εικόνα 3, η ταξινόμηση ορίζεται ομοίως με τη μόνη διαφορά ότι οι έξοδοι τ είναι διακριτές μεταβλητές που λαμβάνουν έναν πεπερασμένο αριθμό πιθανών τιμών. Η τιμή της εξόδου τ για μια δεδομένη είσοδο χ δείχνει την κλάση στην οποία ανήκει το χ. Για παράδειγμα, η ετικέτα τ είναι δυαδική μεταβλητή όπως φαίνεται στην εικόνα 4 για ένα πρόβλημα δυαδικής ταξινόμησης. Με βάση το σύνολο προετοιμασίας (train set) Δ, ο στόχος είναι να προβλεφθεί η ετικέτα, ή η τάξη, τ για μια νέα είσοδο χ.



Εικόνα 2. Απεικόνιση του προβλήματος της παλινδρόμησης



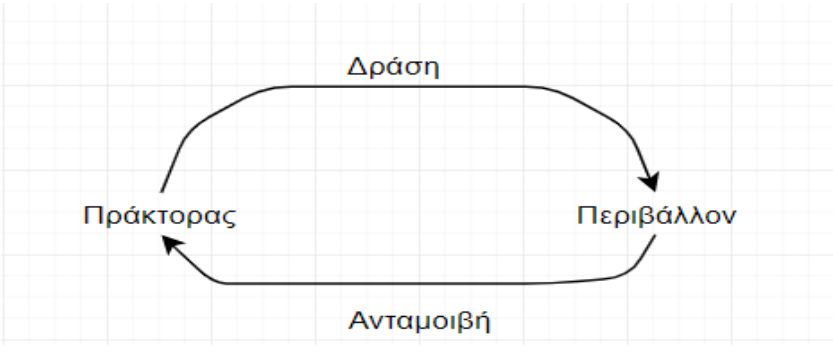
Εικόνα 3. Απεικόνιση του προβλήματος της ταξινόμησης



Εικόνα 4. a) Μάθηση με επίβλεψη και b) μάθηση χωρίς επίβλεψη

Η μάθηση χωρίς επίβλεψη περιλαμβάνει πολλούς αλγόριθμους που χρησιμοποιούνται για την εξαγωγή συμπερασμάτων από σύνολα δεδομένων που δεν περιλαμβάνουν δεδομένα προετοιμασίας. Άρα σε αυτή την περίπτωση, δεν έχουμε σύνολο δεδομένων που περιλαμβάνει δεδομένα με τις σωστές αντίστοιχες κατηγορίες τους. Οι δημοφιλείς αλγόριθμοι μάθησης χωρίς επίβλεψη είναι η κύρια ανάλυση συστατικών στοιχείων, kmeans και η ομαδοποίηση. Στην εικόνα 4(b), οι είσοδοι είναι και πάλι σημεία στο δισδιάστατο επίπεδο, αλλά δεν παρέχεται καμία ένδειξη από τα δεδομένα σχετικά με την αντίστοιχη επιθυμητή έξοδο. Η μη επιτηρούμενη μάθηση στοχεύει γενικά στην ανακάλυψη ιδιοτήτων του μηχανισμού δημιουργίας των δεδομένων. Στο παράδειγμα της Εικ. 4(b), ο στόχος της μη εποπτευόμενης μάθησης είναι να συγκεντρωθούν σημεία εισόδου που είναι κοντά το ένα στο άλλο, ορίζοντας έτσι μια ετικέτα - το δείκτη συμπλέγματος - σε κάθε σημείο εισόδου (οι συστάδες οριοθετούνται από διακεκομμένες γραμμές). Στην μη εποπτευόμενη μάθηση, μας δίνεται ένα σετ εκπαίδευσης Δ που αποτελείται από N δείγματα $\chi_n \sim p(x)$ με $n = 1, ..., N$ που παράγεται από μια άγνωστη πραγματική κατανομή $p(x)$ και στόχος είναι να μάθουμε μερικές χρήσιμες ιδιότητες της κατανομής αυτής.

Οι αλγόριθμοι βασισμένοι στην ενισχυτική μάθηση βρίσκονται, κατά μια έννοια, μεταξύ εποπτευόμενης και μη εποπτευόμενης μάθησης. Σε αντίθεση με τη μη επιτηρούμενη μάθηση, υπάρχει κάποια μορφή εποπτείας, αλλά δεν έρχεται με τη μορφή της προδιαγραφής μιας επιθυμητής εξόδου για κάθε είσοδο στα δεδομένα. Αντί για αυτό, ένας αλγόριθμος εκμάθησης ενίσχυσης λαμβάνει ανατροφοδότηση από το περιβάλλον μόνο αφού επιλέξει μια έξοδο για μια δεδομένη είσοδο. Ο πράκτορας (Agent) είναι η οντότητα που μαθαίνει ενώ η ανατροφοδότηση δείχνει τον βαθμό στον οποίο η παραγωγή,



Εικόνα 5. Ενισχυτική Μάθηση

γνωστή ως δράση στην ενισχυτική μάθηση, εκπληρώνει τους στόχους του agent. Η ενισχυτική μάθηση εφαρμόζεται σε διαδοχικά προβλήματα λήψης αποφάσεων στα οποία ο εκπαιδευόμενος αλληλεπιδρά με ένα περιβάλλον λαμβάνοντας διαδοχικά ενέργειες - τα αποτελέσματα - με βάση τις παρατηρήσεις του - τις εισόδους του - ενώ λαμβάνει σχόλια σχετικά με κάθε επιλεγμένη ενέργεια. Το σύστημα έχει ως στόχο τη μεγιστοποίηση της ”ανταμοιβής” που λαμβάνει.

2.3.2 Αξιολόγηση Κατηγοριοποίησης

Οι δείκτες απόδοσης είναι πολύ χρήσιμοι όταν ο στόχος είναι η αξιολόγηση και σύγκριση διαφορετικών μοντέλων ταξινόμησης ή τεχνικών μηχανικής μάθησης. Η ακρίβεια χρησιμοποιείται συνήθως για τη μέτρηση του ποσοστού των σωστά ταξινομημένων δοκιμαστικών δεδομένων (test instances). Είναι μέχρι τώρα η κύρια μέτρηση για την αξιολόγηση της απόδοσης ενός ταξινομητή.

Πολλές μετρήσεις βασίζονται στον πίνακα σύγχυσης (Confusion Matrix), καθώς περιλαμβάνει όλες τις σχετικές πληροφορίες σχετικά με τον αλγόριθμο και την απόδοση του κανόνα ταξινόμησης. Ο πίνακας σύγχυσης είναι ένας εγκάρσιος πίνακας που καταγράφει τον αριθμό των εμφανίσεων μεταξύ δύο βαθμολογητών, την πραγματική ταξινόμηση και την προβλεπόμενη ταξινόμηση, όπως φαίνεται στο Σχήμα 6 όπου το κάθε στοιχείο $f_{i,j}$ υποδηλώνει το πλήθος των στιγμιότυπων που ανήκουν στην κλάση i και προβλέφθηκαν στην κλάση j.

	Πρόβλεψη Αρνητικής Κλάσης	**Πρόβλεψη Θετικής Κλάσης**
Πραγματική κατηγορία (+)	f_{++} (True Positive)	f_{+-} (False Negative)
Πραγματική κατηγορία (-)	f_{-+} (False Positive)	f_{--} (True Negative)

Πίνακας 2. Πίνακας Σύγχυσης Δυαδικής Κατηγοριοποίησης

Η ακρίβεια (Precision) είναι το κλάσμα των αληθινών θετικών στοιχείων διαιρούμενο με τον συνολικό αριθμό των θετικά προβλεπόμενων μονάδων (άθροισμα στήλης των προβλεπόμενων θετικών).Η Ακρίβεια δηλώνει κατά πόσο μπορούμε να εμπιστευτούμε το μοντέλο όταν προβλέπει ένα στοιχείο ως θετικό. Συγκεκριμένα, το True Positive είναι τα στοιχεία που έχουν χαρακτηριστεί ως θετικά από το μοντέλο και είναι πραγματικά θετικά, ενώ το False Positive είναι τα στοιχεία που έχουν χαρακτηριστεί ως θετικά από το μοντέλο, αλλά στην πραγματικότητα είναι αρνητικά.

$$Precision = \frac{TP}{TP + FP}$$

Η ανάκληση (Recall) είναι το κλάσμα στοιχείων True Positive διαιρούμενο με τον συνολικό αριθμό θετικά ταξινομημένων μονάδων (άθροισμα σειράς των πραγματικών θετικών). Συγκεκριμένα, το False Negative είναι τα στοιχεία που έχουν χαρακτηριστεί ως αρνητικά από το μοντέλο, αλλά είναι πραγματικά θετικά. Η ανάκληση μετρά την προγνωστική ακρίβεια του μοντέλου για τη θετική κατηγορία.

$$Recall = \frac{TP}{TP + FN}$$

Η Πιστότητα (accuracy) είναι μια από τις πιο δημοφιλείς μετρήσεις στην ταξινόμηση πολλαπλών κατηγοριών και υπολογίζεται απευθείας από τον πίνακα σύγχυσης. Η πιστότητα επιστρέφει ένα συνολικό μέτρο για το πόσο σωστά το μοντέλο προβλέπει σε όλο το σύνολο δεδομένων. Το βασικό στοιχείο της μέτρησης είναι τα μεμονωμένα στοιχεία στο σύνολο δεδομένων: κάθε μονάδα έχει το ίδιο βάρος και συμβάλλει εξίσου στην τιμή ακρίβειας. Επομένως, ταιριάζει περισσότερο όταν ενδιαφερόμαστε μόνο για μεμονωμένα στοιχεία αντί για πολλαπλές τάξεις. Αυτό σημαίνει είναι ιδανική μετρική όταν ενδιαφερόμαστε για την πρόβλεψη του υψηλότερου αριθμού στοιχείων στη σωστή τάξη, χωρίς να με νοιάζει η κατανομή των τάξεων.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Η αποτελεσματικότητα της κατηγοριοποίησης δεν είναι αποκλειστική και αντίστοιχη μόνο της εκπαίδευσης του μοντέλου αλλά και στο ίδιο το σύνολο των στιγμιοτύπων. Δεν είναι σπάνια η περίπτωση ύπαρξης θορύβου στην μορφή ενός στιγμιότυπου που έχει ταξινομηθεί σε μία κλάση βάσει των χαρακτηριστικών του, τα οποία τυχαίνει να μην εμφανίζονται σε κανένα άλλο στιγμιότυπο άλλης κλάσης στο σύνολο εκπαίδευσης. Αυτό έχει σαν αποτέλεσμα την λανθασμένη κατηγοριοποίηση ενός στιγμιότυπου στο σύνολο ελέγχου σε μια συγκεκριμένη κλάση απλά και μόνο επειδή περιέχει χαρακτηριστικά ενός δείγματος-θορύβου από το σύνολο εκπαίδευσης. Αυτή η λανθασμένη γενίκευση βασισμένη στα θορυβώδη χαρακτηριστικά των δεδομένων εκπαίδευσης ονομάζεται υπερπροσαρμογή (overfitting), και συνεπάγεται ουσιαστικά την απομνημόνευση των δεδομένων εκπαίδευσης μαζί με τον θόρυβο τους από το μοντέλο με αποτέλεσμα την αύξηση του σφάλματος ταξινόμησης. Το πιο κοινό αίτιο της υπερπροσαρμογής κατά την κατηγοριοποίηση είναι το μικρό μέγεθος του συνόλου εκπαίδευσης (όταν δηλαδή υπάρχουν λίγα αντιπροσωπευτικά στιγμιότυπα για κάθε κλάση ή πολύς θόρυβος, ώστε ο τελευταίος να έχει πολύ μεγάλη πιθανότητα να γίνει η βάση των προβλέψεων του μοντέλου).

Στην πράξη, συνιστάται πάντα να συγκρίνονται διαφορετικά μοντέλα ταξινόμησης για ένα συγκεκριμένο σύνολο δεδομένων και να λαμβάνονται υπόψη οι επιδόσεις πρόβλεψης καθώς και η υπολογιστική απόδοση. Στην περίπτωση της παρούσας έρευνας, πραγματοποιείται κατηγοριοποίηση πολλαπλών τάξεων καθώς το κάθε άρθρο μπορεί να ανήκει σε μία από τις 5 διαθέσιμες κατηγορίες. Δοκιμάζονται διάφοροι ταξινομητές και επιλέγεται ο καλύτερος όσο αφορά την τελική ακρίβεια.

2.3.3 Support Vector Machines

Τα Support Vector Machines (SVMs) είναι σύνολο μεθόδων εποπτευόμενης μάθησης που χρησιμοποιούνται για την ταξινόμηση και την παλινδρόμηση. Ανήκουν σε μια οικογένεια γενικευμένης γραμμικής ταξινόμησης. Μια ειδική ιδιότητα του SVM είναι ότι ελαχιστοποιεί ταυτόχρονα το εμπειρικό σφάλμα ταξινόμησης και μεγιστοποιεί το γεωμετρικό περιθώριο. Το SVM μετατρέπει τα δεδομένα εισόδου σε υψηλότερο διαστατικό χώρο όπου κατασκευάζει ένα διαχωριστικό υπερπλάνο μεγίστου. Δύο παράλληλα υπερπλάνα κατασκευάζονται σε κάθε πλευρά του υπερπλάνου που διαχωρίζουν τα δεδομένα. Το διαχωριστικό υπερπλάνο είναι το υπερπλάνο που μεγιστοποιεί την απόσταση μεταξύ των δύο. Υποτίθεται ότι όσο μεγαλύτερο είναι το περιθώριο ή η απόσταση μεταξύ αυτών των παράλληλων υπερπλάνων τόσο μικρότερο θα είναι το σφάλμα γενίκευσης του ταξινομητή.

Έστω τα σημεία δεδομένων $(x1, y1), (x2, y2), (x3, y3), (x4, y4)..., (xn, yn)$. Όπου $yn =$ 1 ή -1, μια σταθερά που δηλώνει την κλάση στην οποία ανήκει αυτό το σημείο χ_n. n = αριθμός δείγματος. Κάθε χ_n είναι p-διαστατικό πραγματικό διάνυσμα. Η κλιμάκωση είναι σημαντική για την προστασία από μεταβλητές (χαρακτηριστικά) με μεγαλύτερη διακύμανση. Μπορούμε να δούμε αυτά τα δεδομένα εκπαίδευσης, μέσω του διαχωριστικού (ή διαχωρισμού) υπερπλάνου, το οποίο είναι της μορφής

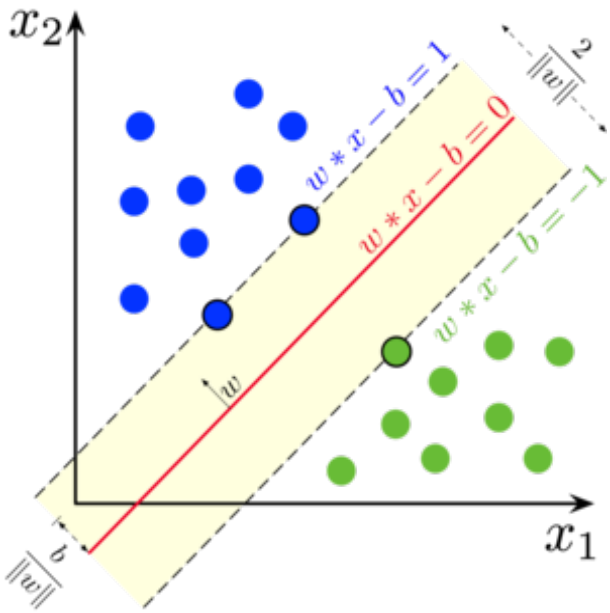
$$wx + b = o$$

Όπου το b είναι βαθμωτό και το w είναι διαστατικό διάνυσμα. Το w δείχνει κάθετα στο διαχωριστικό υπερπλάνο. Η προσθήκη της παραμέτρου b μας επιτρέπει να αυξήσουμε το περιθώριο. Αν απουσιάζει το b, το υπερκείμενο πρέπει να περάσει από την προέλευση, περιορίζοντας τη λύση. Καθώς μας ενδιαφέρει το μέγιστο περιθώριο, μας ενδιαφέρουν επίσης τα παράλληλα υπερπλάνα. Τα παράλληλα υπερπλάνα μπορούν να περιγραφούν με την εξίσωση

$$wx + b = 1$$

$$wx + b = -1$$

Όπως φαίνεται από την εικόνα 6, μπορεί να υπάρχουν πολλά διαχωριστικά υπερπλάνα. Το πρόβλημα της κατηγοριοποίησης μέσω SVM ανάγεται σε πρόβλημα καθορισμού του υπερπλάνου. Εάν υπάρχουν περισσότερα του ενός, θα πρέπει να ευρεθεί το καταλληλότερο. Θεωρούμε αποστάσεις μεταξύ του υπερπλάνου και πρότυπα και στις δύο πλευρές του. Για μια δεδομένη κλάση λαμβάνουμε ένα πρότυπο του οποίου η απόσταση από το υπερπλάνο είναι η ελάχιστη μέσα στην κλάση. Τα πρότυπα που αντιστοιχούν στις ελάχιστες αποστάσεις καλούνται Support Vectors (SVs), δηλαδή διανύσματα που ξεκινούν



Εικόνα 6. Support Vector Machines

από τις αρχές του συστήματος συντεταγμένων και τερματίζουν στο πρότυπο. Αυτά μπορεί να χρησιμοποιηθούν για να κατασκευαστούν τα συνοριακά υπερπλάνα για κάθε κλάση. Η περιοχή μεταξύ αυτών καλείται περιθώριο (margin) δεν περιέχει κάποιο πρότυπο και αντιστοιχεί στη μικρότερη απόσταση από το συνοριακό υπερπλάνο. Η μεγιστοποίηση της απόστασης μεταξύ των συνοριακών υπερπλάνων, δηλαδή με την αύξηση του πλάτους του περιθωρίου, διαμορφώνονται ισομερώς τα συνοριακά υπερπλάνα

Εάν τα δεδομένα εκπαίδευσης διαχωρίζονται γραμμικά, μπορούμε να επιλέξουμε αυτά τα υπερπλάνα έτσι ώστε να μην υπάρχουν σημεία μεταξύ τους και στη συνέχεια να προσπαθήσουμε να μεγιστοποιήσουμε την απόσταση τους. Με γεωμετρία, διαπιστώνουμε ότι η απόσταση μεταξύ του υπερπλάνου είναι 2 / |w|. Έτσι θέλουμε να ελαχιστοποιήσουμε το |w|. Για να διεγείρουν σημεία δεδομένων, πρέπει να διασφαλίσουμε το εξής

$$wxi - b \geqslant 1$$

ή

$$wxi - b \leqslant -1$$

Το οποίο μπορεί να γραφτεί ως εξής

$$yi(wxi - b) \geqslant 1, 1 \leqslant i \leqslant n \tag{2.1}$$

Τα δείγματα κατά μήκος των υπερπλάνων ονομάζονται Support Vectors (SVs). Ένα διαχωριστικό υπερπλάνο με το μεγαλύτερο περιθώριο που ορίζεται από το M = 2 / $\left|\mathrm{w}\right|$ που καθορίζει διανύσματα υποστήριξης σημαίνει σημεία δεδομένων εκπαίδευσης πλησιέστερα σε αυτό. Ικανοποιεί την εξής σχέση

$$y_{\mathrm{i}}[wTx_{\mathrm{i}} + b] = 1, i = 1 \tag{2.2}$$

Το Optimal Canonical Hyperplane (OCH) είναι ένα κανονικοποιημένο υπερπλάνο με μέγιστο περιθώριο. Για όλα τα δεδομένα, πρέπει να ικανοποιεί τους ακόλουθους περιορισμούς

$$y_i[wTx_i + b] \geqslant 1, i = 1, 2...l \tag{2.3}$$

Όπου l είναι Αριθμός σημείου εκπαίδευσης. Προκειμένου να βρεθεί το βέλτιστο διαχωριστικό υπερπλάνο με μέγιστο περιθώριο, μια μηχανή εκμάθησης θα πρέπει να ελαχιστοποιήσει το $||w||^2$ με την επιφύλαξη των περιορισμών ανισότητας

$$[y_i[wTx_i + b] \geqslant, i = 1, 2...l \tag{2.4}$$

Αυτό το πρόβλημα βελτιστοποίησης λύθηκε με την χρήση της συνάρτησης Lagrange

$$L(w, b, a) = \frac{1}{2}||w|| - \sum_{n=1}^{N} \alpha_i[y_i(wx_i + b) - 1] \tag{2.5}$$

Όπου αi είναι ένας πολλαπλασιαστής Lagrange. Η αναζήτηση βέλτιστων σημείων $(w0, b0, \alpha 0)$ είναι απαραίτητη, επειδή οι πολλαπλασιαστές Lagrange πρέπει να ελαχιστοποιηθούν σε σχέση με τα w και b και πρέπει να μεγιστοποιηθεί σε σχέση με τα μη αρνητικά α ($\alpha i \geqslant 0$). Το $L(w, b, a)$ θα μεγιστοποιηθεί θεωρώντας την παρακάτω συνθήκη

$$\alpha_i[y_i(wx_i + b) - 1] = 0, i = 1, 2...N \tag{2.6}$$

που υπονοεί ότι $\alpha_i = 0$ εάν το x_i είναι support vector.

Λαμβάνοντας τη μερική παράγωγο ως προς το w και b προκύπτει ότι

$$\sum_{n=1}^{N} \alpha_i y_i y_i = 0 \tag{2.7}$$

Η (2.5) μπορεί να ξαναγραφτεί ως εξής

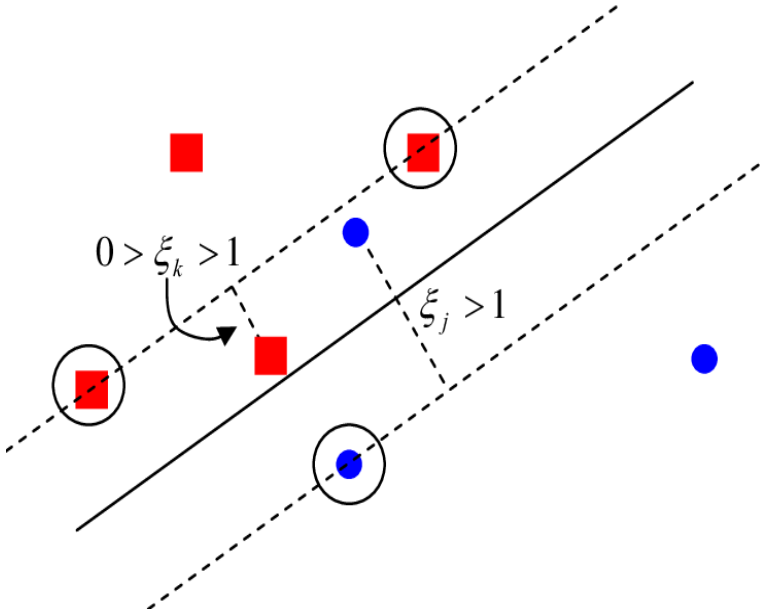
$$L(w, b, a) = \frac{1}{2}wTw - \sum_{i=1}^{N} \alpha_i[y_i wx_i - b\sum_{i=1}^{N} \alpha_i y_i + \sum_{n=1}^{N} \alpha_i \tag{2.8}$$

Η τελική συνάρτηση βελτιστοποίησης είναι

$$L(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\, \alpha_j\, y_i\, y_j\, x_i T x_i,\ \alpha_i \geqslant 0, i = 1, 2...N, \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{2.9}$$

Η παραπάνω ανάλυση αναφέρεται σε προβλήματα που είναι γραμμικώς διαχωρίσιμα. Ο γραμμικός διαχωρισμός ωστόσο, δεν ισχύει πάντοτε. Για παράδειγμα, σε μια περίπτωση που τα πρότυπα δύο κλάσεων είναι τόσο κοντά μεταξύ τους όπου δεν μπορούν να διαχωριστούν με ένα υπερπλάνο. Αν δεν είναι γραμμικώς διαχωρίσιμα τότε έχουμε

$$y_i(wx_i + b) \geqslant 1 - \xi_i, \xi_i \geqslant 0, i = 1, 2...N \tag{2.10}$$



Εικόνα 7. Μη γραμμικά Support Vector Machines

Για $0 \leq \xi_i \leq 1$ το πρότυπο x_i είναι μέσα στο περιθώριο.

Το πρόβλημα της ελαχιστοποίησης στην περίπτωση μη γραμμικώς διαχωρίσιμων προτύπων γράφεται ως εξής

$$\frac{||w||^2}{2} + C\sum_{i=1}^{N} \xi_i \tag{2.11}$$

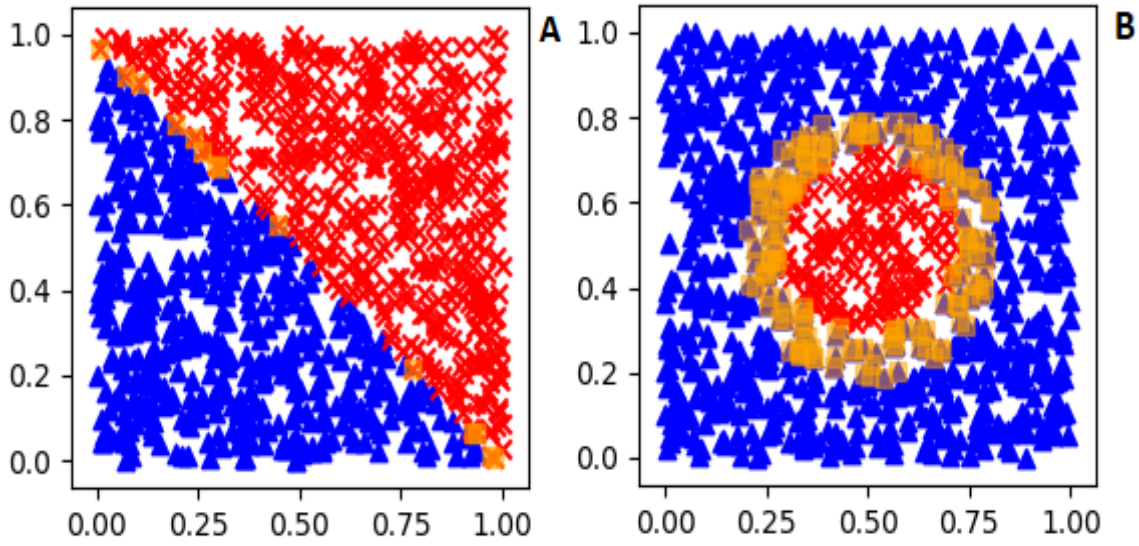
Η μεταβλητή C είναι θετικός πραγματικός αριθμός και καλείται ρυθμιστική παράμετρος (regularization parameter) και ελέγχει τη μορφή του περιθωρίου.

Μια ιδανική κατηγοριοποίηση με μηδενικό σφάλμα είναι εφικτή στην περίπτωση δύο γραμμικώς διαχωρίσιμων κλάσεων. Για πιο σύνθετα προβλήματα, θα πρέπει να γίνει ένας μετασχηματισμός των προτύπων από ένα M − διάστατο χώρο σε ένα χώρο μεγαλύτερης διάστασης, έτσι δημιουργείται μια πιο σύνθετη υπερεπιφάνεια (hypersurface) στον αρχικό χώρο των χαρακτηριστικών. Μια τέτοια υπερεπιφάνεια μπορεί να διαχωρίσει τα πρότυπα σε κλάσεις που δεν είναι γραμμικώς διαχώρισιμα.

2.3.4 Naive Bayes

Οι ταξινομητές Naive Bayes είναι γραμμικοί ταξινομητές που είναι γνωστοί ως απλοί αλλά πολύ αποδοτικοί. Το πιθανοτικό μοντέλο ταξινομητών Bayes βασίζεται στο θεώρημα του Bayes και το όνομα ”αφελές” (naive) προέρχεται από την υπόθεση ότι τα χαρακτηριστικά ενός συνόλου δεδομένων είναι αμοιβαία ανεξάρτητα. Στην πράξη, η παραδοχή ανεξαρτησίας συχνά παραβιάζεται, αλλά οι ταξινομητές Bayes τείνουν να αποδίδουν πολύ καλά ακόμη και κάτω από αυτήν την υπόθεση. Ειδικά για μικρά σύνολα δειγμάτων, οι ταξινομητές Bayes μπορούν να ξεπεράσουν τις πιο ισχυρές εναλλακτικές λύσεις.

Η κατανομή πιθανότητας στο σύνολο των κλάσεων ονομάζεται a priori πιθανότητα ενώ αυτή στο σύνολο των χαρακτηριστικών class conditional πιθανότητα. Για ένα δε-



Εικόνα 8. Όπου (A) Γραμμική και (B) μή γραμμική περίπτωση. Η (B) είναι μια περίπτωση όπου οι γραμμικοί ταξινομητές, όπως ο Naive Bayes, δεν θα ήταν κατάλληλοι δεδομένου ότι οι τάξεις δεν είναι γραμμικά διαχωρισμένες. Σε ένα τέτοιο σενάριο, θα πρέπει να προτιμώνται οι μη γραμμικοί ταξινομητές (π.χ., KNN).

δομένο πρότυπο που χαρακτηρίζεται από τιμές χαρακτηριστικών για κάθε κλάση, υπολογίζουμε την πιθανότητα ότι αυτό το πρότυπο ανήκει σε μια δεδομένη κλάση. Αυτές οι πιθανότητες ονομάζονται a posteriori πιθανότητες. Το πρότυπο αποδίδεται στην κλάση με την υψηλότερη a posteriori πιθανότητα. Αυτός ο συγκεκριμένος ταξινομητής ελαχιστοποιεί την πιθανότητα λανθασμένης ταξινόμησης και που ονομάζεται κατηγοριοποιητής Bayes.

Δεδομένων δύο συμβάντων ενός τυχαίου πειράματος A και B με P (B)> 0, η υπό όρους πιθανότητα του A δεδομένου B ορίζεται ως η ακόλουθη εξίσωση:

$$P(A|B) = \frac{P(AB)}{P(B)}$$

Η ιδιότητα Bayes δηλώνει ότι δεδομένου ενός συμβάντος A σε ένα τυχαίο πείραμα E και ενός πεπερασμένου διαμελισμού $B1, B2, ..., Bn$ του δείγματος χώρου Ω,

$$P(Bj|A) = \frac{P(A|Bj)P(Bj)}{\sum P(A|Bi)P(Bi)}$$

Το $P(h)$ αναφέρεται σε προηγούμενη πιθανότητα, η οποία είναι η αρχική πιθανότητα του h χωρίς εκπαίδευση δεδομένων και αντικατοπτρίζει αυτό που είναι γνωστό για το h ως καθιερωμένη υπόθεση. Χωρίς αυτήν τη γνώση, μπορεί να δοθεί η ίδια προηγούμενη πιθανότητα σε κάθε υπόθεση στην πραγματική επεξεργασία. Ομοίως, το $P(D)$ υποδεικνύει την προηγούμενη πιθανότητα των δεδομένων εκπαίδευσης D που θα παρατηρηθούν. Και το $P(D/h)$ είναι η πιθανότητα εμφάνισης των δεδομένων D όταν το h είναι καθορισμένο. Στη μηχανική εκμάθηση, το $P(h/D)$ είναι αυτό που μας ενδιαφέρει, δηλαδή να υπολογίσουμε την prior πιθανότητα. Σύμφωνα με την ιδιότητα Bayes,

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

Ως σταθερή μεταβλητή ανεξάρτητη από την h, η $P(D)$ μπορεί να αφαιρεθεί. Επομένως, η $P(h|D)$ καθορίζεται από την $P(D|h)P(h)$, η οποία είναι ο πυρήνας του αλγορίθμου Bayes.

Έστω ότι έχουμε ένα σύνολο δεδομένων (εκπαιδευτικά δεδομένα) και θέλουμε να βρούμε την υπόθεση h με την μεγαλύτερη πιθανότητα από τα υποψήφια σύνολα H. Δηλαδή, δεδομένου τα δεδομένα εκπαίδευσης, πως να μάθει ο αλγόριθμος από τα δείγματα δεδομένων ώστε να ταξινομήσουμε νέα δεδομένα σε μια συγκεκριμένη κατηγορία.

Συνήθως τα δεδομένα περιέχουν πολλά χαρακτηριστικά, για παράδειγμα, τα δεδομένα D μπορεί να έχουν χαρακτηριστικά $a1, a2...an$. Ο αλγόριθμος Naïve Bayes βασίζεται σε μια τέτοια υπόθεση. Τα χαρακτηριστικά της δεδομένης τιμής στόχου είναι ανεξάρτητα το ένα από το άλλο. Αυτή η υπόθεση δείχνει ότι όταν δίνεται μια τιμή στόχος, η πιθανότητα από κοινού των χαρακτηριστικών ισούται με το προϊόν της ατομικής πιθανότητας κάθε χαρακτηριστικού. Τότε έχουμε τον ακόλουθο τύπο:

$$P(D|h) = P(a1, a2...an|h) = \prod_{i=1}^{N} P(ai|h)$$

Έτσι, η posterior πιθανότητα στον αλγόριθμο Bayes μπορεί να εκφραστεί ως:

$$P(D|h) = P(h) \prod_{i=1}^{N} P(ai|h)$$

Ο στόχος του ταξινομητή Naïve Bayes είναι να ταξινομήσει το κείμενο στην πλησιέστερη κατηγορία $C(C1, C2, ..., Cj)$ σύμφωνα με το διάνυσμα κειμένου $X(x1, x2, ..., xn)$. Το $X(x1, x2, ..., xn)$ αντιπροσωπεύει το διάνυσμα χαρακτηριστικών του κειμένου που θα ταξινομηθεί, ενώ τα $C1, C2, ..., Cj$ είναι οι κατηγορίες που έχουν καθοριστεί. Επομένως, ο υπολογισμός που αφορά αφορά την πιθανότητα $(P1, P2, ..., Pn)$ όταν το $X(x1, x2, ..., xn)$ ανήκει στο $C1, C2, ..., Cj$, με το Pj να είναι η πιθανότητα όταν το $X(x1, x2, ..., xn)$ ανήκει στο Cj. Τότε το μέγιστο $(P1, P2, ..., Pn)$ είναι το απαιτούμενο αποτέλεσμα.

Σύμφωνα με τον αλγόριθμο naive Bayes, μπορούμε να εφαρμόσουμε τον ακόλουθο τύπο:

$$P(Cj|x1, x2, ..., xn) = P(x1, x2, ..., xn|Cj)P(Cj)$$

Σε παραπάνω τύπο, το $P(Cj)$ είναι η προηγούμενη πιθανότητα όταν το κείμενο ανήκει στο Cj, $P(x1, x2, ..., xn|Cj)$ είναι η prior πιθανότητα το Cj να περιέχει το διάνυσμα κειμένου $(x1, x2, ..., xn)$ όταν το κείμενο στο να ταξινομηθεί ανήκει στο Cj. Έτσι το $max(P1, P2, ..., Pn)$ είναι ίσο με τη μέγιστη τιμή του παρακάτω τύπου:

$$argmaxP(Cj) \prod_{i=1}^{N} P(xi|Cj)$$

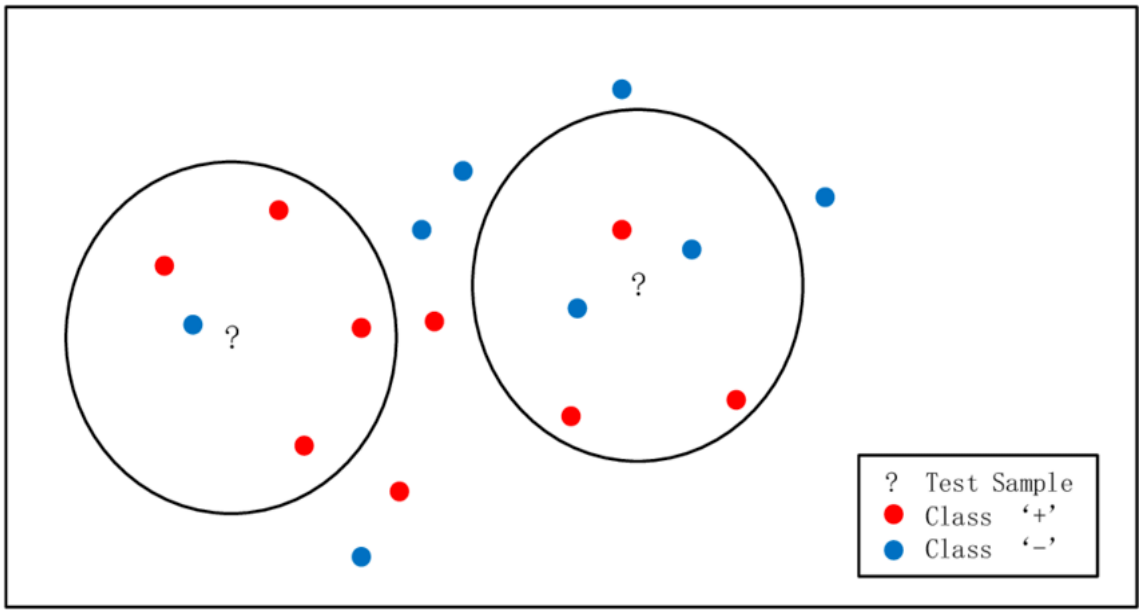
Σε αυτόν τον τύπο, $P(Cj) = \frac{P(C = Cj)}{N}$ είναι η ποσότητα του κειμένου που ανήκει

στο *Cj* στο σύνολο εκπαίδευσης και το N είναι η συνολική ποσότητα του κειμένου στο σύνολο εκπαίδευσης.

2.3.5 K-NN

Ο αλγόριθμος των k-κοντινότερων γειτόνων (K-NN) είναι ένας από τους απλούστερους αλγορίθμους όπου το σύνολο των δεδομένων ταξινομείται με βάση την εξάρτηση από την τάξη του πλησιέστερου γείτονα. Ο K-NN τείνει να αποδίδει πολύ καλά σε μεγάλο αριθμό συνόλων δεδομένων. Είναι κυρίως μη γραμμικό αλλά μπορεί να διακρίνει γραμμικά και μη γραμμικά στοιχεία από ένα σύνολο δεδομένων.

Με δεδομένο ένα δοκιμαστικό έγγραφο κειμένου, ο αλγόριθμος k-NN βρίσκει τους πλησιέστερους γείτονες ανάμεσα στα έγγραφα εκπαιδευτικού συνόλου και χρησιμοποιεί τις κατηγορίες των γειτόνων του για να σταθμίσει τους υποψηφίους της κατηγορίας. Η βαθμολογία ομοιότητας κάθε εγγράφου γείτονα με το έγγραφο δοκιμής χρησιμοποιείται ως το βάρος των κατηγοριών του γειτονικού εγγράφου. Εάν πολλοί από τους k πλησιέστερους γείτονες μοιράζονται μια κατηγορία, τότε τα βάρη ανά γειτονιά αυτής της κατηγορίας προστίθενται μαζί και το προκύπτων σταθμισμένο άθροισμα χρησιμοποιείται ως βαθμολογία πιθανότητας υποψηφίων κατηγοριών. Στην συγκεκριμένη έρευνα χρησιμοποιείται k=5, όσο και οι κατηγορίες εγκλημάτων.



Εικόνα 9. Αλγόριθμος K-NN με K=5

Οι μετρήσεις απόστασης χρησιμοποιούνται για τη μέτρηση της απόστασης μεταξύ των στοιχείων. Η ευκλείδεια απόσταση είναι η πιο κοινή μέτρηση απόστασης που χρησιμοποιείται για τη μέτρηση της απόλυτης απόστασης μεταξύ σημείων σε έναν πολυδιάστατο χώρο.

$$Dist(X, Y) = \sqrt{\sum_{i=1}^{N}(xi - yi)^2}$$

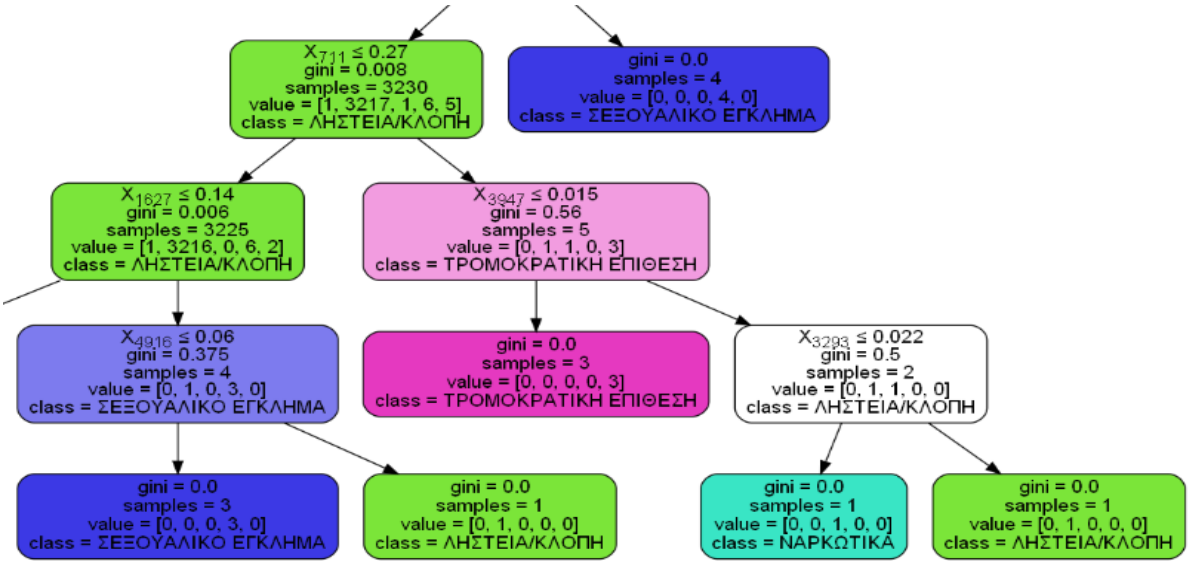
Η απόσταση Minkowski είναι ένα κοινό μέτρο της απόστασης μεταξύ αριθμητικών σημείων, αλλά δεν είναι απόσταση, αλλά ένα σύνολο ορισμών της απόστασης. Η απόσταση Minkowski μεταξύ δύο η-διαστάσεων μεταβλητών $A = (x11, x12, ..., x1n)$ και $B = (x21, x22, ..., x2n)$ ορίζεται ως:

$$Dist(X, Y) = \sqrt[p]{\sum_{i=1}^{N} |\mathrm{X1k} - \mathrm{X2k}|^p}$$

2.3.6 Decision Trees

Τα δέντρα ταξινόμησης (decision Trees) είναι μια από τις τεχνικές ταξινόμησης στην εξόρυξη δεδομένων που χρησιμοποιεί τη μέθοδο κλάδων για να απεικονίσει κάθε πιθανό αποτέλεσμα λήψης αποφάσεων σε κάθε πιθανό αποτέλεσμα. Ένα δένδρο είναι μια γενική δομή δεδομένων ως ακυκλικός γράφος. Σε ένα τέτοιο δένδρο δεν υπάρχουν κυκλικές διαδρομες και κάθε δύο κόμβοι συνδέονται. Δηλαδή, για κάθε δύο κόμβους υπάρχει μια μοναδική διαδρομή. Ένα δένδρο είναι ένας γράφος $T = (V, E)$ όπου V είναι ένα πεπερασμένο σύνολο κόμβων και E $\subset u, v : u, v \exists V$ είναι οι ακμές του δένδρου.

Τα δέντρα ταξινόμησης περιλαμβάνουν τρία είδη κόμβων που πλαισιώνουν ένα καθιερωμένο δέντρο και αποτελείται απο τον «κόμβο ρίζας», τον «εσωτερικό κόμβο» και το «φύλλο». Αναλύει ένα σύνολο δεδομένων σε μικρότερα και μικρότερα υποσύνολα ενώ ταυτόχρονα δημιουργείται ένα συσχετισμένο σύστημα αποφάσεων. Ο ριζικός κόμβος που είναι γνωστός ως αρχικό χαρακτηριστικό ή ο κορυφαίος κόμβος αποφάσεων σε ένα δέντρο που αντιστοιχεί στον καλύτερο αρπακτικό για ένα δέντρο για τη λήψη αποφάσεων που έχουν μηδενικές εισερχόμενες και εξερχόμενες ακμές .Ενώ οι εσωτερικοί κόμβοι έχουν και τα δύο εισερχόμενα και εξερχόμενα άκρα τουλάχιστον ένα ακολουθείται από κόμβο φύλλων που δεν έχει εξερχόμενες ακμές kai αντιπροσωπεύει μια ταξινόμηση ή απόφαση.



Εικόνα 10. Ενα κομμάτι απο το δέντρο ταξινόμησης κατα την υλοποίηση της Ανάλυσης Εγκλημάτων.

Τα δέντρα ταξινόμησης μαθαίνουν να προσεγγίζουν μια ημιτονοειδή καμπύλη με ένα σύνολο κανόνων IFTHEN που χρησιμοποιείται για τη λήψη αποφάσεων. Όσο πιο βαθύ είναι το δέντρο, τόσο πιο περίπλοκο μπορεί να είναι στους κανόνες λήψης αποφάσεων. Για την αυτόματη ταξινόμηση εγγράφων, τα δέντρα αποφάσεων είναι κατάλληλα για εφαρμογή σε ένα απλό πλαίσιο που καθορίζει ένα σύνολο κανόνων και χρησιμοποιείται για τη λήψη αποφάσεων για την ταξινόμηση του εγγράφου βάσει του περιεχομένου του στην κατηγορία του.

Σύγκριση μεταξύ των τεχνικών ταξινόμησης

KNN	**SVM**	**DT**	**NB**
Μπορεί να χρησιμοποιηθεί για συνεχείς εισόδους	Μπορεί να χρησιμοποιηθεί για συνεχείς εισόδους	Μπορεί να χρησιμοποιηθεί για συνεχείς και κατηγορικές εισόδους	Μπορεί να χρησιμοποιηθεί για συνεχείς και κατηγορικές εισόδους
Απλός και εύκολα κατανοητός αλγόριθμος	Μαθηματικά περίπλοκος αλγόριθμος	Ταξινόμηση δεδομένων με λιγότερους υπολογισμούς, απλός στην κατανόηση	Απλός στην κατανόηση
Είναι αυτόματα μη γραμμικός, ικανός να ανιχνεύει γραμμικά και μη γραμμικά εδομένα	Μπορεί να χρησιμοποιηθεί με γραμμικά και μη γραμμικά δεδομένα, καλός με περιορισμένο σύνολο σημείων σε πολλές διαστάσεις	Είναι μη γραμμικός ταξινομητής. Ικανός να απεικονίσει τη σχέση μεταξύ ανεξάρτητων και εξαρτημένων μεταβλητών	Γραμμικό
Κάνει ταξινόμηση καθορίζοντας γειτονιές	Κάνει ταξινόμηση αναζητώντας τα πλησιέστερα σημεία	Κάνει ταξινόμηση με μορφή δέντρου	Υπολογίζει πώς δημιουργήθηκαν τα δεδομένα, δεδομένων των αποτελεσμάτων
Υπολογιστικά ακριβός	Υπολογιστικά ακριβός στην διαδικασία εκμάθηση	Υπολογιστικά χαμηλών προδιαγραφών	Υπολογιστικά αποδοτικός (γρήγορος)
Δεν είναι κατάλληλος για αυτόματη ταξινόμηση	Κατάλληλος για τεχνική αυτόματης ταξινόμησης αλλά λίγο περίπλοκος	Κατάλληλος για τεχνική αυτόματης ταξινόμησης και λιγότερο περίπλοκος	Δεν είναι κατάλληλος για αυτόματη ταξινόμηση

Συνέχεια στην επόμενη σελίδα

Πίνακας 3 – η συνέχεια της προηγούμενης σελίδας

KNN	SVM	DT	NB
Χρονοβόρος	Χρονοβόρος κατά την επεξεργασία μεγάλου όγκου δεδομένων	Χρονοβόρος αν το δέντρο δεδομένων είναι μεγάλο	Γρήγορος για πολυδιάστατα δεδομένα

Πίνακας 3. Σύγκριση των τεχνικών ταξινόμησης

2.4 Elasticsearch

Το Elasticsearch ξεκίνησε το 2004 ως έργο ανοιχτού κώδικα που ονομαζόταν Compass, το οποίο βασίστηκε στο Apache Lucene. Το Elasticsearch είναι μια κατανεμημένη και επεκτάσιμη μηχανή αναζήτησης κειμένου γραμμένη σε Java που είναι ανεξάρτητη από την πλατφόρμα. Αυτές οι λειτουργίες σε συνδυασμό με την απαίτηση ειδικής ευελιξίας και εύκολες επιλογές επέκτασης είναι χρήσιμες για ανάλυση δεδομένων σε πραγματικό χρόνο.

Η προεπεξεργασία δεδομένων σε πραγματικό χρόνο προσθέτει ένα επιπλέον επίπεδο πολυπλοκότητας, ειδικά όταν τα δεδομένα είναι κείμενα και δεν είναι δομημένα. Οι λύσεις για την επεξεργασία μεγάλων συνόλων δεδομένων στους τομείς του cloud computing και της αποθήκευσης αναπτύσσονται με μεγάλη ταχύτητα, αλλά όταν εξετάζουμε μεγάλα δεδομένα σε κλίμακα petabytes, τα analytics που βασίζονται στο cloud περιορίζονται από ανεπάρκειες δικτύου για τη μεταφορά των δεδομένων και επαναλαμβανόμενα κόστη για τους υπολογιστικούς πόρους που απαιτούνται για την εκτέλεση ανάλυσης σε πραγματικό χρόνο. Επί του παρόντος, τα τρία πρώτα εργαλεία που χρησιμοποιούνται για την ανάλυση μεγάλων βάσεων δεδομένων είναι τα Elasticsearch, Hadoop και Spark.

Το Elasticsearch είναι μια κατανεμημένη μηχανή αναζήτησης και ανάλυσης που επιτρέπει μετασχηματισμούς δεδομένων σε πραγματικό χρόνο, ερωτήματα αναζήτησης, επεξεργασία ροής εγγράφων και ευρετηρίαση με σχετικά υψηλή ταχύτητα. Επιπλέον, το Elasticsearch μπορεί να ευρετηριάσει αριθμούς, γεωγραφικές συντεταγμένες, ημερομηνίες και σχεδόν οποιονδήποτε τύπο δεδομένων ενώ υποστηρίζει πολλές γλώσσες (Python, Java, Ruby). Η ταχύτητα του Elasticsearch βασίζεται στην ικανότητά του να εκτελεί aggregation, αναζήτηση και επεξεργασία πάνω στο ευρετήριο (index) των δεδομένων. Είναι ένα εργαλείο αναζήτησης κειμένου και αναλυτικών στοιχείων με ένα πρόσθετο οπτικοποίησης για ανάλυση σε πραγματικό χρόνου.

Οι παραδοσιακές βάσεις δεδομένων απαιτούν την πρώτη μεταφόρτωση των δεδομένων και στη συνέχεια ο διαχειριστής πρέπει ενεργά να αποφασίσει ποια δεδομένα πρέπει να ευρετηριαστούν το οποίο προσθέτει ένα ακόμη επίπεδο επεξεργασίας καθιστώντας το ανέφικτο για ανάλυση σε πραγματικό χρόνο. Το Elasticsearch παρέχει μια λύση σε αυτούς

τους περιοριστικούς παράγοντες παρέχοντας ένα πολύ αποτελεσματικό σύστημα ανάκτησης δεδομένων και ανάλυσης σε πραγματικό χρόνο που πραγματοποιεί προ-ευρετηρίαση πριν από την αποθήκευση των δεδομένων, απαιτεί περιορισμένους πόρους και υπολογιστική ισχύ σε σχέση με τις παραδοσιακές λύσεις και παρέχει ένα σύστημα που διανέμεται και είναι εύκολο να κλιμακωθεί.

Στο πλαίσιο της γενικής λειτουργίας της μηχανής αναζήτησης, το Elasticsearch χρησιμοποιεί μια τρέχουσα παρουσία που ονομάζεται κόμβος που μπορεί να αναλάβει έναν ή περισσότερους ρόλους, συμπεριλαμβανομένου του master ή ενός κόμβου δεδομένων (data node). Τα σύνολα δεδομένων στο Elasticsearch απαιτούν τουλάχιστον έναν master και έναν κόμβο δεδομένων, ωστόσο είναι πιθανό ένα σύνολο να αποτελείται από έναν μόνο κόμβο αφού ένας κόμβος μπορεί να αναλάβει πολλούς ρόλους. Η μόνη μορφή αποθήκευσης δεδομένων που είναι συμβατή με την Elasticsearch είναι η JSON.

Το Elasticsearch χρησιμοποιεί ένα ανεστραμμένο ευρετήριο μέσω της αρχιτεκτονικής Apache Lucene για την αποθήκευση των δεδομένων. Ένα τυπικό ευρετήριο στο Elasticsearch είναι μια συλλογή εγγράφων με διαφορετικές ιδιότητες που έχουν οργανωθεί μέσω μετασχηματισμού που καθορίζεται από τον χρήστη και περιγράφει τους τύπους εγγράφων και τα πεδία για διαφορετικές πηγές δεδομένων. παρόμοιο με έναν πίνακα σε μια βάση δεδομένων SQL. Ο δείκτης (index) στη συνέχεια χωρίζεται σε θραύσματα (shards) που στεγάζονται σε πολλούς κόμβους όπου ένα θραύσμα είναι μέρος ενός ευρετηρίου που κατανέμεται σε διαφορετικούς κόμβους. Ο ανεστραμμένος δείκτης επιτρέπει μια πιο κατηγορηματική αποθήκευση μεγάλων συνόλων δεδομένων σε κόμβους και θραύσματα, έτσι ώστε τα ερωτήματα αναζήτησης σε πραγματικό χρόνο να είναι πιο αποτελεσματικά. Το Elasticsearch χρησιμοποιεί το RESTful API για επικοινωνία με τους χρήστες.

Elasticsearch	**RDBMS**
Ευρετηρίαση (Index)	Βάση Δεδομένων (Database)
Χαρτογράφηση (Mapping)	Σχήμα (Schema)
Έγγραφο (Document)	Σειρα (Row)
Είδος Εγγράφου (Document Type)	Πίνακας (Table)

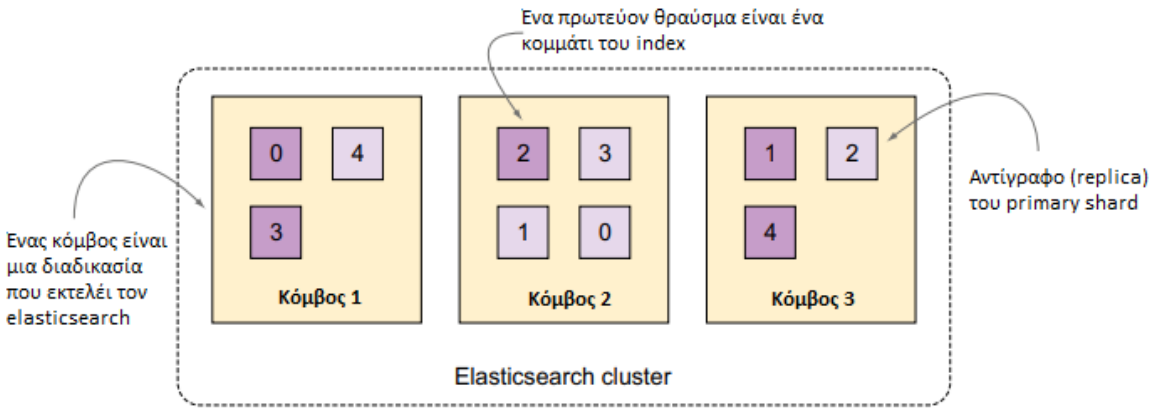
Πίνακας 4. Διαφορά στους όρους μεταξύ του elasticsearch και των παραδοσιακών βάσεων δεδομένων

2.4.1 Κλιμάκωση και Επεκτασιμότητα

Ένας κόμβος είναι ένα στιγμιότυπο (instance) του Elasticsearch. Όταν ξεκινάει το Elasticsearch, δημιουργείται ένας κόμβος. Εάν ξεκινήσει το Elasticsearch σε κάποιον άλλον διακομιστή, τότε θα είναι ένας άλλος κόμβος. Μπορεί να έχουμε πολλούς κόμβους στον ίδιο διακομιστή ξεκινώντας πολλά instances του Elasticsearch. Πολλοί κόμβοι μπορούν να ενταχθούν στο ίδιο σύμπλεγμα (cluster).

Ένα θραύσμα (shard) είναι ένα ευρετήριο της Lucene, δηλαδή, ένας κατάλογος αρχείων που περιέχουν ένα ανεστραμμένο ευρετήριο (inverted index). Το θραύσμα μπορεί να είναι είτε πρωτεύον είτε ένα θραύσμα αντίγραφο, το οποίο χρησιμοποιείται στην θέση του πρωτεύον όταν αυτό χαθεί. Ένας ανεστραμμένος δείκτης είναι μια δομή που επιτρέπει στον Elasticsearch να δει σε ποιο έγγραφο περιέχει έναν όρο (μια λέξη) χωρίς να χρειάζεται να κοιτάξει όλα τα έγγραφα. Ένα ευρετήριο είναι μια δομή δεδομένων που δημιουργείτε μαζί με τα δεδομένα μέσω της οποία επιτυγχάνονται ταχύτερες αναζητήσεις.

Με ένα cluster με πολλαπλούς κόμβους, τα ίδια δεδομένα μπορούν να εξαπλωθούν σε πολλούς διακομιστές. Αυτό βοηθά την απόδοση, επειδή το Elasticsearch έχει περισσότερους πόρους για να χρησιμοποιεί. Βοηθά επίσης την αξιοπιστία καθώς αν υπάρχει τουλάχιστον ένα αντίγραφο ανά θραύσμα (shard), οποιοσδήποτε κόμβος μπορεί να εξαφανιστεί και το Elasticsearch θα εξακολουθεί να εξυπηρετεί όλα τα δεδομένα. Αν και η ομαδοποίηση είναι καλή για απόδοση και διαθεσιμότητα, έχει τα μειονεκτήματά της καθώς πρέπει να ξέρουμε ότι οι κόμβοι μπορούν να επικοινωνούν μεταξύ τους αρκετά γρήγορα και ότι δεν θα υπάρχει κάποιο σημείο στο οποίο δύο κόμβοι του cluster δεν μπορούν να επικοινωνήσουν και άρα καταλήγουν στο συμπέρασμα ότι το άλλο μέρος δεν είναι στο cluster.



Εικόνα 11. Ενα cluster με 3 κόμβους

Όταν ευρετηριάζεται ένα έγγραφο, αποστέλλεται για πρώτη φορά σε ένα από τα primary shards, τα οποία επιλέγονται βάσει κατακερματισμού του ID του εγγράφου. Αυτό το αρχικό θραύσμα μπορεί να βρίσκεται σε διαφορετικό κόμβο, κάτι που γνωρίζει η εφαρμογή. Στη συνέχεια, το έγγραφο αποστέλλεται για ευρετηρίαση σε όλα τα αντίγραφα primary shard. Αυτό διατηρεί τα αντίγραφα σε συγχρονισμό με δεδομένα από τα main shards. Ο συγχρονισμός επιτρέπει στα αντίγραφα να προβάλλουν αναζητήσεις και να προωθούνται αυτόματα σε primary shards σε περίπτωση που το αρχικό shard δεν είναι διαθέσιμο.

Με την αναζήτηση ενός εγγράφου, ο κόμβος που λαμβάνει το αίτημα το προωθεί σε ένα σύνολο θραυσμάτων που περιέχουν όλα τα δεδομένα. Χρησιμοποιώντας τον αλγόριθμο round robin, το Elasticsearch επιλέγει ένα διαθέσιμο θραύσμα (το οποίο μπορεί να είναι πρωτεύον ή αντίγραφο) και προωθεί το αίτημα αναζήτησης σε αυτό.

Όταν ευρετηριάζεται ένα ευρετήριο, το Elasticsearch πρέπει να ψάξει σε ένα πλήρες σύνολο θραυσμάτων για αυτό το ευρετήριο. Αυτά τα θραύσματα μπορεί να είναι είτε πρωτεύοντα είτε αντίγραφα. Το Elasticsearch κατανέμει το φορτίο αναζήτησης μεταξύ των κύριων και των αντίγραφων θραυσμάτων του ευρετηρίου που είναι ύπο αναζήτηση, καθιστώντας τα αντίγραφα χρήσιμα τόσο για την πιο αποδοτική αναζήτηση, όσο και για την αυξημένη ανοχή στα σφάλματα.

Το απλούστερο cluster του Elasticsearch έχει έναν κόμβο. Προσθέτοντας περισσότερους κόμβους στο ίδιο σύμπλεγμα, τα υπάρχοντα shards ισορροπούνται μεταξύ όλων των κόμβων. Ως αποτέλεσμα, αιτήματα ευρετηρίου και αναζήτησης που λειτουργούν με αυτά τα shards επωφελούνται από την επιπλέον ισχύ των κόμβων που έχουν προσθετεί. Η κλιμάκωση με αυτόν τον τρόπο (προσθέτοντας κόμβους σε μια ομάδα) ονομάζεται οριζόντια κλιμάκωση. Με αυτό τον τρόπο γίνεται η προσθήκη περισσότερων κόμβων και τα αιτήματα διανέμονται, έτσι ώστε όλοι να μοιράζονται το έργο. Η εναλλακτική λύση στην οριζόντια κλιμάκωση είναι η κάθετη κλιμάκωση. Σε αυτή την περίπτωση, γίνεται η προσθήκη περισσότερων πόρων στον κόμβο του Elasticsearch, ίσως αφιερώνοντας περισσότερους επεξεργαστές σε αυτόν εάν είναι εικονική μηχανή ή προσθέτοντας RAM σε μια φυσική μηχανή. Αν και η κάθετη κλιμάκωση βοηθά την απόδοση σχεδόν κάθε φορά, δεν είναι πάντα δυνατή ή οικονομικά αποδοτική.

2.4.2 Elasticsearch APIs

Η διεπαφή προγραμματισμού εφαρμογών (API) είναι ένας τρόπος αλληλεπίδρασης μέσω προγραμματισμού με ένα ξεχωριστό στοιχείο ή πόρο λογισμικού. Το Elasticsearch παρέχει APIs ενός εγγράφου και APIs πολλών εγγράφων, όπου η κλήση API στοχεύει ένα μόνο έγγραφο ή και πολλά έγγραφα αντίστοιχα. Πιο συγκεκριμένα, διαθέτει Document API, Search API, Indices API, cat API και Cluster API.

Τα **Document APIs** χρησιμοποιούνται για το χειρισμό εγγράφων στο Elasticsearch. Χρησιμοποιώντας αυτά τα API, μπορεί κάποιος να δημιουργήσει έγγραφα σε ένα ευρετήριο, να τα ενημερώσει, να τα μετακινήσει σε άλλο ευρετήριο ή να τα καταργήσει.

Κατηγορία	Περιγραφή	Query
index	Προσθήκη ή ενημέρωση εγγράφου	PUT /«indexname»
get	Ανάκτηση ενός συγκεκριμένου υπάρχον εγγράφου	GET /«indexname»
Συνέχεια στην επόμενη σελίδα		

Πίνακας 5 – η συνέχεια της προηγούμενης σελίδας

Κατηγορία	Περιγραφή	Query
delete	Διαγραφή ενός εγγράφου	DELETE /«indexname»
reindex	Αντιγράφει ένα έγγραφο από το ένα ευρετήριο στο άλλο	POST /_reindex

Πίνακας 5. Παραδείγματα των Document APIs

Τα **Search APIs** χρησιμοποιούνται για την αναζήτηση δεδομένων ευρετηρίου για συγκεκριμένες πληροφορίες. Τα API αναζήτησης μπορούν να εφαρμοστούν σε όλους τους διαθέσιμους δείκτες και τύπους ή πιο συγκεκριμένα σε ένα ευρετήριο. Οι απαντήσεις θα περιέχουν αντιστοιχίσεις με το συγκεκριμένο ερώτημα.

Κατηγορία	Περιγραφή	Query
search	Επιστρέφει τις απαντήσεις που αντιστοιχούν σε ένα ερώτημα	GET /«targetindex»/_search POST /«targetindex»/_search
validate	Έλεγχος ενός πιθανά βαρύ ερωτήματος, χωρίς αυτό να εκτελεστεί	GET /«targetindex»/_validate/«query»
count	Εμφανίζει τον αριθμό των αντιστοιχιών για το ερώτημα	GET /«targetindex»/_count
explain	Υπολογίζει μια βαθμολογία για ένα ερώτημα σχετικά με το αν ένα έγγραφο ταιριάζει με το ερώτημα ή όχι	GET /«targetindex»/_explain/«id» POST /«targetindex»/_explain/«id»

Πίνακας 6. Παραδείγματα των Search APIs

Τα **Indices APIs** χρησιμοποιούνται για να επιτρέπουν στους χρήστες να διαχειρίζονται

τους δείκτες των εγγράφων και τις αντιστοιχίσεις. Για παράδειγμα, για την δημιουργία ή την διαγραφή ενός ευρετηρίου, για τον έλεγχο αν υπάρχει ένα συγκεκριμένο ευρετήριο ή όχι και για τον ορισμό νέας αντιστοίχισης για ένα ευρετήριο.

Κατηγορία	**Περιγραφή**	**Query**
Index Management	Δημιουργία νέου index	PUT /«indexname»
	Διαγραφή index	DELETE /«indexname»
	Άνοιγμα/κλείσιμο ενός index	POST /«indexname»/_open POST /«indexname»/_close
Mapping Management	Προσθήκη νέου είδους στο mapping	PUT /«indexname»/_mapping POST /«targetindex»/_explain/«id»
	Ανάκτηση του mapping	GET /«indexname»/_mapping

Πίνακας 7. Παραδείγματα των Indices APIs

Τα **Cat APIs** χρησιμοποιούνται για να επιστρέψετε δεδομένα σε πιο φιλική προς το χρήστη μορφή σε αντίθεση με την κανονική απόκριση JSON.

Κατηγορία	**Περιγραφή**	**Query**
Cat Indices	Παρέχει πρόσβαση σε πληροφορίες και μετρήσεις σχετικά με indices	GET /_cat/indices
Cat Health	Επισκόπηση της υγείας του index	GET /_cat/health
Cat Nodes	Πληροφορίες για τους κόμβους	GET /_cat/nodes
		Συνέχεια στην επόμενη σελίδα

Πίνακας 8 – η συνέχεια της προηγούμενης σελίδας

Κατηγορία	Περιγραφή	Query

Πίνακας 8. Παραδείγματα των Cat APIs

Τα **Cluster APIs** χρησιμοποιούνται για την διαχείριση και παρακολούθηση των κόμβων στο Elasticsearch.

Κατηγορία	Περιγραφή	Query
Cluster Health	Επισκόπηση της υγείας των κόμβων	GET _cluster/health/«target»
Cluster State	Αναφορά κατάστασης όλων των κόμβων	GET _cluster/state/«target»
Cluster Stats	Παρέχει βασικές μετρικές όσο αφορά τα indices	GET _cluster/stats/«target»

Πίνακας 9. Παραδείγματα των Cluster APIs

2.4.3 Indexing

Το Elasticsearch χρησιμοποιεί την έννοια του ανεστραμμένου ευρετηρίου για αναζήτηση. Όταν ενεργοποιείται το ερώτημα, το Elasticsearch εξετάζει τον ανεστραμμένο πίνακα ευρετηρίου για να βρει τα απαιτούμενα δεδομένα και θα δείξει το σχετικό έγγραφο στο οποίο περιέχεται ο όρος. Ένα ανεστραμμένο ευρετήριο αποτελείται από μια λίστα με όλες τις μοναδικές λέξεις που εμφανίζονται σε οποιοδήποτε έγγραφο και για κάθε λέξη, μια λίστα με τα έγγραφα στα οποία εμφανίζεται. Το ανεστραμμένο ευρετήριο είναι η πιο συχνά χρησιμοποιούμενη μέθοδος για FTS (Πλήρης αναζήτηση κειμένου). Ο ανεστραμμένος δείκτης αναζητά το σχετικό έγγραφο χαρτογραφώντας τον όρο στο περιέχον έγγραφο. Στο λεξικό ταξινομούνται οι όροι που οδηγούν σε γρήγορη αναζήτηση.

Ένα ευρετήριο αποτελείται από ένα ή περισσότερα έγγραφα και ένα έγγραφο αποτελείται από ένα ή περισσότερα πεδία. Το Elasticsearch είναι σε θέση να επιτυγχάνει γρήγορες απαντήσεις αναζήτησης, επειδή, αντί να αναζητά απευθείας το κείμενο, αναζητά ένα ευρετήριο.

Term	Document #1	Document #2
best	X	
carbonara		X
delicious		X
pasta	X	X
pesto	X	
recipe	X	X
the	X	
with	X	

Εικόνα 12. Η μέθοδος του ανεστραμμένου πίνακα (Inverted index)

Το πλήρες έγγραφο επιστρέφεται ως μέρος όλων των αναζητήσεων. Αυτό αποκαλείται πηγή. Εάν δεν επιστραφεί ολόκληρο το έγγραφο προέλευσης, τότε μπορούν να επιστραφούν μόνο λίγα πεδία από την πηγή. Το Elasticsearch χρησιμοποιεί το ανεστραμμένο ευρετήριο για να αναζητήσει τον όρο. Οι όροι ταξινομούνται σε αύξουσα σειρά.

Το Elasticsearch παρέχει τη δυνατότητα υποδιαίρεσης του ευρετηρίου σε πολλά κομμάτια που ονομάζονται θραύσματα. Όταν ένα νέο έγγραφο αποθηκεύεται και ευρετηριάζεται, το Elasticsearch ορίζει το θραύσμα που είναι υπεύθυνο για αυτό το έγγραφο. Όταν δημιουργείται ένα ευρετήριο, ο χρήστης μπορεί απλά να καθορίσει τον αριθμό θραυσμάτων. Κάθε θραύσμα από μόνο του είναι ένα πλήρως λειτουργικό και ανεξάρτητο ευρετήριο που μπορεί να φιλοξενηθεί σε οποιονδήποτε κόμβο.

2.4.4 Αναλυτές Κειμένου

Οι αναλυτές κειμένου είναι οι ειδικοί αλγόριθμοι που καθορίζουν πώς ένα πεδίο συμβολοσειράς σε ένα έγγραφο μετατρέπεται σε όρους σε ένα ανεστραμμένο ευρετήριο. Μέσα σε έναν αναλυτή υπάρχει μια σειρά απο επεξεργασίες που αποτελείται από το φιλτράρισμα χαρακτήρων, Tokenization και Φιλτράρισμα. Ο τελικός στόχος του αναλυτή είναι η μετατροπή μιας συμβολοσειράς σε μια σειρά από tokens. Ένα παράδειγμα αναλυτή απεικονίζεται στο παρακάτω διάγραμμα.

Η ροή εκτέλεσης ξεκινά με μια συμβολοσειρά που εισέρχεται στον αναλυτή. Αυτή η συμβολοσειρά εκτελείται αρχικά μέσω προαιρετικών φίλτρων χαρακτήρων, καθένα από τα οποία μετατρέπει τη συμβολοσειρά με συγκεκριμένο τρόπο, για παράδειγμα, σε πεζά γράμματα ή πραγματοποιεί αντικατάσταση λέξεων και εξάγει μια μετασχηματισμένη συμβολοσειρά. Στη συνέχεια, η έξοδος συμβολοσειράς των φίλτρων χαρακτήρων μεταφέρεται σε ένα tokenizer. Κάθε token περιέχει τόσο μια τιμή συμβολοσειράς όσο και έναν

Εικόνα 13. Αναλυτής Κειμένου

αριθμό θέσης που υποδεικνύει πού βρίσκεται στη ροή διακριτικών. Τέλος, αυτά τα διακριτικά μεταφέρονται προαιρετικά μέσω φίλτρων τα οποία μπορούν να αλλάξουν περαιτέρω τα tokens.

Το elasticsearch παρέχει ένα μεγάλο σύνολο από αναλυτές αλλά και την επιλογή να φτιάξει κάποιος τον δικό του αναλυτή κειμένου. Παρακάτω αναλύονται κάποιοι από τους πιο σημαντικούς αναλυτές.

Ο **Standard Analyzer** είναι ο προεπιλεγμένος αναλυτής κειμένου όταν δεν έχει καθοριστεί κάποιος συγκεκριμένος αναλυτής. Υποστηρίζει τις περισσότερες ευρωπαϊκές γλώσσες συνδυάζοντας τον προκαθορισμένο tokenizer, το προκαθορισμένο φίλτρο πεζών γραμμάτων και το προκαθορισμένο φίλτρο για λέξεις-κλειδιά (stop-words).

Ο **Simple Analyzer** χρησιμοποιεί το πεζά tokenizer, που σημαίνει ότι εξάγονται μόνο οι χαρακτήρες και μετατρέπονται σε πεζά.

Ο **Stop Analyzer** συμπεριφέρεται όπως ο απλός αναλυτής, αλλά επιπλέον φιλτράρει λέξεις-κλειδιά από τη ροή διακριτικών.

Ο **Whitespace Analyzer** πραγματοποιεί χωρισμό κειμένου όπου βρίσκει κενό ανάμεσα στις λέξεις.

Ακριβώς όπως περιλαμβάνει αναλυτές κειμένου, περιλαμβάνει επίσης έναν αριθμό ενσωματωμένων tokenizer. Το tokenization παίρνει μια σειρά κειμένου και το χωρίζει σε μικρότερα κομμάτια που ονομάζονται tokens. Κάποια παραδείγματα από τους πιο βασικούς tokenizers του elasticsearch αναλύονται παρακάτω.

Ο **Standard Tokenizer** είναι ένα tokenizer που βασίζεται στην γραμματική και είναι καλό για τις περισσότερες ευρωπαϊκές γλώσσες. Χειρίζεται επίσης την τμηματοποίηση κειμένου Unicode, αλλά με προεπιλεγμένο μέγιστο μήκος διακριτικού 255. Καταργεί επίσης σημεία στίξης όπως κόμματα και τελείες.

Ο **Keyword Tokenizer** είναι ένας απλός tokenizer που παίρνει ολόκληρο το κείμενο και το παρέχει ως ένα μοναδικό token στα φίλτρα.

Ο **Letter Tokenizer** παίρνει το κείμενο και το διαιρεί σε tokens στα σημεία που δεν είναι γράμματα.

Ο **Lowercase Tokenizer** συνδυάζει τόσο την ενέργεια του Standard tokenizer γραμμάτων όσο και τη δράση του φίλτρου lowercase. Ο κύριος λόγος για να την εφαρμογή του tokenizer είναι για καλύτερη απόδοση κάνοντας και τα δύο ταυτόχρονα.

Ο **Whitespace Tokenizer** διαχωρίζει τα tokens στο κενό διάστημα, αλλαγή γραμμής και ούτω καθεξής. Δεν αφαιρεί κανένα σημείο στίξης.

Ο Elasticsearch μας δίνει την δυνατότητα να φτιάξουμε δικούς μας αναλυτές που μπο-

ρούν αντίστοιχα να αποτελούνται από custom tokenizers ή custom φίλτρα, όπως φαίνεται στο παρακάτω σχήμα.

Εικόνα 14. Custom Αναλυτής Κειμένου

2.4.5 Εύρεση Κειμένου (IDF)

Ένας τρόπος για την εύρεση ενός εγγράφου είναι να παρατηρήσουμε πόσο συχνά εμφανίζεται ένας όρος στο κείμενο. Έστω ότι έχουμε τις παρακάτω δύο προτάσεις:

*"Το **Elasticsearch** περιέχει πολλούς αναλυτές κειμένου"*

*"Η ομάδα του **Elasticsearch** απαντάει σε ερωτήσεις για τους αναλυτές του **Elasticsearch**"*

Η πρώτη πρόταση αναφέρει τον Elasticsearch μία φορά και η δεύτερη αναφέρει τον Elasticsearch δύο φορές, οπότε ένα έγγραφο που περιέχει τη δεύτερη πρόταση πρέπει να έχει υψηλότερη βαθμολογία από ένα έγγραφο που περιέχει την πρώτη. Εάν μιλούσαμε σε απόλυτους αριθμούς, η πρώτη πρόταση θα είχε συχνότητα όρου (TF) 1 και η δεύτερη πρόταση θα έχει συχνότητα όρου 2.

Λίγο πιο περίπλοκο από την συχνότητα για ένα έγγραφο είναι η αντίστροφη συχνότητα εγγράφου (IDF). Αυτό σημαίνει ότι ένα token (συνήθως μια λέξη, αλλά όχι πάντα) είναι λιγότερο σημαντική όσες περισσότερες φορές εμφανίζεται σε όλα τα έγγραφα του ευρετηρίου. Αυτό είναι πιο εύκολο να εξηγηθεί με μερικά παραδείγματα όπως φαίνεται παρακάτω:

*"**To** elasticsearch περιέχει πολλούς αναλυτές κειμένου"*

*"**Το** Elasticsearch παρέχει τη δυνατότητα υποδιαίρεσης του ευρετηρίου"*

*"**Το** Elasticsearch ορίζει **το** shard που είναι υπεύθυνο για κάποιο έγγραφο"*

Στο πρώτο παράδειγμα, ο όρος «Elasticsearch» έχει συχνότητα εγγράφου 2 (επειδή εμφανίζεται σε δύο έγγραφα). Το αντίστροφο μέρος της συχνότητας του εγγράφου προέρχεται από το σκορ που πολλαπλασιάζεται επί 1 / DF, όπου το DF είναι η συχνότητα του όρου στα έγγραφα. Αυτό σημαίνει ότι επειδή ο όρος έχει υψηλότερη συχνότητα εγγράφου, το βάρος του μειώνεται. Στο δεύτερο παράδειγμα, ο όρος «το» έχει συχνότητα εγγράφου 3 επειδή εμφανίζεται και στα τρία έγγραφα. Σημειώστε ότι η συχνότητα του "το" είναι ακόμα 3, παρόλο που το "το" εμφανίζεται δύο φορές στο τελευταίο έγγραφο, επειδή η αντίστροφη συχνότητα εγγράφου ελέγχει μόνο έναν όρο που εμφανίζεται στο έγγραφο, όχι πόσο συχνά εμφανίζεται στο έγγραφο. Έτσι καταλήγουμε στο ότι η αντίστροφη συχνότητα εγγράφου είναι ένας σημαντικός παράγοντας για την εξισορρόπηση της συχνότητας ενός όρου.

Ο προεπιλεγμένος τρόπος εύρεσης κειμένου είναι γνωστός ως TF-IDF και βασίζεται τόσο στην συχνότητα του όρου όσο και στην αντίστροφη συχνότητα εγγράφου ενός όρου. Η βαθμολογία για ένα δεδομένο ερώτημα q και το ένα έγγραφο d είναι το άθροισμα (για κάθε όρο t στο ερώτημα) της τετραγωνικής ρίζας του όρου συχνότητα του όρου στο έγγραφο d, επί τη συχνότητα του αντίστροφου εγγράφου του όρου τετράγωνο, επί την κανονικοποίηση συντελεστής για το πεδίο στο έγγραφο, επί το χρόνο της ώθησης για τον όρο. Αυτό φαίνεται και στον παρακάτω τύπο.

$$SCORE_{\text{query,document}} = \sum_{t}^{q} \sqrt{TF_{\text{t,d}}} * IDF^2_{\text{t,d}} * norm(d, field) * boost(t)$$

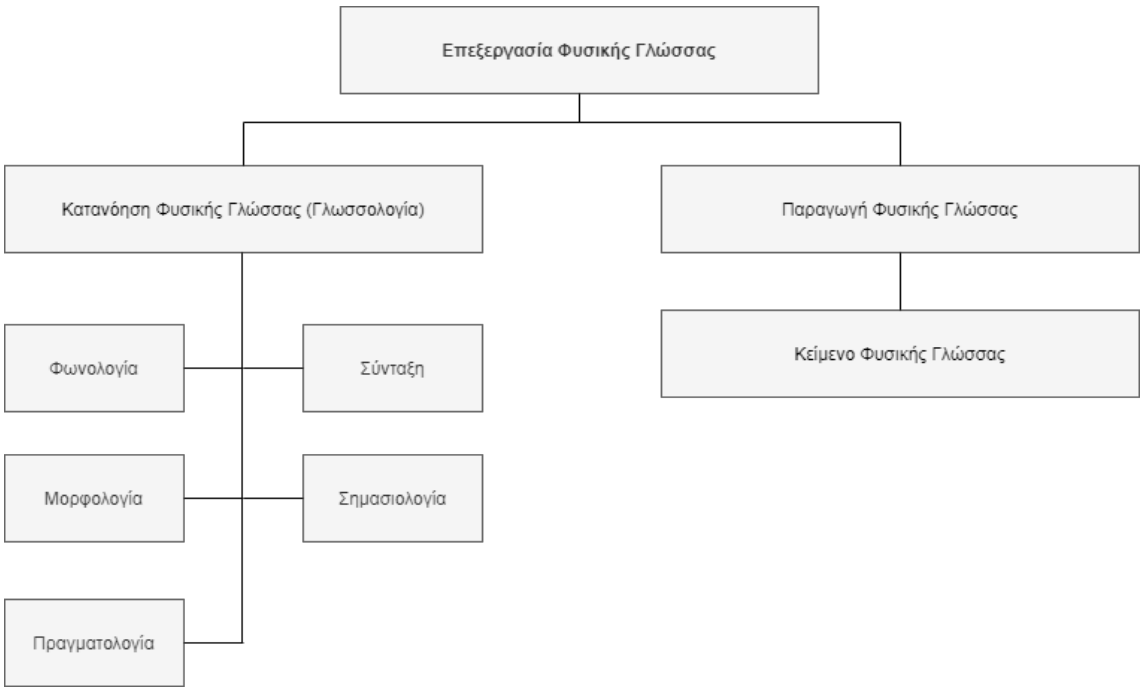
2.5 Επεξεργασία Φυσικής Γλώσσας

Η Επεξεργασία Φυσικής Γλώσσας (NLP) είναι ένα κομμάτι Τεχνητής Νοημοσύνης και Γλωσσολογίας, που επικεντρώνεται στο να κάνει τους υπολογιστές να κατανοήσουν τις δηλώσεις ή τις λέξεις που γράφονται σε διάφορες γλώσσες. Η επεξεργασία φυσικής γλώσσας δημιουργήθηκε για να διευκολύνει την εργασία του χρήστη και να ικανοποιήσει την επιθυμία να επικοινωνήσει με τον υπολογιστή σε φυσική γλώσσα.

Μια γλώσσα μπορεί να οριστεί ως ένα σύνολο κανόνων ή ένα σύνολο συμβόλων. Το σύμβολο συνδυάζεται και χρησιμοποιείται για τη μεταφορά πληροφοριών ή τη μετάδοση των πληροφοριών. Η επεξεργασία της φυσικής γλώσσας μπορεί να ταξινομηθεί σε δύο μέρη, στην κατανόηση της φυσικής γλώσσας και στη δημιουργία της φυσικής γλώσσας, η οποία εξελίσσει την εργασία για την κατανόηση και τη δημιουργία του κειμένου.

Η πιο επεξηγηματική μέθοδος για την παρουσίαση αυτού που πραγματικά συμβαίνει σε ένα σύστημα επεξεργασίας φυσικής γλώσσας είναι μέσω της προσέγγισης «επίπεδα γλώσσας». Αυτό αναφέρεται επίσης ως το σύγχρονο μοντέλο της γλώσσας και διακρίνεται από το προηγούμενο διαδοχικό μοντέλο, το οποίο υποθέτει ότι τα επίπεδα επεξεργασίας της ανθρώπινης γλώσσας ακολουθούν το ένα το άλλο με αυστηρά διαδοχικό τρόπο. Η ψυχογλωσσολογική έρευνα δείχνει ότι η επεξεργασία γλωσσών είναι πολύ πιο δυναμική, καθώς τα επίπεδα μπορούν να αλληλεπιδράσουν με μια ποικιλία σειρών. Στην πραγματικότητα όμως χρησιμοποιούμε συχνά πληροφορίες που αποκτούμε από αυτό που συνήθως

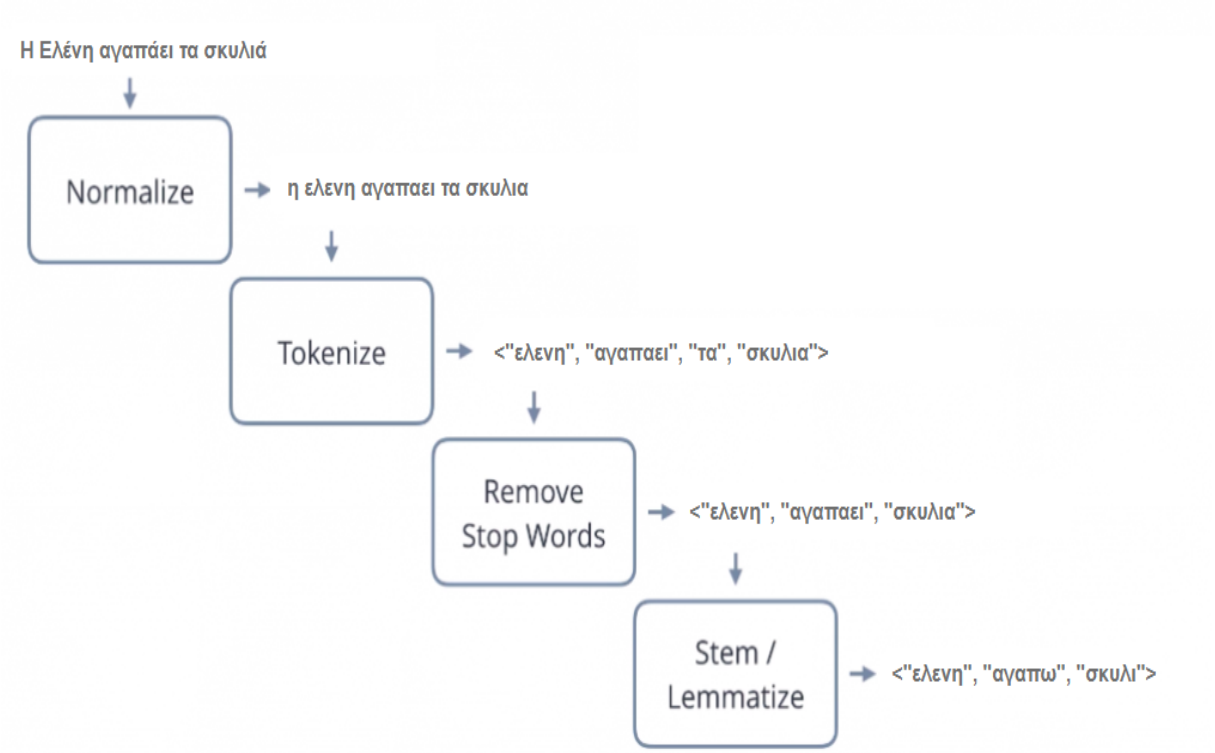
θεωρείται υψηλότερο επίπεδο επεξεργασίας για να βοηθήσουμε σε ένα χαμηλότερο επίπεδο ανάλυσης. Για παράδειγμα, η γνώση ότι ένα έγγραφο αφορά τη βιολογία θα χρησιμοποιηθεί όταν συναντηθεί μια συγκεκριμένη λέξη που έχει αρκετές πιθανές έννοιες και η λέξη θα ερμηνευθεί ως η έννοια της βιολογίας. Αναγκαστικά, η ακόλουθη περιγραφή των επιπέδων θα παρουσιαστεί διαδοχικά. Το βασικό σημείο εδώ είναι ότι το νόημα μεταφέρεται από κάθε επίπεδο γλώσσας και ότι επειδή οι άνθρωποι έχουν αποδειχθεί ότι χρησιμοποιούν όλα τα επίπεδα γλώσσας για να αποκτήσουν κατανόηση, άρα όσα περισσότερα επίπεδα γλώσσας χρησιμοποιεί ένα NLP σύστημα, τόσο πιο ικανό θα είναι.



Εικόνα 15. Επεξεργασία Φυσικής Γλώσσας

2.5.1 Προεπεξεργασία Κειμένου

Η προεπεξεργασία κειμένου είναι σημαντικό μέρος οποιουδήποτε συστήματος NLP, καθώς οι χαρακτήρες, οι λέξεις και οι προτάσεις που προσδιορίζονται σε αυτό το στάδιο είναι οι θεμελιώδεις ενότητες που περνούν σε όλα τα περαιτέρω στάδια επεξεργασίας, από στοιχεία ανάλυσης και επισήμανσης, όπως μορφολογικοί αναλυτές και μέρος του λόγου taggers, μέσω εφαρμογών, όπως η ανάκτηση πληροφοριών και τα συστήματα αυτόματης μετάφρασης. Πρόκειται για μια Συλλογή δραστηριοτήτων στις οποίες τα Έγγραφα κειμένου έχουν υποστεί προεπεξεργασία. Επίσης, μπορούμε να εξαλείψουμε κάποια δεδομένα κειμένου που συχνά περιέχουν ορισμένες ειδικές μορφές όπως μορφές αριθμών, μορφές ημερομηνίας και συχνά επαναλαμβανόμενες λέξεις που είναι απίθανο να βοηθήσουν (stop-words) στην εξόρυξη κειμένου όπως προθέσεις, άρθρα και αντωνυμίες.



Εικόνα 16. Επίπεδα προεπεξεργασίας κειμένου

Το **Tokenization** είναι η διαδικασία διάσπασης μιας ροής κειμένου σε λέξεις, φράσεις, σύμβολα ή άλλα σημαντικά στοιχεία που ονομάζονται tokens. Ο στόχος του tokenization είναι η διερεύνηση των λέξεων σε μια πρόταση. Η λίστα των διακριτικών γίνεται είσοδος για περαιτέρω επεξεργασία, όπως ανάλυση ή εξόρυξη κειμένου. Το Tokenization είναι χρήσιμο τόσο στη γλωσσολογία (όπου είναι μια μορφή τμηματοποίησης κειμένου), όσο και στην επιστήμη των υπολογιστών, όπου αποτελεί μέρος της λεξικής ανάλυσης. Τα δεδομένα κειμένου είναι μόνο ένα μπλοκ χαρακτήρων στην αρχή. Η κύρια χρήση του tokenization είναι ο προσδιορισμός των σημαντικών λέξεων-κλειδιών. Η ασυνέπεια μπορεί να είναι πολλές διαφορετικές μορφές μια λέξης και αριθμοί. Ένα άλλο πρόβλημα είναι οι συντομογραφίες και τα ακρωνύμια που πρέπει να μετατραπούν σε μια τυπική μορφή.

Οι λέξεις που επαναλαμβάνονται πολύ συχνά ή αλλιώς **stop words**, ουσιαστικά δεν έχουν νόημα καθώς χρησιμοποιούνται για να ενώσουν λέξεις σε μια πρόταση. Είναι ευρέως κατανοητό ότι οι πολύ συνηθισμένες λέξεις δεν συμβάλλουν στο περιεχόμενο ή το περιεχόμενο των κειμένων. Λόγω της υψηλής συχνότητας εμφάνισης, η παρουσία τους στην εξόρυξη κειμένων αποτελεί εμπόδιο στην κατανόηση του περιεχομένου των εγγράφων. Οι λέξεις διακοπής χρησιμοποιούνται πολύ συχνά κοινές λέξεις όπως «και», «είναι», «αυτό» κ.λπ. Δεν είναι χρήσιμες στην ταξινόμηση των εγγράφων. Πρέπει λοιπόν να αφαιρεθούν.

Το **Stemming** είναι η διαδικασία του συνδυασμού των παραλλαγών μιας λέξης σε μια κοινή αναπαράσταση. Στην συγκεκριμένη μεθοδολογία θέλουμε να κρατήσουμε μαζί τις παρόμοιες λέξεις που προέρχονται από την ίδια ρίζα και χωριστά αυτές που είναι διαφορετικές μεταξύ τους, και άρα δεν έχουν την ίδια σημασία. Για παράδειγμα, οι λέξεις: «παρουσίασε», «παρουσίασαν», «παρουσιάζω» θα μπορούσαν να μειωθούν σε μια κοινή αναπαράσταση «παρουσίαση». Αυτή είναι μια ευρέως χρησιμοποιούμενη διαδικασία στην επε-

ξεργασία κειμένου για ανάκτηση πληροφοριών με βάση την υπόθεση ότι η τοποθέτηση ερωτήματος με τον όρο «παρουσιάζω» συνεπάγεται ενδιαφέρον για έγγραφα που περιέχουν τις λέξεις «παρουσίασε» και «παρουσίασαν». Υπάρχουν δύο μορφές λαθών στην συγκεκριμένη μεθοδολογία. Η πρώτη περίπτωση είναι όταν δύο λέξεις με διαφορετική σημασία προέρχονται από την ίδια ρίζα και άρα παράγουν το ίδιο αποτέλεσμα μετά το stemming. Αυτό είναι επίσης γνωστό ως ψευδώς θετικό. Και η περίπτωση όπου δύο λέξεις που πρέπει να προέρχονται από την ίδια ρίζα δεν παράγουν το ίδιο αποτέλεσμα ενώ θα έπρεπε. Αυτό είναι γνωστό ως ψευδώς αρνητικό.

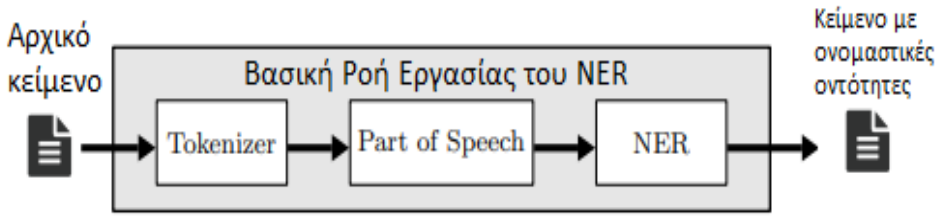
Το **Lemmatization** είναι η διαδικασία συναρμολόγησης των κεκλιμένων τμημάτων μιας λέξης έτσι ώστε να μπορούν να αναγνωριστούν ως ένα ενιαίο στοιχείο, που ονομάζεται λέμμα της λέξης. Αυτή η διαδικασία είναι η ίδια με το stemming αλλά προσθέτει νόημα σε συγκεκριμένες λέξεις. Με απλά λόγια, συνδέει το κείμενο με όμοιες έννοιες με μία λέξη. Ορίζεται ως μια τεχνική αλγορίθμου για την εύρεση του λέμματος μιας λέξης που είναι μια ρίζα της λέξης και όχι ένα ριζικό στοιχείο.

Input	**Stemming**	**Lemmatization**
Γάτες	γατ	γάτα
Εργαζόμενος	εργαζ	εργαζόμενο
Χτυπάω	χτυπα	χτυπάω

Πίνακας 10. Διαφορά μεταξύ Stemming και Lemmatization

2.5.2 Αναγνώριση και Κατηγοριοποίηση Ονομάτων Οντοτήτων

Η αναγνώριση ονομαστικής οντότητας (Named-entity recognition) είναι ο προσδιορισμός ονομαστικών οντοτήτων όπως άτομο, τοποθεσία, οργάνωση, φάρμακο, κ.λπ. σε κείμενο. Το NER περιλαμβάνει δύο εργασίες, που είναι πρώτον ο προσδιορισμός των κατάλληλων ονομάτων σε κείμενο και, δεύτερον, η ταξινόμηση αυτών των ονομάτων σε ένα σύνολο προκαθορισμένων κατηγοριών. Χρησιμοποιείται ευρέως στη μετάφραση, στην ανάκτηση πληροφοριών και στην αυτόματη σύνοψη. Το NER στοχεύει στην αναγνώριση και σημασιολογική ταξινόμηση λέξεων / οντοτήτων από ένα κείμενο. Ένα παραδοσιακό λογισμικό NER αποτελείται από τρεις κύριες εργασίες, όπως φαίνεται στο παρακάτω σχήμα.



Εικόνα 17. Βασική ροή ενος NER

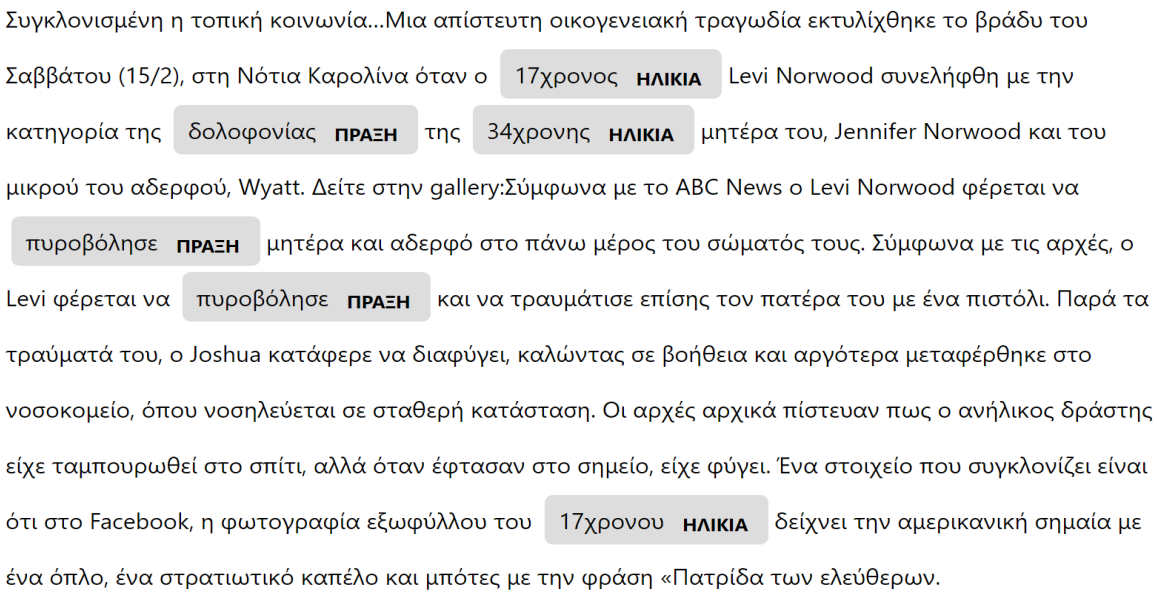
Τα περισσότερα προβλήματα στο NER είναι ότι έχουν σημασιολογική ασάφεια, από την άλλη πλευρά, ένα ουσιαστικό έχει διαφορετικές σημασίες ανάλογα με το πλαίσιο της πρότασης. Για παράδειγμα, πότε είναι ”Ο Λευκός Οίκος” μια οργάνωση, και πότε είναι μια τοποθεσία; Πότε είναι η ”Παρασκευή” ένα όνομα ατόμου και πότε μέρα; Η αυτόματη εξαγωγή κατάλληλων ονομάτων είναι χρήσιμη σε πολλά προβλήματα όπως η αυτόματη μετάφραση, η ανάκτηση πληροφοριών, η απάντηση ερωτήσεων και η συνοπτική παρουσίαση. Για παράδειγμα, το κλειδί για έναν επεξεργαστή ερωτήσεων είναι να προσδιορίσει το σημείο ερώτησης (ποιος, τι, πότε, πού, κ.λ.π.), έτσι σε πολλές περιπτώσεις το σημείο ερώτησης αντιστοιχεί σε μια ονομαστική οντότητα. Στα δεδομένα κειμένου της βιολογίας, το ονομαζόμενο σύστημα οντοτήτων, μπορεί να εξαγάγει αυτόματα τα προκαθορισμένα ονόματα (όπως ονόματα πρωτεϊνών και DNA) από ακατέργαστα έγγραφα. Ο στόχος της αναγνώρισης και εξαγωγής οντοτήτων είναι να εξαχθούν και να ταξινομηθούν τα ονόματα σε ορισμένες συγκεκριμένες κατηγορίες από το κείμενο σε σχέση με την έννοια των ονομάτων.

Υπάρχουν δύο μέθοδοι για την ονομαστική αναγνώριση οντοτήτων. NER με βάση την οντολογία και τη βαθιά μάθηση. Στην πρώτη περίπτωση, μία οντολογία είναι η διαδικασία αναγνώρισης βασισμένη στη γνώση, στην οποία η συλλογή συνόλων δεδομένων περιέχει λέξεις, όρους και τις σχέσεις τους. Και ανάλογα με το επίπεδο λεπτομερειών της οντολογίας, το αποτέλεσμα του NER μπορεί να είναι πολύ περιεκτικό ή συγκεκριμένο για ένα θέμα. Για παράδειγμα, μια εταιρεία που δραστηριοποιείται στην ιατρική επιστήμη χρειάζεται μια πολύ πιο λεπτομερή οντολογία λόγω της πολυπλοκότητας των διαφόρων ιατρικών ορολογιών. Το NER που βασίζεται στην οντολογία είναι σαν μια προσέγγιση μηχανικής εκμάθησης που μπορεί να εντοπίσει γνωστούς όρους και έννοιες σε μη δομημένα ή ημι-δομημένα κείμενα, αλλά ταυτόχρονα βασίζεται και σε ενημερώσεις.

Ενώ από την άλλη πλευρά, το NER με βάση την βαθιά μάθηση είναι πολύ πιο ακριβής από την οντολογία, καθώς είναι σε θέση να συγκεντρώσει λέξεις. Και αυτό οφείλεται σε μια μέθοδο που ονομάζεται ενσωμάτωση λέξεων, η οποία είναι ικανή να κατανοήσει τη σημασιολογική και συντακτική σχέση μεταξύ διαφόρων λέξεων. Επίσης μπορεί να μάθει αυτόματα αναλύσεις συγκεκριμένων θεμάτων καθώς και λέξεων υψηλού επιπέδου. Αυτό καθιστά τη βαθιά μάθηση NER εφαρμόσιμη για την εκτέλεση πολλαπλών εργασιών. Η βαθιά μάθηση μπορεί να κάνει το μεγαλύτερο μέρος της επαναλαμβανόμενης εργασίας από μόνη της.

Για την παρούσα εργασία, αναπτυχθεί ενα ειδικά σχεδιασμένο NER όπου αναγνω-

ρίζει ονομαστικές οντότητες που βοηθούν στη αναγνώριση της πράξης του δράστη, τις ηλικίες που εμπλέκονται, την τοποθεσία του εγκλήματος, την ημερομηνία που συνέβη το έγκλημα. Ένα παράδειγμα φαίνεται παρακάτω.



Εικόνα 18. Παράδειγμα απο τον custom-trained NER

2.5.3 Μέρος του λόγου (POS) και ανάλυση εξάρτησης

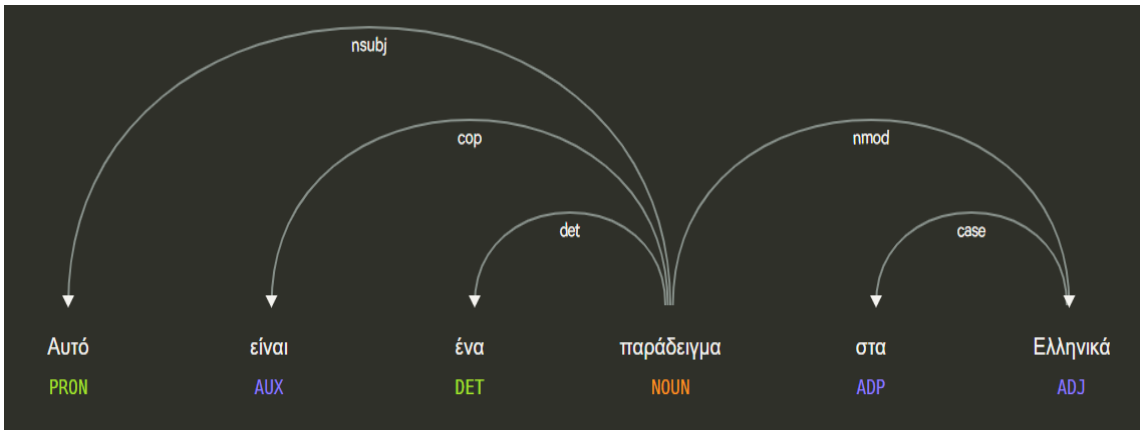
Η εύρεση του μέρος του λόγου (POS) είναι ένα πρόβλημα που αφορά την επισήμανση ακολουθίας, εκχωρώντας σε κάθε λέξη το σαφές μέρος του λόγου (ρήμα, ουσιαστικό, επίθετο κ.λπ.) στο πλαίσιο στο οποίο χρησιμοποιείται η λέξη. Αυτές οι πληροφορίες είναι χρήσιμες για εφαρμογές NLP υψηλότερου επιπέδου, όπως εξαγωγή σημασιολογικών σχέσεων, ανάλυση συναισθημάτων, αυτόματη σύνοψη και μηχανική μετάφραση. Τα περισσότερα παραδοσιακά μοντέλα POS με ετικέτες για περιεχόμενο κοινωνικών μέσων είναι γραμμικά στατιστικά μοντέλα, όπως Hidden Markov Models (HMM), Maximum Entropy Markov Models (MEMMs), Conditional Random Fields (CRF) και γραμμικοί ταξινομητές όπως SVM.

Η βάση της προσθήκης ετικετών POS είναι ότι πολλές λέξεις είναι ασαφείς σχετικά με το μέρος του λόγου τους, στις περισσότερες περιπτώσεις μπορούν να αποσαφηνιστούν πλήρως λαμβάνοντας υπόψη τα συμφραζόμενα της πρότασης.

Η προσθήκη ετικετών POS είναι ιδιαίτερα χρήσιμη, ειδικά εάν υπάρχουν λέξεις που μπορούν να έχουν πολλές διαφορετικές ετικέτες POS. Για παράδειγμα, η λέξη «google» μπορεί να χρησιμοποιηθεί ως ουσιαστικό και ρήμα, ανάλογα απο τα συμφραζόμενα. Κατά την επεξεργασία της φυσικής γλώσσας, είναι σημαντικό να προσδιοριστεί αυτή η διαφορά. Αφού αποσαφηνιστεί το POS της λέξης, μπορει να θέλουμε είτε να την παραβλέψουμε τελείως ή να της δώσουμε μεγαλύτερη βαρύτητα σε σχέση με τις άλλες.

Η ανάλυση εξάρτησης είναι η διαδικασία ανάλυσης της γραμματικής δομής μιας πρότασης βάσει των εξαρτήσεων μεταξύ των λέξεων σε μια πρόταση. Στην ανάλυση εξάρτησης, διάφορες ετικέτες αντιπροσωπεύουν τη σχέση μεταξύ δύο λέξεων σε μια πρόταση.

Αυτές οι ετικέτες είναι οι ετικέτες εξάρτησης. Για παράδειγμα, στη φράση «βροχερός καιρός», η λέξη «βροχή» έχει την έννοια του ουσιαστικού. Επομένως, υπάρχει μια εξάρτηση: καιρό →βροχερό, όπου ο καιρός ενεργεί ως ρίζα και ο «βροχερός» δρα ως εξαρτώμενη λέξη. Αυτή η εξάρτηση αντιπροσωπεύεται από την ετικέτα amod. Παρόμοια με αυτό, υπάρχουν πολλές εξαρτήσεις μεταξύ λέξεων σε μια πρόταση. Μια εξάρτηση περιλαμβάνει μόνο δύο λέξεις στις οποίες η μία ενεργεί ως ρίζα και άλλες ενεργεί ως εξαρτώμενη λέξη.



Εικόνα 19. Παράδειγμα απο τον custom-trained NER

2.5.4 SpaCy

Το spaCy και η NLTK είναι δωρεάν βιβλιοθήκες ανοιχτού κώδικα για προηγμένη επεξεργασία φυσικής γλώσσας (NLP) στην Python. Το NLTK και το spaCy έχουν τα συμπληρωματικά τους πλεονεκτήματα.

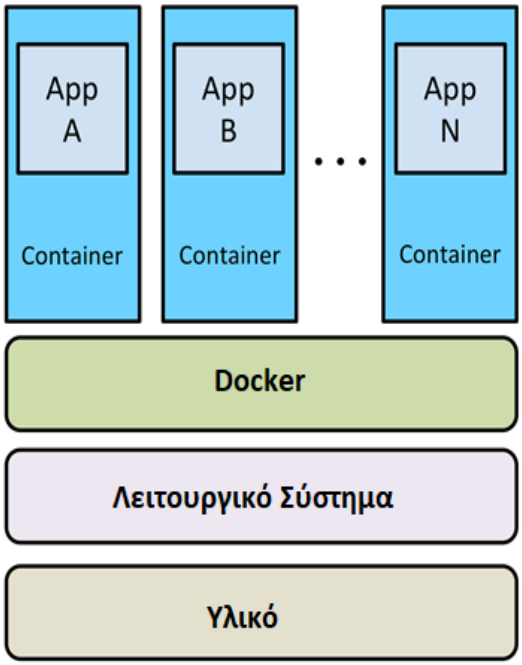
Το spaCy είναι μια ενεργά υποστηριζόμενη και ταχέως αναπτυσσόμενη βιβλιοθήκη που ενημερώνεται για τις εξελίξεις σε αλγόριθμους και μοντέλα NLP. Το spaCy χρησιμοποιείται ενεργά στη βιομηχανία και περιλαμβάνει ένα ισχυρό σύνολο εργαλείων ιδιαίτερα εφαρμοστέων στην εξαγωγή πληροφοριών μεγάλης κλίμακας.

Σε αντίθεση με το NLTK που αντιμετωπίζει διαφορετικά συστατικά της γλωσσικής ανάλυσης ως ξεχωριστά βήματα, το spaCy δημιουργεί έναν αγωγό ανάλυσης (pipeline) από την αρχή και εφαρμόζει αυτόν τον αγωγό στο κείμενο. Ο αγωγός περιλαμβάνει ήδη μια σειρά από χρήσιμα εργαλεία NLP που εκτελούνται με εισαγωγή κειμένου χωρίς να χρειάζεται να καλεστούν ξεχωριστά. Αυτά τα εργαλεία περιλαμβάνουν, μεταξύ άλλων, ένα tokenizer και ένα POS tagger. Εφαρμόζεται το σύνολο των εργαλείων με μια μόνο γραμμή κώδικα που καλείται στον αγωγό επεξεργασίας spaCy και, στη συνέχεια, το πρόγραμμά αποθηκεύει το αποτέλεσμα σε μια βολική μορφή. Αυτό διασφαλίζει επίσης ότι οι πληροφορίες διαβιβάζονται μεταξύ των εργαλείων αυτόματα χωρίς κάποιος να ασχολείται με τις μορφές εισόδου-εξόδου.

2.6 Εικονοποίηση Βασισμένη σε Περιέκτες

Για την ανάπτυξη πολλών εφαρμογών στην ίδια εικονική μηχανή, οι εφαρμογές και οι εξαρτήσεις πρέπει να οργανωθούν και να απομονωθούν. Λόγω της εικονικοποίησης, πολλές εφαρμογές μπορούν να εκτελεστούν στο ίδιο φυσικό υλικό. Τα μειονεκτήματα των τεχνικών εικονικοποίησης είναι: οι εικονικές μηχανές έχουν μεγάλο μέγεθος, μια ασταθή απόδοση λόγω της λειτουργίας πολλών εικονικών μηχανών, η διαδικασία εκκίνησης απαιτεί πολύ χρόνο για να εκτελεστεί και οι εικονικές μηχανές δεν είναι σε θέση να λύσουν δυσκολίες όπως η διαχείριση, οι ενημερώσεις λογισμικού και συνεχής ολοκλήρωση / παράδοση. Αυτά τα προβλήματα οδήγησαν στην εμφάνιση μιας νέας διαδικασίας που ονομάζεται εικονοποίηση βασισμένη σε περιέκτες (Containerization) που οδήγησε περαιτέρω στην εικονικοποίηση σε επίπεδο λειτουργικού συστήματος, ενώ η εικονικοποίηση φέρνει απορρόφηση στο επίπεδο υλικού.

Το Containerization χρησιμοποιεί κεντρικό λειτουργικό σύστημα που μοιράζεται σχετικές βιβλιοθήκες και πόρους. Είναι πιο αποτελεσματικό επειδή δεν υπάρχει λειτουργικό σύστημα επισκέπτη. Στον πυρήνα του κεντρικού υπολογιστή, μπορούν να υποβληθούν σε επεξεργασία τα δυαδικά αρχεία και οι βιβλιοθήκες για την εφαρμογή, στην οποία κάνει την εκτέλεση πολύ γρήγορα. Τα containers διαμορφώνονται με τη βοήθεια της πλατφόρμας Docker, η οποία συνδυάζει εφαρμογές και εξαρτήσεις τους. Αυτά τα container εκτελούνται πάντα πάνω από τον πυρήνα του λειτουργικού συστήματος σε απομονωμένο χώρο. Αυτή η δυνατότητα του Docker διασφαλίζει ότι το περιβάλλον υποστηρίζει οποιαδήποτε σχετική εφαρμογή.



Εικόνα 20. Αρχιτεκτονική εικονοποίησης βασισμένη σε περιέκτες

2.6.1 Μικροϋπηρεσίες

Η Αρχιτεκτονική βασισμένη στις μικροϋπηρεσίες είναι μια προσέγγιση ανάπτυξης μιας σειράς μικρών υπηρεσιών που λειτουργούν ως μία μόνο εφαρμογή. Οι υπηρεσίες επικοινωνούν μέσω ελαφρών μηχανισμών, όπως ένα API και κάθε υπηρεσία λειτουργεί ανεξάρτητα από μόνη της. Από την άλλη πλευρά, η μονολιθική αρχιτεκτονική είναι μια εφαρμογή με μία βάση κώδικα που περιλαμβάνει πολλαπλές υπηρεσίες. Αυτές οι υπηρεσίες επικοινωνούν με εξωτερικά συστήματα ή καταναλωτές μέσω διαφορετικών διεπαφών όπως υπηρεσίες Ιστού, σελίδες HTML ή REST API. Η αρχιτεκτονική αυτή θα διευκολύνει τις διαδικασίες συντηρησιμότητας, επαναχρησιμοποίησης, επεκτασιμότητας, διαθεσιμότητας και αυτόματης ανάπτυξης.

Τα πλεονεκτήματα της αρχιτεκτονικής των μικροεπηρεσιών είναι πολλαπλά. Πρώτον, οι μικροϋπηρεσίες μπορούν να βασίζονται στην ετερογένεια της τεχνολογίας, πράγμα που σημαίνει ότι κάθε υπηρεσία σε ένα σύστημα μπορεί να χρησιμοποιεί διαφορετική τεχνολογία από τις άλλες υπηρεσίες για την επίτευξη των επιθυμητών στόχων και επιδόσεων. Δεύτερον, εάν ένα στοιχείο του συστήματος αποτύχει, τότε δεν επηρεάζει ολόκληρο το σύστημα. Το τρίτο πλεονέκτημα είναι ότι η διαδικασία κλιμάκωσης μπορεί να είναι πιο προσιτή σε σύγκριση με τη μονολιθική κλιμάκωση εφαρμογών, επειδή μόνο οι υπηρεσίες που χρειάζονται πραγματική κλιμάκωση κλιμακώνονται στην αρχιτεκτονική μικροσυσκευών, σε αντίθεση με μια μονολιθική εφαρμογή πρέπει να κλιμακωθεί ως ολόκληρη μονάδα που μπορεί να οδηγήσει σε υψηλότερη χρήση υλικού. Τέταρτον, ευκολία ανάπτυξης, διότι με τις μικροϋπηρεσίες κάθε υπηρεσία μπορεί να αναπτυχθεί ανεξάρτητα χωρίς να επηρεάζεται η απόδοση άλλων υπηρεσιών. Πέμπτον, βοηθά τις εταιρείες να ευθυγραμμίσουν την αρχιτεκτονική της με την οργανωτική της δομή, η οποία θα τους βοηθήσει να ελαχιστοποιήσουν τον αριθμό των ατόμων που εργάζονται σε μια συγκεκριμένη βάση κώδικα. Κατά συνέπεια, οι μικροϋπηρεσίες καθιστούν δυνατή την οργανωτική ευθυγράμμιση. Άλλα πλεονεκτήματα είναι η συνθεσιμότητα και η βελτιστοποίηση για δυνατότητα αντικατάστασης.

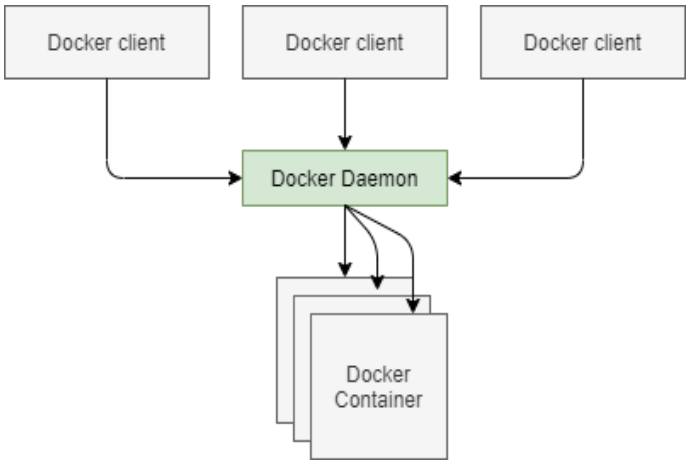
Από την άλλη πλευρά, στη μονολιθική αρχιτεκτονική, εφαρμογές μπορούν να δημιουργηθούν από δεκάδες ή εκατοντάδες διαφορετικές υπηρεσίες που συνδέονται στενά σε μια μονολιθική βάση κώδικα. Αυτό μπορεί να δημιουργήσει πολλές δυσκολίες για ομάδες που εργάζονται στο ίδιο περιβάλλον. Ως εκ τούτου, πολλές εταιρείες κινούνται προς την αρχιτεκτονική των μικροϋπηρεσιών για να επιτρέψουν στις ομάδες ανάπτυξης τους να συντονίζονται εύκολα μεταξύ τους.

2.6.2 Docker

Το Docker παρέχει μια δυνατότητα αυτοματοποίησης των εφαρμογών όταν αναπτύσσονται σε container. Σε ένα περιβάλλον container όπου οι εφαρμογές εικονικοποιούνται και εκτελούνται, το docker προσθέτει ένα επιπλέον επίπεδο ανάπτυξης πάνω από αυτό. Ο τρόπος με τον οποίο έχει σχεδιαστεί το docker είναι να παρέχει ένα γρήγορο και ελαφρύ περιβάλλον όπου ο κώδικας μπορεί να λειτουργεί αποτελεσματικά και επιπλέον παρέχει μια επιπλέον διευκόλυνση της ικανής διαδικασίας εργασίας για να πάρει τον κώδικα από τον υπολογιστή για δοκιμή πριν από την παραγωγή.

Πελάτης - Διακομιστής: Το Docker μπορεί να εξηγηθεί ως εφαρμογή που βασίζεται σε υπολογιστή-πελάτη και διακομιστή, όπως απεικονίζεται στο σχήμα παρακάτω. Ο

διακομιστής docker λαμβάνει το αίτημα από τον πελάτη docker και στη συνέχεια επεξεργάζεται ανάλογα. Το πλήρες RESTful API και ένα δυαδικό πρόγραμμα-πελάτη γραμμής εντολών αποστέλλονται από το docker. Το Docker daemon / server και το docker client μπορούν να εκτελεστούν στον ίδιο υπολογιστή ή ένας τοπικός πελάτης docker μπορεί να συνδεθεί με έναν απομακρυσμένο διακομιστή ή daemon, ο οποίος εκτελείται σε άλλο μηχάνημα.



Εικόνα 21. Πελάτης - Δικομιστής

Docker εικόνες: Οι εικόνες Docker μπορούν να κατασκευαστούν με δύο μεθόδους. Η κύρια μέθοδος είναι με τη βοήθεια ενός αναγνωστικού προτύπου. Το πρότυπο αποτελείται από βασικές εικόνες, είτε μπορεί να είναι λειτουργικό σύστημα όπως centos, ubuntu 16.04 ή fedora, ή οποιεσδήποτε άλλες εικόνες βασικού λειτουργικού συστήματος που είναι ελαφριές. Γενικά, οι βασικές εικόνες είναι το θεμέλιο κάθε εικόνας. Είναι απαραίτητο να δημιουργηθεί μια νέα εικόνα κάθε φορά που δημιουργούνται βασικές εικόνες από το μηδέν. Αυτός ο τύπος δημιουργίας μιας νέας εικόνας ονομάζεται ”πραγματοποίηση αλλαγής”. Η επόμενη μέθοδος είναι να δημιουργηθεί ένα αρχείο σύνδεσης το οποίο έχει όλες τις οδηγίες για τη δημιουργία μιας εικόνας. Όταν η εντολή build docker εκτελείται από το τερματικό, η εικόνα θα δημιουργηθεί με όλες τις εξαρτήσεις που αναφέρονται στο αρχείο docker. Αυτή η διαδικασία είναι γνωστή ως μια αυτοματοποιημένη μέθοδος δημιουργίας μιας εικόνας.

Docker μητρώα: Οι εικόνες τοποθετούνται σε μητρώα του docker. Λειτουργεί αντίστοιχα σε αποθετήρια πηγαίου κώδικα όπου οι εικόνες μπορούν να προωθηθούν ή να τραβηχτούν από μία μόνο πηγή. Υπάρχουν δύο τύποι μητρώων, δημόσιο και ιδιωτικό. Το Docker Hub ονομάζεται δημόσιο μητρώο όπου όλοι μπορούν να τραβήξουν διαθέσιμες εικόνες και να προωθήσουν τις δικές τους εικόνες χωρίς να δημιουργήσουν μια εικόνα από το μηδέν. Οι εικόνες μπορούν να διανεμηθούν σε μια συγκεκριμένη περιοχή (δημόσια ή ιδιωτική) χρησιμοποιώντας τη λειτουργία hub docker.

Containers: Η εικόνα του Docker δημιουργεί ένα container. Τα container κρατούν ότι απαιτείται για μια εφαρμογή, έτσι ώστε η εφαρμογή να μπορεί να εκτελεστεί μεμονωμένα. Για παράδειγμα, ας υποθέσουμε ότι υπάρχει μια εικόνα του scrapy cralwer με μια MongoDB ώς βάση δεδομένων, όταν αυτή η εικόνα εκτελείται με εντολή run docker, τότε θα δημιουργηθεί ένας spider και ξεχωριστά η MongoDB στην οποία θα αποθηκεύεται οτιδήποτε φέρνει ο spider.

Τα κύρια πλεονεκτήματα του docker είναι η ταχύτητα, η φορητότητα, η επεκτασιμότητα, και η γρήγορη παράδοση.

Ταχύτητα: Ο χρόνος που απαιτείται για την κατασκευή τους είναι πολύ μικρός καθώς τα containers είναι μικρά. Η ανάπτυξη και ο έλεγχος μπορούν να γίνουν γρηγορότερα. Τα containers μπορούν να προωθηθούν για δοκιμή όταν έχουν κατασκευαστεί και στη συνέχεια από εκεί, στο περιβάλλον παραγωγής.

Φορητότητα: Οι εφαρμογές που είναι ενσωματωμένες σε container είναι εξαιρετικά φορητές. Αυτές οι φορητές εφαρμογές μπορούν να μετακινούνται εύκολα ως ένα μόνο στοιχείο και η απόδοση παραμένει ίδια.

Επεκτασιμότητα: Το Docker έχει τη δυνατότητα να μπορεί να αναπτυχθεί σε πολλούς φυσικούς διακομιστές, διακομιστές δεδομένων και πλατφόρμες cloud. Μπορεί επίσης να εκτελεστεί σε κάθε υπολογιστή Linux. Τα container μπορούν εύκολα να μετακινηθούν από ένα περιβάλλον cloud σε έναν τοπικό κεντρικό υπολογιστή και από εκεί να επιστρέψουν στο cloud ξανά με γρήγορο ρυθμό. Οι προσαρμογές μπορούν εύκολα να γίνουν. η κλίμακα μπορεί απλώς να ρυθμιστεί από τον χρήστη ανάλογα με τις ανάγκες.

Ταχεία παράδοση: Η μορφή ενός Docker Containers είναι τυποποιημένη. Η ευθύνη του διαχειριστή είναι να αναπτύξει και να συντηρήσει τον διακομιστή με container, ενώ η ευθύνη του προγραμματιστή είναι να φροντίζει τις εφαρμογές μέσα στο container. Τα container μπορούν να λειτουργήσουν σε κάθε περιβάλλον καθώς έχουν ενσωματωμένες όλες τις απαιτούμενες εξαρτήσεις εντός των εφαρμογών.

Πυκνότητα: Το Docker χρησιμοποιεί τους πόρους που είναι διαθέσιμοι πιο αποτελεσματικά επειδή δεν χρησιμοποιεί κάποιον υπεύθυνο (supervisor). Αυτός είναι ο λόγος που μπορούν να εκτελούνται περισσότερα container σε έναν κεντρικό υπολογιστή σε σύγκριση με εικονικές μηχανές (VM). Η απόδοση ενός Docker Container είναι υψηλότερη λόγω της υψηλότερης πυκνότητας και της μη-σπατάλης πόρων.

Το **Dockerfile**: χρησιμοποιείται ως συσκευασία μιας εφαρμογής. Αλλά για τις διανεμημένες εφαρμογές, το Dockerfile είναι πραγματικά μόνο για τη συσκευασία ενός μέρους της εφαρμογής. Για μια εφαρμογή με ένα UI, ένα API backend και μια βάση δεδομένων, θα έπρεπε να υπάρχουν 3 Dockerfiles - ένα για κάθε στοιχείο και να έπρεπε να ενωθούν ώστε να δουλεύουν όλες οι υπηρεσίες μαζί, μέσα σε ένα δίκτυο, τότε θα έπρεπε να υπάρχουν μέσα σε ενα **docker-compose** αρχείο.Το Docker Compose χρησιμοποιεί YAML, το οποίο είναι μια μορφή κειμένου που διαβάζεται από τον άνθρωπο που χρησιμοποιείται ευρέως επειδή μεταφράζεται εύκολα σε JSON. Στο docker-compose κάποια απο τα πιο βασικά στοιχεία που γράφουμε είναι τα εξής:

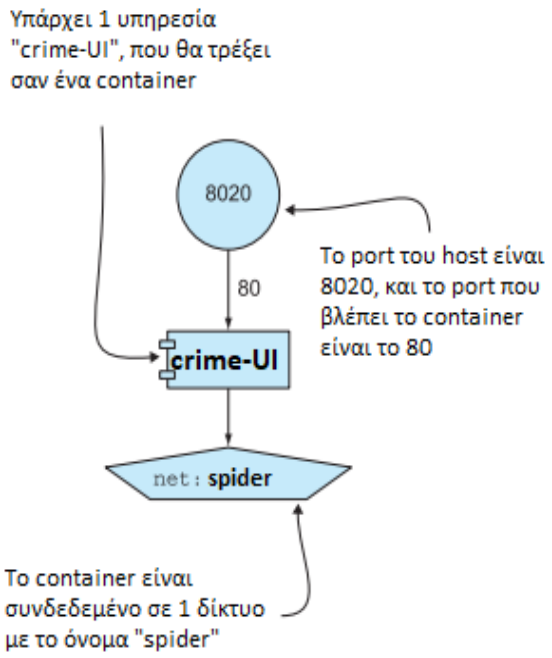
- Η έκδοση της μορφής σύνταξης Docker που χρησιμοποιείται σε αυτό το αρχείο. Το σύνολο χαρακτηριστικών έχει εξελιχθεί σε πολλές κυκλοφορίες, επομένως η έκδοση εδώ προσδιορίζει με ποιες εκδόσεις λειτουργεί αυτός ο ορισμός.

- Όλα τα στοιχεία που αποτελούν την εφαρμογή. Το Docker Compose χρησιμοποιεί την ιδέα των υπηρεσιών αντί των πραγματικών container, επειδή μια υπηρεσία θα μπορούσε να εκτελεστεί σε κλίμακα με πολλά container από την ίδια εικόνα.

- Όλα τα δίκτυα Docker στα οποία μπορούν να συνδεθούν τα δοχεία υπηρεσιών.

Όπως φαίνεται στην παρακάτω εικόνα, κάτω από το όνομα της υπηρεσίας βρίσκονται οι ιδιότητες, οι οποίες είναι αρκετά κοντά στις εντολές εκτέλεσης του docker container. Η

εικόνα είναι το κομμάτι εφαρμογής που πρέπει να εκτελεστεί, και ορίζεται και το δίκτυο στο οποίο θα γίνει η σύνδεση ενός ή πολλαπλών container. Το όνομα υπηρεσίας γίνεται το όνομα container και το όνομα DNS του container, το οποίο μπορούν να χρησιμοποιήσουν άλλα container για να συνδεθούν στο δίκτυο Docker. Το όνομα δικτύου στην υπηρεσία είναι ”spider”.



Εικόνα 22. Η αρχιτεκτονική ενός docker-compose και η σύνδεση του σε 1 δίκτυο

Το Docker Compose καταγράφει όλες τις ιδιότητες που χρειάζονται για να διαμορφωθεί μια εφαρμογή και μπορεί επίσης να καταγράψει άλλους πόρους Docker ανώτατου επιπέδου, όπως volumes. Η παραπάνω εφαρμογή έχει μόνο μία υπηρεσία και ακόμη και σε αυτήν την περίπτωση είναι καλό να υπάρχει ενα docker-compose που μπορεί να χρησιμοποιηθεί για να εκτελεστεί η εφαρμογή και να τεκμηριωθεί η ρύθμισή της. Αλλά ενα docker-compose έχει πραγματικά νόημα όταν εκτελούνται εφαρμογές πολλαπλών container.

Με τον ίδιο τρόπο όπως στην εικόνα 22, μπορούμε να προσθέσουμε ακόμη μια εφαρμογή, έστω μια βάση δεδομένων, να προσδιορίσουμε αρχικά το image της (π.χ. mongo), και έπειτα να εισαχθούν κάποια βασικά configurations όπως το δικό της όνομα (crime-DB), το δικό της port και μετά να εισαχθεί στο ίδιο δίκτυο ”spider”. Αν η εφαρμογή ”crime-UI” εμφανίζει δεδομένα από την βάση ”crime-DB”, τότε μπορούμε να πούμε ότι εξαρτάται από την βάση δεδομένων. Άρα, στο configuration μπορούμε να εισάγουμε ”depends_on” για να δείξουμε αυτή την εξάρτηση και σειρά που θα τρέξουν οι δύο εφαρμογές.

Παρακάτω φαίνεται ένα απλό παράδειγμα ενός docker-compose αρχείο με 2 dummy containers όπου βρίσκονται και οι δύο στο ίδιο network και συνεργάζονται μεταξύ τους καθώς επίσης η μία εφαρμογή (crime-UI) εξαρτάται από την δεύτερη (crime-DB).

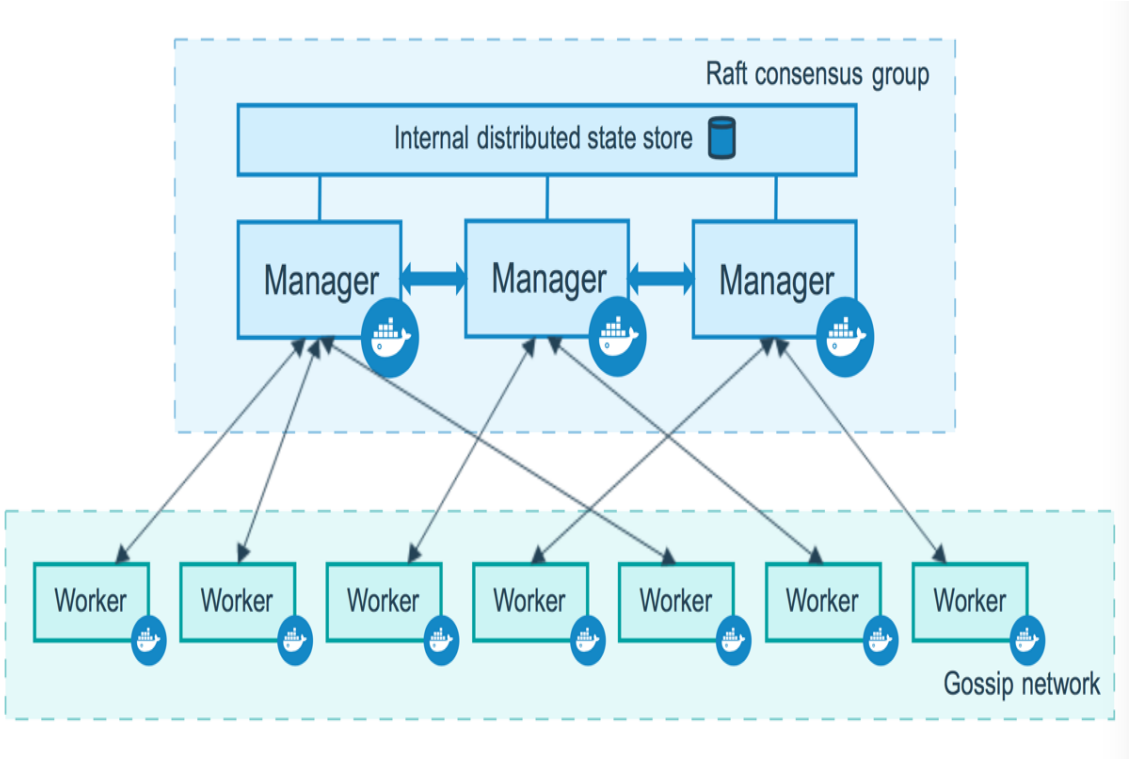
```
services:
    crime-UI:
     image:
        "simonaMnv/newscrawl:latest"
     networks:
        - "spider"
     depends_on:
        - "crime-DB"

    crime-DB:
     image:
        mongo
     ports:
        - "80"
      depends_on:
        - "crime-DB"
```

Ένα container είναι ένα εικονικοποιημένο περιβάλλον με δικό του χώρο δικτύου. Κάθε container έχει μια εικονική διεύθυνση IP που έχει εκχωρηθεί από το Docker και τα container που είναι συνδεδεμένα στο ίδιο δίκτυο Docker μπορούν να φτάσουν μεταξύ τους χρησιμοποιώντας τις διευθύνσεις IP τους. Αλλά τα container αντικαθίστανται κατά τη διάρκεια του κύκλου ζωής της εφαρμογής και τα νέα container θα έχουν νέες διευθύνσεις IP, οπότε το Docker υποστηρίζει επίσης την ανακάλυψη υπηρεσιών με DNS. Το Docker έχει ενσωματωμένη τη δική του υπηρεσία DNS. Οι εφαρμογές που εκτελούνται σε container κάνουν αναζήτηση τομέα όταν προσπαθούν να έχουν πρόσβαση σε άλλα στοιχεία. Η υπηρεσία DNS στο Docker εκτελεί αυτήν την αναζήτηση - εάν το όνομα τομέα είναι όντως όνομα κοντέινερ, το Docker επιστρέφει τη διεύθυνση IP του container και ο καταναλωτής μπορεί να εργαστεί απευθείας μέσω του δικτύου Docker. Εάν το όνομα τομέα δεν είναι container, το Docker μεταφέρει το αίτημα στον διακομιστή όπου λειτουργεί το Docker, επομένως θα κάνει μια τυπική αναζήτηση DNS για να βρει μια διεύθυνση IP στο δίκτυο του οργανισμού ή στο δημόσιο διαδίκτυο.

2.6.3 Μετάβαση σε multi-node περιβάλλον

Το Docker Compose είναι ιδανικό για τη λειτουργία container σε ένα μόνο μηχάνημα, αλλά αυτό δεν λειτουργεί σε περιβάλλον παραγωγής - εάν το μηχάνημα παραμείνει εκτός σύνδεσης, χάνονται όλες οι εφαρμογές. Τα συστήματα παραγωγής χρειάζονται υψηλή διαθεσιμότητα, όπου έρχεται η ενορχήστρωση. Ένας ενορχηστρωτής είναι βασικά πολλές μηχανές όλες συγκεντρωμένες για να σχηματίσουν ένα σύμπλεγμα. Ο ενορχηστρωτής διαχειρίζεται container, διανέμει εργασία σε όλα τα μηχανήματα, εξισορροπεί την κυκλοφορία του δικτύου και αντικαθιστά τυχόν container που είναι προβληματικά.

Δημιουργείται ένα σύμπλεγμα εγκαθιστώντας το Docker σε κάθε ένα από τα μηχανήματά και, στη συνέχεια, τα αυτά ενώνονται μαζί με την πλατφόρμα ενορχήστρωσης - Docker Swarm. Από εκεί και πέρα μπορεί να γίνει η διαχείριση του συμπλέγματος με εργαλεία γραμμής εντολών.



Εικόνα 23. Αρχιτεκτονική Docker Swarm

Ο ενορχηστρωτής προσφέρει ένα σύνολο επιπλέον δυνατοτήτων. Υπάρχει μια κατανεμημένη βάση δεδομένων στο σύμπλεγμα που αποθηκεύει όλες τις πληροφορίες σχετικά με τις εφαρμογές. Στη συνέχεια, υπάρχει ένας χρονοπρογραμματιστής που επεξεργάζεται πού να εκτελέσει τα container και ένα σύστημα για την αποστολή παλμών μεταξύ όλων των διακομιστών του συμπλέγματος. Αυτά είναι τα βασικά δομικά στοιχεία για την αξιοπιστία. Υπάρχει ενα YAML που προσδιορίζει το configuration στο σύμπλεγμα. Αποθηκεύει αυτές τις πληροφορίες και στη συνέχεια προγραμματίζει τα container για την εκτέλεση της εφαρμογής - διανέμοντας το έργο σε διακομιστές με διαθέσιμη χωρητικότητα. Όταν η εφαρμογή εκτελείται, το σύμπλεγμα διασφαλίζει ότι συνεχίζει να λειτουργεί. Εάν ένας διακομιστής παραμείνει εκτός σύνδεσης τότε θα χαθούν containers και το σύμπλεγμα θα ξεκινήσει την αντικατάσταση containers σε άλλους διακομιστές.

Οι ενορχηστρωτές κάνουν όλη την διαχείριση των container ενω πρακτικά χρειάζεται μόνο να οριστεί η επιθυμητή κατάσταση στα αρχεία YAML και δεν χρειάζεται να γνωρίζουμε πόσοι διακομιστές βρίσκονται στο σύμπλεγμα ή πού εκτελούνται τα container. Ο ενορχηστρωτής παρέχει επίσης δυνατότητες δικτύωσης, διαμόρφωσης εφαρμογών και αποθήκευσης δεδομένων. Ο ενορχηστρωτής κρύβει τις λεπτομέρειες των μεμονωμένων μηχανών, δικτύων και συσκευών αποθήκευσης. Βλέπουμε το σύμπλεγμα ως μία μονάδα, στέλνοντας εντολές και εκτελώντας ερωτήματα μέσω του API, στο οποίο συνδέεται η γραμμή εντολών. Το σύμπλεγμα θα μπορούσε να είναι 1.000 μηχανές ή ένα μόνο μηχάνημα. Οι χρήστες της εφαρμογής σας θα μπορούσαν να συνδεθούν με οποιονδήποτε διακομιστή στο σύμπλεγμα και το επίπεδο ενορχήστρωσης φροντίζει για τη δρομολόγηση της κίνησης σε container.

Κεφάλαιο 3

Υλοποίηση της Πλατφόρμας Ανάλυσης Εγλημάτων

3.1 Δομή των αρχείων

Παρακάτω περιγράφεται γενική δομή των αρχείων που περιλαμβάνουν την υλοποίηση.

1. **db_models**
   (α) *article_model.py*
2. **crawling**
   (α) *article_spider.py*
   (β) *pipelines.py*
   (γ) *settings.py*
3. **dash**
   (α) *crime_UI.py*
4. **elasticsearchapp**
   (α) *custom_analyzers.py*
   (β) *documents.py*
   (γ) *query_results_api.py*
   (δ) *serializers.py*
   (ε) *urls.py*
5. **machine_learning**
   (α) classification
      i. *ML_classifiers.py*

(β) **NER**

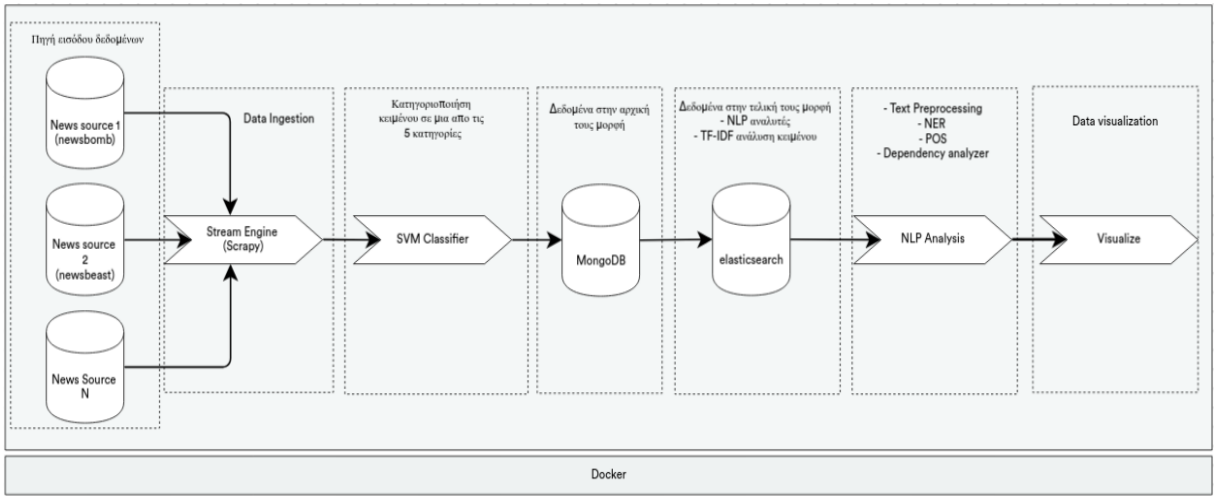
i. custom_model_dir
ii. *custom_NER_trainer.py*
iii. *custom_NER_tester.py*
iv. *EL_NER_updater.py*

(γ) **POS**

i. *define_patterns.py*
ii. *custom_NER_tester.py*
iii. *analysis.py*

6. **docker-compose.yml**

Στο πρώτο κομμάτι της εργασίας γίνεται η συλλογή των raw data (db_models και crawling layer). Έπειτα, ορίζεται ένα train set από τα αρχικά δεδομένα που συλλέχθηκαν και με βάση αυτό φτιάχνουμε έναν custom ταξινομητή, ο οποίο χρησιμοποιείται αργότερα για να διακρίνει τα άρθρα χωρίς κατηγορία. Για την ανάλυση των άρθρων και την εξαγωγή πληροφοριών χρησιμοποιούνται τα elasticsearchapp και machine_learning layers. Για την οπτικοποίηση χρησιμοποιείται το dash layer και τέλος, όλες αυτές οι μικροϋπηρεσίες περιγράφονται στο docker-compose.yml. Η υλοποίηση περιγράφετε στις παρακάτω ενότητες σε μεγαλύτερη λεπτομέρεια.



Εικόνα 24. Η γενική αρχιτεκτονική του συστήματος ανάλυσης εγκλημάτων

3.2 Συγκρότηση Δείγματος & Συλλογή Δεδομένων

Για την συλλογή του δείγματος χρησιμοποιήθηκε το scrapy βασισμένη στην python. Ως πρώτο στρώμα αποθήκευσης, χρησιμοποιείται η mongoDB για να αποθηκεύσει τα δεδομένα στην αρχική τους μορφή. Το δείγμα δεδομένων συλλέχθηκε από ανοιχτά δεδομένα, και πιο συγκεκριμένα, από ιστοτόπους με ειδήσεις που αφορούν την Ελλάδα (π.χ. newsbomb). Παρακάτω φαίνεται το σχήμα της MongoDB, δηλαδή, ποια πεδία θα συλλεχθούν από κάθε άρθρο.

```
    _id = models.ObjectIdField()
    title = models.TextField(null=False, blank=False)
    date = models.DateTimeField(null=False, blank=False)
    body = models.TextField(null=False, blank=False)
    tags = models.TextField(null=True, blank=True)
    author = models.CharField(max_length=255, unique=False)
    link = models.TextField(null=True, blank=True)
    type = models.TextField(null=True, blank=True)
    scope = models.TextField(null=True, blank=True)
```

Το id είναι ένα πεδίο που θα είναι μοναδικό για κάθε άρθρο Στην συγκεκριμένη εργασία, προκύπτει παίρνοντας το hash του link για το κάθε άρθρο. Το title είναι ο τίτλος που φαίνεται σε κάθε άρθρο, το date είναι η ημερομηνία που εκδόθηκε η είδηση, το body είναι το κύριο περιεχόμενο του άρθρου, το πεδίο tags είναι οι λέξεις κλειδιά που περιέχει κάθε άρθρο, το πεδίο author είναι ο συγγραφέας του άρθρου. Αυτό μπορεί να συλλεχθεί και να μελετηθεί μελλοντικά ώστε να δούμε τα στατιστικά του κάθε συγγραφέα και το πόσο έγκυρα είναι τα άρθρα που συγγράφει καθώς και να διακρίνουμε fake-news με βάση τον δημοσιογράφο. Το πεδίο link περιλαμβάνει το url της είδησης, το πεδίο type περιέχει το είδος της είδησης (δολοφονία, ληστεία, τρομοκρατική επίθεση) αν και όταν αυτή υπάρχει ενώ το πεδίο scope περιέχει την τοποθεσία (Ελλάδα/Κόσμος) όπου έγινε το έγκλημα.

Το πεδίο type δεν περιλαμβάνεται στα περισσότερα άρθρα, όμως, θα γίνει η συλλογή από τα άρθρα στα οποία υπάρχει, ξεκινώντας από το πρώτο δυνατό άρθρο της ιστοσελίδας, ώστε να αναπτυχθεί ο classifier.

Παρακάτω φαίνεται ένα δείγμα από την συλλογή των raw data.

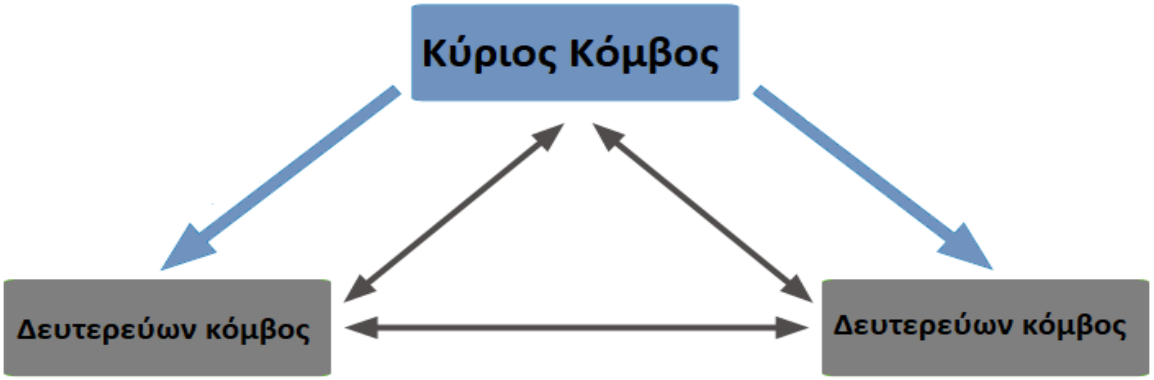

```
{
    "_id": "ef25dd002c71ed8773403e73",
    "author": "Newsroom",
    "body": "Αρπαγή Μαρκέλλας: Συγκλονιστικά είναι τα στοιχεία που έρχονται στο φως της δημοσιότητας μετά
    την αρπαγή και την εκμετάλλευση της 10χρονης Μαρκέλλας - Η γυναίκα που κρατείται προσωρινά επιχείρησε
    να αλλάξει την αρχική της κατάθεση..."
    "date": {"2020-06-22T06:46:13.000Z"},
    "link": "https://www.newsbomb.gr/ellada/astynomiko-reportaz/story/1093192/markella-vrethike-simeiomatar
    io-sto-spiti-tis-33xronis-ti-apokalyptei",
    "scope": "ΚΟΣΜΟΣ",
    "tags": "10ΧΡΟΝΗ, ΑΝΗΛΙΚΟ, ΑΡΠΑΓΗ ΑΝΗΛΙΚΟΥ, ΑΡΠΑΓΗ, ΑΠΑΓΩΓΗ, 33ΧΡΟΝΗ, ΑΣΤΥΝΟΜΙΑ, ΕΙΔΗΣΕΙΣ, ΕΙΔΗΣΕΙΣ
    ΣΗΜΕΡΑ, ΝΕΑ, ΑΣΤΥΝΟΜΙΚΟ",
    "title": "Μαρκέλλα: Βρέθηκε σημειωματάριο στο σπίτι της 33χρονης - Τι αναφέρει"
},
```


Εικόνα 25. Τα αρχικά (raw) δεδομένα, αποθηκευμένα στη mongoDB

3.2.1 Αντίγραφα Δεδομένων

Ένα σύνολο αντιγράφων (replica set) είναι μια ομάδα που διατηρούν το ίδιο σύνολο δεδομένων. Ένα σύνολο αντιγράφων περιέχει πολλούς κόμβους που φέρουν δεδομένα και προαιρετικά έναν κόμβο διαιτητή. Από τους κόμβους δεδομένων, ένα και μόνο ένα μέλος θεωρείται ο κύριος κόμβος, ενώ οι άλλοι κόμβοι θεωρούνται δευτερεύοντες. Ο λόγος που

θέλουμε να κρατήσουμε αντίγραφα είναι για να διατηρηθούν τα δεδομένα ασφαλή, για υψηλή διαθεσιμότητα δεδομένων, και για αποκατάσταση από καταστροφές.



Εικόνα 26. Docker MongoDB Replica

Ο κύριος κόμβος λαμβάνει όλες τις πράξεις εγγραφής. Ένα σύνολο αντιγράφων μπορεί να έχει μόνο ένα κύριο που μπορεί να επιβεβαιώσει εγγραφές. Αν και σε ορισμένες περιστάσεις, μια άλλη περίπτωση mongod μπορεί προσωρινά να πιστεύει ότι είναι επίσης πρωταρχική. Ο κύριος κόμβος καταγράφει όλες τις αλλαγές στα σύνολα δεδομένων του στο αρχείο καταγραφής λειτουργίας του, δηλαδή το oplog. Οι δευτερεύων κόμβοι αναπαράγουν το oplog του πρωτοβάθμιου και εφαρμόζουν τις πράξεις στα σύνολα δεδομένων τους έτσι ώστε τα σύνολα δεδομένων των δευτερευόντων να αντικατοπτρίζουν το σύνολο δεδομένων του πρωτογενούς.

Για να εφαρμόσουμε την αντιγραφή δεδομένων (replica set) χρησιμοποιείται τοπικά docker. Παρακάτω φαίνεται η ρύθμιση των 3 αντιγράφων που εφαρμόζεται στο docker-compose. Συνολικά υπάρχουν 3 κόμβοι απο τους οποίους ο ένας θα είναι ο πρωτεύων.

```
var cfg = {
    "_id": "rs0",
    "protocolVersion": 1,
    "version": 1,
    "members": [
        {
            "_id": 0,
            "host": "\${MONGODB1}:27017",
            "priority": 2
        },
        {
            "_id": 1,
            "host": "\${MONGODB2}:27017",
            "priority": 0
        },
        {
            "_id": 2,
            "host": "\${MONGODB3}:27017",
            "priority": 0,
        }
    ]
}
```

Το docker-compose αρχείο θα αποτελείται συνολικά απο 4 services και 1 network στο οποίο θα ανήκουν τα services. Χρησιμοποιούνται 3 replicas καθώς αυτό είναι και το ελάχιστο που μπορούμε να έχουμε. Τα 3 services με όνομα mongo1, mongo2, mongo3 είναι instances της mongodb. Το τελευταίο service με όνομα mongo-replica-setup, είναι αυτό που εφαρμόζει την ρύθμιση των υπολοίπων, όπως αυτή περιγράφηκε παραπάνω, και είναι σε λειτουργία όση ώρα χρειάζεται ώστε να ρυθμίσει τα replica sets. Στην υλοποίηση ο πρωτεύων κόμβος είναι ο mongo1, ο οποίος πρέπει οπωσδήποτε να είναι σε λειτουργία για να ακολουθήσουν οι υπόλοιποι. Η συγκεκριμένη υλοποίηση είναι με αυθεντικοποίηση άρα το κάθε service διαβάζει ενα file.key (ssh κλειδί) για να υπάρχει εμπιστοσύνη μεταξύ των replicas.

Παρακάτω φαίνεται όλη η υλοποίηση.

```
services:
    mongo-replica-setup:
        container_name: mongo-setup
        image: 'mongo:4.2'
        restart: on-failure
        networks:
            - netSwarmlabMongo
        volumes:
        - ./.docker/mongodb/scripts/mongosetup.sh:/scripts/mongosetup.
                                                    sh
        entrypoint: ["bash", "/scripts/mongosetup.sh" ]
        depends_on:
            - mongo1
            - mongo2
            - mongo3
    mongo1:
        image: 'mongo:4.2'
        command: ["-f", "/etc/mongod.conf", "--keyFile", "/auth/file.
                                            key", "--replSet", "
                                            crime_replica", "--
                                            bind_ip_all"]
        ports:
            - 30001:27017
        networks:
            - netSwarmlabMongo
        volumes:
            - swarmlabmongoData1:/data/db
            - swarmlabmongoLog1:/var/log/mongodb
            - ./.docker/mongodb/initdb.d/:/docker-entrypoint-initdb.d/
            - ./.docker/mongodb/mongod.conf:/etc/mongod.conf
            - ./.docker/mongodb/file.key:/auth/file.key
        environment:
            MONGO_INITDB_DATABASE: "crime_db"
    mongo2:
        image: 'mongo:4.2'
        command: ["-f", "/etc/mongod.conf", "--keyFile", "/auth/file.
                                            key", "--replSet", "
                                            crime_replica", "--
                                            bind_ip_all"]
        ports:
            - 30002:27017
        networks:
            - netSwarmlabMongo
```

```
        volumes:
            - swarmlabmongoData2:/data/db
            - swarmlabmongoLog2:/var/log/mongodb
            - ./.docker/mongodb/mongod.conf:/etc/mongod.conf
            - ./.docker/mongodb/file.key:/auth/file.key
        environment:
            MONGO_INITDB_DATABASE: "crime_db"
        depends_on:
            - mongo1
    mongo3:
        image: 'mongo:4.2'
        command: ["-f", "/etc/mongod.conf", "--keyFile", "/auth/file.
                                            key", "--replSet", "
                                            crime_replica", "--
                                            bind_ip_all"]
        ports:
            - 30003:27017
        networks:
            - netSwarmlabMongo
        volumes:
            - swarmlabmongoData3:/data/db
            - swarmlabmongoLog3:/var/log/mongodb
            - ./.docker/mongodb/mongod.conf:/etc/mongod.conf
            - ./.docker/mongodb/file.key:/auth/file.key
        environment:
            MONGO_INITDB_DATABASE: "crime_db"
        depends_on:
            - mongo1
    volumes:
        swarmlabmongoData1:
        swarmlabmongoData2:
        swarmlabmongoData3:
        swarmlabmongoLog1:
        swarmlabmongoLog2:
        swarmlabmongoLog3:
    networks:
        netSwarmlabMongo:
```

3.3 Υλοποίηση και Ανάλυση Κειμένου μέσω Elasticsearch

Μέσω του elasticsearch, γίνεται η συλλογή όλων των raw δεδομένων από την mongoDB (indexing) και εφαρμόζονται αναλυτές κειμένου για να παραχθεί το επεξεργασμένο κείμενο. Παράγουμε πάνω απο εναν αναλυτή κειμένου και ο καθένας χρησιμοποιείται για διαφορετικές εργασίες. Αυτή η ανάλυση μας βοηθαέι να εξάγουμε με μεγαλύτερη ευκολία τα σημαντικά μέρη των προτάσεων και να αφαιρέσουμε στοιχεία που μπορεί να δυσκολέψουν την ανάλυση (όπως π.χ. τόνους).

3.3.1 Υλοποιήση Τροποποιημένου Αναλυτή Κειμένου

Για την διευκόλυνση μας, θα χρησιμοποιήσουμε έναν ειδικά φτιαγμένο αναλυτή ελληνικού κειμένου ο οποίος θα μεταφράζει το κείμενο σε πεζά, θα αφαιρεί τις λέξεις κλειδιά

που επαναλαμβάνονται πολλές φορές και άρα δεν θεωρούνται σημαντικές, και τέλος, θα εφαρμόζει stemming.

```
greek_analyzer = analysis.analyzer(
    "greek_analyzer",
    type="custom",
    tokenizer="standard",
    filter=[
           analysis.token_filter("greek_lowercase", type="lowercase",
                                                 language="greek"),
           analysis.token_filter('greek_stop', type="stop", stopwords="
                                                 _greek_"),
           analysis.token_filter('greek_stemmer', type="stemmer",
                                                 language="greek"),
           analysis.token_filter('spacy_sw', type="stop", stopwords=
                                                 remove_accent(spacy_sw))
        ],
)
```

Στον elasticseach μπορούμε μέσω API calls να αναφερθούμε στα raw data ή στα analyzed data, εφόσον έχουμε εφαρμόσει κάποιο αναλυτή κειμένου. Για να αναφερθούμε στον παραπάνω αναλυτή, θα πρέπει να ορίσουμε ενα συγκεκριμένο API call μέσω του elasticsearch όπως φαίνεται παρακάτω:

```
    url = "http://127.0.0.1:9200/articles/_analyze"
```

Ενώ αν θέλουμε να αναφερθούμε στα raw data, τότε:

```
    url = "http://127.0.0.1:9200/articles/_search"
```

Οπότε, μπορούμε για παράδειγμα να κάνουμε το εξής API call προς τον elastic:

{

”analyzer”: ”greek_analyzer”,

”text”: ”Σκότωσε την γυναίκα και τον άντρα στο σπίτι.”

}

Το οποίο θα επιστρέψει το εξής:

{

”tokens”:[

”token”:”σκοτωσ”,”start_offset”:0,”end_offset”:7,”type”:”<ALPHANUM>”,”position”:0,

”token”:”γυναικ”,”start_offset”:12,”end_offset”:19,”type”:”<ALPHANUM>”,”position”:2,

”token”:”αντρ”,”start_offset”:28,”end_offset”:33,”type”:”<ALPHANUM>”,”position”:5,

”token”:”σπιτ”,”start_offset”:34,”end_offset”:39,”type”:”<ALPHANUM>”,”position”:6

] }

Για να γίνει το indexing και η μεταφορά των δεδομένων από την mongoDB προς τον elasticsearch θα πρέπει να έχουμε ορίσει την δομή των εγγραφών. Άρα, με παρόμοιο τρόπο όπως και στην mongoDB, ορίζουμε τα πεδία που θα περιέχει η κάθε εγγραφή. Η μόνη διαφορά είναι ότι την στιγμή που ορίζουμε ένα πεδίο, θα πρέπει να συμπεριληφθεί και ο αναλυτής κειμένου σε αυτό - αν αυτός υπάρχει, όπως φαίνεται παρακάτω.

```
class ArticleDocument(Document):
    title = fields.TextField(
        analyzer=greek_analyzer,  # main analyzer
        fields={'simple_analyzer': fields.TextField(analyzer=
                                        greek_simple_analyzer)}
    )
    date = fields.DateField()
    body = fields.TextField(
        analyzer=greek_analyzer,
        fields={'simple_analyzer': fields.TextField(analyzer=
                                        greek_simple_analyzer)}
    )
    tags = fields.TextField()
    author = fields.TextField()
    link = fields.TextField()
    type = fields.TextField()
    scope = fields.TextField()
```

Επιπλέον, θα ορίσουμε και κάποια ακόμη πεδία στα οποία θα αποθηκεύεται η ανάλυση των αρχικών πεδίων.

```
    crime_analysis = fields.NestedField(
        attr='article_analysis',
        properties={
            'acts_committed': fields.TextField(),
            'location_of_crime': fields.TextField(),
            'ages_involved': fields.TextField(),
            'time_of_crime': fields.TextField(),
            'victim_gender': fields.TextField(),
            'criminal_status': fields.TextField(),
            'drug_type': fields.TextField(),
        }
    )
```

Το πεδίο acts_commited περιλαμβάνει λέξεις κλειδιά που αναφέρουν το έγκλημα που έχει διαπραχθεί, το location_of_crime αναφέρει την τοποθεσία του εγκλήματος, το ages_involved αναφέρει την ηλικία των ατόμων που εμπλέκονται στο έγκλημα, το time_of_crime εξάγει το χρονικό πλαίσιο που συνέβη το έγκλημα, το victim_gender εξάγει το γένος του θύματος, το criminal_status εξάγει την πληροφορία σχετικά με το αν ο δράστης συνελήφθη και τέλος, το drug_type εξάγει το είδος των ναρκωτικών που υπάρχουν στο άρθρο - μόνο αν το άρθρο έχει ταξινομηθεί στη αντίστοιχη κατηγορία.

3.4 Ανάπτυξη Ταξινομητή Εγκλημάτων

Για την ανάπτυξη του ταξινομητή, έχουμε στην διάθεση μας ένα dataset με 10.000 εγκλήματα τα οποία έχουν κατηγορία εγκλήματος. Η κατηγορία εγκλήματος έχει εξαχθεί από το crawling layer με βάση τις λέξεις-κλειδιά (tags) που έχουν ανατεθεί από τον δημοσιογράφο. Ο στόχος του ταξινομητή είναι να μπορεί να διακρίνει την κατηγορία του εγκλήματος στα άρθρα που δεν έχουν ήδη κατηγορία.

3.4.1 Εκπαίδευση και Ακρίβεια Μοντέλων

Για την εκπαίδευση του μοντέλου, πραγματοποιείται ένα API call προς τα δεδομένα με τον αναλυτή στον elastic. Μόλις μαζέψουμε όλα τα δεδομένα που έχουν κατηγορία, φτιάχνουμε ένα λεξικό που αποτελείται από τα εξής πεδία:

```
    df = pd.DataFrame({'article_tokens': total_data, 'crime_type':
                                        total_types})
```

Μερικά παραδείγματα φαίνονται παρακάτω:

article_tokens	crime_type
[’ληστ’, ’ξηλων’, ’χρηματοκιβωτ’, ’συνεληφθ’, ’ξημερωμα’, ’πεμπτ’, ’παλαι’, ’ψυχ’, ...]	ΛΗΣΤΕΙΑ/ΚΛΟΠΗ
[’τρ’, ’ατομ’, ’συνεληφθ’, ’βραδ’, ’κυριακ’, ’10’, ’5’, ’2015’, ’ηρακλει’, ’κατηγορουμεν’, ’περιπτωσ’, ’αδικημα’, ’καλλιεργει’, ’δενδρυλλ’, ’κανναβ’, ’κατοχ’, ’διακινησ’, ...]	ΝΑΡΚΩΤΙΚΑ
[’ερευν’, ’δολοφον’, ’ατυχ’, ’ηθοποι’, ’βρισκ’, ’εξελιξ’, ’γυρ’, ’διαδικτυ’, ’καν’, ’δυσαρεστ’, ’ειδησ’, ’70χρον’, ’ηθοποι’, ...]	ΔΟΛΟΦΟΝΙΑ

Πίνακας 11. Train input

Η κάθε εγγραφή θα χρησιμοποιηθεί για την εκμάθηση του μοντέλου.

```
corpus = pd.read_csv('../../dfs/elastic_analyzed_articles.csv')
    train_X, test_X, train_Y, test_Y = model_selection.train_test_split
    (
        corpus['article_tokens'],
        corpus['crime_type'],
        test_size=0.2
    )
```

Το corpus θα χωριστεί σε δύο σύνολα δεδομένων, το Training και το Test. Τα δεδομένα εκπαίδευσης θα έχουν το 80% του σώματος και τα δεδομένα δοκιμής θα έχουν το υπόλοιπο 20%. Η κάθε είσοδος (train και test) θα περάσει από έναν κωδικοποιητή ο

οποίος κωδικοποιεί ετικέτες με τιμή μεταξύ 0 και n_classes-1. Αυτό γίνεται για να μετασχηματίσει τα δεδομένα κατηγορίας του τύπου συμβολοσειράς στο σύνολο δεδομένων σε αριθμητικές τιμές τις οποίες το μοντέλο μπορεί να κατανοήσει.

Συνολικά έχουμε τις εξής κατηγορίες:

{
'ΔΟΛΟΦΟΝΙΑ': 0,
'ΛΗΣΤΕΙΑ/ΚΛΟΠΗ': 1,
'ΝΑΡΚΩΤΙΚΑ': 2,
'ΣΕΞΟΥΑΛΙΚΟ ΕΓΚΛΗΜΑ': 3,
'ΤΡΟΜΟΚΡΑΤΙΚΗ ΕΠΙΘΕΣΗ': 4,
'ΑΛΛΟ ΕΓΚΛΗΜΑ': 5
}

Πριν εφαρμόσουμε κάποιο ταξινομητή στα δεδομένα, θα εφαρμοστεί διανομή λέξεων (Word Vectorization). Πρόκειται για μια γενική διαδικασία μετατροπής μιας συλλογής εγγράφων κειμένου σε αριθμητικά διανύσματα χαρακτηριστικών. Είναι πολλές μέθοδοι για τη μετατροπή δεδομένων κειμένου σε διανύσματα που το μοντέλο μπορεί να κατανοήσει, αλλά μακράν η πιο δημοφιλής μέθοδος είναι η TF-IDF. Άρα, θα μετατρέψουμε το Train_X και το Test_X σε vectorized Train_X_Tfidf και Test_X_Tfidf. Αυτά θα περιέχουν τώρα για κάθε σειρά μια λίστα με μοναδικό ακέραιο αριθμό και τη σχετική σημασία του, όπως υπολογίζεται από το TF-IDF.

Παρακάτω φαίνονται τα μοντέλα που έχουν χρησιμοποιηθεί και η αντίστοιχη ακρίβεια τους.

Classifier	**Ακρίβεια (accuracy_score)**
SVM	97.77%
NB	91.98%
KNN	85.31%
Decision Trees	97.47%

Πίνακας 12. Train input

3.5 Εξαγωγή Πληροφοριών απο τα Άρθρα

Αρχικά φαίνεται η πληροφορία που υπάρχει στον elasticsearch, που έχει προηγουμένως μαζευτεί με crawlers. Η πληροφορία αυτή περιλαμβάνει την ημερομηνία του άρθρου, τον τίτλο του άρθρου, το περιεχόμενο του άρθρου, οι λέξεις-κλειδιά του άρθρου και η κατηγορία του άρθρου που έχει συλλεχθεί είτε μέσω crawlers ή προέκυψε μέσω την κατηγοριοποίησης, όταν δεν βρίσκει ο crawler στο αντίστοιχο πεδίο κάποια διαθέσιμη κατηγορία.

Date	Title	Body	Type	Tags
2012-05-10	Αποπειράθηκαν να απαγάγουν βρέφος από το καροτσάκι του!	Τρεις άγνωστοι αποπειράθηκαν να απαγάγουν ένα δίχρονο ...	ΔΟΛΟΦΟΝΙΑ	ΠΕΛΟΠΟΝΝΗΣΟΣ, ΑΠΑΓΩΓΗ, ΒΡ
2012-11-21	Κρήτη: Ράγισε καρδιές το ξέσπασμα της χήρας του Κυπριωτ...	Μια από τις πλέον συγκλονιστικές υποθέσεις που έχουν κατα...	ΔΟΛΟΦΟΝΙΑ	ΚΡΗΤΗ, ΔΟΛΟΦΟΝΙΑ, ΚΥΠΡΙΩΤΑ
2012-03-07	Τρεις αλλοδαποί πίσω από την απαγωγή του 28χρονου Πακι...	Νέα στοιχεία έφεραν στο φως της δημοσιότητας οι έρευνες τ...	ΔΟΛΟΦΟΝΙΑ	ΣΤΕΡΕΑ ΕΛΛΑΔΑ, ΑΠΑΓΩΓΗ, ΣΥΛ
2012-03-12	Ένοχοι 6 για την απαγωγή του Παναγόπουλου	Με τα λύτρα της απαγωγής του εφοπλιστή Περικλή Παναγόπ...	ΔΟΛΟΦΟΝΙΑ	ΑΤΤΙΚΗ, ΑΠΑΓΩΓΗ, ΔΙΚΑΣΤΗΡΙΟ,
2012-03-22	Ρέθυμνο: Μυστήριο με τη δολοφονία Ινδού	Τραγική κατάληξη είχε η απαγωγή ενός 44χρονου Ινδού στο ...	ΔΟΛΟΦΟΝΙΑ	ΕΛΛΗΝΙΚΗ ΑΣΤΥΝΟΜΙΑ, ΔΟΛΟΦΟ
2012-10-18	Θρίλερ με την απαγωγή 25χρονου	Ένα πρωτοφανές θρίλερ βρίσκεται σε εξέλιξη, καθώς οι αστυ...	ΔΟΛΟΦΟΝΙΑ	ΑΤΤΙΚΗ, ΑΠΑΓΩΓΗ, ΑΠΑΓΩΓΗ 25Χ
2012-10-19	Εγκέφαλος απαγωγής: «Ο 25χρονος τη σκηνοθέτησε μόνος ...	Ένα θρίλερ διαδραματίζεται τις τελευταίες ώρες με την απαγ...	ΔΟΛΟΦΟΝΙΑ	ΑΠΑΓΩΓΗ, ΑΠΑΓΩΓΗ 25ΧΡΟΝΟΥ,
2012-10-19	Στη δημοσιότητα τα στοιχεία των απαγωγέων και του 26χρονου	Στο φως της δημοσιότητας έδωσε η Αστυνομία τα στοιχεία τα...	ΔΟΛΟΦΟΝΙΑ	ΕΛΛΗΝΙΚΗ ΑΣΤΥΝΟΜΙΑ, ΣΥΛΛΗΨ

Εικόνα 27. Βασική Πληροφορία Άρθρων

Για να εξαχθούν οι υπόλοιπες πληροφορίες, χρησιμοποιούνται διάφορες τακτικές μηχανική μάθησης. Οι υπόλοιπες πληροφορίες φαίνονται παρακάτω.

Victim	Criminal Status	Acts	Locations	Ages
ΑΓΝΩΣΤΟ	ΑΓΝΩΣΤΟ	-	Πύργο	-
ΓΥΝΑΙΚΑ	ΟΜΟΛΟΓΗΣΕ	απαγωγή	Κρήτη, Σύρου	25χρονου, 33χρονος
ΑΝΤΡΑΣ	ΟΜΟΛΟΓΗΣΕ	απαγωγή	Πακιστάν, Οινόφυτα	33χρονου, 28χρονου
ΑΓΝΩΣΤΟ	ΑΓΝΩΣΤΟ	απαγωγής, δολοφονίας	Γαλάτσι, Αθήνας, Κορυδαλλού	-
ΑΝΤΡΑΣ	ΑΓΝΩΣΤΟ	δολοφονία, απαγωγής	Λατζμάς, Ινδία, Ρέθυμνο	44χρονου, 24χρονος
ΓΥΝΑΙΚΑ	ΣΥΝΕΛΗΦΘΗ	απαγωγής	Νέα Σμύρνη, Παλαιό Φάληρο, Καλλιθέα	25χρονος, 72χρονος
ΓΥΝΑΙΚΑ	ΑΓΝΩΣΤΟ	δολοφονία, απαγωγής	-	25χρονος, 72χρονος
ΑΓΝΩΣΤΟ	ΑΓΝΩΣΤΟ	απαγωγή	Βαρβάρας	26χρονου

Εικόνα 28. Εξαγωγή πληροφορίας με ML

Για να εξαχθούν οι **ηλικίες των εμπλεκομένων** (Ages), το **μέρος του εγκλήματος** (Location) και οι **ενέργειες του δράστη** εκπαιδεύτηκε ένα μοντέλο NER χρησιμοποιώντας ένα annotation εργαλείο.

Για να εξαχθεί το **φύλο του θύματος** (Victim) χρησιμοποιείται εξόρυξη λέξεων για τις πολύ βασικές λέξεις που υποδηλώνουν ένα συγκεκριμένο φύλο ενώ εάν η εξόρυξη κειμένου αποτύχει τότε χρησιμοποιείται ανάλυση εξάρτησης (dependency collector/analysis) για τον προσδιορισμό του φύλου. Η ανάλυση εξάρτησης εφαρμόζεται σε ένα ”συνοπτικό” σύνολο προτάσεων που εξάγονται από τον τίτλο και το σώμα του κάθε άρθρου. Ο elasticsearch διαθέτει ενσωματωμένο TF-IDF το οποίο χρησιμοποιείται για τη συλλογή όλων των πιο χρησιμοποιημένων ρημάτων (βάσει ενός προκαθορισμένου ορίου) σε κάθε κατηγορία εγκλημάτων. Από όλες τις X περιπτώσεις δολοφονιών που έχουμε, βρίσκουμε τα κορυφαία Y ρήματα που επαναλαμβάνονται συχνότερα σε όλη τη βάση δεδομένων. Μετά από αυτό, συλλέγουμε επίσης όλα τα τοπικά ρήματα από το τρέχον άρθρο και τέλος, εντοπίζουμε τις προτάσεις στις οποίες ταιριάζουν αυτά τα ρήματα και εξάγουμε αυτές τις συγκεκριμένες προτάσεις. Με αυτόν τον τρόπο, δημιουργούμε μια περίληψη του άρθρου που περιλαμβάνει μόνο προτάσεις με τα σημαντικότερα ρήματα. Αυτό μας βοηθά να κρατήσουμε μόνο τις πολύ σημαντικές πληροφορίες για να εξαγάγουμε το φύλο του θύματος, όπως είναι πιθανότερο να υπάρχει σε αυτές τις προτάσεις λόγω του γεγονότος ότι τα ρήματα υπονοούν μια πράξη. Για να περιοριστούν περαιτέρω οι προτάσεις, η αντιστοίχιση βάσει κανόνων (spacy matcher) χρησιμοποιείται για να ταιριάζει με τους γραμματικούς κανόνες στο pattern.py που θα έδειχνε ότι μια πράξη γίνεται από κάποιον ή μια πράξη γίνεται σε κάποιον.

Για να εξαχθεί η **κατάσταση του δράστη** (Criminal Status) πρώτα γίνεται μια εσωτερική ανάλυση στον elasticsearch και συλλέγονται όλα τα ρήματα του συγκεκριμένου άρθρου. Πραγματοποιείται ένα mapping (key:value) λέξεων που δηλώνουν αν κάποιος ομολόγησε ή συνελήφθη, όπως για παράδειγμα οι λέξεις ”Ομολόγησε”, ”Συνελήφθη”, όπου αυτές θα είναι το key και value όλα τα ρήματα που βρέθηκαν στο άρθρο. Σε όλους τους συνδυασμούς (key:value) εφαρμόζεται ένας δείκτης ομοιότητας που αν ξεπεράσει ένα προκαθορισμένο όριο, τότε επιστρέφουμε ως κατάσταση το συγκεκριμένο key. Παρακάτω φαίνεται ένα απλό και μεμονωμένο παράδειγμα της τακτικής αυτής.

```
import spacy

nlp = spacy.load('el_core_news_lg')

x = nlp("άντρας")
y = nlp("γυναίκα")
z = nlp("τίποτα")

print(x.similarity(y))
print(x.similarity(x))
print(x.similarity(z))

## αποτελέσματα ##
# 0.87
# 1.0
# 0.25
```

Κάθε φορά που συλλέγεται ένα νέο άρθρο, εφαρμόζεται και η ανάλυση που περιέχει όλα τα παραπάνω σε αυτό. Ο κώδικας που συλλέγει το άρθρο φαίνεται παρακάτω. Η συνάρτηση που εφαρμόζει την ανάλυση (analyse_victim()) καλείται για να εξάγει πληρο-

φορίες από τον τίτλο και το σώμα του τρέχων άρθρου.

```
class DjangoPipeline(object):
    collection_name = 'scrapy_articles'

    def process_item(self, item, spider):
        return deferToThread(self._process_item, item, spider)

    def _process_item(self, item, spider):
        body = item["body"]

        hashed_id = hashlib.md5(item["link"].encode()).hexdigest()

        # run the ML, POS, Elastic analysis on the title+body
        content = item["title"] + min(final_body, key=len)
        article_summary, victim_gender,
        criminal_status, act,
        age, date,
        specific_person, location = analyse_victim(content, item["type"
                                                ])

        if item["type"] == 'OTHER CRIME' and item["scope"] == 'GREECE':
            # SVM Classification applied on non-tagged articles
            ML_crime_type = classify_crime_type(content)

            article = ArticleOfInterest(_id=str(ObjectId(hashed_id[:24]
                                                    )),
            title=item["title"], date=item["date"],
            body=min(final_body, key=len), tags=item["tags"],
            author=item["author"],link=item["link"],
            type=ML_crime_type, scope=item["scope"],
            victim_gender=victim_gender, criminal_status=
                                                    criminal_status,
            acts_committed=act,ages_of_involved=age,
            time_of_crime=date, person_involved=specific_person,
            location_of_crime=location, drug_type=specific_drug)
        else:
            article = ArticleOfInterest(_id=str(ObjectId(hashed_id[:24]
                                                    )),
            title=item["title"], date=item["date"],
            body=min(final_body, key=len), tags=item["tags"],
            author=item["author"],link=item["link"],
            type=item["type"], scope=item["scope"],
            victim_gender=victim_gender, criminal_status=
                                                    criminal_status,
            acts_committed=act, ages_of_involved=age,
            time_of_crime=date, person_involved=specific_person,
            location_of_crime=location, drug_type=specific_drug)

        article.save()
        return item
```

3.5.1 Εκπαίδευση και Ακρίβεια NER

Για να γίνει η εκπαίδευση ενός NER μοντέλου, πρέπει να έχουμε μια πηγή πληροφορίας. Αρχικά, πραγματοποιήθηκε η εξαγωγή 20-30 άρθρων από κάθε κατηγορία. Πρώτου ξεκινήσει η εκμάθηση του μοντέλου, πρέπει να δημιουργηθεί μια συγκεκριμένη δομή εισόδου για αυτό. Η δομή δημιουργήθηκε με ένα annotation εργαλείο και περιλαμβάνει το αρχικό κείμενο, την τοποθεσία των οντοτήτων που μας ενδιαφέρουν καθώς και το είδος της κατηγορίας. Μερικά παραδείγματα από τα δεδομένα εκπαίδευσης φαίνονται παρακάτω.

”Την Παραμονή της Πρωτοχρονιάς δυο γυναίκες δολοφονήθηκαν στην Αλβανία και σύμφωνα με τους συγγενείς των θυμάτων...”

```
{
  ”entities”:    [1624, 1631, ΠΡΑΞΗ], [1531, 1537, ΠΡΑΞΗ],
                 [361, 368, ΠΡΑΞΗ], [316, 328, ΗΜΕΡΟΜΗΝΙΑ],
                 [272, 282, ΠΡΑΞΗ], [43, 56, ΠΡΑΞΗ],
                 [17, 29, ΗΜΕΡΟΜΗΝΙΑ],
}
```

”Φρίκη και οργή προκαλεί στην ρουμανική κοινή γνώμη η υπόθεση απαγωγής και δολοφονίας δύο νεαρών κοριτσιών...”

```
{
  ”entities”:    [900, 907, ΗΜΕΡΟΜΗΝΙΑ], [859, 866, ΗΛΙΚΙΑ],
                 [758, 765, ΗΛΙΚΙΑ],[718, 726, ΗΛΙΚΙΑ],
                 [559, 564, ΠΡΑΞΗ], [547, 557, ΠΡΑΞΗ],
                 [409, 416, ΗΛΙΚΙΑ],
}
```

Όπου οι αριθμοί (απο αριστερά προς τα δεξιά) υποδηλώνει την αρχή και το τέλος της λέξης και μετά ακολουθεί η κατηγορία στην οποία ανήκει η λέξη αυτή.

Ο κώδικας για την εκμάθηση του μοντέλου φαίνεται παρακάτω.

```
nlp = spacy.load('el_core_news_lg')
ner = nlp.get_pipe("ner")

# training data
from ner_train_data import TRAIN_DATA

# spacy Training an additional entity type to existing el model
def main(model=None, new_model_name="crime-analysis", n_iter=30):
    random.seed(0)
    if model is not None:
        nlp = spacy.load(model)  # load existing spaCy model
        print("Loaded model '%s'" % model)
    else:
        nlp = spacy.blank("el")  # create blank Language class
        print("Created blank 'el' model")
    # Add entity recognizer to model if it's not in the pipeline
    # nlp.create_pipe works for built-ins that are registered with
                                        spaCy
    if "ner" not in nlp.pipe_names:
        ner = nlp.create_pipe("ner")
```

```
        nlp.add_pipe(ner)
    # otherwise, get it, so we can add labels to it
    else:
        ner = nlp.get_pipe("ner")

    # add new entity label to entity recognizer
    ner.add_label("GR_ACT")      # custom-ner-1
    ner.add_label("GR_AGE")      # custom-ner-2
    ner.add_label("GR_DATE")     # custom-ner-3
    if model is None:
        optimizer = nlp.begin_training()
    else:
        optimizer = nlp.resume_training()

    # get names of other pipes to disable them during training
    pipe_exceptions = ["ner", "trf_wordpiecer", "trf_tok2vec"]
    other_pipes = [pipe for pipe in nlp.pipe_names if pipe not in
                                        pipe_exceptions]

    # only train NER
    with nlp.disable_pipes(*other_pipes), warnings.catch_warnings():
        # show warnings for misaligned entity spans once
        warnings.filterwarnings("once", category=UserWarning,
        module='spacy')

        sizes = compounding(1.0, 4.0, 1.001)
        # batch up the examples using spaCy's minibatch
        for itn in range(n_iter):
            random.shuffle(TRAIN_DATA)
            batches = minibatch(TRAIN_DATA, size=sizes)
            losses = {}
            for batch in batches:
                texts, annotations = zip(*batch)
                nlp.update(texts, annotations, sgd=optimizer,
                drop=0.35, losses=losses)
            print("Losses", losses)

    output_dir = 'custom_model/'

    # save model to output directory
    if output_dir is not None:
        output_dir = Path(output_dir)
        if not output_dir.exists():
            output_dir.mkdir()
        nlp.meta["name"] = new_model_name
        nlp.to_disk(output_dir)
```

Για την μέτρηση της ακρίβειας του μοντέλου, για την κάθε κατηγορία ξεχωριστά, χρησιμοποιήθηκε η κλάση nlp.evaluate() του SpaCy. Με τον ίδιο τρόπο που συλλέχθηκε το σύνολο δεδομένων εκμάθησης, μαζεύτηκε και ένα μικρό σύνολο δεδομένων, της ίδια μορφής, για τον έλεγχο της ακρίβειας του μοντέλου. Η κλάση του spaCy που πραγματοποιεί την αξιολόγηση του μοντέλου, παρέχει 3 ξεχωριστές μετρικές (precision, recall, f-score) και την τελική ακρίβεια του μοντέλου που διαμορφώνεται από τα αποτελέσματα της κάθε κλάσης διαιρεμένη με το σύνολο των κλάσεων.

Για κάθε μία από τις 3 κλάσεις (ηλικία, πράξη, ημερομηνία) έχει συλλεχθεί ένα ίσο

δείγμα από δεδομένα εισόδου προς δοκιμή. Συνολικά 50 δεδομένα εισόδου (ετικέτες οντοτήτων) για την κάθε κλάση.

Παρακάτω φαίνεται η ακρίβεια της κάθε κλάσης ξεχωριστά.

	Precision	**Recall**	**F-Score**
ΠΡΑΞΗ	0.72	0.73	0.77
ΗΛΙΚΙΑ	0.84	0.88	0.86
ΗΜΕΡΟΜΗΝΙΑ	0.59	0.62	0.6

Πίνακας 13. Αποτελέσματα ακρίβειας custom-trained NER

Η συνολική ακρίβεια του μοντέλου φαίνεται παρακάτω.

```
Precision:  0.72
Recall:  0.78
F-Score:  0.74
```

Το μοντέλο έχει ικανοποιητικά αποτελέσματα για την κάθε κλάση ξεχωριστά καθώς και συνολικά. Πρέπει να σημειωθεί ότι το σύνολο δεδομένων αποτελείται από περίπου 50 άρθρα και στο κάθε άρθρο παρατηρείται διαφορετικός αριθμός λέξεων που αντιστοιχούν σε μια κλάση. Δηλαδή, σε ένα άρθρο που ανήκει στο σύνολο εκπαίδευσης, η κλάση ”ηλικία” μπορεί να επαναληφθεί πολλές ή καμία φορές, άρα θα έχουμε ένα ανισόρροπο δείγμα εισόδου προς εκπαίδευση. Επίσης, η κάθε κλάση έχει διαφορετική δυσκολία εκπαίδευσης. Δηλαδή, η κλάση ”πράξη” μπορεί να αποτελείται από πολλές διαφορετικές λέξεις όπως και η κλάση ”ημερομηνία” από πολλές διαφορετικές μορφές ημερομηνίας αλλά η κλάση ”ηλικία” έχει σχεδόν την ίδια μορφή σε κάθε άρθρο και εμφανίζεται σε σχεδόν όλα τα δείγματα εισόδου και συνήθως πολλαπλές φορές, έτσι εξηγείται και ο λόγος που έχει τις καλύτερες μετρικές σε σχέση με τις άλλες 2 κλάσεις.

3.5.2 Εξαγωγή πληροφοριών με POS

Κάθε token σχολιάζεται με μια ετικέτα POS (μέρος της ομιλίας), η οποία είναι μια γραμματική ετικέτα όπως ουσιαστικό, προσδιοριστικό ή επίρρημα.

Το POS χρησιμοποιείται σε συνδιασμό με μια τακτική εξαγωγής των πιο σημαντικών προτάσεων του κειμένου που περιέχουν τα πιο σημαντικά ρήματα μέσω του elasticsearch, όπως αναφέρθηκε παραπάνω. Αφού συλλέξουμε τις πιο σημαντικές προτάσεις από το άρθρο, έχουμε στην διάθεση μας ένα μικρό σύνολο από προτάσεις που περιέχουν ρήμα (με βάση το TF-IDF από τον elasticsearch και την κατηγορία ”ΠΡΑΞΗ” από τον NER). Εφαρμόζεται σε αυτές POS matching ώστε να βρεθεί το φύλο του θύματος. Μπορούμε έτσι να

βρούμε αν μια πράξη γίνεται από κάποιον ή σε κάποιον. Επιπλέον, στις πιο σημαντικές προτάσεις που εξάγονται, εφαρμόζεται NLP matcher που εξάγει προτάσεις που είναι ακόμη πιο πιθανό να μας υποδείξουν το θύμα, με βάση τους παρακάτω γραμματικούς κανόνες.

Ο NLP matcher ορίζεται όπως φαίνεται παρακάτω και με βάση τους παρακάτω γραμματικούς κανόνες γίνεται το εξάγονται οι τελικές προτάσεις πάνω στις οποίες εφαρμόζεται ο κώδικας εξαγωγής φύλου.

```
victim_grammatical_patterns = [
    [
        {"POS": "VERB"}, {"DEP": "det", "OP": "?"},
        {"DEP": "nummod", "OP": "?"},
        {"DEP": "nmod", "OP": "?"},
        {"DEP": "obj"}
    ],
    [
        {"POS": "VERB"}, {"DEP": "det", "OP": "?"},
        {"DEP": "amod"}, {"DEP": "obj", "OP": "?"}
    ],
    [
        {"DEP": "amod"}, {"DEP": "nmod"}
    ],
    [
        {"DEP": "nsubj:pass"}, {"POS": "VERB"}
    ],
    [
        {"DEP": "det"}, {"DEP": "nmod"}, {"DEP": "nmod"}
    ],
]
```

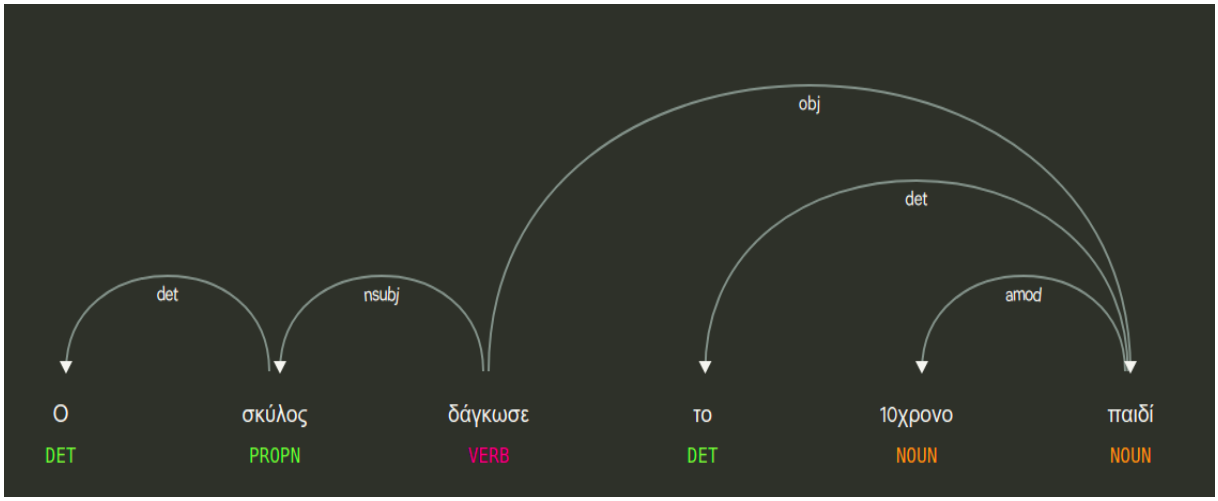
Για να βρεθεί το φύλο θα λάβουμε υπόψιν μας τα ρήματα από τον elasticsearch (important _sentences_from_elastic) καθώς και τα ρήματα που θα βρεθούν από το NER μοντέλο (extra _important_sentences) και ποιες λέξεις συνδέονται με αυτά καθώς περιμένουμε ότι το ρήμα θα μας οδηγήσει στο θύμα. Η κάθε σημαντική πρόταση αναλύεται ανεξάρτητα η μια από την άλλη.

Για την εύρεση των σημαντικών προτάσεων:

```
# all the verbs in dictionary from elastic
for v in verbs_dictionary:
    doc_v = nlp(v)
    for token in doc_v:
        if token.pos_ == "VERB":
            elastic_dict_verbs.append({
                'raw': token.text,
                'analyzed': get_specific_analyzed(token.text)[0][0]
            })

# all the matching from the above two
matched_verbs = [v2['raw'] for v1 in elastic_dict_verbs for v2 in
                                    article_verbs if
                 (v1['analyzed'] == v2['analyzed'])
```

```
                and v2['raw'] not in VERBS_TO_EXCLUDE]

# break article to sentences and find object and subject only where
                                    matched verb is located
tokens = nltk.sent_tokenize(data)

for verb in set(matched_verbs_from_elastic):
    for t in tokens:
        if verb in t:
            important_sentences.append(t)

if important_sentences == []:
    important_sentences.extend(extra_important_sentence)
    matched_verbs_from_elastic.extend(extra_important_sentences)
for sent in important_sentences:
    doc = nlp(sent)
    # POS Pattern matching based on important sentences and based on
                                        matching verbs
    matches = matcher(doc)
    for match_id, start, end in matches:
        string_id = nlp.vocab.strings[match_id]
        span = doc[start:end]
        victim_gender = dependency_collector(span.text, type="person")
        victim_genders.append(victim_gender)
```

Έστω η παρακάτω πρόταση ως μια απο τις ”σημαντικές” προτάσεις:



Εικόνα 29. POS Ανάλυση

Για να βρούμε το φύλο του θύματος, με βάση την ανάλυση που γίνεται στην πλατφόρμα ανάλυσης εγκλημάτων, αρχικά θα ακολουθήσουν τα παρακάτω βήματα:

	Πρόταση
Make lowercase	ο σκύλος δάγκωσε το 10χρονο παιδί
Remove accent	ο σκυλος δαγκωσε το 10χρονο παιδι
Remove stopwords	σκυλος δαγκωσε 10χρονο παιδι

Πίνακας 14. Επεξεργασία πρότασης

Επιπλέον, με βάση το NER μοντέλο μας, το ”δάγκωσε” δηλώνει πράξη ενώ το 10χρονο την ηλικία. Με βάση την ανάλυση και συλλογή που γίνεται από τον elasticsearch, η παρακάτω πρόταση δεν θα συμπεριληφθεί στις σημαντικές προτάσεις καθώς περιλαμβάνει τον όρο ”δάγκωσε” ως ρήμα ο οποίος δεν βρίσκεται στην λίστα με τα πιο συχνά επαναλαμβανόμενα ρήματα με βάση το threshold που έχει δοθεί. Όμως, λόγω του NER μοντέλου που έχει εκπαιδευτεί ώστε να αναγνωρίζει εγκληματικές πράξεις, ο όρος θα συμπεριληφθεί. Το NER και τα πιο σημαντικά ρήματα από τον elasticsearch με βάση την TF-IDF ανάλυση αλληλοκαλύπτονται για να δημιουργήσουν ένα ακόμη πιο πλούσιο λεξικό. Ο παρακάτω κώδικας θα εφαρμοστεί στο ρήμα ”δάγκωσε” για να βρεθεί το φύλο του θύματος.

```
for token in doc:
    check = re.findall("Gender=[^|]*", token.tag_)
    if not check == []:
        if token.dep == 'nmod' or
        token.pos_ == 'DET' or
        token.dep_ == 'iobj' or
        (token.pos_ =='NOUN' and token.dep_ == 'obj'):

            gender = check[0].split("=")[1]

            if gender == 'Masc':
                return "MALE"
            elif gender == 'Fem':
                return "FEMALE"
            elif gender == "Neut":
                return "CHILD"
```

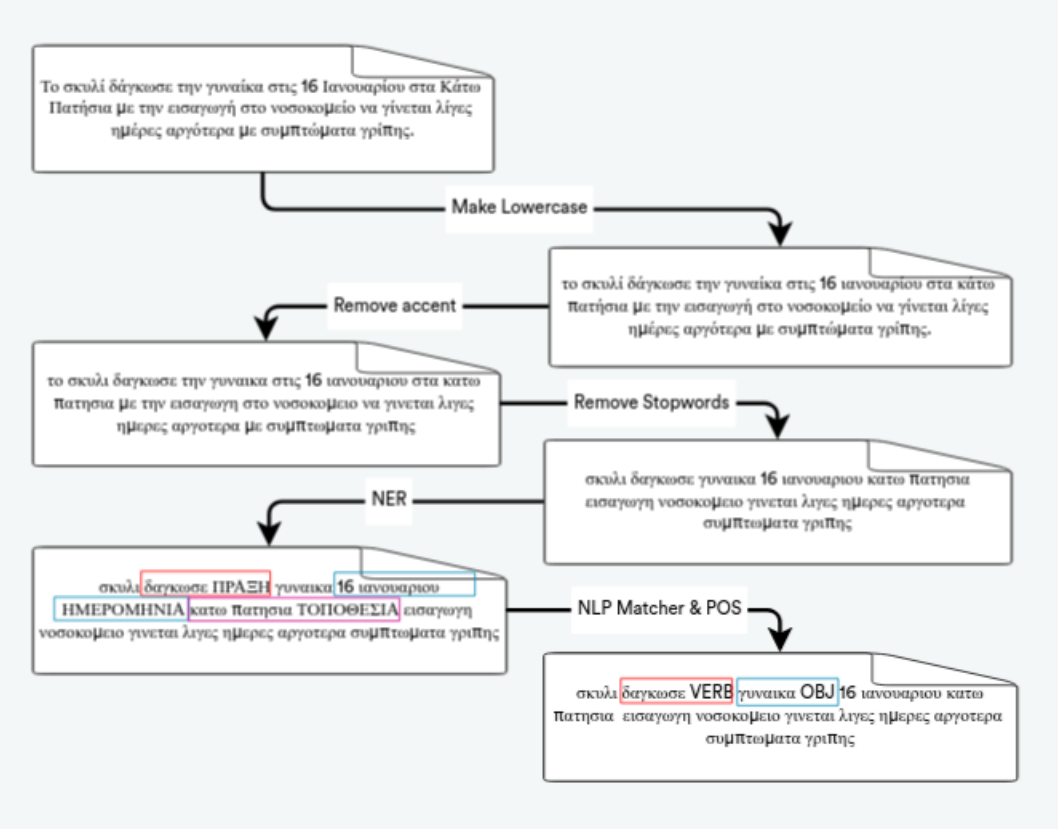
Το spaCy χρησιμοποιεί τους όρους κεφαλή και παιδί για να περιγράψει τις λέξεις που συνδέονται με ένα μόνο κομμάτι στο δέντρο εξάρτησης. Ο όρος dep περιγράφει τον τύπο της συντακτικής σχέσης που συνδέει το παιδί με την κεφαλή. Ακόμη πιο συνοπτικά, περιγράφει την συντακτική εξάρτηση, δηλαδή η σχέση μεταξύ των tokens.

Και το τελικό αποτέλεσμα του παραπάνω θα είναι ”ΠΑΙΔΙ” ως το φύλο του θύματος. Για να μετρηθεί η ακρίβεια της τεχνικής, εφαρμόστηκε σε 50 άρθρα με σωστά αποτελέσματα σε 36 απο αυτά (72%). Η ανάλυση φαίνεται παρακάτω.

Είσοδος	POS/DEP TAG
σκυλος	NOUN nsubj [’Gender=Masc’]
δαγκωσε	VERB ROOT []
10χρονο	NUM nummod []
παιδι	NOUN obj [’Gender=Neut’]

Πίνακας 15. Εξαγωγή θύματος

Με βάση όσα έχουν προαναφερθεί, έχουμε συνολικά το παρακάτω NLP pipeline ανάλυσης.

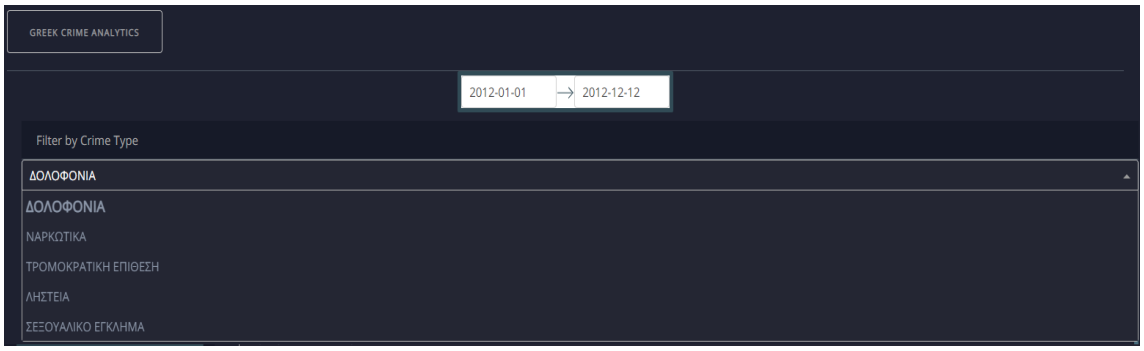


Εικόνα 30. NLP Pipeline

3.6 Οπτικοποίηση Δεδομένων

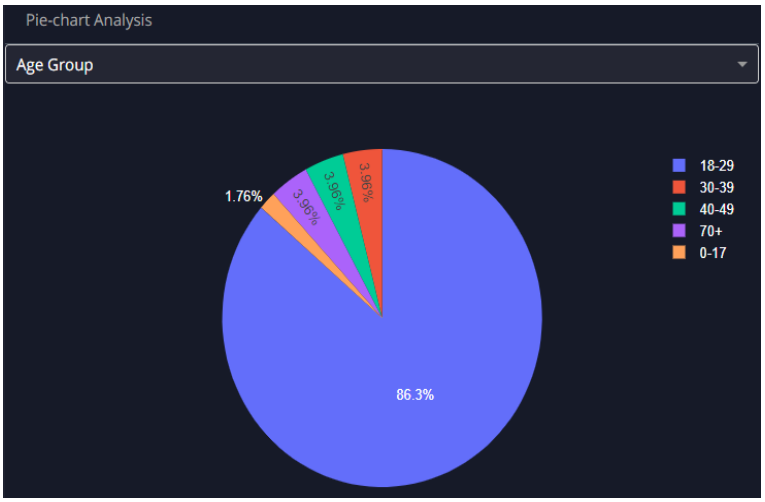
Η οπτικοποίηση δεδομένων επηρεάζει θετικά τη διαδικασία λήψης αποφάσεων ενός οργανισμού με διαδραστική οπτική αναπαράσταση δεδομένων. Οι Οργανισμοί μπορούν να αναγνωρίσουν τα μοτίβα πιο γρήγορα επειδή μπορούν να ερμηνεύσουν δεδομένα σε γραφικές ή εικονογραφικές μορφές. Χρειαζόμαστε οπτικοποίηση δεδομένων επειδή ο ανθρώπινος εγκέφαλος δεν είναι καλά εξοπλισμένος για να καταλάβει γρήγορα τόσες πρώτες, ανοργάνωτες πληροφορίες και να τις μετατρέψει σε κάτι χρήσιμο και κατανοητό. Χρειαζόμαστε γραφήματα και γραφήματα για να κοινοποιήσουμε τα ευρήματα δεδομένων, ώστε να μπορούμε να εντοπίσουμε μοτίβα και τάσεις για να αποκτήσουμε εικόνα και να πάρουμε καλύτερες αποφάσεις γρηγορότερα.

Έστω ότι επιλέγουμε χρονική περίοδο και κατηγορία όπως φαίνεται στην εικόνα παρακάτω.

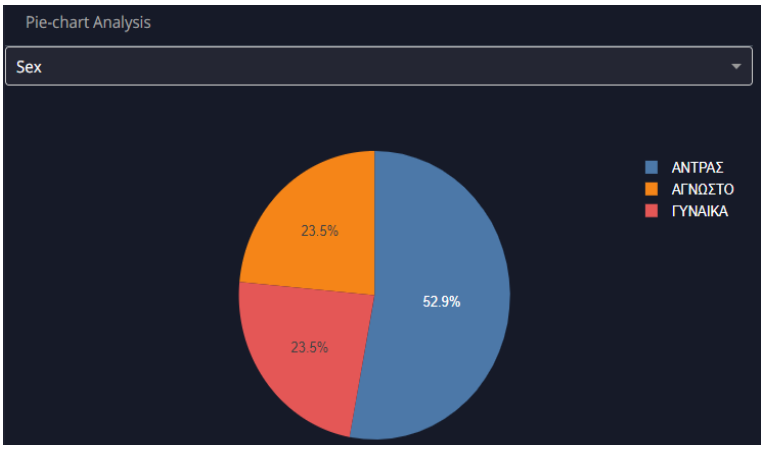


Εικόνα 31. Μενού Επιλογής κατηγορίας και χρονικής περιόδου

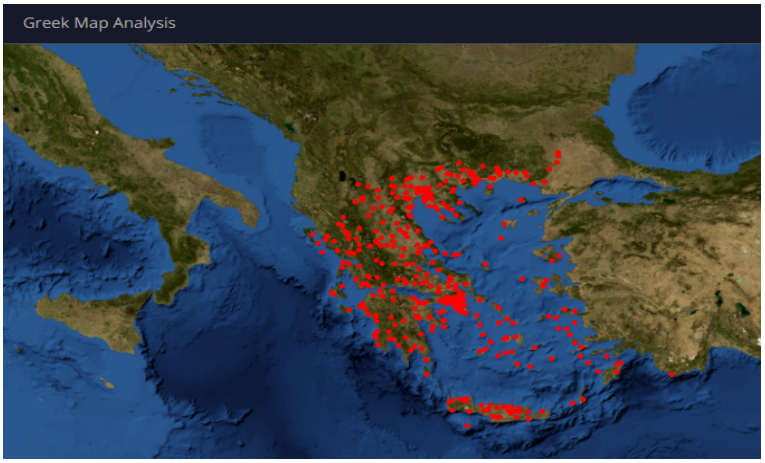
Τώρα μπορούμε να δούμε κάποιες αναλύσεις που εξάγονται με βάση τις πληροφορίες που παράγουν τα NLP μοντέλα. Όπως αναφέρθηκε στην ενότητα 3.5, από το αρχικό κείμενο εξάγουμε κάποιες πληροφορίες (ηλικίες των εμπλεκομένων, μέρος του εγκλήματος, ενέργειες του δράστη, φύλο του θύματος, κατάσταση του δράστη) τις οποίος θα οπτικοποιήσουμε.



Εικόνα 32. Pie-chart ανάλυση των ηλικιακών ομάδων

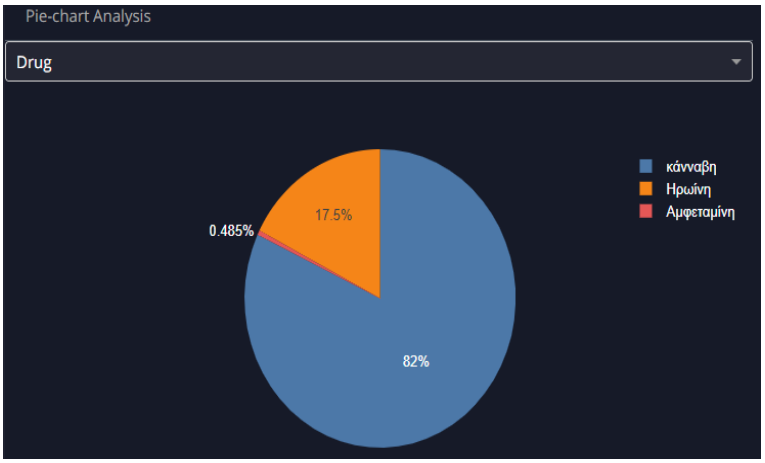


Εικόνα 33. Pie-chart ανάλυση του φύλου του θύματος



Εικόνα 34. Όλα τα εγκλήματα στον χάρτη της Ελλάδας

Κάθε διαφορετική κατηγορία εγκλήματος ενδέχεται να έχει διαφορετική οπτικοποίηση δεδομένων, για παράδειγμα:



Εικόνα 35. Pie-chart ανάλυση των ναρκωτικών που αναφέρονται σε κάθε άρθρο

3.7 Υλοποίηση Μικροϋπηρεσιών και Εικονοποίησης βασισμένη σε Περιέκτες

Υλοποιήθηκε ένα κατανεμημένο πρόγραμμα λήψης άρθρων χρησιμοποιώντας docker. Η εφαρμογή χωρίστηκε σε μικροϋπηρεσίες που συνολικά λειτουργούν σε συνεργασία στο ίδιο docker δίκτυο. Οι μικροϋπηρεσίες που αποτελούν την εφαρμογή είναι mongodb, redis, elasticsearch, app όπου η τελευταία μικροϋπηρεσία αποτελεί την εφαρμογή, δηλαδή, το scrapy component, την ταξινόμηση με SVM και την ανάλυση με NLP και elasticsearch.

Το κάθε component απο τα παραπάνω θέλει μερικές παραμετροποιήσεις για να μπορεί να επικοινωνεί με τα υπολοίπα. Το Redis έχει ενσωματωθεί ως εξής στο settings.py αρχείο:

```
# Redis
DUPEFILTER_CLASS = "scrapy_redis.dupefilter.RFPDupeFilter"
SCHEDULER = "scrapy_redis.scheduler.Scheduler"
SCHEDULER_PERSIST = True
REDIS_ENCODING = 'utf-8'
REDIS_HOST = 'localhost'
REDIS_PARAMS = {'password': "toor"}
REDIS_PORT = 6379
```

Και η ακόλουθη αλλαγή πραγματοποιείται στον spider.

```
class NewsbombSpider(RedisCrawlSpider):
    name = 'newsbomb'
    redis_key = 'redisSpider'
```

Για την μικροϋπηρεσία ”app”, η οποία αποτελεί το κύριο μέρος της εφαρμογής θα έχουμε ένα dockerfile μέσα στο οποίο θα δηλώσουμε όλες τις βιβλιοθήκες που χρησιμοποιεί η εφαρμογή καθώς και το σημείο έναρξης της.

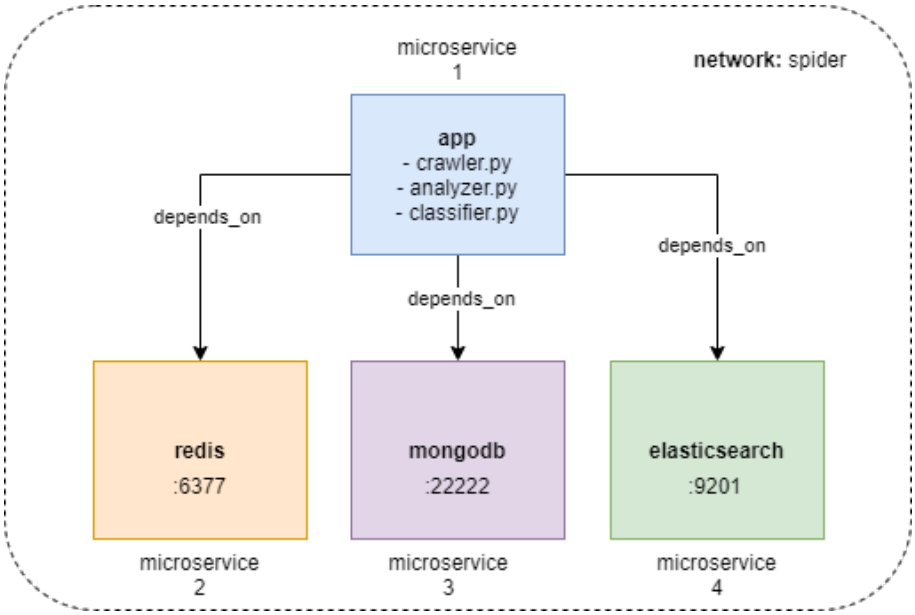
```
FROM python:3.8
MAINTAINER simonamnv
ADD . /root/
RUN pip3 install scrapy
RUN pip3 install pymongo
RUN pip3 install scrapy_redis
RUN pip3 install django
RUN pip3 install flask
RUN pip3 install django-elasticsearch-dsl
RUN pip3 install django-elasticsearch-dsl-drf
RUN pip3 install djongo
RUN pip3 install nltk
RUN pip3 install spacy
RUN pip3 install pandas
RUN pip3 install sklearn
RUN python3 -m spacy download el_core_news_lg
WORKDIR /root/crawling/crawling
CMD ["scrapy", "crawl", "newsbomb"]
```

Η σύνθεση docker είναι απαραίτητη για την ανάπτυξη πολλαπλών container ενός μηχανήματος. Το Docker compose είναι ένα εργαλείο για τη δημιουργία πολλαπλών container σε έναν διακομιστή ή κεντρικό υπολογιστή. Θέλουμε να εισάγουμε το ”app” σε ένα δίκτυο μαζί με τις υπόλοιπες μικροϋπηρεσίες. Παρακάτω φαίνεται το τελικό docker-compose.yml.

```
version: "3.5"
services:
        app:
                build: .
                volumes:
                        - code:/root/crawling
                environment:
                  ELASTICSEARCH_URL: "http://elastic:9201"
                  REDIS_URL: "redis://redis:6377"
                depends_on:
                        - "mongodb"
                        - "redis"
                        - "elastic"
                deploy:
                        mode: global
                networks:
                        - "spider"
        mongodb:
                image: "mongo"
                volumes:
                        - mongodb:/data/db
                ports:
                        - "22222:27017"
                networks:
                        - "spider"
                environment:
                        MONGO_INITDB_ROOT_USERNAME: root
                        MONGO_INITDB_ROOT_PASSWORD: toor
                command: --wiredTigerCacheSizeGB 0.3
                deploy:
                        replicas: 1
                        placement:
                                constraints: [ node.role == manager ]
        redis:
                image: "redis"
                ports:
                        - "6377:6379"
                networks:
                        - "spider"
                command: redis-server --requirepass toor
                deploy:
                        replicas: 1
                        placement:
                        constraints: [node.role == manager]
        elastic:
                image: "elasticsearch:7.12.0"
                ports:
                      - "9201:9200"
                      - "9301:9300"
                volumes:
                    - ./Elasticsearch/data:/etc/elasticsearch/data
                    - ./Elasticsearch/config/elasticsearch.yml:
                    /etc/elasticsearch/config/elasticsearch.yml
                tty : true
                environment:
                        - cluster.name=docker-elastic-cluster
```

```
                        - bootstrap.memory_lock=true
                        - http.cors.enabled=true
                        - http.cors.allow-origin=*
                        - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
                        - node.name=node-1
                        - cluster.initial_master_nodes=node-1
                ulimits:
                        memlock:
                                soft: -1
                                hard: -1
                command: elasticsearch
volumes:
        code:
        mongodb:
        elastic:
networks:
        spider:
```

Η αρχιτεκτονική της εφαρμογής φαίνεται στο παρακάτω σχήμα.



Εικόνα 36. Αρχιτεκτονική της πλατφόρμας ανάλυσης δεδομένων

Συνολικά έχουμε 4 μικροϋπηρεσίες όπου η πρώτη αποτελείται από το dockerfile που περιέχει την εφαρμογή ανάλυσης εγκλημάτων, ”app”, το microservice ”mongodb” ώς το κύριο στοιχείο αποθήκευσης δεδομένων, το ”elasticsearch” για την αποθήκευση και ανάλυση των δεδομένων και το ”redis” για να πραγματοποιηθεί πολλαπλή κατανομή του crawler.py. Το κάθε microservice έχει το δικό του εσωτερικό port για να επικοινωνεί στο δίκτυο με τα υπόλοιπα και όλα μαζί βρίσκονται σε ένα κοινό δίκτυο ”spider”. Το ”app” εξαρτάται από τα υπόλοιπα τρία που σημαίνει ότι αν υπάρχει κάποια βλάβη κατά την έναρξη των υπόλοιπων εφαρμογών, τότε δεν θα υπάρχει νόημα να ξεκινήσει και το κύριο microservice καθώς αντλεί δεδομένα από την mongodb και χρειάζεται το component ”elasticsearch” για την ανάλυση δεδομένων και τέλος, το αρχείο ”crawler.py” χρειάζεται το ”redis” για την κατανεμημένη λήψη των δεδομένων καθώς συντονίζει την ουρά με τις

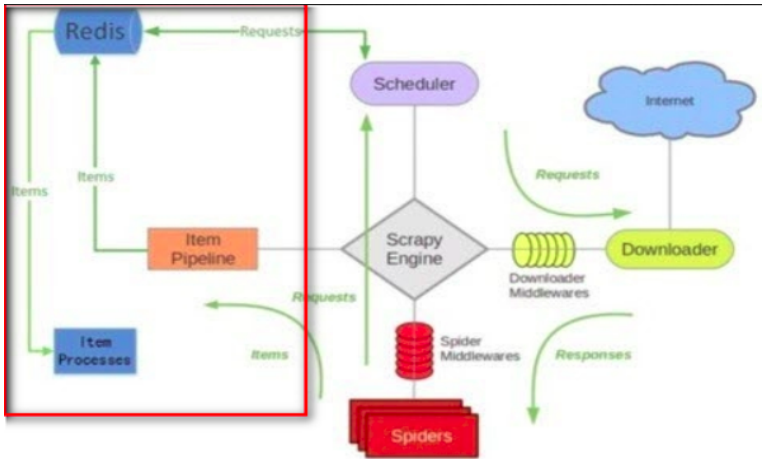
ιστοσελίδες προς λήψη και βοηθάει στο να αποφύγουμε την λήψη της ίδιας ιστοσελίδας πολλαπλές φορές.

Έχοντας την mongodb ως μια διαφορετική υπηρεσία στο docker δίκτυο μας προσφέρει να έχουμε πολλαπλά αντίγραφα της βάσης δεδομένων και να μην χάνουμε τα δεδομένα σε περίπτωση βλάβης ή μη παροχής υπηρεσιών. θέλουμε να κρατήσουμε αντίγραφα είναι για να διατηρηθούν τα δεδομένα ασφαλή, για υψηλή διαθεσιμότητα δεδομένων, και για αποκατάσταση από καταστροφές.

Έχοντας τον elasticsearch ως ξεχωριστό component, διατηρούμε σε αυτόν ένα δικό του αντίγραφο δεδομένων σε περίπτωση που δεν υπάρχει διαθεσιμότητα απο την mongodb. Ο elasticsearch επικοινωνεί άμεσα με την μικροϋπηρεσια ”mongodb” ανά τακτά χρονικά διαστήματα και ανανεώνει τα δεδομένα του σε περίπτωση που έχουν προσθετεί παραπάνω γραμμές την mongodb. Ο elasticsearch προσφέρει πολλαπλές υπηρεσίες, όπως τους αναλυτές κειμένου που έχουν οριστεί για την συγκεκριμένη ανάλυση καθώς και τον τρόπο που κάνει index τα άρθρα Έχοντας όλα αυτά σε μια διαφορετική μικρουπηρεσία ξεχωριστά από την εφαρμογή εξασφαλίζουμε υψηλή διαθεσιμότητα δεδομένων, ακόμη και σε περίπτωση που δεν υπάρχει επικοινωνία με την βάση δεδομένων. Πέρα από την διαθεσιμότητα των δεδομένων, εξασφαλίζουμε ευκολία στην αλλαγή των αναλυτών και του τρόπου επεξεργασία και indexing των άρθρων που υπάρχουν σε αυτόν. Με ανεξάρτητα API calls γίνονται άμεσες τροποποιήσεις στην ανάλυση και ανάγνωση των δεδομένων χωρίς να υπάρχουν εξαρτήσεις από τα άλλα microservices.

Όταν ξεκινήσει η υπηρεσία, θα δημιουργηθεί αυτόματα ένα δίκτυο bridge. Επομένως, η διεύθυνση IP που συνδέει το redis και το mongo στον πηγαίο κώδικα ανίχνευσης μπορεί να γράψει απευθείας το όνομα της υπηρεσίας. Το όνομα υπηρεσίας επιλύει την ip του container και όχι το ip του κεντρικού υπολογιστή. Αυτό που επιτυγχάνεται με αυτόν τον τρόπο είναι να γίνεται η λήψη άρθρων με πολλαπλούς spiders την ίδια στιγμή, χρησιμοποιώντας το redis για να ελέγχει την ροή μηνυμάτων ώστε να μην υπάρχει διπλοεγγραφή. Το τελικό αποτέλεσμα είναι να έχουμε παράλληλη λήψη πολλαπλών άρθρων από διάφορες πηγές.

Για την έναρξη της κατανεμημένης λήψης, πρέπει να ανοίξουμε το redis CLI και να εισάγουμε <lpush redis_key_value> <URL>. Παράλληλα, μπορεί να θέσουμε πάνω απο έναν με τον ίδιο ακριβώς τρόπο.



Εικόνα 37. Η υλοποίηση του κατανεμημένου crawler

Κεφάλαιο 4

Συμπεράσματα και Επεκτάσεις της Εφαρμογής

Η παρούσα εργασία έδειξε ότι ένα σύστημα ανάλυσης εγκλημάτων από άρθρα μπορεί να φανεί χρήσιμο για να εξάγει με καλή ακρίβεια σημαντικές πληροφορίες καθώς και μέσω της οπτικοποιήσης της πληροφορίας να επισημάνει κρυφά μοτίβα στα δεδομένα. Σε αυτό το κεφάλαιο πραγματοποιείται η συνολική ανάλυση για να προσδιοριστεί σε πιο γενικό επίπεδο το σύνολο των αποτελεσμάτων ούτως ώστε να κριθεί η βέλτιστη χρήση τους καθώς και οι επεκτάσεις ή οι βελτιώσεις που μπορούν να εφαρμοστούν.

Στην προηγούμενη ενότητα γίνεται η καταγραφή των αποτελεσμάτων για κάθε εφαρμογή που αναπτύχθηκε και εξετάστηκε η επίδοση και της κατηγοριοποίησης καθώς και του NER μοντέλου καθώς και NLP τεχνικών για να εξαχθούν οι επιθυμητές πληροφορίες. Για λόγους πληρότητας από τη μεριά της αποτίμησης των μετρήσεων, κρίνεται αναγκαία η σύνθεση αυτών των αποτελεσμάτων με τέτοιο τρόπο ώστε για κάθε υλοποίηση να γνωστοποιηθεί η επίδοση της σε συνάρτηση με το σύνολο των κριτηρίων αξιολογώντας την εφαρμογή τους σε υποθετικά σενάρια χρήσης.

Όσο αφορά την κατηγοριοποίηση των άρθρων, στην παρούσα φάση συλλέγεται ενα σύνολο δεδομένων με 10.000 άρθρα ως train set. Η ακρίβεια μπορεί να θεωρηθεί βέλτιστη καθώς με την χρήση του SVM, αγγίζει τα 97.77% και άρα δεν χρειάζεται να αλλάξει το μοντέλο κατηγοριοποίησης και ούτε να προστεθούν επιπλέον δεδομένα στο train set, για τις κατηγορίες που χρησιμοποιούνται στην εργασία. Μια επέκταση της συγκεκριμένης λειτουργίας είναι να αυξηθούν οι κατηγορίες των εγκλημάτων.

Όσο αφορά την λήψη και συλλογή των άρθρων, λόγω της δυνατότητας κατανεμημένης λήψης των άρθρων, επιτυγχάνεται η βέλτιστη ταχύτητα αφού μπορούμε να θέσουμε πολλαπλές πηγές παράλληλα. Η λήψη μπορεί να γίνει χωρίς την ιδιότητα της κατανομής για μια πηγή μόνο με ρυθμούς λήψης 1 άρθρο/δευτερόλεπτο ή μπορεί να γίνει παράλληλα για πολλές πηγές ή σελίδες εκκίνησης χρησιμοποιώντας το docker-compose.yml οπού το redis αναλαμβάνει την ουρά μηνυμάτων ώστε να μην υπάρχουν διπλοεγγραφές. Στην παρούσα εργασία χρησιμοποιούνται το πολύ 2 πηγές δεδομένων, μια πιθανή επέκταση θα ήταν να προστεθούν και άλλες πηγές δεδομένων.

Όσο αφορά την ανάλυση που πραγματοποιείται μέσω του elasticsearch, αυτό μπορεί να χωριστεί σε δυο διαφορετικές εργασίες. Η πρώτη εργασία αποτελείται από τους custom analyzers οι οποίοι είναι υπεύθυνοι για την προεπεξεργασία του κειμένου και αποτελού-

νται από όλα τα απαραίτητα εργαλεία που χρειάζονται για να απλοποιήσουν το κείμενο και άρα δεν χρειάζονται περαιτέρω προσθήκες. Η δεύτερη εργασία αφορά την λειτουργία TF-IDF που χρησιμοποιείται όταν κάνουμε εύρεση της πράξης του δράστη και όταν εξάγουμε τις πιο σημαντικές προτάσεις από το συνολικό άρθρο. Ο δείκτης αυτός δείχνει λέξεις που επαναλαμβάνονται πολλές φορές για μια ορισμένη κατηγορία άρθρου. Άρα όσα παραπάνω άρθρα έχουμε, τόσο πιο πλούσιο και το λεξικό που δημιουργείται. Έχει λοιπόν νόημα το να έχουμε όσα παραπάνω άρθρα γίνεται για κάθε κατηγορία. Ο αριθμός των άρθρων μπορεί να θεωρηθεί μη βέλτιστος και αρκετά μικρός καθώς υπάρχουν συνολικά 8808 άρθρα εκ των οποίων 7349 αφορούν την Ελλάδα και πιο συγκεκριμένα, 1366 αναφέρονται σε δολοφονίες, 1717 σε υποθέσεις ναρκωτικών 3075 σε κλοπές/ληστείες, 161 σε τρομοκρατικές υποθέσεις και 1030 σε σεξουαλικά εγκλήματα. Με την αύξηση των άρθρων για κάθε κατηγορία, θα βελτιστοποιηθεί και το λεξικό των σημαντικών λέξεων.

Όσο αφορά το NER μοντέλο, στην παρούσα εργασία χρησιμοποιήθηκαν 50 άρθρα συνολικά για την εκμάθηση του. Συνεπώς, μπορεί να θεωρηθούν πολύ λίγα, με αποτέλεσμα έχουμε ακόμη καλύτερα ποσοστά ακρίβειας με ένα μεγαλύτερο δείγμα δεδομένων εκμάθησης. Επιπλέον, δεν υπήρχε η ίδια συχνότητα των κατηγοριών σε κάθε άρθρο εκμάθησης. Δηλαδή, η κατηγορία ”ΗΛΙΚΙΑ” θα εμφανιστεί σε κάθε άρθρο πολλαπλές φορές σε αντίθεση με την κατηγορία ”ΗΜΕΡΟΜΗΝΙΑ”. Άρα θα πρέπει να λάβουμε υπόψιν μας ότι όταν γίνει ξανά η εκμάθηση του μοντέλου, τότε το train set θα πρέπει να έχει τον ίδιο αριθμό δεδομένων για κάθε κατηγορία και να μην μετράμε τα δεδομένα εκμάθησης μόνο με βάση τον αριθμό άρθρων.

Όσο αφορά τις NLP τεχνικές που χρησιμοποιούνται σε συνδυασμό με άλλες, όπως για παράδειγμα την εύρεση του φύλου του θύματος ή την κατάσταση του δράστη η επίδοση τους μπορεί να θεωρηθεί ικανοποιητική καθώς αγγίζουν ποσοστά ακρίβειας γύρω στα $65 - 70\%$. Οι τεχνικές αυτές υλοποιήθηκαν για την παρούσα εργασία και θα πρέπει να σημειωθεί ότι εκτελούνται σε κείμενο που είναι στα Ελληνικά και δεν υπάρχουν παρόμοιες υλοποιήσεις. Η επέκταση και βελτιστοποίηση τους είναι δυνατή με παραπάνω έρευνα.

Όσο αφορά την οπτικοποίηση, ως πιθανή επέκταση είναι η αύξηση των σχημάτων για να καλύπτουν κάθε πιθανή πληροφορία που εξάγεται.

4.1 Συμπεράσματα και Επεκτάσεις της διανομής

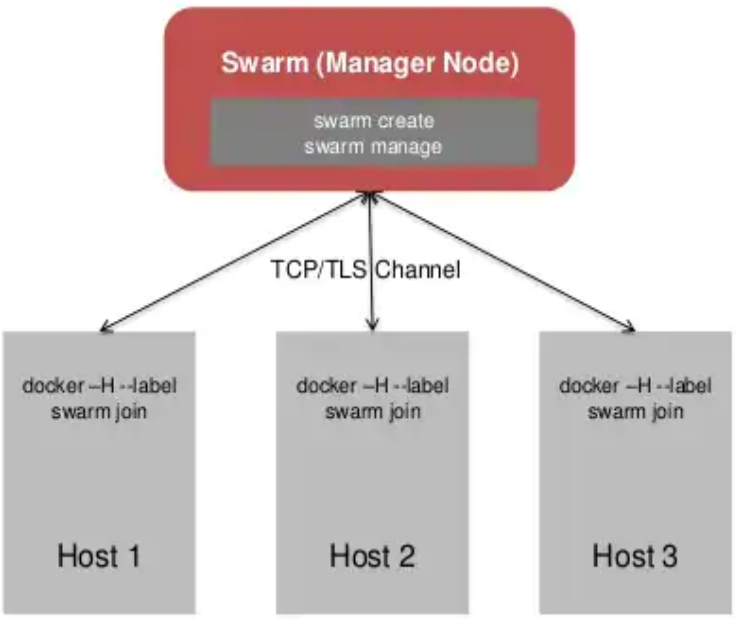
Το Docker Compose είναι ιδανικό για τη λειτουργία container σε ένα μόνο μηχάνημα, αλλά αυτό δεν λειτουργεί σε περιβάλλον παραγωγής εάν το μηχάνημα παραμείνει εκτός σύνδεσης, χάνονται όλες οι εφαρμογές. Η εκκίνηση του docker swarm είναι κάτι που γίνεται μία φορά και στη συνέχεια μπορούμε να ενώσουμε οποιονδήποτε αριθμό μηχανών - το Docker καλεί μηχανές στους κόμβους Swarm. Για να ενώσουμε έναν κόμβο στο Swarm, πρέπει να βρίσκεται στο ίδιο δίκτυο και χρειάζεται το διακριτικό σύνδεσης από τον διαχειριστή, το οποίο λειτουργεί σαν κωδικός πρόσβασης για να προστατεύσει το Swarm από ”ξένους” κόμβους.

Το σμήνος ενός κόμβου λειτουργεί με τον ίδιο ακριβώς τρόπο όπως το σμήνος πολλαπλών κόμβων, εκτός από το ότι δεν υπάρχει υψηλή διαθεσιμότητα από την ύπαρξη εφεδρικών μηχανών ή την επιλογή κλιμάκωσης container για χρήση της χωρητικότητας πολλών μηχανών. Στην παρούσα εργασία, έχουμε την ύπαρξη ενός μόνο μηχανήματος,

άρα πολλαπλά containers σε ένα δίκτυο που λειτουργεί σε ένα μηχάνημα.

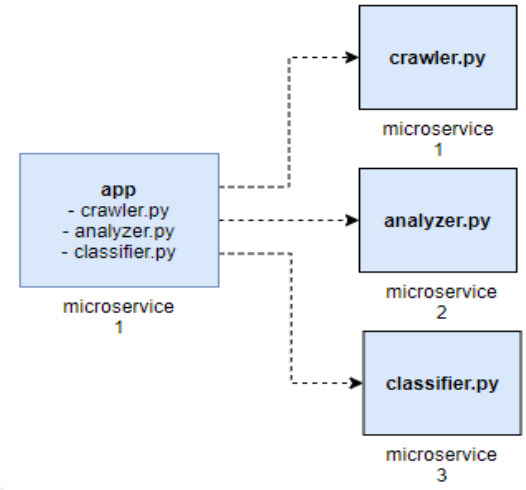
Άρα μια επέκταση όσο αφορά την διανομή των containers είναι η δημιουργία ενός σμήνους με πάνω απο έναν κόμβο εγκαθιστώντας το Docker σε κάθε διακομιστή, εκτελώντας μια φορά το docker swarm init και ενώνοντας το docker swarm για όλους τους άλλους κόμβους. Πιο συγκεκριμένα, θα επιλεγεί ένας διακομιστής ως κόμβος διαχειριστή του swarm. Μετά την προετοιμασία, ο διακομιστής θα συμμετάσχει αυτόματα στο swarm ως κόμβος διαχειριστή και οι υπόλοιποι θα είναι worker nodes. Αφού δημιουργηθεί το σύμπλεγμα, μπορούν να δημιουργηθούν υπηρεσίες με εντολή, η οποία είναι παρόμοια με την εντολή εκτέλεσης. Συνοπτικά, τα container ”mongodb” και ”redis” αναπτύσσονται σε έναν κόμβο διαχειριστή και κάθε κόμβος αναπτύσσει ένα container ”app” που έχει τον crawler με τον οποίον γίνεται η επικοινωνία και πάνω στον οποίον γίνεται η κατανομή.

Ένα από τα βασικά οφέλη που συνδέονται με τη λειτουργία του docker swarm είναι το υψηλό επίπεδο διαθεσιμότητας που προσφέρει καθώς και αλλαγή στην απόδοση καθώς προσφέρει αυτόματη εξισορρόπηση φόρτου. Ο διαχειριστής διασφαλίζει ότι ο φόρτος εργασίας του container ανατίθεται να λειτουργεί στον πιο κατάλληλο κεντρικό υπολογιστή για βέλτιστη απόδοση και αυτό θα φέρει τα βέλτιστα αποτελέσματα σε περίπτωση που έχουμε πολλαπλές πηγές λήψης δεδομένων στον κατανεμημένο crawler.



Εικόνα 38. Η γενική αρχιτεκτονική του docker swarm

Όσο αφορά τις μικροϋπηρεσίες, ως πιθανή επέκταση είναι η περαιτέρω διάσπαση της κύριας εφαρμογή σε microservices. Η κύρια εφαρμογή ”app” περιέχει τον crawler, analyzer καθώς και τον classifier. Το καθένα από αυτά θα μπορούσε να εμπεριέχεται σε ένα ξεχωριστό container και να επικοινωνούν μεταξύ τους μέσω API calls. Αυτό θα εξασφάλιζε λιγότερες εξαρτήσεις μεταξύ των εφαρμογών και ευκολότερη επίλυση προβλημάτων καθώς θα ξέραμε σε ποιό από τα 3 επίπεδα υπάρχει κάποιο πρόβλημα. Επίσης, είναι ευκολότερη και η ανάπτυξη και κλιμάκωση της εφαρμογής.



Εικόνα 39. Επέκταση των μικροϋπηρεσιών

Βιβλιογραφία

[1] MohammadReza Keyvanpour, Mostafa Javideh, Mohammad Reza Ebrahimi, “Detecting and investigating crime by means of data mining: a general crime matching framework” WCIT 2010.

[2] Zhuo Chen, Lan Jiang Zhou, Xuan Da Li, Jia Nan Zhang, Wen Jie Huo, “The Lao Text Classification Method Based on KNN”, 3rd International Conference on Mechatronics and Intelligent Robotics (ICMIR-2019).

[3] Fernando Rodrıguez-Haroa, Felix Freitagb, Leandro Navarrob, Efraın Hernandez, Nicandro Farıas-Mendoza, Juan Antonio Guerrero-Ibanez, Apolinar Gonzalez-Potes, “A summary of virtualization techniques”, The 2012 Iberoamerican Conference on Electronics Engineering and Computer Science.

[4] DURGESH K. SRIVASTAVA, LEKHA BHAMBHU, “DATA CLASSIFICATION USING SUPPORT VECTOR MACHINE”, 2010

[5] Said A. Salloum, Mostafa Al-Emran, Azza Abdel Monem, Khaled Shaalan, “Using Text Mining Techniques for Extracting Information from Research Articles”, Chapter in Studies in Computational Intelligence, January 2018

[6] Dr. S. Vijayarani, Ms. J. Ilamathi, Ms. Nithya, M. Phil Research Scholar, “Preprocessing Techniques for Text Mining - An Overview”, Chapter in Studies in Computational Intelligence, 2015

[7] Sebastian Raschka, “Naive Bayes and Text Classification I”, October 4 2014

[8] Aurangzeb Khan, Baharum Baharudin, Lam Hong Lee, Khairullah khan, “A Review of Machine Learning Algorithms for Text-Documents Classification”, 2010

[9] Osvaldo Simeone, “A Very Brief Introduction to Machine Learning With Applications to Communication Systems”, arXiv:1808.02342v4 [cs.IT] 5 Nov 2018

[10] Omar Al-Debagy, Peter Martinek, “A Comparative Review of Microservices and Monolithic Architectures”, 18th IEEE International Symposium on Computational Intelligence and Informatics, 2018

[11] Margherita Grandini, Enrico Bagli, Giorgio Visani, “METRICS FOR MULTI-CLASS CLASSIFICATION: AN OVERVIEW”, arXiv:2008.05756v1 [stat.ML] 13 Aug 2020

[12] Yangguang Liu, Yangming Zhou, Shiting Wen, Chaogang Tang, “A Strategy on Selecting Performance Metrics for Classifier Evaluation”, International Journal of Mobile Computing and Multimedia Communications, October 2014

[13] Lawrence McClendon and Natarajan Meghanathan, “USING MACHINE LEARNING ALGORITHMS TO ANALYZE CRIME DATA”, International Journal of Mobile Computing and Multimedia Communications, Machine Learning and Applications: An International Journal (MLAIJ) Vol.2, No.1, March 2015

[14] Darshita Kalyani, Dr. Devarshi Mehta, “Paper on Searching and Indexing Using Elasticsearch”, International Journal of Mobile Computing and Multimedia Communications, International Journal Of Engineering And Computer Science ISSN:2319-7242, Volume 6 Issue 6 June 2017, Page No. 21824-21829

[15] Babak Bashari Rad, Harrison John Bhatti, Mohammad Ahmadi, “An Introduction to Docker and Analysis of its Performance”, IJCSNS International Journal of Computer Science and Network Security, VOL.17 No.3, March 2017 No. 21824-21829

[16] Ginger Saltos, Mihaela Cocea, “AN EXPLORATION OF CRIME PREDICTION USING DATA MINING ON OPEN DATA”, 7 May 2015

[17] Duan Li-guo, Di peng, Li Ai-ping, “A New Naive Bayes Text Classification Algorithm”, TELKOMNIKA Indonesian Journal of Electrical Engineering, Vol.12, No.2, February 2014, pp. 947 952

[18] Hossin, M. and Sulaiman, M.N., “A REVIEW ON EVALUATION METRICS FOR DATA CLASSIFICATION EVALUATIONS”, International Journal of Data Mining Knowledge Management Process (IJDKP) Vol.5, No.2, March 2015

[19] Park R. Young, Srinarayan Sharma, “VIRTUALIZATION: A REVIEW AND FUTURE DIRECTIONS”, May 2011

[20] Divya Khyani, Siddhartha B S, Niveditha N M, Divya B M, “An Interpretation of Lemmatization and Stemming in Natural Language Processing”, Shanghai Ligong Daxue Xuebao/Journal of University of Shanghai for Science and Technology, January 2021

[21] Matthew L. Williams, Pete Burnap and Luke Sloan, “CRIME SENSING WITH BIG DATA: THE AFFORDANCES AND LIMITATIONS OF USING OPEN-SOURCE COMMUNICATIONS TO ESTIMATE CRIME PATTERNS”, BRIT. J. CRIMINOL. (2017) 57, 320–340, Advance Access publication 31 March 2016

[22] Bo Zhao, “Web Scraping”, May 2017

[23] Yong Wang, Julia Hodges, Bo Tang, “Classification of Web Documents Using a Naive Bayes Method”, Conference Paper, December 2003

[24] Nahid Jabeen and Parul Agarwal, “Data Mining in Crime Analysis”, D. Goyal et al. (eds.), Proceedings of Second International Conference on Smart Energy and Communication, Algorithms for Intelligent Systems, January 2021

[25] ELTON STONEMAN, “Learn Docker in a Month of Lunches”, ISBN: 9781617297052, 2020 by Manning Publications

[26] RADU GHEORGHE, MATTHEW LEE HINMAN, ROY RUSSO, “Elasticsearch in Action”, ISBN: 9781617291623, 2016 by Manning Publications

[27] Vishaal Jatav, Ravi Teja, Srini Bharadwaj, “Improving Part-of-Speech Tagging for NLP Pipelines”, 2017

[28] Hema Krishnan, M.Sudheep Elayidom, T.Santhanakrishnan, “MongoDB – a comparison with NoSQL databases”, International Journal of Scientific Engineering Research, Volume 7, Issue 5, May-2016, ISSN 2229-5518

[29] Prakash M Nadkarni, Lucila Ohno-Machado, Wendy W Chapman, “Natural language processing: an introduction”, Journal of the American Medical Informatics Association, September 2011

[30] Suhong Kim, Param Joshi, Parminder Singh Kalsi, and Pooya Taheri, “Crime Analysis Through Machine Learning”, November 2018

[31] Isuru Jayaweera, Chamath Sajeewa, Sampath Liyanage, Tharindu Wijewardane, Indika Perera, Adeesha Wijayasiri, “Crime Analytics: Analysis of Crimes Through Newspaper Articles”, April 2015

[32] Neel Shah, Darryl Willick, Vijay Mago, “A framework for social media data analytics using Elasticsearch and Kibana”, Springer Science+Business Media, LLC, part of Springer Nature 2018

[33] Xavier Schmitt, Sylvain Kublert, Jer´ emy Robert, Mike Papadakis, Yves LeTraon, “A Replicable Comparison Study of NER Software: StanfordNLP, NLTK, OpenNLP, SpaCy, Gate”, 2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)

[34] Xavier Schmitt, Sylvain Kublert, Jeremy Robert, Mike Papadakis, Yves LeTraon, “A Replicable Comparison Study of NER Software: StanfordNLP, NLTK, OpenNLP, SpaCy, Gate”, 2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)

[35] Johnson Kolluri, Shaik Razia, Soumya Ranjan Nayak, “Text classification using Machine Learning and Deep Learning Models”, International Conference on Artificial Intelligence in Manufacturing Renewable Energy, 2019

[36] Srinath Reddy Meadusani, “Virtualization Using Docker Containers: For Reproducible Environments and Containerized Applications”, May, 2018

Παράρτημα: Κώδικας Υλοποιήσεων στο Πλαίσιο της Εργασίας

Ο κώδικας της εφαρμογής αντλεί μόνο ανοιχτά δεδομένα και είναι open-source και ο καθένας μπορεί να συνεισφέρει στην ανάπτυξη και εξέλιξη του. Ο Κώδικας της εφαρμογής μπορεί να βρεθεί εδώ.