ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ Μ/Υ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΣΧΟΛΗ ΝΑΥΤΙΛΙΑΣ ΚΑΙ ΒΙΟΜΗΧΑΝΙΑΣ
ΤΜΗΜΑΤΟΣ ΒΙΟΜΗΧΑΝΙΚΗΣ ΔΙΟΙΚΗΣΗΣ & ΤΕΧΝΟΛΟΓΙΑΣ
ΔΙΑΠΑΝΕΠΙΣΤΗΜΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
«ΤΕΧΝΟ-ΟΙΚΟΝΟΜΙΚΑ ΣΥΣΤΗΜΑΤΑ»

Σύγχρονες Διαδικτυακές Εφαρμογές

Δημιουργία εφαρμογής ανάλυσης φωτογραφιών κοινωνικών δικτύων μέσω της πλατφόρμας Google Vision AI

Project 4ου εξαμήνου

Ακαδημαϊκό Έτος 2021-2022

Σωτήρης Κούκιος-Πανόπουλος

Εισαγωγή

Τα τελευταία χρόνια έχει εμφανιστεί η τάση στα κοινωνικά δίκτυα (social networks) μιας ραγδαίας αύξησης στο πλήθος των μηνυμάτων που διαμοιράζονται τα οποία περιέχουν πολυμεσικό περιεχόμενο (multimedia), όπως εικόνες ή βίντεο. Αποτέλεσμα αυτού είναι η εμφάνιση κοινωνικών δικτύων (πχ Instagram, TikTok) τα οποία έχουν σχεδόν αποκλειστικά και μόνο περιεχόμενο σε αυτή τη μορφή. Είναι αρκετά συνήθης η προσπάθεια των χρηστών να χαρακτηρίσουν με διάφορες τεχνικές το περιεχόμενο των μηνυμάτων αυτών. Για παράδειγμα, μέσω hashtags ή άλλων λεκτικών περιγραφών, λειτουργία που υποστηρίζεται από την πλειοψηφία των κοινωνικών δικτύων, και σε ορισμένα είναι πολύ σημαντική η χρήση τους. Ωστόσο, αυτό δεν είναι επαρκές καθώς δεν υιοθετείται μια καθολική τεχνική, και οι περιγραφές είναι συχνά υποκειμενικές.

Σε επέκταση του παραπάνω φαινομένου, υπάρχει μια μεγάλη συμμετοχή στα κοινωνικά δίκτυα από ειδησεογραφικούς οργανισμούς και από μέσα μαζικής ενημέρωσης (ΜΜΕ). Γίνονται αναρτήσεις νέων ειδήσεων στα κοινωνικά δίκτυα, με σκοπό τη γρήγορη ενημέρωση των χρηστών από αυτά. Ένα υποσύνολο αυτών είναι οι λογαριασμοί οι οποίοι αναρτούν πρωτοσέλιδα εφημερίδων στους λογαριασμούς τους, σε μορφή εικόνων. Επίσης αξίζει να σημειωθεί πως για τέτοιο περιεχόμενο έχει επικρατήσει το Twitter να είναι ο μεγαλύτερος πόλος έλξης από τα κοινωνικά δίκτυα για ενημέρωση των χρηστών.

Σκοπός

Σκοπός της παρούσας εργασίας είναι η δημιουργία ενός συστήματος αυτόματης ανάλυσης και αναγνώρισης κειμένου σε εικόνες από κοινωνικά δίκτυα, και συγκεκριμένα το Twitter, μέσω της πλατφόρμας Google Vision API. Το Google Vision AI προσφέρει προ-εκπαιδευμένα μοντέλα μηχανικής μάθησης για άμεση χρήση σε εικόνες, με πολλές διαφορετικές μορφές. Μερικές από αυτές είναι: αναγνώριση προσώπου, ετικετών, ορόσημων, ανίχνευση μη ασφαλούς περιεχομένου, αντίστροφη αναζήτηση εικόνων στο διαδίκτυο, και για την προσέγγισή μας, οπτική αναγνώριση χαρακτήρων (Optical Character Recognition – OCR). Το περιεχόμενο θα αντλείται από το API του κοινωνικού δικτύου της επιλογής μας, το Twitter, και θα εμπλουτίζεται μέσω του Google Vision API.

Προσέγγιση προβλήματος

Η συλλογή των δεδομένων πραγματοποιήθηκε από το κοινωνικό δίκτυο Twitter, και έγινε χρησιμοποιώντας τη .NET βιβλιοθήκη `Tweetinvi`[1], ενώ η επικοινωνία έγινε με τη νέα έκδοση v2 του Twitter API. Η έκδοση αυτή έγινε διαθέσιμη σε παραγωγικό περιβάλλον το Νοέμβριο του 2021. Στην έκδοση αυτή το Twitter στην αναζήτηση των περιεχομένων των αναρτήσεών του (εφεξής tweets) προσφέρει μια πιο πλούσια απάντηση με ευκολότερη ανάκτηση των πολυμεσικών περιεχομένων του κάθε tweet. Για την επικοινωνία με το API, χρειάζεται μια διαδικασία απόκτησης πρόσβασης μέσω του Twitter Developer Platform[2], ενώ έχει περιορισμούς στο πλήθος των requests ανά μήνα και των εφαρμογών που μπορούν να είναι ενεργές ανά χρήστη. Στον περιορισμό επίσης της δωρεάν πρόσβασης είχαμε αναζήτηση tweets με μέγιστη παλαιότητα 7 ημερών πριν τη στιγμή της αναζήτησης.

Για την αναγνώριση κειμένου στο περιεχόμενο των εικόνων, χρησιμοποιήθηκε η λειτουργία αναγνώρισης πυκνού κειμένου (dense text detection) του Google Vision API. Η λειτουργία αυτή είναι κατάλληλη για ανάκτηση κειμένου από εικόνες που περιέχουν μεγάλο ποσοστό κειμένου, όπως σελίδες με κείμενο, και ταιριάζει περισσότερο στο πρόβλημά μας. Για την επικοινωνία με το Google Vision API χρειάζεται επίσης ένας developer λογαριασμός στο Google Cloud, και αντίστοιχη δημιουργία εφαρμογής και λήψη κλειδιών πρόσβασης στο Cloud Vision API. Από πλευρά κώδικα, χρησιμοποιήθηκε η .NET βιβλιοθήκη `Google.Cloud.Vision.V1`[3].

Η εφαρμογή αναπτύχθηκε σε τεχνολογίες .NET και συγκεκριμένα με τη γλώσσα προγραμματισμού C#. Χρησιμοποιήθηκε το framework .NET 6, και για το σχεδιασμό και τη διαχείριση του σχήματος των πινάκων της βάσης δεδομένων χρησιμοποιήθηκε το Entity Framework Core, μια τεχνολογία ORM (object – relational mapper) που βοηθά στο σχεδιασμό και την επικοινωνία με τη βάση δεδομένων. Ως βάση δεδομένων υπάρχει μια βάση σε MySQL server.

Η εφαρμογή ονομάστηκε **Octweet**, ως συνδυασμός των όρων “OCR” και “Tweet”.

---

[1] GitHub repository: https://github.com/linvi/tweetinvi

[2] Twitter Developer Platform: https://developer.twitter.com/

[3] Library: https://www.nuget.org/packages/Google.Cloud.Vision.V1/
Repository: https://github.com/googleapis/google-cloud-dotnet

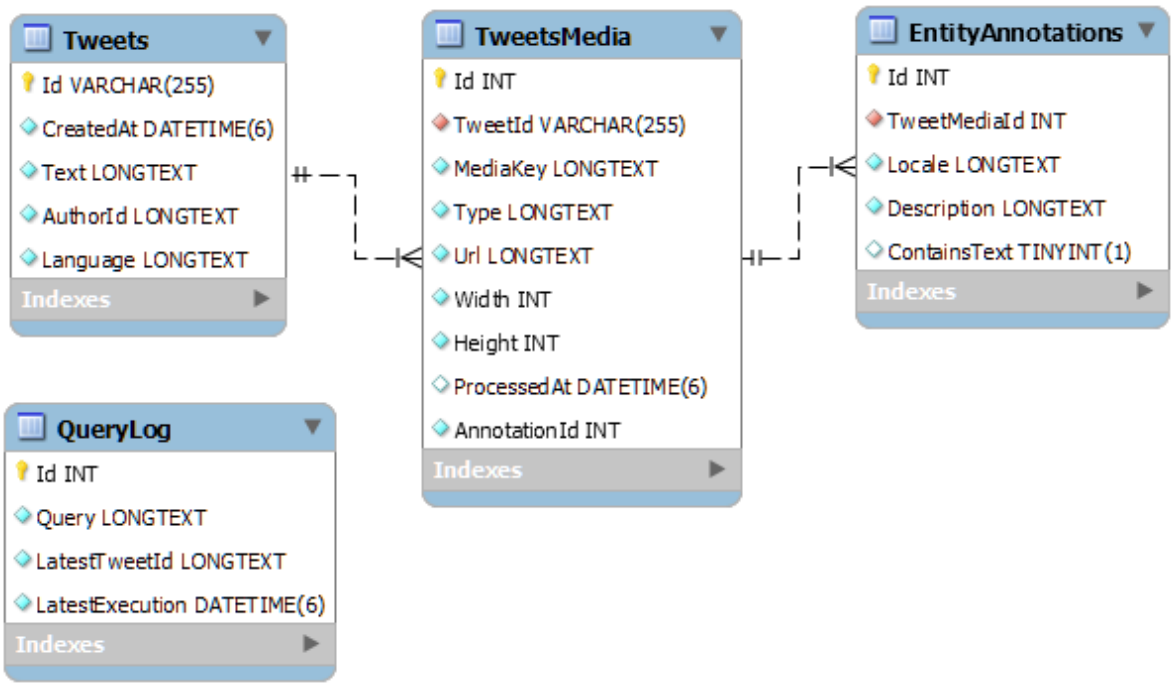
Βάση δεδομένων

Όπως αναφέρθηκε παραπάνω, η βάση δεδομένων που χρησιμοποιήθηκε, «ζει» σε έναν MySQL server, ο οποίος με τη σειρά του εγκαταστάθηκε σε ένα Docker container, για αμεσότερη χρήση και εγκατάσταση όλης της εφαρμογής από τον εκάστοτε χρήστη.

Το σχήμα (schema) της βάσης δεδομένων παράγεται με γνώμονα τον κώδικα, χάρη στο ORM Entity Framework Core. Οι πίνακες σχεδιάζονται ως κλάσεις, με τη κάθε ιδιότητα (property) να αντιστοιχεί σε μια κολώνα του πίνακα, ενώ υπάρχει και επιπλέον προδιαγραφή για τις ρυθμίσεις των πεδίων. Παράδειγμα τέτοιων ρυθμίσεων είναι ο ορισμός μιας κολώνας ως πρωτεύον κλειδί, ο καθορισμός σχέσεων μεταξύ ιδιοτήτων δυο κλάσεων/πινάκων, ο καθορισμός του τύπου και του μήκους των πεδίων κ.ο.κ.

Οι αλλαγές αυτές στο σχήμα της βάσης δεν εφαρμόζονται αυτόματα, αλλά δημιουργούνται τα καλούμενα “Migrations” που ουσιαστικά αποτελούν ένα σύνολο εντολών SQL για την μεταβολή του σχήματος της βάσης δεδομένων. Στην τελική εφαρμογή, ωστόσο, γίνεται έλεγχος κατά την αρχικοποίηση της εφαρμογής αν υπάρχει η βάση δεδομένων. Αν δεν υπάρχει, εκτελούνται όλα τα migrations και βεβαιωνόμαστε ότι έχουμε τη βάση δεδομένων με το τελευταίο σχήμα, που αντιστοιχεί και στον κώδικα που θα τρέξει.

Σχήμα Βάσης OctweetDB



Για την αποθήκευση των δεδομένων έχουμε τους τρεις πίνακες `Tweets`, `TweetsMedia`, και `EntityAnnotations`. Ο πίνακας `QueryLog` βοηθάει στην εκτέλεση της εφαρμογής και την συνέχιση αναζήτησης ώστε να μην υπάρχουν πολλαπλές εγγραφές για το ίδιο tweet στη βάση μας.

Ο πίνακας `Tweets` αποθηκεύει τα αποτελέσματα των αναζητήσεων από το Twitter API. Αποθηκεύουμε ένα υποσύνολο από τις ιδιότητες του κάθε tweet, με κυριότερα το `Id`, το μοναδικό

χαρακτηριστικό που μας επιστρέφει το Twitter API, το `CreatedAt`, για τη χρονοσφραγίδα δημιουργίας του tweet, και το `Text`, που είναι όλο το περιεχόμενο του αναρτημένου tweet.

Ο πίνακας `TweetsMedia` περιέχει τα πολυμέσα τα οποία είναι συνδεδεμένα με τα tweets που αποθηκεύουμε, χρησιμοποιώντας την κολώνα `TweetId` (ξένο κλειδί προς το `Id` του πίνακα `Tweets`). Εν γένει, τα αντικείμενα πολυμέσων μπορούν να είναι φωτογραφίες, GIFs, ή βίντεο, αλλά για τους σκοπούς της εργασίας μας κάνουμε φιλτράρισμα στις αναζητήσεις μας και καταχωρούμε τα tweets και τα αντίστοιχα πολυμέσα τους που έχουν ως `Type`: `photo`. Η κολώνα `MediaKey` περιλαμβάνει το μοναδικό αναγνωριστικό ενός αντικειμένου πολυμέσου που μας επιστρέφει το TwitterAPI, και η κολώνα `Url` περιέχει το στατικό Url της κάθε εικόνας. Τα πεδία `AnnotationId` και `ProcessedAt` έχουν σχέση με τον πίνακα που περιγράφεται στη συνέχεια, τον `EntityAnnotations`. Το `AnnotationId` δείχνει στην αντίστοιχη εγγραφή του `EntityAnnotations` που περιέχει το αποτέλεσμα του OCR, ενώ το `ProcessedAt` αποθηκεύει τη στιγμή την οποία αποθηκεύτηκε το αποτέλεσμα στον `EntityAnnotations` πίνακα.

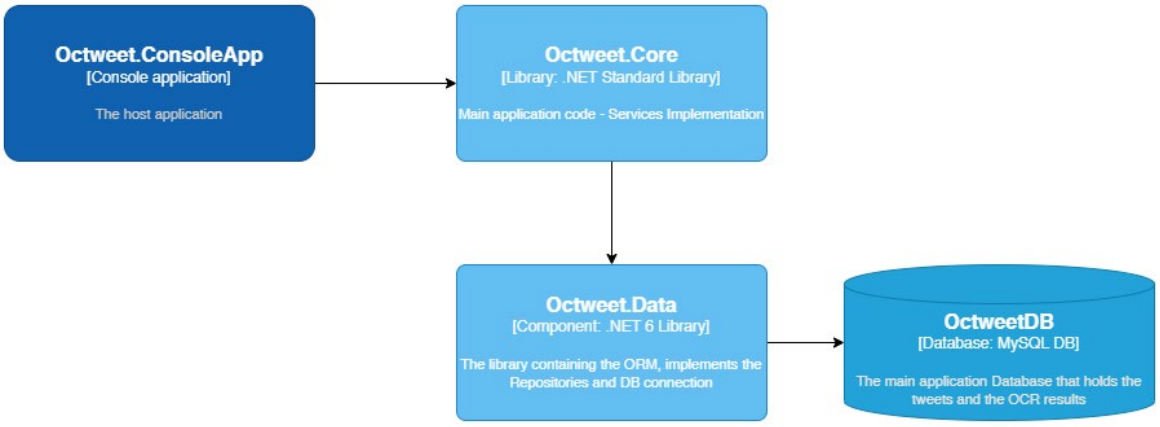
Ο πίνακας `EntityAnnotations` περιέχει τα δεδομένα που ανακτώνται από το Google Vision API για κάθε εικόνα που επεξεργαζόμαστε. Αν μια εικόνα δεν περιέχει κείμενο, σημειώνεται στη κολώνα `ContainsText`, ώστε να μην ληφθεί υπόψη σε περαιτέρω ανάλυση. Η κολώνα `Description` περιέχει το κείμενο που ανιχνεύθηκε στην εικόνα, ενώ η κολώνα `Locale` τη γλώσσα που αναγνωρίστηκε από το Google Text Detection API.

Ο πίνακας `QueryLog` χρησιμοποιείται για καταγραφή πληροφοριών γύρω από τις αναζητήσεις που εκτελούμε. Η κολώνα `Query` περιέχει το τι αναζητούμε, και για κάθε αναζήτηση, ενημερώνουμε τα `LatestExecution` και το `LatestTweetId` που αποθηκεύσαμε για τη κάθε αναζήτηση. Με αυτόν τον πίνακα υποστηρίζουμε ένα μηχανισμό που υλοποιήθηκε στην εφαρμογή για συνέχιση των αναζητήσεων, όπως θα περιγράψουμε στη συνέχεια.

Εφαρμογή

Η εφαρμογή Octweet έχει γραφεί σε γλώσσα προγραμματισμού C#, πάνω στο περιβάλλον .NET 6. Είναι μια εφαρμογή κονσόλας, και αποτελείται από δύο διαφορετικές υπηρεσίες-εργάτες (worker services[4]). Οι υπηρεσίες αυτές εκτελούνται στην διεργασία της εφαρμογής και δεν χρειάζονται κάποιο περιβάλλον χρήστη (user interface – UI), και είναι ιδανικές για εκτέλεση εργασιών μεγάλης διάρκειας στο παρασκήνιο. Υποστηρίζουν πολλά χρήσιμα χαρακτηριστικά του οικοσυστήματος .NET, όπως παραμετροποίηση (configuration), καταγραφή (logging) και dependency injection.

Τα δυο διαφορετικά services που εκτελούνται, εκτελούνται ανεξάρτητα μεταξύ τους, και το καθένα έχει τη δική του διακριτή εργασία. Το ένα service (Twitter Scrapper) έχει σαν εργασία να εκτελέσει την αναζήτηση των tweets, χρησιμοποιώντας το Twitter API, ενώ το δεύτερο service (Google Vision Worker) εκτελεί την αναγνώριση κειμένου στις διαθέσιμες εικόνες, χρησιμοποιώντας το Google Vision API. Και στις δυο περιπτώσεις, έχουμε την κάθε εργασία να εκτελείται εκ νέου κάθε 1 λεπτό, με τον χρόνο αυτό να είναι παραμετροποιήσιμος από configuration files ή από arguments κατά την εκτέλεση της εφαρμογής.



Η αρχιτεκτονική της εφαρμογής ακολουθεί την δομή στρωμάτων, με το κάθε στρώμα να επικοινωνεί μόνο με το αμέσως χαμηλότερο πάντα, διασφαλίζοντας έτσι την χαμηλή σύνδεση (coupling) των συνιστωσών (components) της εφαρμογής, με το καθένα από αυτά να έχει υψηλή συνεκτικότητα. Το πρώτο στρώμα της εφαρμογής είναι το εκτελέσιμο «`Octweet.ConsoleApp`» που αποτελεί το επίπεδο παρουσίασης της εφαρμογής, και είναι υπεύθυνο τόσο για την εκτέλεση της εφαρμογής, όσο και για τη συγκέντρωση και εγγραφή των εξαρτήσεων (dependency registration) και για την φόρτωση των παραμέτρων που θα καθορίζουν την ροή εκτέλεσης της εφαρμογής. Το δεύτερο στρώμα είναι η βιβλιοθήκη `Octweet.Core` η οποία περιέχει την πλειοψηφία του κώδικα της εφαρμογής και είναι το στρώμα όπου είναι υλοποιημένη η λογική για το κάθε service. Το τελευταίο στρώμα (`Octweet.Data`), με το οποίο επικοινωνεί το προηγούμενο, είναι το στρώμα όπου είναι υλοποιημένη η λογική για την επικοινωνία με την βάση δεδομένων. Περιέχει αποθετήρια (repositories) και τον ορισμό των μοντέλων και την επικοινωνία με την βάση δεδομένων.

[4] Worker Services documentation: https://docs.microsoft.com/en-us/dotnet/core/extensions/workers

Twitter Service

Κάθε φορά που εκτελείται το worker service που αφορά το Twitter, εκτελούνται τα εξής βήματα, με στόχο να ανακτηθούν και να αποθηκευτούν στην βάση δεδομένων τα tweets:

- Ανακτάται από τις παραμέτρους της εφαρμογής το κείμενο αναζήτησης που θα πρέπει να εκτελεστεί. Σε αυτό προστίθεται ένα επιπλέον φίλτρο ώστε να ανακτούμε μόνο tweets που περιέχουν εικόνες.
- Ανακτώνται από την βάση δεδομένων και συγκεκριμένα τον `QueryLog` πίνακα οι πληροφορίες εκτέλεσης για το συγκεκριμένο κείμενο αναζήτησης, προκειμένου να εφαρμοστεί σωστά η λογική συνέχισης της εφαρμογής.
- Κατασκευάζουμε το request για το Twitter API, έχοντας επιπλέον ως «οδηγία» να ανακτηθούν tweets μετά από το τελευταίο που έχουμε αποθηκεύσει (πληροφορία από το προηγούμενο βήμα). Έτσι διασφαλίζουμε πως δεν θα έχουμε πολλαπλές εγγραφές στη βάση δεδομένων για το ίδιο tweet.
- Χρησιμοποιούμε την εξωτερική βιβλιοθήκη Tweetinvi για την εκτέλεση της αναζήτησης, αφού πρώτα έχει γίνει αρχικοποίηση με τα credentials της εφαρμογής μας για το Twitter API.
- Συγκεντρώνονται τα αποτελέσματα και προσαρμόζονται στο μοντέλο που αντιστοιχεί στη βάση δεδομένων που έχουμε σχεδιάσει, και αποθηκεύονται. Αποθηκεύουμε μαζί τόσο το ίδιο το tweet (στον πίνακα `Tweets`) όσο και τις εικόνες που σχετίζονται με αυτό (στον πίνακα `TweetsMedia`)
- Ενημερώνουμε τον πίνακα `QueryLog` με το τελευταίο tweet που αποθηκεύσαμε, για να το χρησιμοποιήσουμε στην επόμενη εκτέλεση του service.

Google Service

Το service αυτό έχει ως στόχο να εντοπίσει ποιες εικόνες που έχουμε αποθηκεύσει στη βάση δεδομένων δεν έχουν επεξεργαστεί ακόμα, και αν υπάρχουν τέτοιες, επικοινωνεί με το Google Vision API για να λάβει το αποτέλεσμα. Αφού το λάβει, αποθηκεύεται στη βάση μας στον πίνακα `EntityAnnotations` και αντίστοιχα μαρκάρονται οι εικόνες στον `TweetsMedia` πίνακα ως επεξεργασμένες για να μην ανακτηθούν κατά την επόμενη εκτέλεση.

Docker

Για ευκολότερη ανάπτυξη της εφαρμογής αλλά και για λόγους διαμοιρασμού της, έχει δημιουργηθεί ένα docker-compose αρχείο που περιέχει την εφαρμογή και τις απαραίτητες εφαρμογές στις οποίες εξαρτάται. Μέσα στην docker εφαρμογή, υπάρχουν οδηγίες για διαχείριση containers μιας βάσης δεδομένων MySQL (με κατάλληλη παραμετροποίηση για τον κωδικό του χρήστη root με τον οποίο εκτελείται η εφαρμογή), και ένα container της υπηρεσίας Seq, η οποία προσφέρει ένα περιβάλλον για αποθήκευση και αναζήτηση των logs της εφαρμογής.

Η εκτέλεση της εφαρμογής με αυτό το τρόπο γίνεται πολύ εύκολη χωρίς να χρειάζεται εγκατάσταση διαφορετικών frameworks και εφαρμογών. Χρειάζεται ωστόσο μια σωστή παραμετροποίηση για τα credentials προς τις υπηρεσίες Google Vision API και Twitter API, καθώς τα credentials αυτά δεν θα μπορούσαν να διαμοιραστούν στο source control management σύστημα που χρησιμοποιήθηκε (GitHub). Στο readme της εφαρμογής (https://github.com/sokopa/Octweet/blob/main/README.md) περιέχονται αναλυτικές οδηγίες για το πώς μπορεί κάποιος να αποκτήσει credentials για τις υπηρεσίες και πώς μπορεί να παραμετροποιήσει σωστά την εφαρμογή.

Αποτελέσματα

Για τη συγκέντρωση των δεδομένων, έγινε η κατάλληλη παραμετροποίηση της εφαρμογής ώστε να αναζητηθούν tweets τα οποία έχουν αναρτηθεί από τον χρήστη Front Pages Today - ukpapers. Σύμφωνα με το τρόπο κατασκευής των παραμέτρων αναζήτησης για το Twitter API, αυτή η αναζήτηση θα πρέπει να γραφεί ως «`from: ukpapers`», και προσθέσαμε επίσης το φίλτρο «`has images`» για να μας επιστραφούν αποτελέσματα που περιέχουν εικόνες. Ο λογαριασμός αυτός αναρτά καθημερινά tweets με τα πρωτοσέλιδα εφημερίδων αγγλόφωνων χωρών: Μ. Βρετανία, Η.Π.Α., Αυστραλία, Καναδάς.

Έγινε συλλογή και αποθήκευση στη βάση δεδομένων 1640 Tweets, με 1618 από αυτά να περιέχουν στατικές εικόνες που αποθηκεύτηκαν στον πίνακα `TweetsMedia`. Η διαφορά αυτή, ενώ είχαμε ήδη ζητήσει από τις παραμέτρους αναζήτησης να μας επιστραφούν tweets που περιέχουν εικόνες, φαίνεται να εξηγείται από το γεγονός πως οι εικόνες που περιέχονταν στα 22 tweets αυτά ήταν κινούμενες (GIFs), και στην επιπλέον διαλογή που εκτελέστηκε κατά την αποθήκευση στην πλευρά της εφαρμογής, τις αγνοήσαμε. Σε ορισμένα tweets υπήρχαν και πολλαπλές εικόνες, και από την διαδικασία της αναγνώρισης κειμένου μέσω του Google Vision API βρέθηκαν και εικόνες που δεν περιείχαν κείμενο. Τελικά στην ανάλυσή μας χρησιμοποιήθηκαν 1660 εικόνες, και θα αποτιμήσουμε την ομοιότητα του περιεχομένου της αρχικής ανάρτησης με το περιεχόμενο της εκάστοτε εικόνας.

Για την ανάλυση των αποτελεσμάτων, χρησιμοποιήθηκε η μετρική απόστασης Levenshtein[5], η οποία είναι μια μετρική για την αποτίμηση της διαφοράς μεταξύ δύο συμβολοσειρών. Ουσιαστικά αναπαριστά το πλήθος των αλλαγών ενός χαρακτήρα (προσθήκες, διαγραφές ή αντικαταστάσεις) που χρειάζεται να γίνουν για να μετασχηματίσουμε μια λέξη στην άλλη. Με την παρακάτω σχέση μπορούμε επίσης να υπολογίσουμε μια μετρική ομοιότητας μεταξύ δυο συμβολοσειρών, την οποία θα χρησιμοποιήσουμε για τα αποτελέσματά μας:

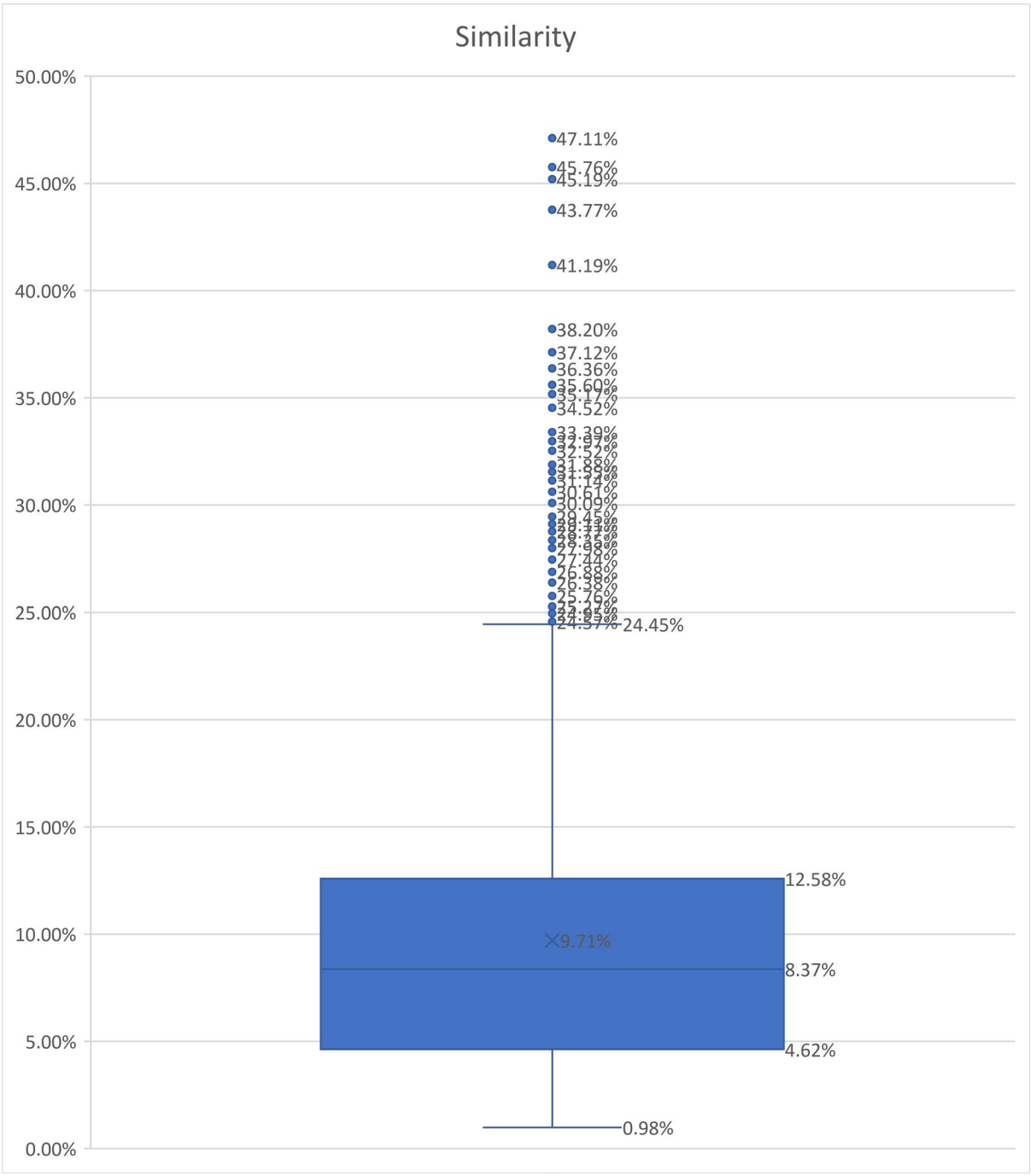
$$Similarity = \frac{Max\big(len(a), len(b)\big) - LevenshteinDistance(a, b)}{Max(len(a), len(b))}$$

όπου a, b δυο συμβολοσειρές (strings), $len(.)$ η συνάρτηση μήκους της συμβολοσειράς, και $LevenshteinDistance(a, b)$ η απόσταση Levenshtein μεταξύ των συμβολοσειρών.

Στο διάγραμμα της επόμενης σελίδας υπάρχει μια σύνοψη των ποσοστών ομοιότητας για κάθε αποτέλεσμα. Παρατηρούμε πως η ομοιότητα παραμένει σε χαμηλά επίπεδα, με μια διάμεση τιμή 8.37%, ενώ μέση 9.71%. Υπάρχουν περιπτώσεις που προσεγγίζουν και το 50%, όμως η πλειοψηφία των τιμών κυμαίνεται από 4.62% έως 12.58%.

Η χαμηλή ομοιότητα οφείλεται στο γεγονός ότι συχνά στα tweets αναγράφονται οι κύριες ειδήσεις του πρωτοσέλιδου, είτε ως αυτούσια φράση, είτε με μικρή παραλλαγή, και ότι το πρωτοσέλιδο περιλαμβάνει και άλλο κείμενο. Για παράδειγμα, περιέχει τον τίτλο της εφημερίδας, ημερομηνία, άλλες ειδήσεις, περίληψη μια είδησης κ.ο.κ. Αυτό έχει σαν αποτέλεσμα η συμβολοσειρά με την οποία συγκρίνουμε το αρχικό tweet είναι κατά πολύ μεγαλύτερη, και η σύγκριση αυτή δεν επιτυγχάνει μεγάλο ποσοστιαίο αποτέλεσμα.

[5] Levenshtein distance: https://en.wikipedia.org/wiki/Levenshtein_distance


Similarity
50.00%
45.00%
40.00%
35.00%
30.00%
25.00%
20.00%
15.00%
10.00%
5.00%
0.00%
47.11%
45.76%
45.19%
43.77%
41.19%
38.20%
37.12%
36.36%
35.60%
35.17%
34.52%
24.45%
12.58%
9.71%
8.37%
4.62%
0.98%