Τεχνικές Εξόρυξης Διαδικτύου – Εργασία 1

Μεντζέλος Ηλίας 1115201400106

Φιλιππίδης Αθανάσιος 1115201400215

Δημιουργία WordCloud

Για το Wordcloud χρησιμοποιήσαμε κυρίως την βιβλιοθήκη "wordcloud", το οποίο αν δεν έχει γίνει install πρέπει να γίνει με την εντολή pip install wordcloud. Επίσης, χρησιμοποιήσαμε τα stop words που παρέχονται από αυτήν, ενώ προσθέσαμε και μερικά ακόμα δικά μας stop words που κρίναμε αναγκαία για την παραγωγή μιας πιο αντιπροσωπευτικής εικόνας.

Επιπρόσθετα, αποφασίσαμε να δώσουμε έναν πιο ιδιαίτερο τόνο στην εργασία μας προσθέτοντας και μια μάσκα στο wordcloud, η οποία για να λειτουργήσει σωστά στον ίδιο φάκελο με το πρόγραμμα πρέπει να υπάρχει και το αρχείο "stormtrooper_mask.png" το οποίο και επισυνάπτουμε μαζί με τα υπόλοιπα αρχεία της εργασίας. Τέλος, για την τελική παραγωγή της επιθυμητής εικόνας έγινε χρήση της matplotlib.pyplot.

Αποτελέσματα

Politics

Business

Films

Football

Technology

Υλοποίηση Συσταδοποίησης (Clustering)

Υλοποήσαμε το Clustering με χρήση του αλγόριθμου K-Means. Ανοίγουμε το csv, βάζουμε το content σε μια list και μετά τα μετατρέπουμε σε διανύσματα μέσω του vectorizer.fit_transform. Στην συνέχεια, βρίσκουμε 5 τυχαία κέντρα, το οποίο προφανώς μεταβαλλεί τον αριθμό των προσπαθειών που θα χρειαστεί ο αλγόριθμος. Έπειτα ξεκινάμε τον αλγόριθμο με μέγιστο τις 100 προσπάθειες. Για κάθε προσπάθεια, υπολογίζουμε για την κάθε εγγραφή την απόσταση του με το κάθε κέντρο και το εισάγουμε σε αυτό με την μικρότερη απόσταση, η οποία υπολογίζεται με την cosine similarity. Για κάθε append που κάνουμε στο cluster, εισάγουμε και την κατηγορία του στοιχείου σε μια λίστα, όπου κρατάμε τα categories των στοιχείων του cluster. Όταν γίνει αυτό για όλα τα στοιχεία, πάμε και υπολογίζουμε το νέο κέντρο για κάθε cluster βασισμένο στα διανύσματα που ανήκουν σε αυτό. Έπειτα συγκρίνουμε τα νέα κέντρα με τα παλιά και αν είναι διαφορετικά πάμε στην επόμενη προσπάθεια, όπου επαναλαμβάνουμε την ίδια διαδικασία. Αν τα κέντρα είναι όλα ίδια με τα αντίστοιχα προηγούμενα τους, τότε ο αλγόριθμος K-Means τερματίζει. Υπολογίζονται τα στατιστικά για κάθε κέντρο, δημιουργείται το csv αρχείο και έπειτα το πρόγραμμα τερματίζει.

Average Efforts: 28, Max Efforts: 38

Παραδείγματα Εκτέλεσης:

	A	B	C	D	E	F
1		Politics	Film	Football	Business	Technology
2	Cluster 1	0.9	0	0	0.1	0
3	Cluster 2	0	0	0.9	0	0
4	Cluster 3	0.1	0	0	0.9	0
5	Cluster 4	0.1	0	0	0.4	0.4
6	Cluster 5	0	1	0	0	0

	A	B	C	D	E	F
1		Politics	Film	Football	Business	Technology
2	Cluster 1	0.9	0	0	0.1	0
3	Cluster 2	0	1	0	0	0
4	Cluster 3	0	0	0.9	0	0
5	Cluster 4	0.1	0	0	0.9	0
6	Cluster 5	0.1	0	0	0.4	0.4

Παρατηρούμε ότι σχεδόν κάθε cluster έχει μαζέψει το μεγαλύτερο κομμάτι του από μια μόνο κατηγορία, εκτός από ένα cluster το οποίο μαζεύει από δυο κατηγορίες, οι οποίες είναι αρκετά συνδεδεμένες μεταξύ τους, το Business και το Technology.

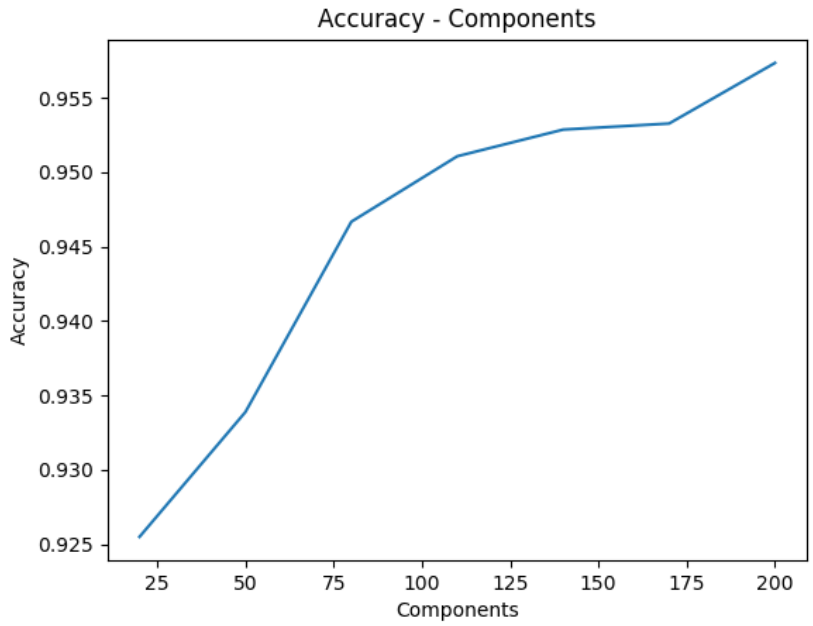
Υλοποίηση Κατηγοριοποίησης (Classification)

Starting the Classification

Για την υλοποιήση του Classification υλοποιήσαμε ένα dictionary το οποίο αντιστοιχεί την κάθε κατηγορία σε έναν αριθμό από το 0 εώς το 4. Έπειτα, με βάση αυτό το dictionary, πήραμε την στήλη category από το data αρχείο μας και φτιάξαμε έναν πίνακα με αριθμούς (0-4) όπου περιέχει την κατηγορία για κάθε εγγραφή.

Τεχνική LSI – Σχέση Components με Accuracy

Στο επόμενο κομμάτι για να ελέγξουμε την σχέση του αριθμού των Components του LSI με το Accuracy, χρησιμοποιήσαμε μια επανάληψη, στην οποία αλλάζαμε τον αριθμό των Components, καλούσαμε τον TruncatedSVD για να παράξουμε το X_LSI, ενώ τέλος καλούσαμε τον αλγόριθμο SGD για να μας κάνει predict τα αποτελέσματα και να υπολογίσουμε το Accuracy. Παρακάτω είναι το σχήμα πού παράχθηκε με τα αποτέλεσματα του της διαδικασίας.



Όπως είναι φανερό από το σχήμα, όσα περισσότερα components τόσο μεγαλύτερο το Accuracy.

10-fold Cross Validation

Χρησιμοποιούμε την KFold με αριθμό από splits = 10, όπως ορίσθηκε στην εκφώνηση. Ακολουθώντας το tutorial, κάνουμε μια for στο kf.split(train_data[category_criteria]), μέσα στην οποία για κάθε κομμάτι του train data καλώντας τον classifier υπολογίζουμε τα metrics που ζητήθηκαν από την άσκηση και εκτυπώνουμε τον μέσο όρο τους και τον εκτυπώνει στο αρχείο EvalutionMetric_10fold.csv . Αυτή η διαδικασία γίνεται 3 φορές, μια με classifier Naïve Bayes, μια με classifier SVM και μια με classifier Random Forests. Τα αποτελέσματα των metrics φαίνονται παρακάτω:

	A	B	C	D
1	Statistic Measure	Naive Bayes	Random Forest	SVM
2	Accuracy	0.959480809	0.9572799877	0.9080371495
3	Precision	0.9570057601	0.9529742967	0.9095825584
4	Recall	0.9557339216	0.9545273306	0.890197305
5	F-Measure	0.9561679882	0.9536337423	0.8966873455
6	AUC	0.5978502446	0.6029476081	0.5916051568

Test Set Categories

Έχουμε δημιουργήσει 3 διαφορετικά csv αρχεία TestSetCategories όπου το καθένα έχει δημιουργηθεί με διαφορετικό Classifier. Επειδή τα αποτελέσματα είναι πολλα θα εκτυπωθούν τα 10 πρώτα από κάθε Classifier.

testSet_categories_NaiveBayes.csv:

	A	B
1	ID	Category
2	2	Politics
3	10	Technology
4	25	Technology
5	28	Business
6	29	Business
7	33	Business
8	34	Business
9	37	Technology
10	39	Technology
11	40	Technology

testSet_categories_SVM.csv:

	A	B
1	ID	Category
2	2	Politics
3	10	Technology
4	25	Technology
5	28	Business
6	29	Business
7	33	Business
8	34	Business
9	37	Business
10	39	Technology
11	40	Business

testSet_categories_RandomForest.csv:

	A	B
1	ID	Category
2	2	Politics
3	10	Technology
4	25	Film
5	28	Business
6	29	Business
7	33	Business
8	34	Business
9	37	Business
10	39	Technology
11	40	Business

K-Nearest Neighbors:

Για την υλοποίηση του K nearest neighbors, έπειτα από λίγη έρευνα στο διαδίκτυο συγκεντρώσαμε μερικούς αλγορίθμους που βρήκαμε σχετικούς με τον KNN, κρατήσαμε όποια ιδεά βρήκαμε χρήσιμη από τον καθένα και τελικά το μεταφράσαμε σε python. Η τελική μας υλοποίηση είναι σχετικά απλή για αυτό και δεν έχει καθόλου καλό χρόνο εκτέλεσης, με πολυπλοκότητα υλοποίησης ψευδοπολυωνυμικού χρόνου.

Πρακτικά, αρχικά κάνουμε διανύσματα τα δεδομένα μας, μετά αφαιρούμε όσο το δυνατόν περισσότερο περιττό data, έπειτα παίρνουμε ένα ένα τα στοιχεία που θέλουμε να προβλέψουμε, ψάχνουμε τους κ κοντινότερους τους γείτονες με βάση την ήδη υπάρχουσα γνώση, και βλέποντας σε ποιά κατηγορία ανήκει ο καθένας απο αυτούς "ψηφίζουν" μεταξύ τους για το ποιανού το "μέρος" θα πάρει το νέο στοιχείο και κρατάμε την πρόβλεψη αυτή.

Αφού κάνουμε το ίδιο για το κάθε ένα προς εξέταση στοιχείο ελέγχουμε τις προβλέψεις μας για να δούμε τι ποσοστό ακρίβειας είχαμε.

Σε σχόλια υπάρχει και υλοποίηση με χρήση KFolds η οποία όμως δεν καταφέραμε να την κάνουμε να λειτουργήσει σωστά λόγω πίεσης χρόνου. Για αυτόν ακριβώς τον λόγο, δεν έχει συμπεριληφθεί στο αρχείο EvalutionMetric_10fold.csv .

Η υλοποιήση του βρίσκεται στο αρχείο KNN.py

Beat the Benchmarks

Αν και δεν υλοποιήσαμε κάποιο συγκεκριμένο κομμάτι κώδικα για αυτό το ερώτημα, μετά από διάφορες δοκιμές νομίζω πως μπορούμε να προτείνουμε την λύση την οποία εμείς θεωρούμε ό,τι θα οδηγήσει στην μέγιστη απόδοση.

Αρχικά, ήταν εύκολο να αναγνωρίσουμε ότι ο Random Forests αποτελεί τον ιδανικό classifier με μεγάλο μάλιστα αριθμό από trees. Ο Random Forest μας παρέχει την καλύτερη απόδοση από τους άλλους αλλά και υψηλότερα metrics. Αξίζει να σημειωθεί πως κατάλληλη θα ήταν και μια καλή υλοποίηση του K-Nearest Neighbors με μεγάλο αριθμό K, παρόλο που η εκτέλεση του θα αργούσε πολύ.

Πριν από την επιλογή αυτή όμως, θα πρέπει να χρησιμοποιήσουμε έναν vectorizer με ενσωματωμένα stop words, όπως και έχουμε κάνει. Τέλος, απαραίτητη είναι η μείωση των components και η χρήση του TruncatedSVD.

Περιεχόμενα ZIP:

- ReadMe.pdf
- Data-Mining-Project-1-1.py
- Data-Mining-Project-1-2.py
- Data-Mining-Project-1-3.py
- Accuracy-Components.png
- stormtrooper_mask.png
- clustering-KMeans.csv
- EvalutionMetric_10fold.csv
- testSet_categories_NaiveBayes.csv
- testSet_categories_SVM.csv
- testSet_categories_RandomForest.csv
- KNN.py