

ΤΕΧΝΙΚΕΣ ΕΞΟΡΥΞΗΣ ΔΕΔΟΜΕΝΩΝ



ΟΜΑΔΙΚΗ ΕΡΓΑΣΙΑ

Μέλη : Μιτροφάν Κριστιάν (1115201200112)

Βακαλόπουλος Ξενοφών (1115201200011)

ΔΗΜΙΟΥΡΓΙΑ WORDCLOUD

Στο σημείο αυτό καλείστε να δημιουργήσετε ένα Wordcloud για τις πέντε κατηγορίες άρθρων. Για την δημιουργία ενός WordCloud θα χρησιμοποιείτε όλα τα άρθρα κάθε κατηγορίας. Παράδειγμα ενός WordCloud παρουσιάζεται στην ακόλουθη εικόνα. Για την δημιουργία του WordCloud μπορείτε να χρησιμοποιήσετε όποια βιβλιοθήκης της Python επιθυμείτε.

Προκειμένου να υλοποιήσουμε τα ζητούμενα Wordclouds για κάθε κατηγορία άρθρου(Football, Business, Film, Technology, Politics) χρησιμοποιήσαμε τις βιβλιοθήκες Wordcloud, Pandas & Matplotlib. Κάθε wordcloud περιέχει λέξεις που βρίσκονται στον τίτλο των άρθρων της αντίστοιχης κατηγορίας. Το όνομα του προγράμματος είναι wordcloud_gen.py και παράγει τις 5 παρακάτω εικόνες (.png files).

Football.png



Film.png



Business,png



Technology.png



Politics.png



Υλοποίηση Συσταδοποίησης (Clustering)

Σε αυτό το ερώτημα θα πρέπει να υλοποιήσετε *clustering* στα διάφορα αρχεία κειμένου χρησιμοποιώντας τον αλγόριθμο *clustering KMeans*. Η συνάρτηση απόστασης η οποία πρέπει να χρησιμοποιηθεί είναι η *Cosine Similarity*. Ο αριθμός των *clusters* για κάθε ερώτημα θα είναι 5. Ο *KMeans* θα εφαρμοστεί στα δεδομένα εκπαίδευσης (*training set*). Το *clustering* θα πρέπει να υλοποιηθεί χωρίς να χρησιμοποιήσει η μεταβλητή *category*.

Σημείωση: Δεν θα πρέπει να χρησιμοποιήσετε κάποια υλοποίηση του αλγορίθμου η οποία παρέχεται από το *ScikitLearn*. Η υλοποίηση του αλγορίθμου θα πρέπει να γίνει από εσάς.

- Στο συγκεκριμένο ερώτημα ο κώδικας σας θα πρέπει να βγάζει σαν έξοδο ένα αρχείο *csv* με τίτλο: *clustering_KMeans.csv*

- Το αρχείο αυτό θα περιέχει το ποσοστό των δεδομένων κάθε κατηγορίας μέσα στο *cluster*.

Για την υλοποίηση του αλγορίθμου *K-Means* ακολουθήσαμε τα εξής βήματα :

>Επιλέξαμε τυχαία 5 άρθρα στο *.csv data file* για αρχικά *centroids* τα οποία κρατάμε σε μια λίστα.

>Υπολογίζουμε το *cosine similarity* κάθε άρθρου με αυτά τα πέντε *centroids*

>Με βάση το *cosine similarity* τοποθετούμε τα άρθρα σε ένα συγκεκριμένο *cluster* (παίρνουμε το μεγαλύτερο *cosine similarity* του κάθε άρθρου)

>Βρίσκουμε ένα νέο *centroid* για κάθε *cluster* με βάση τα άρθρα που βρίσκονται σε αυτό το *cluster* (Παίρνουμε το μέσο(*mean*) όρο των διαστάσεων.

>Επαναλαμβάνουμε την διαδικασία από το δεύτερο βήμα με τα καινούρια *centroids*

>Ο αλγόριθμος τερματίζει όταν τα καινούρια *centroids* είναι ίδια με τα παλιά (*clusters are stable*) ή όταν φτάσει το ανώτερο όριο των επαναλήψεων (*maxiterations=100*)

n_components=Default

Clusters	Politics	Business	Football	Film	Technology
Cluster0	0.565107913669	0.00323741007194	0.0140287769784	0.386690647482	0.0309352517986
Cluster1	0.0030534351145	0.501984732824	0.38320610687	0.00152671755725	0.110229007634
Cluster2	0.0779922779923	0.0	0.000772200772201	0.919691119691	0.0015444015444
Cluster3	0.0509209100758	0.173708920188	0.539183820874	0.0216684723727	0.21451787649
Cluster4	0.424778761062	0.0493712156497	0.15510013973	0.163949697252	0.206800186306

n_components=Default with stemming & neutral words removal

Clusters	Politics	Business	Football	Film	Technology
Cluster0	0.0405571487095	0.237607537894	0.440393281442	0.0266284309709	0.254813600983
Cluster1	0.00167177486765	0.435775982168	0.480913903594	0.000835887433826	0.0808024519365
Cluster2	0.271752837327	0.0	0.00126103404792	0.722572509458	0.00441361916772
Cluster3	0.362363919129	0.0456194919647	0.134784862623	0.212026956973	0.245204769311
Cluster4	0.551267916207	0.00294009555311	0.02131569276	0.389562660786	0.0349136346931

n_components=200

Clusters	Politics	Business	Football	Film	Technology
Cluster0	0.0157170923379	0.0024557956778	0.00343811394892	0.975933202358	0.0024557956778
Cluster1	0.00136472193791	0.00102354145343	0.989082224497	0.000341180484476	0.00818833162743
Cluster2	0.0782414307004	0.00670640834575	0.000745156482861	0.0134128166915	0.900894187779
Cluster3	0.702267140126	0.0213056541928	0.0546298825458	0.169625785305	0.0521715378312
Cluster4	0.0100174216028	0.934233449477	0.00609756097561	0.0243902439024	0.0252613240418

n_components=200 with stemming & neutral words removal

Clusters	Politics	Business	Football	Film	Technology
Cluster0	0.0237780713342	0.00308234258036	0.00836635843241	0.959489211801	0.00528401585205
Cluster1	0.0228884590587	0.70051579626	0.00773694390716	0.0306254029658	0.238233397808
Cluster2	0.00220750551876	0.0110375275938	0.873436350258	0.00147167034584	0.111846946284
Cluster3	0.00159659393294	0.00266098988824	0.990420436402	0.00266098988824	0.00266098988824
Cluster4	0.71244870041	0.0109439124487	0.00820793433653	0.109986320109	0.158413132695

Υλοποίηση Κατηγοριοποίησης (Classification)

Σε αυτό το ερώτημα θα πρέπει να δοκιμάσετε τις παρακάτω 4 μεθόδους Classification:

- NaiveBayes (Multinomial και Binomial)
- KNearest Neighbor
- Support Vector Machines (SVM)
- Random Forests

Επίσης θα πρέπει να αξιολογήσετε και να καταγράψετε την απόδοση κάθε μεθόδου χρησιμοποιώντας 10fold Cross Validation χρησιμοποιώντας τις παρακάτω μετρικές:

- Accuracy
- ROC plot

Beat the Benchmark Τέλος θα πρέπει να πειραματιστείτε με όποια μέθοδο Classification θέλετε, κάνοντας οποιαδήποτε προ επεξεργασία στα δεδομένα επιθυμείτε με στόχο να ξεπεράσετε όσο περισσότερο μπορείτε την απόδοση σας στο προηγούμενο ερώτημα. Επίσης μπορείτε να χρησιμοποιήσετε ένα συνδυασμό από classifier. Θα πρέπει αναλυτικά να τεκμηριώσετε τα βήματα που ακολουθήσατε

ΒΙΒΛΙΟΘΗΚΕΣ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΗΘΗΚΑΝ ΓΙΑ ΤΟΥΣ CLASSIFIERS

```
import csv
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy import interp
from sklearn.metrics import roc_curve, auc
from sklearn import metrics, preprocessing
from sklearn.linear_model import SGDClassifier
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.decomposition import TruncatedSVD
from sklearn.pipeline import Pipeline
from sklearn.grid_search import GridSearchCV
from sklearn.preprocessing import label_binarize
from nltk.stem.porter import *
from nltk.corpus import stopwords

#Classifier imports
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
from sklearn.cross_validation import Kfold
```

Για την υλοποίηση του *Classification* ακολουθήσαμε το εξής σκεπτικό:

Αρχικά, ορίζουμε όλους τους *classifiers* που θα χρησιμοποιήσουμε, δηλαδή:

- `svc=SVC(probability=True)`
- `mnb=MultinomialNB()`
- `bnb=BernoulliNB()`
- `knn=KNeighborsClassifier()`
- `rfc=RandomForestClassifier()`
- `vcl = VotingClassifier(estimators=[('svc', svc), ('knn', knn), ('rfc', rfc)], voting='soft')`

*Χρησιμοποιήσαμε *pipelines* για να επεξεργαστούμε το *DataFrame* (χρήση *CountVectorizer* & *TFIDF*), να μειώσουμε τις διαστάσεις (χρήση *SVD*), να εκπαιδεύσουμε το μοντέλο (*Pipeline.fit*).

*Όσον αφορά το *10-fold cross validation*, εκπαιδεύουμε το μοντέλο στα 9/10 του *DataSet* και τεστάρουμε στο υπόλοιπο (1/10) με χρήση *Kfold* που παρέχεται από την βιβλιοθήκη *sklearn.cross_validation* βάζοντας τις κατάλληλες παραμέτρους. Αυτή η διαδικασία επαναλαμβάνεται 10 φορές, κάθε φορά για διαφορετικό συνδυασμό *train* & *test* όπου υπολογίζουμε το *accuracy* και το *roc curve*. Βγαίνοντας από την επανάληψη υπολογίζουμε το *mean accuracy* και το *mean roc* και στη συνέχεια δημιουργούμε το *.csv* file και το *.png* file για τις 2 αυτές μετρικές αντίστοιχα. Για την εύρεση των κατηγοριών στο *test_set.csv*, χρησιμοποιήσαμε την μέθοδο που είχε το καλύτερο *accuracy* σε εκείνη την εκτέλεση. Οι κατηγορίες αυτές αποθηκεύονται στο ζητούμενο αρχείο *testSet_categories.csv*.

*Για την δημιουργία του *CSV* file χρησιμοποιούμε την συνάρτηση *csv_writer* που υλοποιήσαμε. Όσον αφορά το *.png* file χρησιμοποιήσαμε την συνάρτηση *savefig* της βιβλιοθήκης *Matplotlib*.

Beating the Benchmark

Data Parsing (pre-processing)

Προκειμένου να βελτιώσουμε την απόδοση, εστίασαμε κυρίως στην προεπεξεργασία των δεδομένων. Αρχικά, αποθηκεύουμε τα άρθρα του *train_set.csv* σε μια μεταβλητή τύπου *DataFrame* και σε αυτήν εφαρμόζουμε τεχνικές για να μειώσουμε το μέγεθος του *Content* κάθε αρχείου και να διατηρήσουμε μόνο σημαντικές λέξεις-κλειδιά.

*Εφαρμόσαμε δύο τέτοιες τεχνικές :

> *Stemming* : Χρησιμοποιήσαμε τον *PortStemmer* της βιβλιοθήκης *nlTK* προκειμένου να μειώσουμε τον αριθμό λέξεων στο *Content*. Δηλαδή, εφαρμόζουμε τον *Stemmer* σε κάθε λέξη κάθε *Content* άρθρου επομένως μειώνεται ο αριθμός ίδιων λέξεων σε ένα συγκεκριμένο *stem*.

>Neutral words removal : Χρησιμοποιήσαμε το stopwords της βιβλιοθήκης nltk προκειμένου να ψάξουμε λέξεις οι οποίες εμφανίζονται συχνά και δεν έχουν βαρύτητα στην σημασία του Content. Μερικές από αυτές τις λέξεις είναι το “the”, ”this”, ”a”, ”an” κτλ.

Αποφασίσαμε να χρησιμοποιήσουμε έναν συνδυασμό μεθόδων(SVC, KneighborsClassifier, RandomForestClassifier) όπου με βάση διαφόρων δοκιμών που κάναμε, είχαν την καλύτερη απόδοση. Τις μεθόδους αυτές τις συνδυάσαμε με χρήση του VotingClassifier που παρέχεται από την βιβλιοθήκη sklearn. Σε συνδυασμό με χρήση της 10-fold , ο βελτιωμένος πλέον αυτός αλγόριθμος επιλέγει την καλύτερη μέθοδο για κάθε επανάληψη μέσα στον 10-fold. Παρακάτω παραθέτουμε κάποια αποτελέσματα από τις διάφορες δοκιμές που έγιναν γι την βελτιώση της απόδοσης:

n_components=50

Statistic Measure	SVM	Naive Bayes(MNB)	Naive Bayes(BNB)	KNN	Random Forest	My Method
Accuracy	0.931354674793	0.936572443565	0.910240231017	0.957605454224	0.95369221074	0.967930787834
ROC	0.498458460714	0.310999085677	0.335966147595	0.377894734472	0.369714403787	0.455300631995

n_components=Default (.png file)

