Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

Τεχνικές Ανάλυσης Δεδομένων Υψηλής Κλίμακας

Δημήτριος Γουνόπουλος

Final Project Report

Χαλβατζάρας Αθανάσιος M1611

Κατούνας Γεώργιος cs3180003

Περιεχόμενα

Περιγραφή

Στην παρούσα εργασία εξετάζονται κάποιες τεχνικές ανάλυσης δεδομένων υψηλής κλίμακας, και πιο συγκεκριμένα:

- Δημιουργία WordCloud
- Εύρεση ομοιότητας μεταξύ κειμένων
- Classification κειμένων με χρήση συγκεκριμένων αλγορίθμων και τεχνικών διανυσματοποίησης:
    - Random Forests (Bag of Words)
    - Random Forests (SVD)
    - Random Forests (Average Word Vector(W2V))
    - SVM (Bag of Words)
    - SVM (SVD)
    - SVM (Average Word Vector(W2V))
    - Custom Method
- Classification ενός test_set κειμένων μετά την εκπαίδευση μοντέλου με την custom μέθοδο που εφαρμόστηκε.

Για την υλοποίηση την άσκησης χρησιμοποιήθηκε η γλώσσα προγραμματισμού Python και μία ευρεία γκάμα βιβλιοθηκών της, με σκοπό την διευκόλυνση υλοποίησης καθώς και για αύξηση της ταχύτητας. Κάποιες από αυτές τις βιβλιοθήκες είναι:

- Numpy
- Pandas
- Nltk
- Wordcloud
- Sklearn
- String
- Genism

WordClouds

Το πρώτο ζητούμενο είναι η δημιουργία WordClouds από τα κείμενα και πιο συγκεκριμένα ένα wordcloud για κάθε κατηγορία κειμένου. Δηλαδή θα δημιουργηθούν 5 WordClouds για τις εξής κατηγορίες:

- Football
- Politics
- Business
- Film
- Technology

Για το σκοπό αυτό δημιουργήθηκε το script wordclouds.py, το οποίο βρίσκεται κάτω από τον κατάλογο scripts του αρχείου που παραδώθηκε. Το output αυτού του script θα παραδωθεί σε αρχέια png κάτω από τον κατάλογο output. Εκεί μέσα μπορούν να παρατηρηθούν τα 5 wordclouds που αναφέρθηκαν παραπάνω.

Βιβλιοθήκες:

Οι βιβλιοθήκες που χρησιμοποιήθηκαν για τη δημιουργία των WordClouds είναι οι εξής:

- Numpy
- Pandas
- Wordcloud
- Matplotlib
- Sys

Επεξήγηση Βημάτων:

1. Άνοιγμα του αρχείου που χρειάζεται και διάβασμα του με την χρήση της βιβλιοθήκης pandas.

```
datadf = pd.read_csv("../data/train_set.csv", sep='\t',encoding='utf-8')

contentdf = datadf['Content']
categorydf = datadf['Category']
titledf = datadf['Title']
iddf = datadf['Id']
```

2. Δημιουργία dataframes για κάθε στήλη του αρχείου που διαβάστηκε. (Content, Category, Title, Id)

```
datadf = pd.read_csv("../data/train_set.csv", sep='\t',encoding='utf-8')

contentdf = datadf['Content']
categorydf = datadf['Category']
titledf = datadf['Title']
iddf = datadf['Id']
```

3. Δημιουργία λιστών για κάθε κατηγορία αρχείου. Έπειτα δημιουργείται μία μεγάλη συμβολοσειρά, που είναι όλα τα κείμενα για κάθε κατηγορία ενωμένα. Οπότε η κατάληξη είναι να έχουμε λίστες, όπου κάθε εγγραφή τους είναι ένα κείμενο.

```
listBusiness = []
listFilm = []
listFootball = []
listTechnology = []
listPolitics = []

l = len(contentdf)-1
while(l >= 0):
    if(categorydf[l] == 'Business'):
        listBusiness.append(str(contentdf[l]))

    elif(categorydf[l] == 'Film'):
        listFilm.append(str(contentdf[l]))

    elif(categorydf[l] == 'Technology'):
        listTechnology.append(str(contentdf[l]))

    elif(categorydf[l] == 'Politics'):
        listPolitics.append(str(contentdf[l]))

    else:
        listFootball.append(str(contentdf[l]))
    l=l-1
#end while


textBusiness = str(listBusiness)
textPolitics = str(listPolitics)
textTechnology = str(listTechnology)
textFilm = str(listFilm)
textFootball = str(listFootball)
```

4. Δημιουργία tuple που περιέχει stopwords. Για αυτό το σκοπό βρέθηκε στο διαδίκτυο ένα αρχείο όπου περιέχει μία ποικιλία από συχνές stopwords. Εμπλουτίστηκε manually και από special character codes όπου εμφανίζονταν στα wordclouds.

```
#create stopwords tuple
stopfile = open("../data/stopwords.txt","r")
more_stopwords = stopfile.readlines()
more_stopwords = map(lambda s: s.strip(), more_stopwords)
special_stopwords = ('x80' , 'xe2' , 'x9d','x93','x99','x99s','xc2','us',
stopfile.close()
more_stopwords.extend(special_stopwords)
more_stopwords_tuple = tuple(more_stopwords)
stop_words = STOPWORDS.union(more_stopwords_tuple)
```

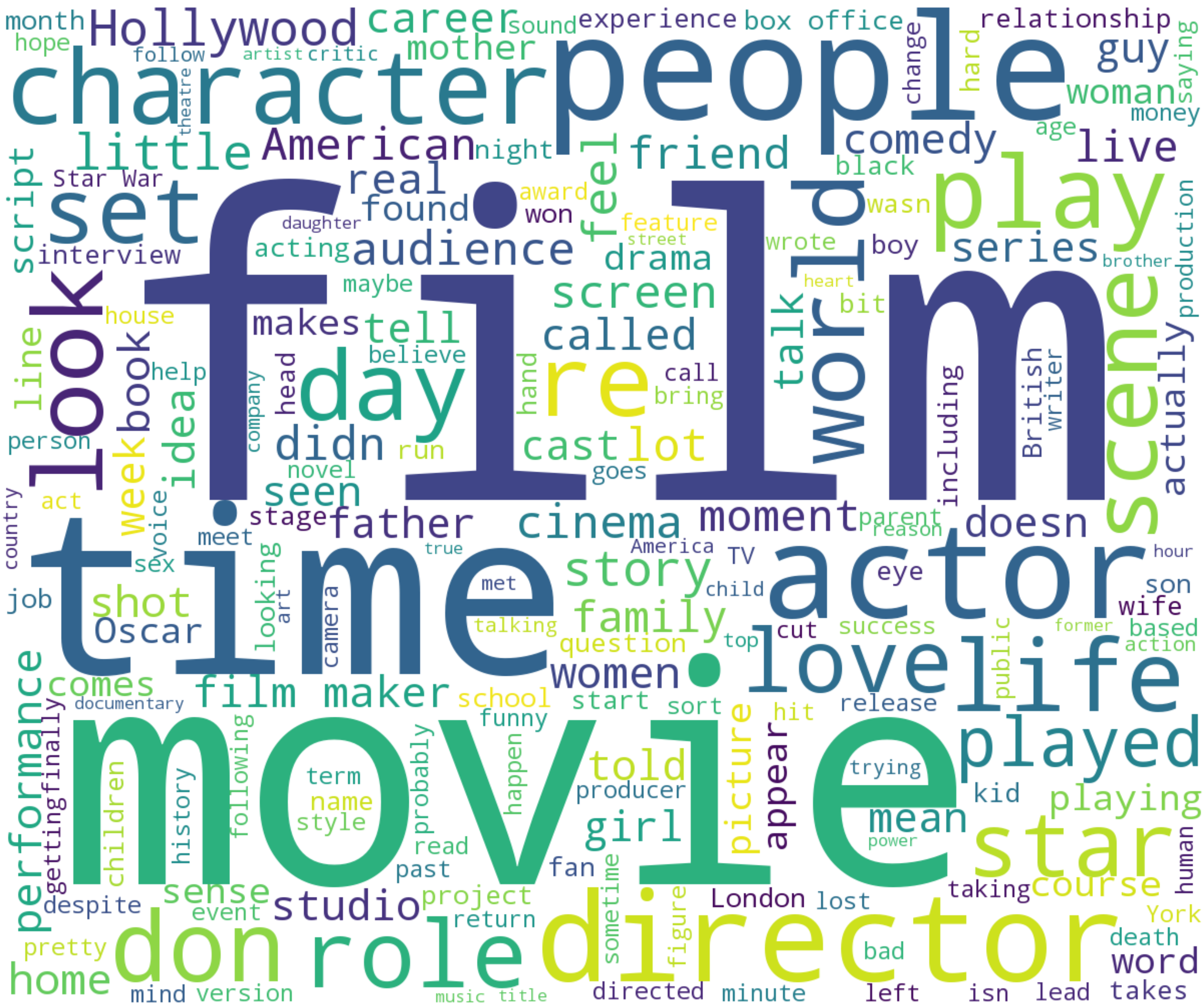
5. Δημιουργία των wordclouds σειριακά.

```
#create the wordclouds
print "Creating Business WordCloud..."
businessWordCloud = WordCloud(stopwords=stop_words, background_color='white', width=1200, height=1000).generate(textBusiness)
name = '../output/wordclouds/BusinessWordcloud.png'
plt.imshow(businessWordCloud)
plt.title("BusinessWordcloud")
businessWordCloud.to_file(name)
plt.axis('off')
```

Output του script wordclouds.py

Business:

Film :


film
movie
time
people
character
director
actor
love
life
play
scene
world
look
set
day
re
star
don
role
played
Hollywood
career
American
little
friend
comedy
audience
series
screen
called
story
family
cinema
father
moment
doesn
women
film maker
studio
performance
girl
mean
told
picture
appear
cast
lot
didn
seen
real
found
drama
feel
talk
book
idea
week
shot
Oscar
comes
sense
home
course
playing
word
mother
guy
woman
live
month
hope
experience
box office
relationship
sound
follow
artist
critic
theatre
change
hard
saying
money
age
night
black
script
Star War
award
won
feature
street
wasn
wrote
boy
brother
production
acting
interview
daughter
maybe
heart
bit
house
makes
tell
believe
line
help
call
bring
hand
head
company
person
run
including
British
writer
actually
novel
goes
act
stage
voice
meet
parent
reason
country
true
America
TV
hour
sex
met
eye
child
son
wife
job
art
camera
looking
talking
question
cut
success
top
former
based
action
public
school
start
sort
hit
release
documentary
funny
finally
getting
children
history
following
probably
happen
term
name
style
producer
trying
kid
power
human
read
past
fan
sometime
figure
taking
despite
event
project
London
lost
return
York
death
pretty
bad
mind
version
music
title
directed
minute
left
isn
lead
takes

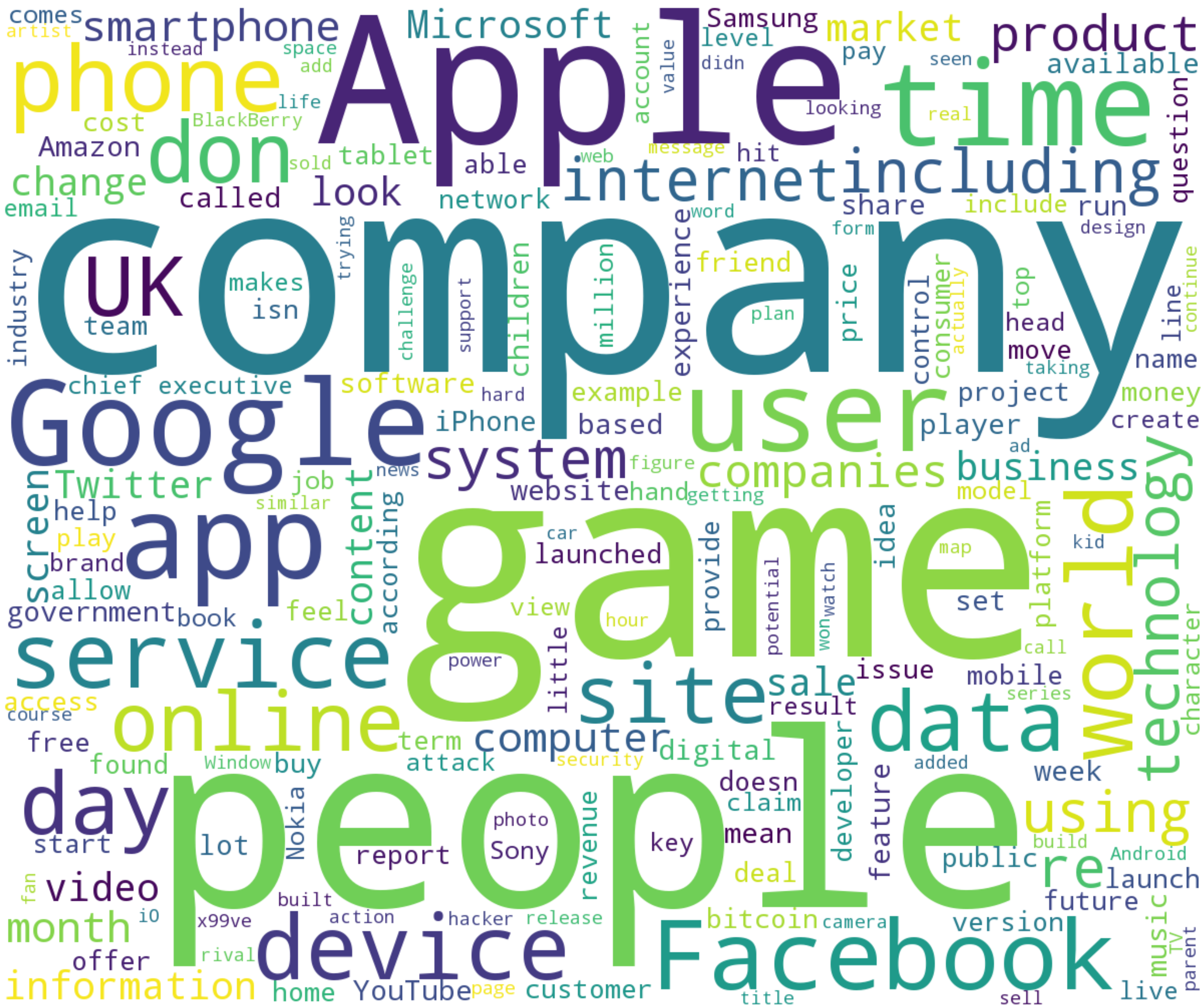
Football :


player
game
team
club
time
season
goal
Liverpool
Arsenal
World Cup
United
manager
Chelsea
play
Premier
League
England
football
striker
Manchester United
win
week
Fifa
Rooney
home
ball
match
fan
chance
people
squad
City
little
half
minute
summer
Barcelona
decision
day
move
left
don
moment
played
playing
start
level
lot
Tottenham
position
Champions League
tournament
Mourinho
midfielder
despite
defender
forward
hope
final
Van Gaal
country
performance
look
month
Brazil
goalkeeper
pitch
job
deal
Moyes
Germany
hard
looked
believe
following
Everton
term
result
Manchester City
penalty
lost
won
feel
Blatter
Spain
coach
added
challenge
Su xa1rez
signing
opportunity
France
five
West Ham
contract
change
Hodgson
talk
shot
front
mean
third
plan
victory
close
saying
lead
able
found
happy
role
charge
coming
sport
FA
life
late
expected
break
form
called
happen
bit
ahead
seen
Sunday
offer
opponent
Spur
stadium
x99ve
captain
injury
hour
bid
past
referee
scored
difficult
remain
supporter
pressure
line
including
looking
support
question
free kick
future
team mate
Southampton
Real
taking
corner
matches
involved
situation
competition
half time
career
x99m
pass
told
head
Wednesday
stand
didn
defeat
Saturday
re
draw
hand
January
maybe
set
campaign
score
help
cross
night
issue
international
confirmed
try
run
Wenger
sent
former
beat
started
midfield
leave
top
stay
stage
December
return
defence
course
continue
Sunderland

Politics :


Labour
party
people
government
time
prime minister
Cameron
Britain
Conservative
election
Miliband
campaign
UK
Corbyn
support
country
vote
week
Tories
Scotland
saying
voter
issue
change
MP
Lib Dem
Ukip
plan
day
David Cameron
Liberal Democrat
Labour party
Ed Miliband
decision
leader
Clegg
parliament
month
position
including
parties
told
help
seat
job
term
don
children
business
deal
minister
school
bill
little
course
Labour MP
local
tell
key
re
former
offer
life
matter
hope
lead
talk
left
despite
policies
Blair
feel
opposition
chancellor
real
stand
Farage
role
EU
increase
continue
Conservative party
risk
described
action
shadow cabinet
instead
view
event
letter
x99ve
programme
force
include
won
family
Common
Europe
past
claimed
current
home
Salmond
seen
economic
office
Scottish
Tory
speech
agreed
law
win
idea
remain
called
candidate
allow
policy
evidence
coalition
money
start
believe
question
expected
following
politic
set
run
report
referendum
comment
rule
live
message
future
suggested
warned
look
figure
cost
debate
politician
hard
able
held
meeting
taking
police
reform
power
call
moment
five
found
result
Osborne
London
political
Street
challenge
women
didn
NHS
actually
public
economy
immigration
Speaking
stop
British
lot
hand
poll
world
budget
Green
level
strong
added
claim
system
statement
wrong
attack
union
supporter
official
chance
act
move
mean
proposal
SNP
Westminster
Labour leader
trying
benefit
cut
pay

Technology :


comes
artist
smartphone
instead
Microsoft
Samsung
level
market
product
Apple
phone
time
space
add
pay
seen
available
life
value
account
didn
BlackBerry
looking
real
cost
question
Amazon
don
tablet
sold
web
message
hit
able
internet
including
change
email
called
look
network
share
include
run
word
form
design
company
trying
UK
makes
isn
team
industry
challenge
support
children
million
experience
friend
plan
price
control
consumer
actually
top
head
move
line
continue
taking
name
chief executive
software
hard
example
project
money
Google
user
iPhone
based
player
create
system
figure
ad
companies
business
Twitter
job
news
website
hand
getting
model
help
similar
screen
app
content
according
game
idea
technology
world
car
play
launched
map
kid
brand
allow
platform
government
book
feel
view
hour
provide
potential
watch
won
set
service
power
little
call
site
issue
mobile
sale
series
character
access
course
online
result
data
free
term
computer
digital
found
Window
buy
attack
security
developer
added
week
day
doesn
claim
photo
Nokia
revenue
feature
using
start
lot
report
Sony
key
mean
people
build
public
Android
re
fan
video
deal
launch
built
future
iO
x99ve
action
hacker
release
bitcoin
camera
version
month
music
TV
parent
offer
rival
device
Facebook
information
home
YouTube
page
customer
title
sell
live

Εύρεση όμοιων κειμένων

Το δεύτερο ζητούμενο είναι η εύρεση όμοιων ή πανομοιώτυπων κειμένων, χρησιμοποιώντας το αρχείο train.csv ως αρχείο κειμένων. Δηλαδή η έξοδος του αρχείου θα είναι δυάδες κειμένων όπου η ομοιότητα τους θα έιναι μεγαλύτερη από 0.7 μονάδες.

Για το σκοπό αυτό δημιουργήθηκε το script duplicates.py, το οποίο βρίσκεται κάτω από τον κατάλογο scripts του αρχείου που παραδώθηκε. Το output αυτού του script θα παραδωθεί σε ένα αρχείο csv κάτω από τον κατάλογο output.

Βιβλιοθήκες:

- Numpy
- Pandas
- Nltk
- Sklearn
- String

Βήματα:

1. Δημιουργία λιστών που περιέχουν τα κείμενα ανα κατηγορία. Όπως και στο πρώτο ερώτημα.
2. Lemmatization των κειμένων, δηλαδή αποκοπή των λέξεων στην ρίζα τους με σκοπό την αναγνώριση ως πανομοιότυπες λέξεις με την ίδια ρίζα.
3. Αφαίρεση της στίξης.
4. Δημιουργία λεξικού με τα κείμενα της κατηγορίας.
5. Vectorization των λέξων του κάθε κειμένου με σκοπό των υπολογισμό της απόστασης μεταξύ αυτών.
6. Υπολογισμός του cos_similarity_matrix με βλαση των πίνακα vector των κειμένων.
7. Αν κάποια εγγραφη του πίνακα αυτού είναι μεγαλύτερη το 0.7 τότε τα κείμενα με τα index αυτης της εγγραφής είναι παρόμοια.

Κώδικας:

```
LemVectorizer = CountVectorizer(tokenizer=LemNormalize, stop_words='english')
LemVectorizer.fit_transform(listBusiness)

tf_matrix = LemVectorizer.transform(listBusiness).toarray()

tfidfTran = TfidfTransformer(norm="l2")
tfidfTran.fit(tf_matrix)

tfidf_matrix = tfidfTran.transform(tf_matrix)

cos_similarity_matrix = (tfidf_matrix * tfidf_matrix.T).toarray()

x = 0
y = 0

for lis in  cos_similarity_matrix:
    y = 0
    for value in lis:
        if value > 0.7 and x != y:
            res = str(listBusinessId[x]) + "\t" + str(listBusinessId[y]) + "\t" + str(value) + "\n"
            csvfile.write(res)

        y = y + 1
    x = x + 1
    #end_for
#end_for
```

Δίνεται ως παράδειγμα εδώ η σύγκριση μεταξύ των κειμένων κατηγορίας business. Δεν έχει νόημα η σύγκριση μεταξύ κειμένων διαφορετικών κατηγοριών και επίσης η χρονική και χωρική πολυπλοκότητα θα αυξανόταν πολύ.

Output:

Document_ID1	Document_ID2	Similarity
1117	1118	0.7809318886558196
895	888	0.7381888695714799
888	895	0.7381888695714799
8369	1336	0.9999999999999999
7319	6843	0.7382893107266093
1785	1766	0.7350059406146175
1785	14183	0.7777424632822943
1785	1795	0.7750006210729188
8281	8282	0.9408195472553628
5954	9912	0.764674373089905
3484	3444	0.7424266704300777
3484	3471	0.7094625945286243
3484	3496	0.7363565855223003
3444	3484	0.7424266704300777
3444	3471	0.7736635652235201
3444	3496	0.7034543690467162
10098	10097	0.7117032547371459
14707	14702	0.9492140309690263
5340	5342	0.961111173966638
299	304	0.8512844007928716
876	867	0.804283642598485
149	169	0.7184073762424523
10815	10822	0.7434536806813467
3471	3484	0.7094625945286243
3471	3444	0.7736635652235201
14702	14707	0.9492140309690263
5342	5340	0.961111173966638
6932	6922	1.0000000000000002
13926	7885	0.8073064567156764

Classification

Σε αυτό το σημείο το ζητούμενο ήταν κατηγοριοποίηση των κειμένων που δίνονται στο train_set με συγκεκριμένους classifiers και με συγκεκριμένες τεχνικές vectorization.

Ζητούμενοι classifiers και τεχνικές :

- Random Forests (Bag of Words)
- Random Forests (SVD)
- Random Forests (Average Word Vector(W2V))
- SVM (Bag of Words)
- SVM (SVD)
- SVM (Average Word Vector(W2V))

Για το σκοπό αυτό δημιουργήθηκαν τα scripts randomforests.py, svm.py, word2vecrf.py, word2vecsvm.py, τα οποία βρίσκονται κάτω από τον κατάλογο scripts του αρχείου που παραδώθηκε. Το output αυτών των scripts θα παραδωθεί σε ένα αρχείο csv κάτω από τον κατάλογο output.

Random Forests Classifier:

Σύντομη περιγραφή του αλγορίθμου από την Wikipedia:

"Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set."

SVM Classifier:

"In machine learning, support-vector machines (SVMs, also support-vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces."

Για να μπορέσουμε να κάνουμε classify κείμενα, θα πρέπει με κάποιο τρόπο να μαθηματικοποιηθούν, δηλαδή να τα γίνουν διανύσματα, αυτό ζητείται να γίνει με τις παρακάτω τεχνικές:

Bag of Words:

"The bag-of-words model is a simplifying representation used in natural language processing and information retrieval (IR). In this model, a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity. The bag-of-words model has also been used for computer vision.

The bag-of-words model is commonly used in methods of document classification where the (frequency of) occurrence of each word is used as a feature for training a classifier."

Average Word Vector:

"Word2vec is a two-layer neural net that processes text. Its input is a text corpus and its output is a set of vectors: feature vectors for words in that corpus. While Word2vec is not a deep neural network, it turns text into a numerical form that deep nets can understand. Deeplearning4j implements a distributed form of Word2vec for Java and Scala, which works on Spark with GPUs."

Λίγα λόγια για τον SVD:

From Wikipedia:

In linear algebra, the **singular-value decomposition** (**SVD**) is a factorization of a real or complex matrix. It is the generalization of the eigendecomposition of a positive semidefinite normal matrix (for example, a symmetric matrix with positive eigenvalues) to any $m \times n$ matrix via an extension of the polar decomposition. It has many useful applications in signal processing and statistics.

Formally, the singular-value decomposition of an $m \times n$ real or complex matrix $\mathbf{M}$ is a factorization of the form $\mathbf{U\Sigma V^*}$, where $\mathbf{U}$ is an $m \times m$ real or complex unitary matrix, $\mathbf{\Sigma}$ is an $m \times n$ rectangular diagonal matrix with non-negative real numbers on the diagonal, and $\mathbf{V}$ is an $n \times n$ real or complex unitary matrix. The diagonal entries σ_i of $\mathbf{\Sigma}$ are known as the **singular values** of $\mathbf{M}$. The columns of $\mathbf{U}$ and the columns of $\mathbf{V}$ are called the **left-singular vectors** and **right-singular vectors** of $\mathbf{M}$, respectively.

The singular-value decomposition can be computed using the following observations:

- The left-singular vectors of $\mathbf{M}$ are a set of orthonormal eigenvectors of $\mathbf{MM^*}$.
- The right-singular vectors of $\mathbf{M}$ are a set of orthonormal eigenvectors of $\mathbf{M^*M}$.
- The non-zero singular values of $\mathbf{M}$ (found on the diagonal entries of $\mathbf{\Sigma}$) are the square roots of the non-zero eigenvalues of both $\mathbf{M^*M}$ and $\mathbf{MM^*}$.

Applications that employ the SVD include computing the pseudoinverse, least squares fitting of data, multivariable control, matrix approximation, and determining the rank, range and null space of a matrix.

Output:

Statistic Measure	SVM(BoW)	SVM(SVD)	SVM(W2V)	RF(BoW)	RF(SVD)	RF(W2V)
Accuracy	0.9420	0.9370	0.9480	0.8920	0.8940	0.9120
Precision	0.9410	0.9340	0.9430	0.8940	0.8950	0.9060
Recall	0.9340	0.9280	0.9430	0.8710	0.8770	0.9010

Beat the benchmark (Stemming + W2V + SVC)

Σε αυτό το σημείο ζητείται να ξεπεραστούν σε απόδοση οι παραπάνω αλγόριθμοι, με τη χρήση οποιουδήποτε αλγορίθομου και τεχνικής.

Το κλειδί σε αυτό το στάδιο είναι η προεπεξεργασία των δεδομένων κειμένου, ώστε να επιτευχθεί η ζητούμε απόδοση.

Η επιλογή ήταν η χρήση της τεχνικής stemming. Ένας απλός ορισμός του stemming είναι ο ακόλουθος:

«In linguistic morphology and information retrieval, stemming is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form—generally a written word form. The stem need not be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root. Algorithms for stemming have been studied in computer science since the 1960s. Many search engines treat words with the same stem as synonyms as a kind of query expansion, a process called conflation.»

Με λίγα λόγια είναι η μείωση της κάθε λέξης στη ρίζα της ώστε να θεωρείται ίδια λέξη. Για παράδειγμα οι λέξεις Playing και Playable, θα μειωθούν στην λέξη Play.

Αυτό έχει ως αποτέλεσμα να πετυχαίνονται καλύτεροι χρόνοι, καθώς και αποδόσεις στα μοντέλα που εκπαιδεύονται.

Η επόμενη τεχνική που επιλέχθηκε για να κάναι τα δεδομένα κειμένου να εκπαιδεύσουν ένα μοντέλο κατηγοριοποίησης ήταν η τεχνική Word2Vector που έχει περιγραφεί παραπάνω.

Τέλος ο classifier που επιλέχθηκε ήταν ο SVC με γραμμικό πυρήνα. Επιλέχθηκε διότι είχε γενικότερα καλύτερη απόδοση από τον RandomForests και πριν την εφραμογή του stemming.

Τα αποτελέσματα της εκτέλεσης είναι τα ακόλουθα:

Statistic Measure	My Method
Accuracy	0.955
Precision	0.951
Recall	0.951

Παρατηρείται ότι η μέθοδος που δημιουργήσαμε ξεπερνάει τις υπόλοιπες σε απόδοση.

Calssification test_set.csv

Τέλος με τη μέθοδο που δημιουργήθηκε πριν, κάνουμε κατηγοριοποίση το test_set. Τα αποτελέσματα που προέκυψαν υπάρχουν κάτω από τον κατάλογο output. Η μορφή τους είναι η εξής:

Document_ID	Category
2	Politics
10	Technology
25	Technology
28	Business
29	Business
33	Business
34	Business
37	Business
39	Technology
40	Technology
44	Business
56	Business
66	Technology
75	Business
77	Business
85	Business
89	Business
92	Technology
107	Technology