ΕΘΝΙΚΟ ΚΑΙ
ΚΑΠΟΔΙΣΤΡΙΑΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΑΘΗΝΩΝ

ΤΕΧΝΙΚΕΣ ΕΞΟΡΥΞΗΣ ΔΕΔΟΜΕΝΩΝ

Εφαρμογή τεχνικών εξόρυξης δεδομένων και αξιολόγηση

WordCloud, Clustering, Classification και Beat the Benchmark

Κοντόπουλος Παναγιώτης ΑΜ: 1115201100124

Τσακριλής Αλέξανδρος-Παναγιώτης

ΑΜ: 1115201100092

ΑΘΗΝΑ 2016

Περιεχόμενα

Δημιουργία WordCloud

Δομή Κώδικα (+ οδηγίες για εκτέλεση κώδικα)

Στον παρακάτω πίνακα φαίνεται η διάρθρωση της εργασίας σε αρχεία και φακέλους.

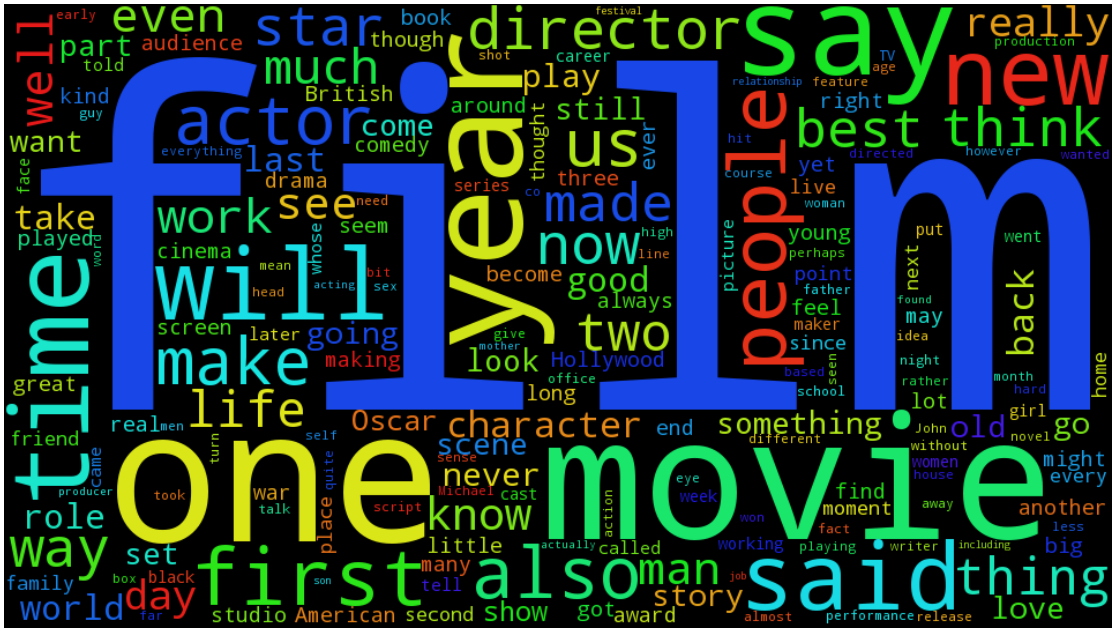
ΦΑΚΕΛΟΣ	ΑΡΧΕΙΟ	ΛΕΠΤΟΜΕΡΕΙΕΣ
./	data_csv_functions.py	Αποτελείται από το σύνολο των συναρτήσεων για την εισαγωγή και εξαγωγή δεδομένων από τα αρχεία csv.
./	data_wordcloud.py	Αποτελεί την υλοποίηση των συναρτήσεων για την παραγωγή του WordCloud.

Επίσης στον φάκελο data υπάρχουν τα αρχεία train_set.csv και test_set.csv.

<u>Οδηγίες για την εκτέλεση του προγράμματος</u>

Για την εκτέλεση στα μηχανήματα linux της σχολής ο χρήστης τρέχει την εντολή python data_wordcloud.py path_to_file. Ένα παράδειγμα χρήσης του είναι το ακόλουθο: python data_wordcloud.py ./data/train_set.csv

Αφού εκτελεστεί το πρόγραμμα, θα παραχθούν πέντε .png αρχεία στο φάκελο data και θα εμφανιστούν στην οθόνη, όπου το καθένα περιέχει το wordcloud μίας κατηγορίας.



Για την παραγωγή του wordcloud χρησιμοποιήθηκε η βιβλιοθήκη:
https://github.com/amueller/word_cloud

Υλοποίηση Συσταδοποίησης (Clustering)

Δομή Κώδικα (+ οδηγίες για εκτέλεση κώδικα)

Στον παρακάτω πίνακα φαίνεται η διάρθρωση της εργασίας σε αρχεία και φακέλους.

ΦΑΚΕΛΟΣ	ΑΡΧΕΙΟ	ΛΕΠΤΟΜΕΡΕΙΕΣ
./	data_csv_functions.py	Αποτελείται από το σύνολο των συναρτήσεων για την εισαγωγή και εξαγωγή δεδομένων από τα αρχεία csv.
./	data_clustering.py	Αποτελεί την υλοποίηση των συναρτήσεων για την εφαρμογή του

		clustering με τον αλγόριθμο K-Means.

Επίσης στον φάκελο data υπάρχουν τα αρχεία train_set.csv και test_set.csv.

Οδηγίες για την εκτέλεση του προγράμματος

Για την εκτέλεση στα μηχανήματα linux της σχολής ο χρήστης τρέχει την εντολή python data_clustering.py path_to_file. Ένα παράδειγμα χρήσης του είναι το ακόλουθο: python data_clustering.py ./data/train_set.csv

Αφού εκτελεστεί το πρόγραμμα, θα παραχθεί το αρχείο clustering_KMeans.csv στον φάκελο data, το οποίο περιέχει τα ποσοστά των δεδομένων κάθε κατηγορίας.

	Politics	**Football**	**Technology**	**Business**	**Film**
Cluster1	0.01	0.00	0.90	0.08	0.01
Cluster2	0.01	0.01	0.02	0.01	0.95
Cluster3	0.09	0.04	0.01	0.85	0.00
Cluster4	0.97	0.00	0.00	0.02	0.00
Cluster5	0.00	0.99	0.00	0.00	0.00

Για την υλοποίηση του αλγορίθμου χρησιμοποιήθηκε κώδικας από:

The Data Science Lab: https://datasciencelab.wordpress.com/2013/12/12/clustering-with-k-means-in-python/

Και για την Cosine Similarity από το Scikit-learn: http://scikit-learn.org/dev/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html

Υλοποίηση κατηγοριοποίησης (Classification)

Δομή Κώδικα (+ οδηγίες για εκτέλεση κώδικα)

Στον παρακάτω πίνακα φαίνεται η διάρθρωση της εργασίας σε αρχεία και φακέλους.

ΦΑΚΕΛΟΣ	ΑΡΧΕΙΟ	ΛΕΠΤΟΜΕΡΕΙΕΣ
./	data_classification.py	Αποτελεί την υλοποίηση των συναρτήσεων για την εφαρμογή του classification.
./	data_csv_functions.py	Αποτελείται από το σύνολο των συναρτήσεων για την εισαγωγή και εξαγωγή δεδομένων από τα αρχεία csv.

Επίσης στον φάκελο data υπάρχουν τα αρχεία train_set.csv και test_set.csv.

<u>Οδηγίες για την εκτέλεση του προγράμματος</u>

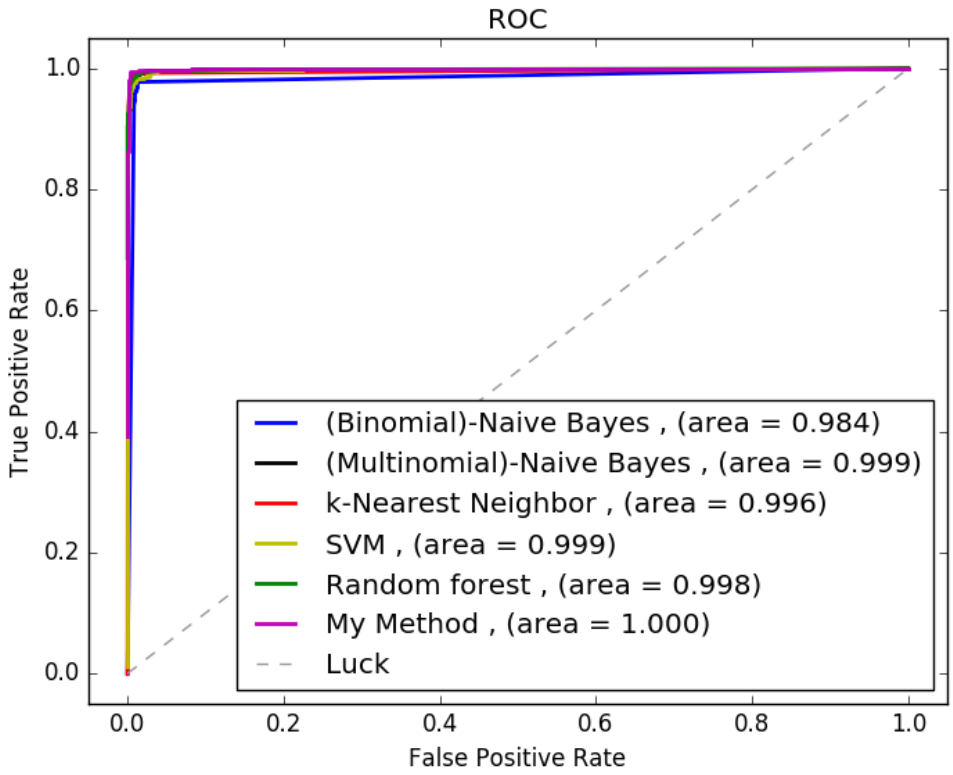
Για την εκτέλεση στα μηχανήματα linux της σχολής ο χρήστης τρέχει την εντολή python data_classification.py path_to_train_file path_to_test_file. Ένα παράδειγμα χρήσης του είναι το ακόλουθο: python data_classification.py ./data/train_set.csv ./data/test_set.csv

Αφού εκτελεστεί το πρόγραμμα, θα παραχθούν τα αρχεία στον φάκελο data:

- **EvaluationMetric_10fold.csv, το οποίο περιέχει τον πίνακα με τις ακρίβειες και τα αποτελέσματα του ROC AUC.**
- **roc_10fold.png, το οποίο περιέχει τα αποτελέσματα του ROC plot.**
- **testSet_categories.csv, το οποίο περιέχει τις κατηγορίες των άρθρων που περιέχονται στο Test Set.**

Δοκιμές

Στις δοκιμές μας χρησιμοποιήσαμε το 75% του train_set για train των αλγορίθμων και το υπόλοιπο 25% σαν test_set, ώστε να ελέγξουμε την απόδοση των αλγορίθμων και να βρούμε τις βέλτιστες ρυθμίσεις.



Από τις δοκιμές μας γίνεται εμφανές ότι η πρόβλεψη είναι καλή καθώς πλησιάζει το 1.0 και γενικά βρίσκεται πάνω από τη καμπύλη (διαγώνιος) της τύχης (Luck) 0.5.

Statistic Measure	K-Nearest-Neighbor	(Binomial)-Naive Bayes	SVM	(Multinomial)-Naive Bayes	Random Forest	My Method
Accuracy	0.952	0.943	0.934	0.958	0.949	0.965
ROC	0.996	0.984	0.999	0.999	0.998	1.0

Από τον πίνακα παρατηρούμε ότι ο (Multinomial)-Naive Bayes έχει την καλύτερη ακρίβεια. Αλλά η μέθοδος (My Method) που υλοποιεί τον SGDClassifier και αποτελεί το Beat the Benchmark μας δίνει καλύτερα αποτελέσματα και από την (Multinomial)-Naive Bayes, την οποία αναλύουμε στην επόμενη παράγραφο.

Για το 10fold cross-validation χρησιμοποιήσαμε την GreadSearchCV, η οποία παρόλο που προσπαθεί και βρίσκει τις καλύτερες παραμέτρους για το cross-validation προσφέρει παράλληλα αρκετές δυνατότητες και ευκολία ως προς το pipeline.

Μέθοδοι

- **NaiveBayes – Multinumial**: Σε αυτόν αλλάζαμε τιμές στο όρισμα alpha (The smoothing priors $\alpha \geq 0$ accounts for features not present in the learning samples and prevents zero probabilities in further computations. Setting $\alpha = 1$ is called Laplace smoothing, while $\alpha < 1$ is called Lidstone smoothing.), ώστε να μελετήσουμε τις διαφορές για τους περιορισμούς. Στο δικό μας train_set παρατηρήσαμε ότι για την τιμή 0.05 τα αποτελέσματα, εν συγκρίσει με την default τιμή 1, ήταν καλύτερα. Επομένως η περιοχή Lidstone smoothing από 0 ως 1, είχε τα βέλτιστα αποτελέσματα. (Για την παραγωγή του **testSet_categories.csv** χρησιμοποιήσαμε αυτό τον αλγόριθμο, επειδή μας έδωσε τα καλύτερα αποτελέσματα ακρίβειας, με εκπαίδευση στο 100% του train_set.csv.)
- **NaiveBayes – Binomial (Bernoulli)**: Σε αυτόν αλλάζαμε τιμές στο όρισμα alpha, όπως και πριν οι αλλαγές είναι παρόμοιες.
- **K-Nearest Neighbor**: Σε αυτόν αλλάζαμε το όρισμα K ώστε να επηρεάσουμε τον αριθμό των γειτόνων. Σε αυτή την περίπτωση μετά από δοκιμές η βέλτιστη τιμή στο set που είχαμε ήταν το 9 (Larger values of k generalize better, and smaller values may tend to overfit.). Στις τιμές 8 και 10 η ακρίβεια μειωνόταν, οπότε είχαμε κορυφή επίδοσης στη τιμή 9.
- **Support Vector Machines (SVM)**: Σε αυτή την περίπτωση επιλέξαμε την default τιμή με C=1.
- **Random Forest**: Σε αυτή την περίπτωση επιλέξαμε ως τιμή του ορίσματος n_estimators το 100, καθώς μετά από δοκιμές για τιμές πολύ κοντά στο >100 και <100 υπήρχαν ελάχιστες διαφορές προς το χειρότερο.

Beat the Benchmark

Στο beat the benchmark χρησιμοποιήσαμε τον SGDClassifier. Τα αποτελέσματα, όπως φαίνονται και στην παράγραφο Classification->Δοκιμές, είναι καλύτερα από τις υπόλοιπες μεθόδους, γιατί κάναμε preprocessing του dataset με τη βοήθεια της βιβλιοθήκης NLTK και μετά εκπαιδεύσαμε τον αλγόριθμο. Πιο συγκεκριμένα κατά το preprocessing ακολουθήσαμε την εξής διαδικασία:

- Μετατροπή όλων των upper letters -> lower letters
- Κάναμε tokenization στις λέξεις

- Τέλος χρησιμοποιήσαμε τον Lancaster stemmer, ώστε να μειώσουμε την περιττή πληροφορία

Η παράμετρος loss='modified_huber' που χρησιμοποιήσαμε στον SGDClassifier μετέτρεψε τον classifier σε linear method και μας έδωσε επέτρεψε να κάνουμε χρήση της predict_proba, που ήταν απαραίτητη για το ROC curve.

Επιπλέον δεν χρησιμοποιήσαμε την μέθοδο LSI γιατί έκοβε σημαντική πληροφορία από τα δεδομένα και μας μείωνε την επίδοση της ακρίβειας του αλγορίθμου.

Συμπεράσματα

Από τις δοκιμές που έγιναν καταλήξαμε στα εξής συμπεράσματα:

- Το train_set παίζει σημαντικό ρόλο στην εξέλιξη του αλγορίθμου ώστε να μπορέσει να ανταπεξέλθει στα επόμενα προβλήματα που έχει να αντιμετωπίσει, αλλά και το μέγεθός επηρεάζει τα αποτελέσματα. Για παράδειγμα όταν κάναμε εκπαίδευση των αλγορίθμων με ποσοστό του train_set 75% είχαμε (μικρές) διαφορές στα αποτελέσματα από όταν κάναμε εκπαίδευση με το 67%.
- Ο αλγόριθμος που χρησιμοποιείται σε κάθε σύνολο δεδομένων μπορεί να αλλάξει σημαντικά τα αποτελέσματα και τα ποσοστά επιτυχίας της ακρίβειας πρόβλεψης πάνω στο set που θα δοκιμαστεί.