****

**ΤΕΧΝΙΚΕΣ ΕΞΟΡΥΞΗΣ ΔΕΔΟΜΕΝΩΝ**

Εφαρμογή τεχνικών εξόρυξης δεδομένων και αξιολόγηση

Classification με χρήση του LDA

(**Latent Dirichlet Allocation**)

Κοντόπουλος Παναγιώτης ΑΜ: 1115201100124

Τσακριλής Αλέξανδρος-Παναγιώτης

ΑΜ: 1115201100092

ΑΘΗΝΑ 2016

# **Περιεχόμενα**

[Περιεχόμενα 1](#_Toc453024055)

[Υλοποίηση κατηγοριοποίησης (Classification) με LDA 2](#_Toc453024056)

[Δομή Κώδικα (+ οδηγίες για εκτέλεση κώδικα) 2](#_Toc453024057)

[Δοκιμές 3](#_Toc453024058)

[Συμπεράσματα 4](#_Toc453024059)

[Παραδοχές 5](#_Toc453024060)

# Υλοποίηση κατηγοριοποίησης (Classification) με LDA

## Δομή Κώδικα (+ οδηγίες για εκτέλεση κώδικα)

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

|  |  |  |
| --- | --- | --- |
| ΦΑΚΕΛΟΣ | ΑΡΧΕΙΟ | ΛΕΠΤΟΜΕΡΕΙΕΣ |
| ./ | data\_classification\_lda.py | Αποτελεί την υλοποίηση των συναρτήσεων για την εφαρμογή του classification με LDA και ex1 features. |
| ./ | data\_csv\_functions.py | Αποτελείται από το σύνολο των συναρτήσεων για την εισαγωγή και εξαγωγή δεδομένων από τα αρχεία csv. |
| ./ | data\_feature\_functions.py | Αποτελείται από τις συναρτήσεις για τα tokenizer, preprocessing και τη δημιουργία features |
| ./ | data\_predict.py | Αποτελείται από την συνάρτηση που χρησιμοποιείται για την πρόβλεψη των κατηγοριών |

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\_classification\_lda.py path\_to\_train\_file path\_to\_test\_file. Ένα παράδειγμα χρήσης του είναι το ακόλουθο: python data\_classification\_lda.py ./data/train\_set.csv ./data/test\_set.csv**

**Κατά την εκτέλεση ζητείται από το χρήστη να δώσει μία από τις παρακάτω επιλογές:**

* **1: για εκτέλεση μόνο με LDA features**
* **2: για εκτέλεση με LDA features + ex1 features**
* **3: για πρόβλεψη κατηγοριών**
* **0: για έξοδο από το πρόγραμμα**

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

* **EvaluationMetric\_10fold\_lda\_only.csv, το οποίο περιέχει τον πίνακα με τις ακρίβειες για το LDA μόνο. (για την επιλογή 1)**
* **EvaluationMetric\_10fold\_ex1\_features.csv, το οποίο περιέχει τον πίνακα με τις ακρίβειες για το LDA + ex1 features. (για την επιλογή 2)**
* **testSet\_categories.csv, το οποίο περιέχει τις κατηγορίες των άρθρων που περιέχονται στο Test Set. (για την επιλογή 3)**

## 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.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| EvaluationMetric\_10fold\_lda\_only.csv | | | | | | |
| Statistic Measure | K-Nearest-Neighbor | (Binomial)-Naive Bayes | SVM | (Multinomial)-Naive Bayes | Random Forest | My Method |
| Accuracy=10 | 0.916 | 0.733 | 0.909 | 0.897 | 0.922 | 0.903 |
| Accuracy=50 | 0.936 | 0.858 | 0.925 | 0.927 | 0.947 | 0.927 |
| Accuracy=100 | 0.928 | 0.878 | 0.904 | 0.928 | 0.945 | 0.924 |
| Accuracy=1000 | 0.898 | 0.907 | 0.249 | 0.916 | 0.930 | 0.918 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| EvaluationMetric\_10fold\_ex1\_features.csv | | | | | | |
| Statistic Measure | K-Nearest-Neighbor | (Binomial)-Naive Bayes | SVM | (Multinomial)-Naive Bayes | Random Forest | My Method |
| Accuracy=10 | 0.949 | 0.941 | 0.934 | 0.957 | 0.956 | 0.964 |
| Accuracy=50 | 0.947 | 0.942 | 0.925 | 0.958 | 0.956 | 0.965 |
| Accuracy=100 | 0.946 | 0.942 | 0.907 | 0.959 | 0.956 | 0.966 |
| Accuracy=1000 | 0.936 | 0.942 | 0.248 | 0.959 | 0.957 | 0.966 |

Για τα features χρησιμοποιήσαμε tokenization διαφορετικό του παραδείγματος, που δόθηκε, διότι για Κ=1000 είχε θέμα η μνήμη ακόμα και σε υπολογιστή με 8GB RAM.

# Συμπεράσματα

Από τις δοκιμές που έγιναν καταλήξαμε στα εξής συμπεράσματα:

Ερώτημα 1ο:

* Το LDA μόνο δεν δίνει γενικά καλά αποτελέσματα, σε σύγκριση με την πρώτη εργασία.
* Από την άλλη όσο μεγαλώνει το Κ έχουμε σχετική βελτίωση, αλλά όχι σε όλες τις περιπτώσεις. Σε ορισμένες έχουμε και μείωση των επιδόσεων.

Ερώτημα 2ο:

* Ο συνδυασμός του LDA με τα ex1 features προσφέρει ακόμα καλύτερη βελτίωση στην απόδοση, σε σύγκριση με το LDA features only.
* Και εδώ υπάρχουν περιπτώσεις, όπου το LDA δεν βελτιώνει, αλλά μπορεί να μειώσει την απόδοση.
* Μετά από δοκιμές και τους πίνακες παραπάνω, γίνεται εμφανές ότι το Κ=1000 με LDA features + ex1 features δίνει την καλύτερη απόδοση.

Ερώτημα 3ο:

* Λόγω τις καλύτερης απόδοσης παραπάνω για τη σύγκριση χρησιμοποιούμε το Κ=1000 με LDA features + ex1 features στην δικιά μας μέθοδο με τον SGDClassifier, η οποία έδωσε τα βέλτιστα αποτελέσματα από τις άλλες.

Γενικά:

* Ο αλγόριθμος SVC με kernel linear αποδίδει καλύτερα, από όταν χρησιμοποιήσουμε το kernel rbf. Μία πιθανή αιτιολόγηση για αυτό είναι ότι, όσο μεγαλώνει ο αριθμός των features με το training sample, ο linear kernel είναι αρκετά καλύτερος. Πηγή: <http://stackoverflow.com/questions/20566869/where-is-it-best-to-use-svm-with-linear-kernel> 07-06-2016

# Παραδοχές

* Στο φάκελο data περιέχονται και τα αρχεία που δείχνουν τη διαφορά του SVC linear kernel με τον SVC rbf kernel για Κ=1000 και στις δύο περιπτώσεις:
  + LDA features only
  + LDA features + ex1 features