Τεχνικές Εξόρυξης Δεδομένων - Εργασία 1η

Εκπονήθηκε από τους: sdi1500084 και sdi1500176

Ερώτημα 1:

Στον φάκελο Wodclouds υπάρχουν 3 αρχεία: wordclouds.py, stopwords.py, clouds dir. Το wordclouds.py φτιάχνει τα wordclouds, χρησιμοποιώντας ως stopwords το set στο stopwords.py. Αν τρέξει το wordclouds.py θα φτιαχτούν .png αρχεία ανά κατηγορία που περιέχουν τα wordclouds στον φάκελο clouds. Στο εκπονημένο αρχείο τα .png είναι έτοιμα, αλλά το πρόγραμμα τρέχει κάνοντας overwrite τα προηγούμενα .png

Ερώτημα 2, 3:

- Στην στατιστική μέτρηση 10 fold έγινε χρήση LSI με 100 components (Η επιλογή του αριθμού components έγινε μέσω πειραματισμού. Στον φάκελο Classifiers, υπάρχουν 2 αρχεία .png, τα οποία παρουσιάζουν ένα γράφημα accuracy/components, κρατώντας σταθερό classifier και αυξάνοντας τα components στο διάστημα [2-497]. Παρατηρούμε οτι στα 100 περίπου components σταθεροποιείται το accuracy και δεν αυξομειώνεται πλέον δραστικά. Το ένα γράφημα αφορά τον αλγόριθμο Naive Bayes και το άλλο τον KNearestNeighbours. Το αρχείο LSIaccuracytest.py είναι το αρχείο κώδικα, με το οποίο φτιάχτηκαν τα γραφήματα). Ο κύριος λόγος για τον οποίο επιλέξαμε να γίνεται το LSI είναι για να τρέχουν πιο γρήγορα οι αλγόριθμοι.

- Χρησιμοποιήθηκε η βιβλιοθήκη Kfold.

- Όλοι οι αλγόριθμοι τρέχουν κάνοντάς τους truncate τις διαστάσεις, εκτός απο τον Naive Bayes, επειδή το truncation παρήγαγε αρνητικές τιμές, με αποτέλεσμα να μην τρέχει. Προσπαθήσαμε κάνοντας χρήση του minimax scaler να αποκτήσουμε δείγμα χωρίς μηδενικές τιμές αλλά η απόδοση του Naive Bayes μειωνόταν σημαντικά, οπότε αποφασίσαμε να μην έχουμε truncation σε αυτόν.
- Ακολουθεί ο συμπληρωμένος πίνακας της εκφώνησης

Statistic Measure	Naive Bayes	Random Forest	SVM	KNN	My Method
Accuracy	0.95793	0.93649	0.95907	0.86645	0.96233
Precision	0.95793	0.93649	0.95907	0.86645	0.96233
Recall	0.95793	0.93649	0.95907	0.86645	0.96233
F - Measure	0.95793	0.93649	0.95907	0.86645	0.96233

- Το .csv αρχείο που περιέχει τα στοιχεία του πίνακα είναι το EvaluationMetric_10fold.csv και βρίσκεται στον φάκελο datasets. Το αρχείο αυτό κατασκευάζεται απο το αρχείο ./Classifiers/classifiers.py, το οποίο κάνει cross-validation για όλους τους classifiers και γράφει τα scores στο .csv
- Η επιλογή των parameters του SVM έγινε με χρήση του GridSearchCV και έγινε από το αρχείο gridSearch.py. Το param_grid φτιάχτηκε κάνοντας διαδικτυακή έρευνα.

- Ο αλγόριμος K-Nearest-Neighbours υλοποιήθηκε με υπόβαθρο τη δομή kd-tree. Οποιαδήποτε εξήγηση του κώδικα αναφέρεται στα σχόλια στα αρχεία knearest.py, kd_tree.py. **Θεωρούμε άξιο να αναφερθεί:** Έχουμε φτιάξει 2 υλοποιήσεις του αλγορίθμου με kd-tree που δηλώνονται δίνοντας τιμή στην bool μεταβλητή “balanced” του constructor του classifier. Η επιλογή αυτή αφορά στο χτίσιμο του δέντρου και στον υπολογισμό της ενδιάμεσης τιμής. Αν balanced=true, τότε το δέντρο θα είναι ισορροπημένο, αλλιώς μπορεί και να μην είναι. Ο τρόπος που χτίζεται το δέντρο είναι ο εξής: Για κάθε διάσταση των διανυσμάτων των δεδομένων, υπολογίζεται η ενδιάμεση τιμή της διάστασης αυτής. Έπειτα, τα δεδομένα σπάνε στα 2, έτσι ωστε δεδομένα με τιμή μικρότερη της ενδιάμεσης να πάνε αριστερά, και δεδομένα με τιμή **μεγαλύτερη ή ίση** της ενδιάμεσης να πάνε δεξιά. Αναδρομικά προχωράμε στις επόμενες διαστάσεις. 1 διάσταση = 1 επίπεδο. Σε περιπτώσεις που η ενδιάμεση τιμή είναι η ελάχιστη του δείγματος (συγκεκριμένα αν έχουμε πολλά μηδενικά, πχ σε περιπτώσεις μη truncation) τότε όλα τα δεδομένα πάνε στα δεξιά παιδιά, και έτσι έχουμε ένα μη ισορροπημένο δέντρο. Έτσι, όμως, μπορεί μεν η αποτελεσματικότητα του αλγορίθμου να είναι καλή, αλλά η απόδοση πέφτει σημαντικά. Για αυτό το λόγο φτιάξαμε τον “balanced” αλγόριθμο, ο οποίος σπάει τη λίστα των δεδομένων στο index στο οποίο βρίσκεται η ενδιάμεση τιμή. Έτσι, πάντα υπάρχουν αριστερά και δεξιά παιδιά, και έχουμε ενα ισορροπημένο δέντρο. Ο “balanced” αλγόριθμος είναι σαφέστατα πιο γρήγορος, αλλά έχει μικρότερη αποτελεσματικότητα, καθώς ενδέχεται κάποιοι κοντινοί γείτονες να πάνε σε λάθος κλαδί.
- Η υλοποίηση του αλγορίθμου έγινε εξ’ολοκλήρου από εμάς.

- Η υλοποίηση και ο υπολογισμός των μετρικών της ζητούμενης My Method βρίσκεται στο αρχείο ./Classifiers/mymethod.py με τη μορφή της συνάρτησης mymethod(), η οποία κάνει το cross-validation και επιστρέφει μια λίστα με scores. Τη συνάρτηση αυτή καλεί το αρχείο classifiers.py για την κατασκευή του πίνακα του ερωτήματος 2. Χρησιμοποιείται ο αλγόριθμος Naive Bayes, διότι βάσει του παραπάνω πίνακα είναι ο αποτελεσματικότερος και αποδοτικότερος αλγόριθμος. Η τελική προεπεξεργασία που κάναμε στα δεδομένα είναι η εξής:
  Titling: Αποφασίσαμε να δώσουμε σημασία στον τίτλο του κάθε document προσθέτοντας ν φορές τον τίτλο στο ίδιο το document. Το ν υπολογίζεται από τον τύπο (#λέξεων_doc / #λέξεων_τίτλου) / k. Το k υπολογίσθηκε έπειτα από πολλούς πειραματισμούς να είναι ο αριθμός 10.

Stopwords Removal: Αφαιρέσαμε από το κάθε document τα stopwords που γίνονται import από το stopwords.py (custom stopwords set)
Stemming: Χρησιμοποιήσαμε τον PorterStemmer του nltk.stem για να κάνουμε stem τις λέξεις. Το tokenization έγινε με τον word_tokenizer του nltk. Χρειάστηκε να κάνουμε το κείμενο decode σε utf-8 για να χρησιμοποιηθεί ο tokenizer
Δεν κάναμε **truncation,** καθώς, όπως εξηγήθηκε παραπάνω, ο Naive Bayes υπολειτουργεί. Δοκιμάσαμε να χρησιμοποιήσουμε **TFIDF-vectorizer,** αλλά ο **CountVectorizer** ήταν καλύτερος. Τέλος, δοκιμάσαμε να κάνουμε **Normalization,** με τον normalizer του scikit, αλλά πάλι η απόδοση έπεφτε.

- Η ίδια μέθοδος χρησιμοποιήθηκε και για την κατηγοριοποίηση του test_set. Τα αποτελέσματα αυτής βρίσκοντα στο αρχείο ./datasets/testSet_categories. Δεν παρουσιάζονται σε πίνακα στο παρόν αρχείο, καθώς ο πίνακας έχει 3000 γραμμές. Το πρόγραμμα που υλοποιεί τις προβλέψεις είναι το ./Classifiers/benchmark.py