Duta Teodor Ionut
Grupa 243

Documentatie

Modelul ales

In acest proiect am ales sa reprezint datele cu modelul bag-of-words. Am ales sa folosesc modelul bag-of-words pentru ca datele erau de tip text. Aceasta metoda de reprezentare a datelor se bazeaza pe frecventa de aparitie a cuvintelor. Am reprezentat modelul in fisierul Bag_of_words.py, am creat o clasa care contine o lista de cuvinte reprezentand vocabularul si o variabila care retine numarul de cuvinte al vocabularului. Clasa prezinta doua metode:

1. Build_vocabulary(self, data) care primeste setul de antrenare, parcurge fiecare linie a vectorului data, unde linia reprezinta o fraza, se aplica functia .split() care imparte fraza in cuvinte, se testeaza daca cuvantul nu este deja in vocabular si se adauga in cazul in care nu este deja prezent.
2. Get_features(self, data) unde se initializeaza cu 0, o matrice de dimensiune len(data) x vocabulary_len, se parcurge data cu ajutorul document_idx care retine numarul liniei si document care retine textul corespunzator liniei document_idx. Document se fragmenteaza in cuvinte cu functia .split(), daca cuvintele apartin vocabularului se creste cu 1 features [document_idx, idx], unde features[document_idx, idx] reprezinta numarul de aparitii al cuvantului cu id-ul idx in documentul document_idx. La sfarsit se returneaza features.

SVM (Support Vector Machines) classifier

Pentru implementarea acestui algoritm am folosit biblioteca ScikitLearn. Algoritmul implementat din ScikitLearn are o abordare one-vs-one ceea ce reprezinta ca pentru fiecare 2 clase este antrenat un clasificator binar care sa diferenteze intre acestea. Abordarea one-vs-one reprezinta faptul ca sunt antrenati (nr_clase)*(nr_clase-1)/2 clasificatori, cate unul corespunzator fiecarei perechi de doua clase. Eticheta finala pentru un nou exemplu va fi cea care are cele mai multe voturi pe baza acestor clasificatori. Am ales ca parametrii de initializare C=100, kernel=”linear”, am lucrat la fel ca in laboratorul 5 cu kernel=”linear” si am ales C=100 pentru a nu se ajunge la overfitting sau underfitting.

Normalizarea datelor

Normalizarea datelor a fost efectuata cu functia normalize_data(train_data, test_data). Am folosit package-ul preprocessing din sklearn si functia .Normalizer(norm=’l2’) din preprocessing.

Am ales sa folosesc norma “L2” datorita rezultatelor mai bune. Dupa normalizare scalam datele cu functia .transform(data). La sfarsit returnam scaled_train_data si scaled_test_data.

Incarcarea datelor

Am incarcat datele cu ajutorul functiei np.genfromtxt(), unde am pus ca delimitator un tab si am preluat datele pe 2 coloane, etichetele si textul.

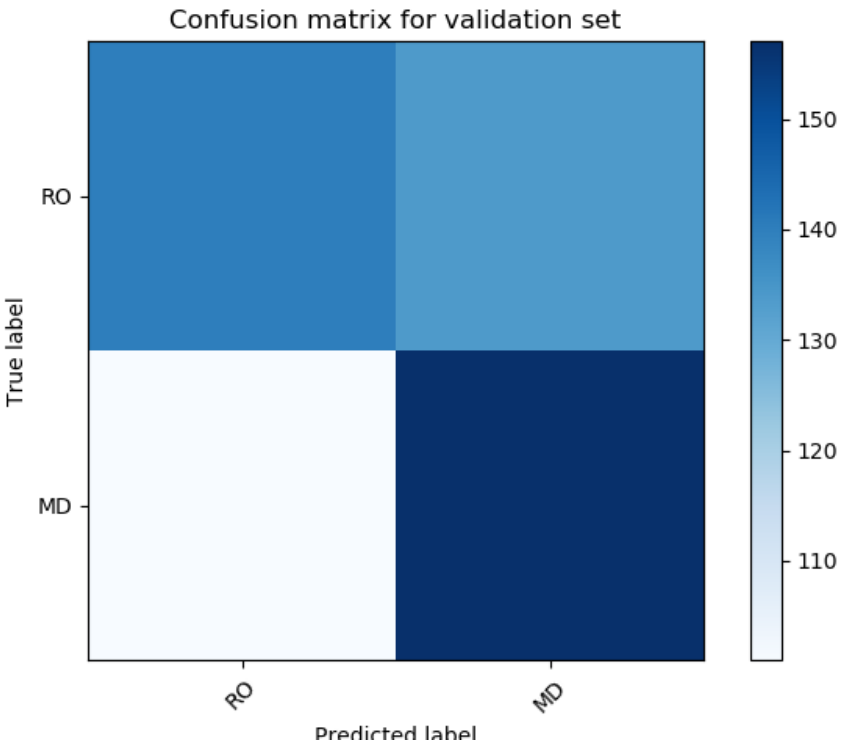
Train_samples reprezinta datele de antrenare, test_samples reprezinta datele de testare, etichetele au fost stocate in variabila etichete, train_labels reprezinta labelurile de antrenare.

Algoritmul

Este instantiat un obiect Model de tip Bag_of_words(), se apeleaza functia build_vocabulary, care primeste ca parametru setul de antrenare, doar 5000 de linii din acesta, deoarece mai tarziu apare o eroare referitoare la spatiul prea mare de alocare pentru array-ul din functia get_features din clasa Bag_of_words(). Probabil din cauza memoriei insuficiente RAM. Se apeleaza metoda get_features pe train_samples si test_samples, se normalizeaza si se scaleaza datele obtinute din get_features.

Initializez clasificatorul SVM, apelez functia .fit pe datele scalate impreuna cu train_labels . Folosesc functia .predicted pe clasificator dupa ce acesta a invatat si obtin predictiile, dupa care le scriu in fisierul csv. Acesta este algoritmul din fisierul ProiectML.py care a calculat predictiile trimise pentru competitie.

In fisierul Validation.py se foloseste acelasi algoritm, dar este rulat pe datele de validare pentru a obtine f1 score si confusion matrix. Se aplica functia train _test_split pe datele validation_samples si validation_labels pentru a avea date de antrenare si de testare, la fel si labeluri. Dupa calcularea predictiilor se foloseste functia f1_score din biblioteca sklearn.metrics care primeste labelurile de test si predictiile pentru a calcula F1 score corespunzator pentru setul de validare. Se foloseste si functia confusion_matrix din biblioteca sklearn.metrics pentru a calcula matricea de confuzie corespunzatoare pentru setul de validare primind ca parametrii labelurile de test si predictiile. La sfarsit se apeleaza functia plot_confusion_matrix care afiseaza matricea de confuzie ca un grafic, aceasta primeste matricea de confuzie, un titlu si lista dialectelor (RO, MD). In aceasta functie se folosesc diferite functii din biblioteca matplotlib.pyplot. Tot ce am spus mai sus despre Validation.py este scris in functia validation_doc(). Mai jos este reprezentata matricea de confuzie si este afisat F1 score.



Confusion Matrix:

[[140 134]

[101 157]]

F1 score:
0.5719489981785063