Draghioti Andreea-Maria
Grupa 262

# Translate Source Dialect Identification

The goal of the competition is to predict the native-dialect of a text based on its translation in different languages

Etapele solutiei:

- **importuri necesare:** biblioteci precum os, numpy, pandas, nltk, re; TfidTransformer si TfidVectorizer din sklearn.feature_extraction.text; train_test_split din sklearn.model_selection; svm din sklearn (pentru MultinomialNB am mai utilizat si MultinomialNB din libraria sklearn.naive_bayes)
- **citirea datelor:** prin intermediul modulului os citim intr-o prima etapa fisierul *train_data*, ulterior citim si fisierul *test_data*
- **codificarea etichetelor din string in int:** codificam etichetele / labels in valori cu numere intregi dela 0 la N, dupa care aplicati dictionarul label2id peste toate etichetele din train
- **preprocesarea datelor:** definim o functie *proceseaza* ce are ca rol modificarea textului pentru a creste performanta algoritmului prin urmatoarele moduri:
  - → elimina semnele de punctatie prin functia sub
  - → elimina stopwords prin libraria nltj.corpus
  - → elimina majusculele prin functia lower
  - → elimina cifrele pe care le-am stocat in prealabil in variabila pattern
  - → separa textul in cuvinte prin functiile strip si split
  - → returneaza la final textul ca un vector de cuvinte
- **aplicarea functiei de preprocesarea datelor pe intregul set de date:** utilizand un for aplicam functia *proceseaza* pe train_data rezultand un vector de cuvinte *processed_text*
- **implementarea TF-IDF:** TF-IDF este o metoda des utilizata in Natural Language Processing; aceasta masoara importanta unui cuvant intr-o anumita structura in raport cu o colectie de structuri prin intermediul unor

formule matematice; precum sugereaza si numele, TF-IDF puncteaza un cuvant prin inmultirea Term Frequency cu Inverse Document Frequency

- **impartirea setului de date in date train si test:** ne folosim de metoda train_test_split pentru a imparti datele intr-o ordine aleatoare avand 80% dintre acestea rezervate pentru train si 20% pentru test (la final vom avea , dintr-un total de 41570, 33256 exemple pentru train si 8314 exemple pentru test)
- **LinearSVC:** Linear Support Vector Classification; acest pas ne testeaza si antreneza algoritmul pe *processed_text* impartit in date de *train* si *test*

Intr-o prima faza aplicam pasii descrisi mai sus fisierului train_data, dupa care le aplicam si pe test_data pentru a testa modelul si pe un set de date nemaiintalnit si a genera submisiile necesare pentru a le incarca pe platforma Kaggle.

Submisia acestui model a inregistrat atat pe cele 40% de date din setul de date *test_data* , cat si pe partea de date privata un scor de 0.68434. Cea de-a doua submisie a fost realizata cu ajutorul clasificatorului Multinomial NB (acesta inregistrand o performanta mai redusa, avand scorul 0.54244).