Universitatea din București
Facultatea de Matematică și Informatică
Specializarea Calculatoare și Tehnologia Informației

Proiect la Inteligență Artificială

Profesor coordonator:
Alexe Bogdan

Student:
Vînaga Mădălina

1. Descrierea proiectului:

Scopul acestui proiect este de a identifica complexitatea unui cuvânt. Un cuvânt este considerat ca fiind complex dacă are eticheta cu valoarea 1 sau este considerat simplu (necomplex) dacă valoarea etichetei este 0.

Cuvintele au fost clasificate de către persoane care nu vorbesc nativ engleză și pentru fiecare exemplu din datele de antrenare și testare s-a făcut o medie a răspunsurilor, medie care a fost transformată in valori de 1 (complex) sau 0 (simplu) pentru a transforma această problemă în una de clasificare binară.

Datele acestui proiect sunt colectate din trei surse din domenii diferite (biblice, biomedicale și euro-parlamentare). Pentru datele de antrenare există 7662 de exemple, fiecare reprezentând o linie din tabel, având ca atribute id, corpus, sentence, token și complex (ultimul atribut reprezentând eticheta care arată dacă acel cuvânt este clasificat compelx sau simplu). Datele de testare au aceeași structură, cu excepția coloanei complex și sunt în număr de 1338.

Ca metodă de evaluare s-a folosit *balanced accuracy score*, metodă adecvată pentru a calcula acuratețea unui model de clasificare binară pe seturi de date nebalansate.

2. Codurile dezvoltate

2.1 Instalări și importuri

În această parte istalăm și importăm pachete (numpy, pandas, PyPhen) și funcții prestabilite (balanced_accuracy_score, confusion_matrix etc.)

Dale_chall reprezintă o listă ce conține cele mai frecvente cuvinte folosite în limba engleză, deci putem să presupunem că aceste cuvinte sunt știute de majoritatea persoanelor. De asemenea, în fișierul wordForms.xlsx se află mai multe forme ale unor cuvinte folosite în mod mai frecvent, date adunate de COCA (Corpus of Contemporany American English).

2.2 Funcții pentru caracteristici

- Words_forms() – verific apariția cuvântului în lista de cuvinte din COCA
- Corpus_feature() – atribui o valoare pentru fiecare corpus
- Length() – lungimea cuvântului
- Nr_vowels() – numărul de vocale ale cuvântului
- Nr_consonant() – numărul de consoane
- Is_title() – verifică dacă token-ul începe cu majusculă
- Is_dale_chall() – verifică apartenența cuvântului în lista DALE_CHALL
- Nr_syllables() – numărul de silabe aflat cu ajutorul pachetului PyPhen

→ De-a lungul testării acestui algoritm am folosit mai multe funcții precum numărul de hyponims și hypernyms (cu ajutorul pachetului wordnet), lungimea frazei pentru fiecare intrare, numărul de caractere speciale și de litere grecești în frază, frecvența cuvintelor din datele de train și test, numărul de înțelesuri ale cuvintelor (cu ajutorul funcției synsets), veificare dacă în fața cuvântului se află un cuvânt de legătură (of, from, at etc), normalizarea datelor. Aceste funcții nu au îmbunătățit eficența algoritmului.

2.3 Modelul folosit

- SVM (Supported Vector Machine)
    - Funcția importată și folosită din pachetul sklearn este *SCV()*
    - Parametrii și hiperparametrii
        - *Kernel* – am ales funcția *RBF (Radical Basis Function).*
        - *C* – parametru de regularizare, care reprezintă clasificarea greșită sau termenul de eroare, ce indică eroarea maximă suportată. Valoarea aleasă a fost de 7.
        - *Gamma* – folosit pentru kernel „rbf", „poly" și „sigmoid". Pentru valori mare ale lui *gamma*, punctele trebuie să fie foarte apropiate unele de altele pentru a fi considerate în același grup, deci au tendința de a face overfit. Valoarea aleasă a fost „*auto"* (= $\frac{1}{nr\_features}$).
        - *Class_weight* – valoarea aleasă a fost *„balanced"* ce folosește valorile lui y pentru a ajusta automat greutățile invers proporționale cu frecvențele de clasă.
    - Antrenare și verificare parametrii
        - O metodă folosită pentru a antrena și verifica parametrii aleși a fost funcția de cross_validation pentru 10 *Kfold* împreună cu *GridSearchCV* ce îmi testează modelul pentru mai multe combinații de hiperparametrii. Pe baza rezultatului salvat în *grid* am testat și local datele, împărțind datele de train în date de antrenare locale (80% din cele totale = 6129) și în date de validare (20% din cele totale = 1533).
        - Deoarece durata de testare cu *GridSearchCV* este destul de mare, de-a lungul testelor am păstrat ca opțiuni pentru C și gamma doar câteva valori pentru care am avut rezultate mai bune. Dacă doream să testez alte valori decât cele alese în urma *GridSearchCV*, antrenam local cu parametrii aleși și comparam cu rezultatele de pe antrenarea locală cu parametrii selectați de *Grid.*
        - Rezultatele în urma antrenării cu 10-fold validation pe toate datele de antrenare au fost un *balanced score* de 0.813589, iar matricea de confuzie asociată pentru toate datele de train este următoarea: $\begin{bmatrix} 4972 & 1940 \\ 32 & 718 \end{bmatrix}$.
    - Timpul de antrenare
        - Am folosit funcția *timer()* pentru a calcula timpul de antrenare și prezicere pentru datele noastre, find de ≈ 02.067114s.
    - Performanța pe Kaggle (Public Leaderboard)
        - Scorul obținut pe cele 40% de date din setul de test public de pe platforma Kaggle a fost: 0,82845.

- Naïve Baye

  - Funcția importată și folosită din pachetul sklearn este *GaussianNB()*

  - Parametrii și hiperparametrii
    - Aceștia au fost lăsați la valorile default și pentru a afla probabilitatea likelihood pentru fiecare clasă, am afișat atributul *class_prior_* (clasa 0 ≈ 0.90211433 și clasa 1≈ 0.09788567).

  - Antrenare și verificare locală
    - Pentru antrenare și verificare am folosit atât împărțirea datelor de antrenare în antrenare locală și verificare (în proporție de 80%-20%) cât și *cross_validation* pentru a compara *balanced_scorul* obținut local cu media scorului obținut în urma *cross_validation*-ului cu 10 Kfold.

  - Timpul de antrenare
    - Timpul de antrenare este de ≈ 0.004612s. Am folosit funcția *timer()* din pachetul *timeit.*

  - Performanța pe platforma Kaggle (Public Leaderboard)
    - Scorul obținut pe cele 40% de date din setul de test public de pe platforma Kaggle a fost: 0,79752.