UNIVERSITATEA DIN BUCUREȘTI

FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

DEPARTAMENTUL CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI

PROIECT

INTELIGENȚĂ ARTIFICIALĂ

COORDONATORI ȘTIINȚIFICI:

DIACONU ALEXANDRA

ALEXE DUMITRU BOGDAN

STUDENT:

MAXIM TIBERIU

BUCUREȘTI

2021

UNIVERSITATEA DIN BUCUREȘTI

FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

DEPARTAMENTUL CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI

PROIECT KAGGLE

COMPLEX WORD CLASSIFICATION

COORDONATORI ȘTIINȚIFICI:

DIACONU ALEXANDRA

ALEXE DUMITRU BOGDAN

STUDENT:

MAXIM TIBERIU

BUCUREȘTI

2021

CUPRINS

1. Descrierea Proiectului

Acest proiect se desfășoară sub forma unei competiții pe platforma Kaggle[1]. Proiectul constă în clasificarea cuvintelor în funcție de complexitatea acestora. Clasificarea cuvintelor (complex – eticheta 1; simplu – eticheta 0) a fost făcută de către persoane a căror limbă maternă nu este engleza. Există trei domenii din care au fost colectate aceste date: Biblie, articole biomedicale și dezbateri din Parlamentul European.

Există 7662 de exemple pentru datele de antrenare și 1338 de exemple pentru date de testare[2]. Fiind vorba de clasificarea binară pe seturi de date nebalansate, ca metodă de evaluare se va folosi *balanced accuracy score* $\left(balanced\ accuracy = \frac{sensitivity + specificity}{2}\right)$.

2. Biblioteci utilizate pentru implementarea proiectului

Pentru a implementa proiectul, am folosit o serie de biblioteci Pyhton. Câteva dintre ele au necesitat instalarea folosind comanda `! pip install <bibliotecă>`.

Bibliotecile utilizate sunt:

- biblioteci standard: numpy, pandas, os;
- timeit – pentru calcularea timpilor de executare;
- pyphen – pentru despărțirea în silabe a unui cuvânt;
- nltk – pentru caracteristici wordnet;
- openpyxl – pentru citirea/scrierea fișierelor .xlsx;
- gensim – pentru tokenizarea propozițiilor;
- Inflector – pentru transformarea cuvintelor de la forma plurală la cea singulară;
- dale_chall – pentru importarea listei DALE_CHALL;
- scikit-learn – pentru importarea clasificatorilor, a unor funcții modelatoare și a metricilor.

[1] https://www.kaggle.com/c/ub-fmi-cti-2021-2022
[2] https://www.kaggle.com/c/ub-fmi-cti-2021-2022/data

3. Descrierea caracteristicilor implementate

3.1. Generalități

Majoritatea caracteristicilor au discutate la orele de laborator și proiect. Au apărut mici modificări la unele dintre ele din motive de eficiență asupra rezultatelor obținute.

3.1. Caracteristici referitoare la corpus

- **corpus_feature()** – returnează domeniul din care provine cuvântul supus analizei.

3.2. Caracteristici referitoare la structura cuvântului supus analizei

- **word_length()** – returnează lungimea cuvântului supus analizei;
- **number_of_vowels()** – returnează numărul de vocale din cuvântul supus analizei;
- **number_of_consonants()** – returnează numărul de consoane din cuvântul supus analizei;
- **number_of_special_chars()** – returnează numărul de caractere speciale din cuvântul supus analizei;
- **number_of_syllables()** – returnează numărul de silabe ale cuvântului supus analizei;

  Pentru această caracteristică am avut nevoie de biblioteca pyphen și de funcția *inserted()* din aceasta. Am verificat eficiența acesteia făcând comparații cu funcția *hyphenate_word()* din biblioteca hyphenate și cu funcția *estimate()* din biblioteca syllables. Cele mai bune rezultate le-am obținut folosind biblioteca pyphen.
- **is_title()** – returnează valoarea 1, dacă cuvântul supus analizei este un titlu, iar în caz contrar, valoarea 0;
- **is_dale_chall()** – returnează valoarea 1, dacă cuvântul supus analizei se află în lista DALE_CHALL, iar în caz contrar, valoarea 0;

  Lista DALE_CHALL reprezintă o colecție de cuvinte pe care 80% din copii de clasa a 5-a o cunosc. Datorită faptului că această listă conține doar cuvinte la forma singulară, am adus cuvintele de la forma plurală la cea singulară folosind funcția *singularize()* din biblioteca Inflector.
- **is_word_forms() –** returnează valoarea 1, dacă cuvântul supus analizei se află în lista word_forms, iar în caz contrar, valoarea 0;

  Lista word_forms reprezintă o colecție de forme ale unor cuvinte din limba engleză.

- **is_stopword() –** returnează valoarea 1, dacă cuvântul supus analizei face parte din categoria cuvintelor de tip stopword, iar în caz contrar, valoarea 0;
- **get_word_frequency() –** returnează frecvența apariției cuvântului supus analizei în textele din datele de antrenare și testare.

  Pentru a genera această listă, am folosit funcția *tokenize()* din biblioteca gensim aplicată pe fiecare propoziție în parte. Am descoperit că această funcție este mult mai eficientă, comparativ cu funcția corespondentă din NLTK, deoarece aceasta nu adaugă în lista de cuvinte semnele de punctuație și are o acuratețe mai mare atunci când împarte cuvintele care folosesc semne precum <‘>, <->, etc.

3.3. Caracteristici bazate pe funcționalități wordnet

- **synsets() –** returnează numărul de synset-uri aferente cuvântului supus analizei;
- **hypernyms() –** returnează numărul de hipernime aferente cuvântului supus analizei;
- **hyponyms() –** returnează numărul de hiponime aferente cuvântului supus analizei;
- **meronyms() –** returnează numărul de meronime aferente cuvântului supus analizei;
- **holonyms() –** returnează numărul de holonime aferente cuvântului supus analizei.

4. Clasificatori

4.1. Clasificatorul Support Vector Machines (SVM)

4.1.1. Caracteristicile folosite

Caracteristicile folosite în acest model sunt cele descrise la secțiunea 3 din această documentație. Vectorul de caracteristici, în acest caz, a fost format din următoarele caracteristici: **corpus_feature(), word_length()**, **number_of_vowels()**, **number_of_consonants()**, **number_of_syllables(), is_title()**, **is_dale_chall()**, **is_word_forms()**, **get_word_frequency()** (cu mențiunea că în varianta aceasta de submisie se folosea o variantă veche a funcției, bazată pe *word_tokenize()* din NLTK) și **synsets()** . Acestea nu au fost normalizate, deoarece s-a constatat că nu este eficientă în problema dată.

4.1.2. Parametrii și hiperparametrii modelului

Modelul a fost construit pe baza clasificatorului SVM din librăria sckit-learn:

```
from sklearn.svm import SVC
```

- **kernel** – a fost folosită valoarea default (RBF: Radial Basis Function);
- **gamma** – valoarea acestuia este ‘auto’. Pot fi date diverse valori pentru gamma, însă, din testele efectuate, valoarea ‘auto’ a fost cea mai bună;
- **C** – valoarea parametrului de regularizare este 4;
- **class_weight** – cu valoarea ‘balanced’.

4.1.3. Antrenarea parametrilor și a hiperparametrilor

Pentru acest model, au fost două variante de antrenare și găsire a celor mai bune valori pentru parametrii/hiperparametrii amintiți anterior.

Una dintre variante presupunea generarea unui set de valori pentru fiecare element component în apelul clasificatorului și rularea unor instrucțiuni repetitive de tip *for* în care se antrena modelul pe datele de antrenare (20% din numărul total). Această metodă a fost ineficientă, mai ales din punct de vedere al timpului necesar aflării celor mai bune combinații de valori. Însă, cu ajutor acestei metode am reușit să reduc acest interval de valori la unul cât mai mic posibil.

A doua variantă, cea prezentă în script, folosește *GridSearchCV(),* prin care se găsește setul de valori cu care s-a obținut cel mai bun *balanced accuracy score*. Într-un final, doar valorile pentru **C** au mai rămas de testat, celelalte fiind adăugate în apelul clasificatorului SVM (**gamma**=’auto’, **kernel**=’rbf’, **class_weight**=’balanced’). Rulând această variantă pentru câteva valori ale lui **C**, a rezultat că valoarea potrivită ar fi 4.

4.1.4. Timpul de antrenare

Timpul mediu de antrenare rezultat în urma a 100 de apeluri la antrenare de către clasificator este de aproximativ **1.621495 secunde**.

4.1.5. Performanță pe Kaggle

Submisia aferentă acestui model a înregistrat, pe cele 40% de date din setul de date de test public, un scor de **0.82185**. Pe setul de date de test privat, s-a obținut un scor de **0.82808**.

4.1.6. 10-Folds Cross Validation

Valoarea medie obținută în urma acestei proceduri este de **0.8120430050393163**.

4.1.7. Matricea de confuzie

Pentru a afișa matricea de confuzie aferentă modelului, am folosit *cross_val_predict()* pentru a genera predicțiile. Aceasta se prezintă sub următoarea formă:

$$\begin{bmatrix} 5052 & 1860 \\ 80 & 670 \end{bmatrix}$$

4.2. Clasificatorul Naïve-Bayes

4.2.1. Caracteristicile folosite

Caracteristicile folosite în acest model sunt cele descrise la secțiunea 3 din această documentație. Vectorul de caracteristici, în acest caz, a fost format din toate cele 16 elemente. Acestea nu au fost normalizate, deoarece s-a constatat că nu este eficientă în problema dată.

4.2.2. Parametrii și hiperparametrii modelului

Modelul a fost construit pe baza clasificatorului GaussianNB din librăria sckit-learn:

```
from sklearn.naive_bayes import GaussianNB
```

Acest clasificator nu a fost apelat cu valori custom, ci sunt calculate automat de către acesta. Valoarea probabilităților likelihood se poate obține prin intermediul atributului **class_prior_** aferent clasificatorului definit. Valoarea poate fi accesată doar după ce modelul este antrenat.

4.2.3. Antrenarea parametrilor și a hiperparametrilor

Nu a fost necesară o antrenare, în cazul acestui model.

4.2.4. Timpul de antrenare

Timpul mediu de antrenare rezultat în urma a 1000 de apeluri la antrenare de către clasificator este de aproximativ **0.002907 secunde**.

4.2.5. Performanță pe Kaggle

Submisia aferentă acestui model a înregistrat, pe cele 40% de date din setul de date de test public, un scor de **0.80546**. Pe setul de date de test privat, s-a obținut un scor de **0.78150**.

4.2.6. 10-Folds Cross Validation

Valoarea medie obținută în urma acestei proceduri este de **0.7953526001038833**.

4.2.7. Matricea de confuzie

Pentru a afișa matricea de confuzie aferentă modelului, am folosit *cross_val_predict()* pentru a genera predicțiile. Aceasta se prezintă sub următoarea formă:

$$\begin{bmatrix} 4503 & 2409 \\ 46 & 704 \end{bmatrix}$$

5. Concluzii

Pe baza rezultatelor obținute de la cele două modele, se poate observa faptul că numărul de caracteristici și gradul de importanță a acestora sunt definitorii pentru un scor cât mai bun. De asemenea, valorile parametrilor/hiperparametrilor trebuie să fie calculate cu mare atenție pentru că influențează foarte mult scorul final.