Proiect Inteligența Artificială

Bahrim Dragoș

November 22, 2021

Cuprins

1 Cerință

Enunt "Predicting which words are considered hard to understand for a given target population is a vital step in many Natural Language Processing applications such as text simplification. A system could simplify texts for second language learners, native speakers with low literacy levels, and people with reading disabilities. This task is commonly referred to as Complex Word Identification. Usually, this task is approached as a binary classification task in which systems predict a complexity value (complex vs. non-complex) for a set of target words in a text. In this challenge, the task is to predict the lexical complexity of a word in a sentence. A word which is considered to be complex has label 1, a word is considered to be simple (non-complex) has label 0."[1]

Date "The data comes from three sources: biblical text, biomedical articles and proceedings of the European Parliament. These sources were selected as they contain a natural mixture of common language and difficult to understand expressions, whilst each containing vastly different domain-specific vocabulary. The training data consists of 7662 training examples, each training example is a row of the form (id, corpus, sentence, token, complex). The testing data consists of 1338 test examples, each test example is a row iof the form (id, corpus, sentence, token). Notice that you have to infer the label 0 (the token is not complex) or 1 (the token is complex)."[2]

Platforma Proiectul se desfasoara pe platforma Kaggle

2 Dezvoltare cod

Structura codului este cea preluata de la laborator iar cateva caracteristici sunt utilizate deoarece au fost dezvoltate la laborator

2.1 Caracteristici

Număr de silabe Inițial am folosit un regex ce separă dupa grupuri de vocale însă am rămas la folosirea unui pachet numit PyPhen

Aparține listei Dale Chall Dale Chall este o listă de cuvinte pe care 80% din elevii din Grade 5 le cunosc, am alterat aceasta functie prin alterarea cuvântului daca este la plural folosind WordNetLemmatizer din NLTK.

Lungimea cuvântului

Numărul de vocale

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

Numărul de consoane

Titlu Verificarea dacă numărul este titlu, adică dacă începe cu majusculă

Abreviere Verificarea dacă cuvântul este format doar din caractere cu majusculă

Caractere repetate Dacă cuvântul conține caractere care se repetă

Term Frequency–Inverse Document Frequency O metodă de a măsura importanța unui cuvânt într-o colecție. Valoarea crește proporțional cu numărul de apariții ale cuvantului si este balansat de numărul de documente în care se gasește. Am folosit implementarea din SKLearn

Caracteristici din WordNet

- Numărul de sinonime
- Dacă se găsește o definiție pentru cuvant (inițial era lungimea definiției)
- Dacă exista exemple în care cuvantul este folosit (initial era numarul de exemple)

Poziția token-ului in propozitie

Propozitie complexa Dacă propoziția conține elemente non ASCII, corpusul find in limba engleza, ne putem aștepta ca toate cuvintele sa aiba doar caractere standard.

Complexitatea medie a propozitiei Aplică funcția ce stabilește caracteristicile token-ului pe fiecare cuvant din propozitie si stabilim o medie

Numărul de cuvinte unice

Tipul de corpus din care provine cuvantul

Pronunția Numărul de grupuri ce se pronunță atunci când spunem cuvântul, am folosit pachetul "pronouncing"

2.2 Testare caracteristici si stabilire parametrii

Calcularea acurateței am facut-o folosind balanced accuracy score din SK Learn ce calculeaza media aritmetica dintre recall(sensitivitate) si rata de negativitate adevarata $balanced\_accuracy = \frac{sensitivitate + specificitate}{2}$

Inițial am folosit o funcție de generare a datelor proprite generate_data(percentage) ce avea rolul de a genera la intamplare o lista de indecsi cu un anumit procent din fiecare tip de corpus pe care îl avem și generam un set de date de antrenare si de testare. Pentru a testa un algoritm și un set de caracteristici stabileam un număr de iterații, unde la fiecare iterație generam un set nou de date de antrenare și validare pe care aplicăm algoritmul și calculăm balanced accuracy score și la finalul iteratiilor calculăm media tuturor rezultatelor. Ulterior am folosit cross validation.

După folosirea funcției proprii as obține un rezultat de tipul acesta pentru un KNN unde primul număr este numărul de vecini iar al doilea media scorurilor obținute, cu o astfel de listă pot să decid ce caractersitici, parametrii și hiper-parametrii sa folosesc.

- 1 0.668257413183145
- 2 0.6126109670655668
- 3 0.6774073128977224
- 4 0.6339482434381662
- 5 0.6680421902150359
- 6 0.6356128388286275
- 7 0.6540611829398212
- 8 0.6241868282579967
- 9 0.6428127641844336
- 10 0.6171953886598779
- 11 0.6298598263970768
- 12 0.618206900202975
- 13 0.6238640555630808
- 14 0.6136176874161184

2.3 Modele

2.3.1 Cei mai apropiați K vecini

K-Nearest-Neighbours(KNN) este un model ce clasifica datele de test în funcție de distanța calculata fata de datele de antrenare iar label-ul este luat la fel ca cel al celor k cei mai apropiati vecini. Exista diferite moduri în care putem sa rezolvăm egalitățile, in SK Learn, algoritmul folosit la proiect se ia prima valoarea intalnita din vecini.

Caracteristici folosite

- Tipul de corpus
- Numărul de Synsets
- Prezinta definiție
- Prezintă exemple
- Numărul de cuvinte unice
- Poziția în propoziție
- Numărul de silabe
- Daca este găsit in Dale Chall
- Numărul de vocale
- Lungimea cuvântului
- Este titlu
- Este abreviere
- Prezintă caractere care se repetă
- Numărul de consoane

Parametrii si hiperparametrii Modelul KNN nu are parametrii. Hiperparametrii sunt numărul de vecini luați în considerare în clasificare, în cazul meu dupa testare în diferite moduri am observat ca acuratețea cea mai bună o obțin atunci când iau 3 vecini.

Cum antrenez hiperparametrii Creez o bucla for ce ia in considerare mai mulți vecini si aplic fie cross validate fie procedura creata de mine si la submisie folosesc hiperparametrii ce obțin acuratețea cea mai buna

Cross validate

- 1 0.5846458931095925
- 3 0.5879787510212504
- 5 0.5951311466724666
- 7 0.5962992925837006

Cross validate cu Scale

- 1 0.5998953946836425
- 3 0.6193787354062276
- 5 0.6180504588307136
- 7 0.620784033081541

Durație de antrenare Antrenarea cu caracteristicile specificate și cu 3 vecini pe setul de date de antrenare dureaza 2.922588 secunde, însă datorită cache-ului Python prima antrenare durează in jur de 06.009618 secunde.

Cross Validate Cross Validate este o metoda ce împarte setul de date de antrenare in n parti, antrenează pe n-1 parti și testează pe partea rămasă, eroarea finală se obține ca media erorilor. Am folosit metoda din SK Learn și am obținut date pentru fiecare split, cât durează antrenarea, testarea si acuratețea, în cazul meu pentru 3 vecini folosind scale.

Split	Fit Time	Score Time	Test Score
1	0.003632068634033203	0.08950114250183105	0.5059441233140656
2	0.003999233245849609	0.09296703338623047	0.49366088631984584
3	0.002999544143676758	0.10604405403137207	0.5760154365653642
4	0.0029883384704589844	0.10300064086914062	0.5442836468885673
5	0.0029990673065185547	0.11272335052490234	0.6981379643029426
6	0.0019996166229248047	0.12099814414978027	0.6387554269175109
7	0.0039997100830078125	0.11499905586242676	0.7947226242161118
8	0.0029985904693603516	0.12003874778747559	0.7911625663289918
9	0.0039632320404052734	0.12251639366149902	0.5981379643029425
10	0.002518177032470703	0.11400032043457031	0.5529667149059334

Obținem eroarea medie 0.6193787354062276.

Performanță Pe niste date de test generate automat si separat de cele de antrenare din 20% din setul total de date obținem o acuratețe balansată de 0.6789562891242662 și matricea de confuzie asociată este:

1338	61
79	53

Interpretarea matricei de confuzie ne arată ca 1338 de cuvinte non complexe au fost clasificare ca find non-complexe(true positive), 61 de cuvinte non complexe au fost clasificate ca fiind complexe(false negative), 79 de elemente complexe au fost clasificate ca non complexe(false positive), 53 de elemente complexe au fost clasificate ca find complexe (true negatives).

Pe Kaggle avem fără Scale pe Public 0.56556 si pe Private 0.59246 iar cu Scale pe Public 0.59350 si pe Private 0.64657.

Observam ca o data cu standardizarea datelor avem o acuratețe mai bună pe setul de date complet.

2.3.2 Mașini cu vectori suport

Support Vector Machines(SVMs) este un model ce încearcă să clasifice datele separându-le folosind un hiperplan maximal. Din definiție este un clasificator binar însă se poate aplica și pentru mai multe clase folosind metodele One vs All (învățarea unui clasificator împotriva tuturor celorlalte clase) One vs One(învățarea unui clasificator pentru fiecare alta clasa). Uneori insa elementele nu sunt linear separabile, în acest caz identificăm soluții precum folosirea unei funcții Kernel ce mapează datele într-un plan mai mare decât cel inițial în care sunt separabile, sau crearea de margini soft prin care permitem algoritmului să clasifice greșit anumite elemente.

Caracteristici folosite

- Tipul de corpus
- Numărul de Synsets
- Poziția în propoziție
- Daca este găsit in Dale Chall
- Numărul de vocale
- Lungimea cuvântului
- Este titlu
- Numărul de consoane

Parametrii si hiperparametrii SVMs are ca parametrii weights si bias ce definesc hiperplanul de separare. Implicit clasificatorul din SK Learn are weigths-urile setate pe 'balanced' si eu am folosit aceasi setare. Funcția kernel folosită este radial basis ce are ca parametrii un gamma ce implicit este setat 'auto', cand am testat caracteristicele l-am luat în calcul însă am decis sa rămână în final pe 'auto'. Hiperparametrul modelului este C ce controlează trade-off-ul dintre margine si acuratețe.

Cum antrenez hiperparametrii La fel ca la K-NN am antrenat parametrii și hiperparametrii apeland o bucla for în care schimbăm valoarea lui C, in bucla for am fie un cross-validate si calculăm eroare în urma lui, fie functia mea de generare de date apelată de cateva ori și eroarea finală fiind considerată media valorilor obținute. De asemenea am încercat alternarea printre celelalte funcții kernel și parametrilor lor. Primul număr este valoarea lui C iar al doilea este acuratețea calculata de cross validate.

Cross validate

- 0.01 0.70875520105736
- 0.1 0.7139451466557362
- 1 0.6995767980280457
- 3 0.6896426697227497
- 5 0.6762009667372132
- 10 0.6736878529064856

Cross validate cu Scale

- 0.01 0.5980818129599111
- 0.1 0.665018587453887
- 1 0.6553484589924408
- 3 0.661095081546668
- 5 0.6512908521062156
- 10 0.6332385682697161

Durație de antrenare Antrenarea cu caracteristicile specificate și cu C = 3, funcția kernel rbf, gamma auto și weights balansat durează 02.490917 secunde, însă la prima rulare este 04.983216 secunde.

Cross Validate La fel ca la K-NN am folosit metoda din pachetul SK Learn și am obținut date pentru fiecare split, cât durează antrenarea, testarea și acuratețea split-ului, în cazul meu pentru C = 3, funcția kernel rbf, gamma = auto și weights balansat obțin folosind StandardScale printr-un pipe obțin.

Split	Fit Time	Score Time	Test Score
1	1.2505707740783691	0.27596092224121094	0.7421965317919075
2	1.1491479873657227	0.24999690055847168	0.6574277456647399
3	1.1926369667053223	0.22300124168395996	0.574278822961891
4	1.1621947288513184	0.23200225830078125	0.6265412445730825
5	1.205249547958374	0.24204659461975098	0.7759575494452484
6	1.1470437049865723	0.23896241188049316	0.7187843704775687
7	1.1749987602233887	0.2440502643585205	0.7957935359382537
8	1.2814092636108398	0.2579505443572998	0.9089339122045346
9	1.1769964694976807	0.2779994010925293	0.6887795465508925
10	1.1594951152801514	0.2291882038116455	0.5270718765074771

Obținem eroarea medie 0.7015765136115595.

Performanță Pe niste date de test generate automat si separat de cele de antrenare din 20% din setul total de date obținem o acuratețe balansată de 0.7503489835240454 și matricea de confuzie asociată este:

966	418
29	118

Interpretarea matricei de confuzie ne arată ca 966 de cuvinte non complexe au fost clasificare ca find non-complexe(true positive), 418 de cuvinte non complexe au fost clasificate ca fiind complexe(false negative), 29 de elemente complexe au fost clasificate ca non complexe(false positive), 118 de elemente complexe au fost clasificate ca find complexe (true negatives).

Pe Kaggle avem fără Scale pe Public 0.77216 și pe Private 0.76369 iar cu Scale pe Public 0.78773 si pe Private 0.77808.

Observam ca o data cu standardizarea datelor avem o acuratețe mai bună pe setul de date complet.

3 Concluzie

În urma proiectului am observat că este foarte importantă combinația de caracteristici, nu este suficient sa existe un număr mare de caracteristici ci să fie căutate cele care au impactul cel mai bun, dacă sunt multe caracteristici atunci greutatea fiecăruia este mai mică ceea ce duce la clasificări eronate. Nu este suficient să ne luăm după clasamentul public întrucât acesta ne poate duce în eroare, chiar dacă am obținut note mai mici pe clasamentul public atunci când am folosit Standard Scale, pe cel privat s-a observat fie o creștere sau fie o creștere mult diminuată. Este important modul în care stocăm submisiile

precedente, întrucât nu am vrea sa pierdem submisii pe aceeași combinație de caracteristici și parametrii, de asemenea putem vedea și impactul unei noi submisii folosind 'diff' sau să numărăm elementele comune ca sa putem vedea ce fel de impact ar apărea.

Notă: Cele doua notebook-uri de documentatie prezinta metodele prezentate aici, submisia 1 reprezinta una din submisiile mele iar submisia 2 este modificarea script-ului din documentatia 2, adica SVM. Eu stiu ce am folosit in mare parte in submisie adica SVM cu C = 1 si cu niste caracteristici mai putine decat cele care sunt insa functiile de calculare de caracteristici s-au schimbat, am folosit PyPhen pentru silabe in loc de regex, am folosit WordNetLemmatizer in loc de splice pentru a calcula pluralul, in loc de lungimea definitiei am spus daca are sau nu, in loc de numarul de exemple am pus daca are exemplu sau nu, din motivele astea, codul s-a schimbat foarte mult si nu pot sa ajung la aceasi submisie intrucat combinatiile sunt foarte multe, am atasat o submisie care este mult mai buna decat cea a mea(obtine 0.78561 pe privat si 0.77639 pe public).