Stanescu Maria-Raluca

Grupa 232

Documentatie proiect Inteligenta artificiala

Alien Language Classification

30.05.2022

1.Cerinta proiectului

Proiectul consta in construirea unui model care sa clasifice corect un set de date constand in texte clasificate in 3 limbaje diferite : 1 – Bellerophon , 2 – Ymir , 3 – Nanook (multiclass text classification). Se dau date de antrenare constand in 15 000 samples si 5000 samples pentru validare.

2.Datele

Seturile de date constau in :

1. Datele de antrenare si etichetele lor : train_samples.txt , train_labels.txt
2. Datele de validare si etichetele lor : validation_samples.txt , validation_labels.txt
3. Datele de testare : test_samples.txt

Valorile din toate fisierele sunt separate prin TAB . Fisirele train_samples.txt si validation_samples.txt au aceeasi forma : ID + data sample. Fisirele train_labels.txt si validation_labels.txt contin ID + eticheta corespunzatoare.

3.Prelucrarea datelor

Preprocesarea datelor a fost efectuata cu ajutorul a 2 metode diferite : CountVectorizer si Tf – Idf.

3.1 CountVecotrization (Bag of Words)

Pentru implementarea conceptului Bag of Words am folosit CountVectorizer.

Prin Bag of Words numaram de cate ori apare un cuvant /combinatie de cuvinte intr-un text.

CountVectorizer foloseste anumiti parametri in mod default care nu sunt potriviti pentru limbajele cu care lucram. Astfel , am modificat urmatorii parametri :

1. Lowercase = False -> pentru a nu transforma toate literele in litere mici , din moment ce lucram cu limbje necunoscute si nu stim regulile limbajelor.

2. Ngram_range -> parametrul primeste in mod default valoarea (1,1) . Parametrul ajuta la intelegerea modelului despre context(functioneaza la nivel de cuvinte) . Am obtinut cele mai bune rezultate pentru ngram_range = (1,2) .
3. Tokenizer -> parametru care descrie cum se formeaza un token . Am impartit textul doar prin whitespace pentru a fi considerate token si cuvintele continand caractere speciale si pentru a nu fi ignorata punctuatia. Altfel toate caracterele speciale si semnele de punctuatie din date sunt ignorate in mod default.

Limitarile principale ale metodei Bag of words sunt reprezentate de faptul ca pentru samples mari cu multe token-uri diferite este generat un vector de dimensiuni mari . De asemenea , Bag of Words este limitant in ceea ce priveste contextul , iar pentru cele 3 limbaje nu stim exact cat de important este contextul pentru clasificare.

3.2 Term Frequency Inverse Document Frequency (TFIDF)

Term Frequency = numarul de aparitii ale unui cuvant in text / numarul total de cuvinte din text

Inverse document Frequency = log(numarul total de limbaje / numarul de limbaje care contin cuvantul).

TFIDF = Term Frequency * Inverse document Frequency

Ex : Daca un cuvant apare de multe ori in multe limbaje nu este atat de important. In schimb, daca apare de multe ori intr-un singur limbaj este important intrucat consideram ca sugereaza o regula pentru limbajul respectiv.

Folosind CountVectorizer , obtinem o matrice in care fiecare coloana este un token si fiecare rand un text. Fiecare celula contine numarul de aparitii ale tokenului in text.

Cu Tf-Idf obtinem o matrice in care fiecare celula contine valoarea tfidf corespunzatoare.

Fiind vorba de 3 clase , am considerat ca este potrivita alegerea greutatii cuvintelor in matrice ca fiind count / tfidf .

Dupa testare pe mai multe modele , incluzand MultinomialNaiveBayes , ComplementNaiveBayes , LogisticRegression , LogisticRegressianCV ,MLPClassifier , rezultatele au fost similare atat pentru prelucrarea prin CountVectorizer , cat si pentru prelucrarea cu TFIDF. Pentru modelul antrenat cu MultinomialNaiveBayes am ales sa folosesc CountVectorizer , pentru ca functioneaza mai bine in lucrul cu numere intregi , iar pentru cel antrenat cu LogisticRegressionCV , TFIDF .

4.Clasificatori

Clasificatorii incercati au fost MLPClassifier ,MultinomialNaiveBayes ,GaussianNaiveBayes ,ComplementNaiveBayes, LogisticRegression , LogisticRegressionCV.Modelul a fost initial antrenat pe datele de antrenare si testat pe datele de validare.

Rezultate initiale :

- MLPClassifier cu 1/2/3/4 hidden layers de dimensiuni diferite , max_iter intre 1 si 300 -> acuratete **68%** in competitie
- MultinomialNaiveBayes cu parametri default -> acuratete **68%**
- GaussianNaiveBayes -> acuratete **55%**

Rezultate dupa modificarea parametrilor , antrenare pe train data:

- ComplementNaiveBayes (alpha = 0.6) -> **74.5%**
  Acuratete CNB : **0.7452**
  Matrice acuratete :

  [[1502 312 186]
  [303 1081 116]
  [274 83 1143]]

- MultinomialNaiveBayes -> **74.2%**
  Acuratete MNB : **0.7428**
  Matrice acuratete :

  [[1521 309 170]
  [312 1092 96]
  [292 107 1101]]

Rezultate finale , antrenare pe train data :

- **LinearRegression CV** , solver = `newton-cg` , max_iter = 30 , cv = 10 -> **74.9%**
  Solver = `newton-cg` - > potrivit pentru mai multe clase , regularizare L2(Ridge)
  Max_iter = 30 -> Pentru max_iter < 30 , acuratetea a fost mai scazuta
  Cv = 10 -> default = 5-fold , dar am obtinut cele mai bune rezultate pentru date impartite in 10 grupuri.

  Matrice acuratete , LinearRegressionCV :

  [[1549 275 176]
  [311 1097 92]
  [308 92 1100]]

- MultinomialNaiveBayes , alpha = 0.3 -> **74.4%**
  Matrice acuratete , MNB :

[[1541 298 161]
 [320 1089   91]
 [304 106 1090]]

Desi diferentele nu sunt diferente mari in matricele de confuzie , rezultatele cele mai bune sunt obtinute pentru LinearRegressionCV().

Bibliografie :

1. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
2. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
3. https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.ComplementNB.html
4. https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html
5. https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html
6. https://www.analyticsvidhya.com/blog/2021/11/implementation-of-gaussian-naive-bayes-in-python-sklearn/