

Detectarea optimismului si pesimismului



Rusti Emilia Noemi 341

Farcasi George Octavian 333

Diaconescu Alexandra 344

Introducere

Am analizat sentimentele exprimate in postari de tip tweet, pe care le-am clasificat in trei categorii: pozitiv, neutru si negativ, folosind 2 metode:

- a. Algoritmul clasic: Naive Bayes
- b. Novel approach: BERTweet

Setul de date

- Set-ul de date folosit este Sentiment Analysis Dataset ([Shrivastava, 2021](#)), preluat de pe Kaggle. Datele au fost colectate automat din tweet-uri publice si etichetate manual, pe baza sentimentului exprimat
- Pentru antrenare s-au folosit 26k date (train.csv), iar pentru test aproximativ 3500 (test.csv)
- O intrare contine:
 - text – tweet-ul propriu-zis, in format text
 - sentiment – eticheta : pozitiv/ negativ/ neutru

Preprocesare

Pentru Naive Bayes:

- Eliminarea URL-urilor din tweet-uri.
- Eliminarea caracterelor repetitive si consecutive.
- Modificarea literelor mari în litere mici.
- Transformarea caracterelor speciale, cum ar fi semnele de punctuatie, într-un spatiu.
- Înlocuirea emoticoanelor si emoji-urilor cu descrierea lor textuala.
- Îndepartarea mentiunilor si a hashtag-urilor.
- Transformarea numerelor în cuvinte.
- Lematizarea cuvintelor pentru a le reduce la forma de baza.

Pentru BERTweet se adauga:

- Eliminarea duplicatelor
- Tokenizarea textelor, folosind vinai/bertweet-base
- Stratificare dupa etichetele de sentiment
- Construirea unui dataset compatibil cu PyTorch

```
48 ig = "i've i* ;* been :D :)))))@abc sick +++ for #yes123 the past few days?? and thus,----- my hair looks wierd. @asf:))) if ! 1124 i didn't have a ""hat"" on it would look... http://tinyurl.com/wnfaku"
49
50 iStringPreprocessing))
51 iStringPreprocessing))
52
53 :
54 ts[i])
55 neutralTweets[i], end='\n\n')
56
57
58
59
```

i've kiss kiss been happy laugh sick for yesone hundred and twenty three the past few days and thus my hair looks wierd laugh if one thousand one hundred and twenty four i didn't have a hat on

Naive Bayes

- Pentru fiecare categorie de sentiment, se construiește un dicționar care reține frecvența fiecărui cuvânt.
- Se calculează probabilitățile (și log-probabilitățile) fiecărui cuvânt într-o clasă, folosind smoothing Laplace pentru a evita probabilitățile zero pentru cuvintele nevazute.
- Pentru un tweet nou, se adună log-probabilitățile tuturor cuvintelor aparținând fiecărei clase, iar tweetul este clasificat în funcție de scorul maxim.

```
11
12 negativeTweetsWordsFrequency = getWordsFrequency(negativeTweets)
13 neutralTweetsWordsFrequency = getWordsFrequency(neutralTweets)
14 positiveTweetsWordsFrequency = getWordsFrequency(positiveTweets)
15
16 print("negative words freq:", len(negativeTweetsWordsFrequency))
17 print("neutral words freq:", len(neutralTweetsWordsFrequency))
18 print("positive words freq:", len(positiveTweetsWordsFrequency))
```

```
➔ negative words freq: 10438
neutral words freq: 13505
positive words freq: 10806
```

```
def predictSentiments(tweet):
    tokens = preprocess(tweet)
    scores = {}
    for typeOfSentiment in ['negative', 'neutral', 'positive']:
        scores[typeOfSentiment] = sentimentPriorLog[typeOfSentiment]
        for token in tokens:
            scores[typeOfSentiment] += logLikelihoods[typeOfSentiment].get(token, 0)
    return max(scores, key=scores.get), scores
```

Acuratetea obtinuta este the 65%

BERTweet

- Am ales vinai/bertweet-base, bazat pe BERT, dar antrenat special pentru limbajul informal de pe Twitter.
- Folosește tokenul [CLS] din last_hidden_state care rezuma tweet-ul.
- Se aplica Dropout pentru regularizare si prevenirea overfitting-ului.
- Linear Layer produce scoruri (logits) pentru cele 3 clase: negativ, neutru, pozitiv.
- Se foloseste CrossEntropyLoss pentru antrenare, pe durata a 3 epoci.

```
class TweetClassifier(nn.Module):
    def __init__(self, dropout_rate=0.3):
        super(TweetClassifier, self).__init__()
        self.backbone = AutoModel.from_pretrained("vinai/bertweet-base") # without c
        self.dropout = nn.Dropout(dropout_rate) # dropout layer to prevent overfitti
        self.classifier = nn.Linear(self.backbone.config.hidden_size, 3) #Linear lay

    def forward(self, input_ids, attention_mask):

        outputs = self.backbone(input_ids=input_ids, attention_mask=attention_mask)
        hidden_state = outputs.last_hidden_state
        pooled_output = hidden_state[:, 0]
        x = self.dropout(pooled_output)
        x = self.classifier(x)

        return x
```

Acuratetea obtinuta este the 80%

References

- [Attention Is All You Need](#), Vaswani, Ashish and Shazeer, Noam and Parmar, Niki and Uszkoreit, Jakob and Jones, Llion and Gomez, Aidan N and Kaiser, Lukasz and Polosukhin, Illia, 2017
- [BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding](#), Devlin, Jacob and Chang, Ming-Wei and Lee, Kenton and Toutanova, Kristina, 2019
- [BERTweet: A pre-trained language model for English Tweets](#), Nguyen, Dat Quoc and Vu, Thanh and Nguyen, Anh Tuan, 2020
- [XLNet: A Multilingual Language Model for Twitter](#), Barbieri, Francesco and Espinosa-Anke, Luis and Camacho-Collados, Jose, 2021
- [Sentiment Analysis Dataset](#), Abhinav Shrivastava, 2021
- [Codul pe github](#)

Multumim



Rusti Emilia Noemi 341

Farcasi George Octavian 333

Diaconescu Alexandra 344