Gender identification

Michal Abramov (ID. 301834297) Arie Kfir Hayne (ID. 205601172)

Submitted as a final project report for Natural Language Processing, IDC, 2019

1 Introduction

Our project is "Tweet Author Gender Classification" - based on a tweet, determine the gender of the author of the tweet. Originally we thought about user's accounts identification in several social media networks base on user's basic data and posts. We changed it because We were unable to obtain information from the various APIs.

1.1 Related Works

We started by searching for related articles on the same topic. We came across an article on "Gender Classification" that uses Machine Learning Models. In this article they used the same dataset we used in our project, and achieved 67% accuracy using RFC(Random Forest Classifier). In this related article they also classified tweets that had "brand" as gender the same an tweets that had "male" as gender(there is a link in Code References section).

2 Solution

2.1 General approach

Our general approach was trying to run different model with different data processing in order to find the one that gives us better results.

starting with some statistic - Most common words (the number is how many time this word appears in all the tweets) -

Top 10 words for Female: [('like', 451), ('get', 353), ('__', 333), ('one', 332), ('love', 315), ('day', 296), ('go', 278), ('people', 249), ('time', 241), ('know', 210)]

Top 10 words for Male: [('like', 352), ('get', 346), ('one', 267), ('time', 232), ('new', 216), ('love', 210), ('go', 205), ('people', 197), ('good', 186), ('day', 186)]

Top 10 words for Brand: [('weather', 2279), ('get', 1326), ('channel', 1169), ('15', 1166), ('updates', 1147), ('40', 727), ('39', 424), ('new', 244), ('amp', 191), ('us', 167)]

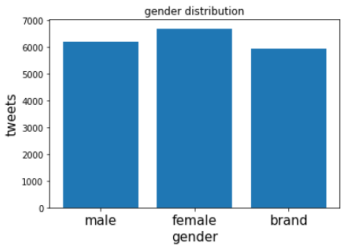

Figure 1: how many valid tweets we had in the data set per gender

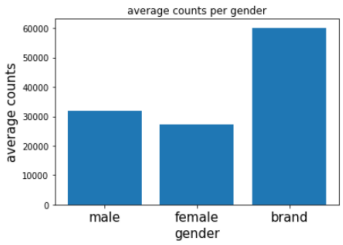

Figure 2: average tweet count per gender - we can see that brand values are almost doubled

2.2 Design

We used an open source project that implements Text Classification Models in Pytorch(there is a link in Code References section). We used pytorch and pytorchtext with SGD optimizer. We also used nltk for word lemmatization, word tokenize and stopwords, and GloVe for word representation with vectors. We chose 3 models and modified them to fit with the data processing configuration we wanted to test.

1. FastText - is a simple, and yet very powerful model for text classification, proposed by Facebook research. It gives comparable performance to much complex neural network based models.

2. Seq2Seq - models have been widely used in problems such as machine translation, document summarizing due to their capability to generate new sequence based on the already seen data. Here, we used Seq2Seq model for text classification task.

3. TextRNN - we use an model that were implemented a Bi-directional LSTM(Long short-term memory) network in PyTorch. LSTMs have been very popular for solving text classification problems due to their theoretical property to capture the entire context while representing a sentence.

2.3 Configurations

For each model we used two types of configurations for data processing:

- text_process_args_vars: determains the type of processing we are doing on the text fields -

    1. Keep_letters_only - removes all chars that are not english letters
    2. Lemmatize
    3. Remove_stopwords
    4. To_lower_case (we ended up running to_lower_case = True everytime)

- dataset_configs: This configuration refers to what fields to keep from the dataset.
- Gender and Text (tweet) are always used.
- Description and Tweet Count are not always used.
- DEBUG flag - this flag refers to debug log.

3 Experimental results

We ran each model with all 64 configuration variation (8 text processing configurations, 8 dataset configurations). Full results files can be found in the Code Reference section.

model	FastText	Seq2Seq	TextRNN
Final training accuracy	0.6270	0.5995	0.5648
Final validation accuracy	0.5826	0.6279	0.5623
Final test accuracy	0.5901	0.6001	0.5776

Figure 3: best results for each mode

in Figures 4 and 5 we can see the best run per model and per run type. each model with difference data configuration:

Seq2Seq - the data includes gender,text,tweet_counts gender without brand and the variables 'keep_letters_only': False, 'to_lower_case': True, 'lemmatize': True, 'remove_stopwords': False

TextRNN - the data includes gender,text,description,tweet_counts gender without brand and the variables 'keep_letters_only': False, 'to_lower_case': True, 'lemmatize': True, 'remove_stopwords': False

FastText - the data includes gender,text gender without brand and the variables 'keep_letters_only': True, 'to_lower_case': True, 'lemmatize': True, 'remove_stopwords': True

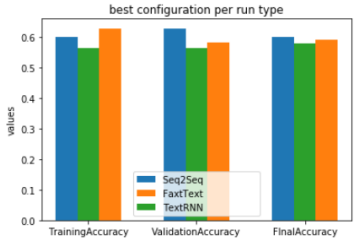

Figure 4: the best run of each type

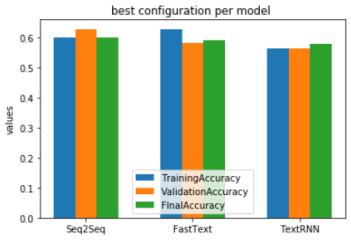

Figure 5: the best run of each model

4 Discussion

As we can see from the results, we were not able to reproduce the results that were shown in the related work we reffered to. There are probably more improvements that can be done in order to get better results, and different model we could have used to get better results. We tried 3 different model with 64 configuration variations. The configuration variations we tried were supposed to determine what variation best suited each model and what model and configuration produced the best results.

We also tried figuring out if there is configuration parameters that improve all of the models, and while most parameters value differ between models, removing all tweets related to "brand" gender (remove_brand parameter) improved the results in all 3 models.

The model that ended up producing the best result is Seq2Seq.

5 Code References

- Colab notebook - `https://colab.research.google.com/drive/1YEYEiS5PR3SAi8yZTPlHZVfLjb-XYYh`
- Results - `https://drive.google.com/open?id=1VxQ6a3kjCwDketaqS_fGrJn8QPYWPR86`
- Database - `https://www.kaggle.com/crowdflower/twitter-user-gender-classification`
- ML work - `https://www.kaggle.com/mehmetcekic/twitter-gender-prediction-with-nlp`
- Models - `https://github.com/AnubhavGupta3377/Text-Classification-Models-Pytorch`
- Word2vector file - `https://www.kaggle.com/takuok/glove840b300dtxt\ #glove.840B.300d.txt`
- Stop words file - `http://www.nltk.org/nltk_data`