Faculty of Mathematics and Information Science
Warsaw University of Technology

Project documentation

Predicting Reddit flairs

Subject: Natural Language Processing

Authors:
Konrad Komisarczyk, Patryk Wrona

Warsaw
2022

Contents

1. Introduction

1.1. Description of the research problem

The main goal of this project is to predict reddit tags (called flairs) using Natural Language Processing techniques. There are compared baseline models (like RandomForest or SupportVectorClassifier) along with more sophisticated BERT model. Moreover, the influence of Term-frequency Inverse-document-frequency vectorization on multilabel classification for baseline models is investigated.

1.2. Used data

The reddit submissions data is divided into months and years and contains all reddit submissions divided by subreddits. Each submission (user's post) is assigned a tag (called flair) specific to given subreddit. In this project, the data was filtered out to subreddits *teenagers* and *AmItheAsshole*. All steps of initial data preprocessing will be described in 2. We chose aforementioned subreddits because of the relatively high number of submissions, number of flairs and unusual rules of flair selection in the *AmItheAsshole* subreddit.

1.3. Instruction of the application (how to reproduce results)

In order to reproduce the results covered in this report, one must use python notebooks inside directories that are available on github:

— EDA – exploratory data analysis directory
— TfIdf* – directories containing results of training baseline models using TfIdf vectorization
— acquiring_data – scripts permitting to initially filter out downloaded data
— bert – scripts from Google Colab used for data tokenization, training and evaluating of BERT model
— train_test – scripts allowing to create train-test splits of balanced and unbalanced data meant for both baseline and BERT models

2. Preprocessing of submissions data

2.1. Data filtering

All reddit submissions dataset for the period from 2015 to 2021 contained **5 TB** of data.

For the purpose of this project, only 2 subreddits were chosen – *AmItheAsshole* (AITA) and *teenagers*. They contained numerous tags (called flairs) which seemed distinguishable and worth of attention. After this preprocessing step only **2.3 GB** of data was left.

Moreover, data cleaning step was also performed – only posts fulfilling at least one of the following conditions were kept:

— title of the post containing at least 3 words
— text of the post containing at least 5 words

Finally, all the words from title and text were lowercased.

Using such data, only significant columns were chosen for further analysis:

— flair – target variable (a tag)
— title – title of a post
— text – text of a post
— is_self – boolean variable telling if is self (it does not refer to an existing post)
— url – url of a post
— created_utc – number of milliseconds since the beginning of UNIX epoch
— score – number of upvotes of a post
— num_comments – number of comments of a post
— num_crossposts – number of posts being not self and referring to this post

2.2. Exploratory Data Analysis

From text and title columns, another 4 columns were drawn and analyzed for EDA as well as used in classification problems:

— title_wordCount – number of words in post's title
— title_characterCount – number of characters in post's title
— text_wordCount – number of words in post's text
— text_characterCount – number of characters in post's text

Basing on Exploratory Data Analysis, several distributions were analyzed across all posts within a given subreddit:

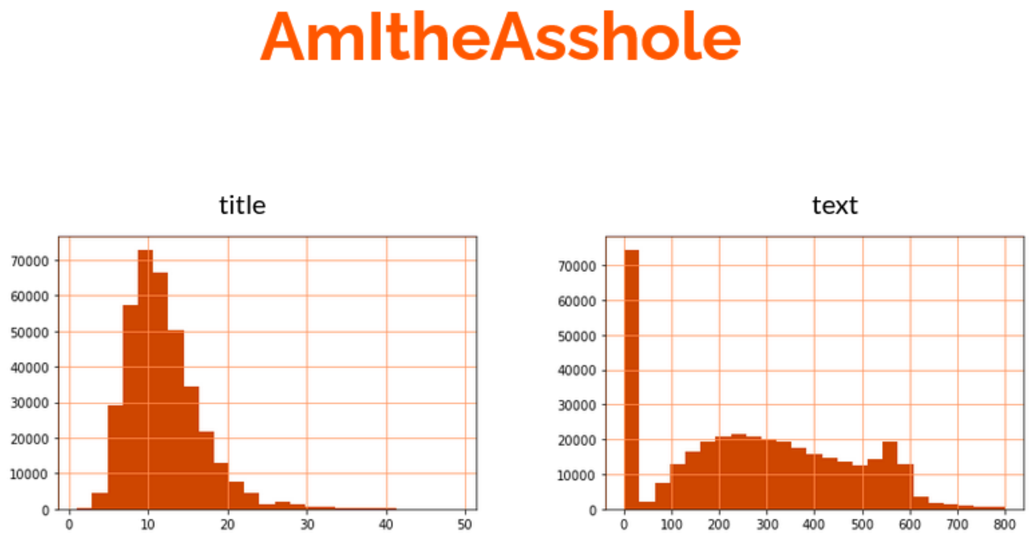

Figure 2.1: The distribution of number of words within posts' title and text for AITA subreddit

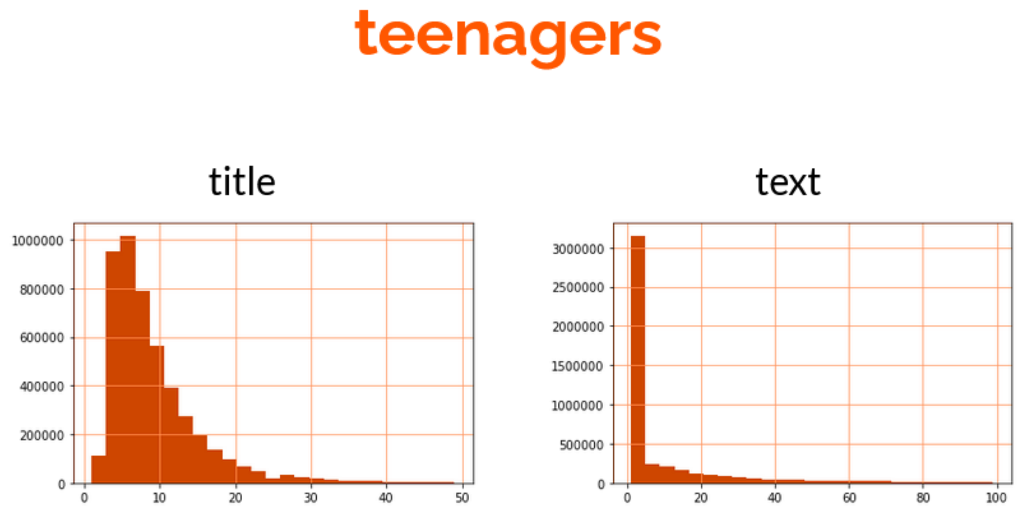

Figure 2.2: The distribution of number of words within posts' title and text for teenagers subreddit

2.3. Train test split

In order to prepare convenient data for classification task, it was split into train-test split. After trying to fit the data on Berta model, it was decided to lower the number of observations because of lacking RAM errors.

Each subreddit (AITA and teenagers) dataset was divided into balanced and unbalanced splits. In unbalanced split, the proportions of the classes were kept and in balanced dataset the number of occurrences of each class was the same. Number of observations in each created dataset is as follows:

— **AITA balanced** train-test - 18,000-1,800 (2,000 observations per class in training dataset, 200 observations per class in testing dataset)
— **AITA unbalanced** train-test - 20,000-2,000

— **teenagers balanced** train-test - 20,000-2,000 (2,000 observations per class in train dataset, 200 observations per class in test dataset)
— **teenagers unbalanced** train-test - 20,000-2,000

All above datasets chosen for machine learning tasks weighted about **65 MB**. The code for train-test split could be investigated in repository's *train_test* directory.

3. Machine Learning Tasks

3.1. Classification

3.1.1. Chosen classes (flairs)

In case of *AmItheAsshole* dataset, chosen labels for classes were (in descending order of occurrences):

— not-the-a-hole
— asshole
— no a-holes here
— everyone sucks
— not enough info
— update
— meta
— tl;dr

It is a 8 labels classification problem.

In case of *teenagers* dataset, chosen labels for classes were (in descending order of occurrences):

— other
— meme
— discussion
— social
— advice
— rant
— serious
— relationship
— media
— art

It is a 10 labels classification problem.

3.1.2. Dimensionality reduction

There were attempts to obtain better predictive power (less overfitting) along with decreasing the computation time by using different dimensionality reduction techniques. Calculations can be observed in *old_results_on_bigger_datasets/fs_dr.ipynb*. It was observed that manifold learning (being unsupervised learning meant for dimensionality reduction) had no sense because of lacking RAM and computational power. Furthermore, results of the only working dimensionality reduction method – PCA – were much worse than in case of fitting baseline models (baseline models will be covered in the next subsection) on the whole data. That is why the main focus was on performing Term-frequency inverse-document-frequency analysis using baseline models.

3.1.3. Term-frequency inverse-document-frequency analysis

In directories referring to *TfIdf*, we performed analysis of the impact of parameter *max_features* on accuracy and f1-scores obtained by baseline models on testing datasets.

Baseline models covered in this analysis were as follows:

— **RandomForestClassifier** from *sklearn.ensemble* python package
— **LinearSVC** from *sklearn.svm* python package
— **GaussianNB** from $sklearn.naive_{b}ayespythonpackage$

The analyzed range of the *max_features* parameter of TfidfVectorizer was from 5 to 1300. The results in form of weighted F1 Scores and Accuracies could be found in respective *TfIdf* folders (having the names describing the analyzed subreddit and answering whether used dataset was balanced or not). The best models were also saved and f1 scores for each class and predicted probabilities were saved for every best performing model from set of baseline models (results for the best RandomForest, SVC and NB for given dataset were saved in *best_results directories*).

3.1.4. Uncased BERT model

Bidirectional Encoder Representations from Transformers (BERT) pretrained model was also downloaded and further trained on each case of dataset. For the purpose of training the BERT model, each dataset must have been tokenized to fulfill BERT input requirements. All tokenization, training and evaluation steps could be investigated following jupyter notebooks in *bert* directory on the repository. The tokenized data for BERT increased in size from about 65 MB to about **250 MB**.

4. Results

4.1. TfIdf results

With the use of 3 baseline models: *1) Linear Support Vector Classifier*, *2) Random Forest* and *3) Gaussian Naive Bayes Classifier*, the influence of max features of post's title and text (TfIdf vectorizer's hyperparameter *max_features*) on obtained averaged f1 scores on test datasets (average was made using classes' weights) was investigated.

For all plots in this section, *NB*, *RF*, *SVM* stand for *Naive Bayes*, *Random Forest*, *Support Vector Machine* respectively.

In case of unbalanced *amItheAsshole* subreddit data:

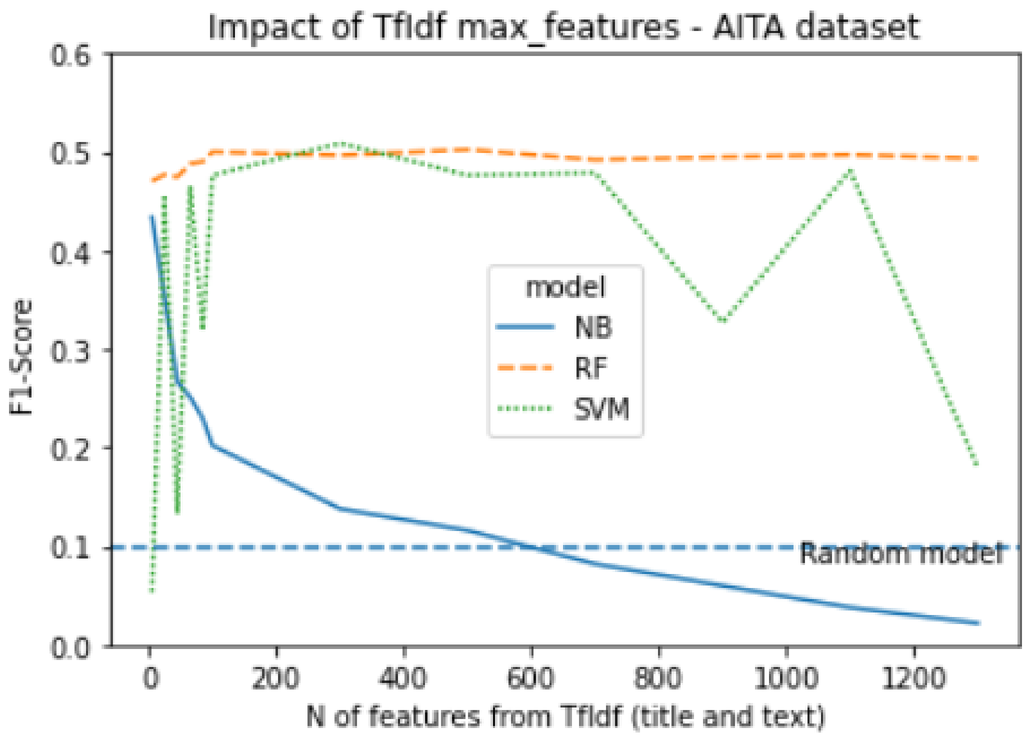

Figure 4.1: Influence of the number of features chosen by TfIdf vectorizer on obtained f1 scores, categorized by baseline model – unbalanced AITA subreddit

In case of balanced *amItheAsshole* subreddit data:

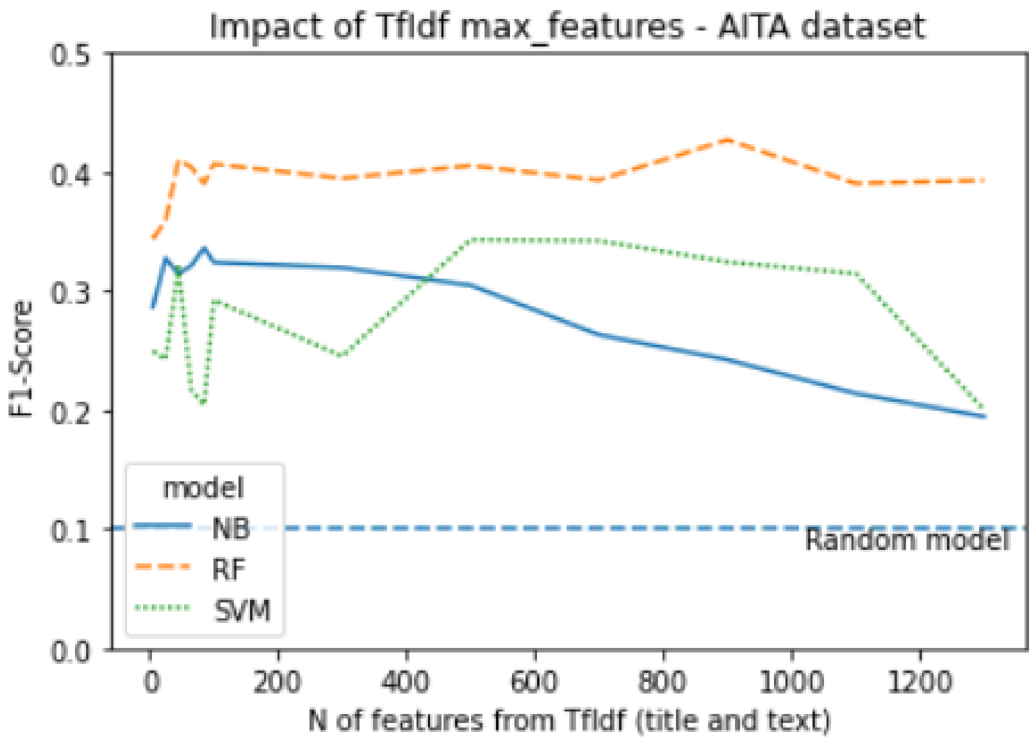

Figure 4.2: Influence of the number of features chosen by TfIdf vectorizer on obtained f1 scores, categorized by baseline model – balanced AITA subreddit

In case of unbalanced *teenagers* subreddit data:

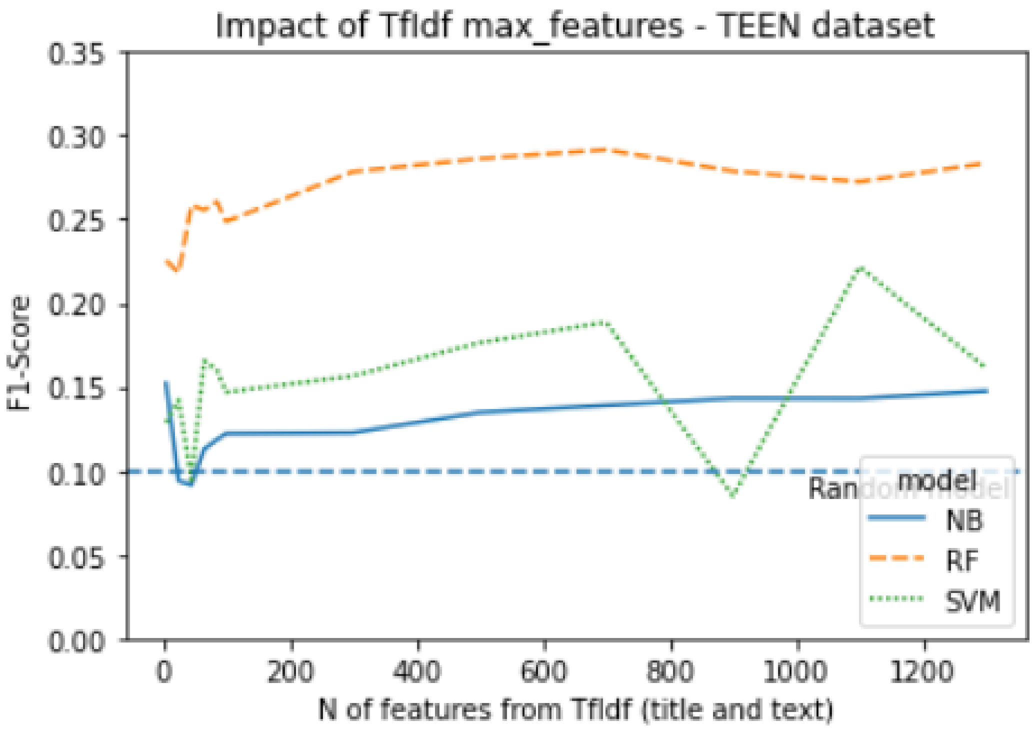

Figure 4.3: Influence of the number of features chosen by TfIdf vectorizer on obtained f1 scores, categorized by baseline model – unbalanced teenagers subreddit

In case of balanced *teenagers* subreddit data:

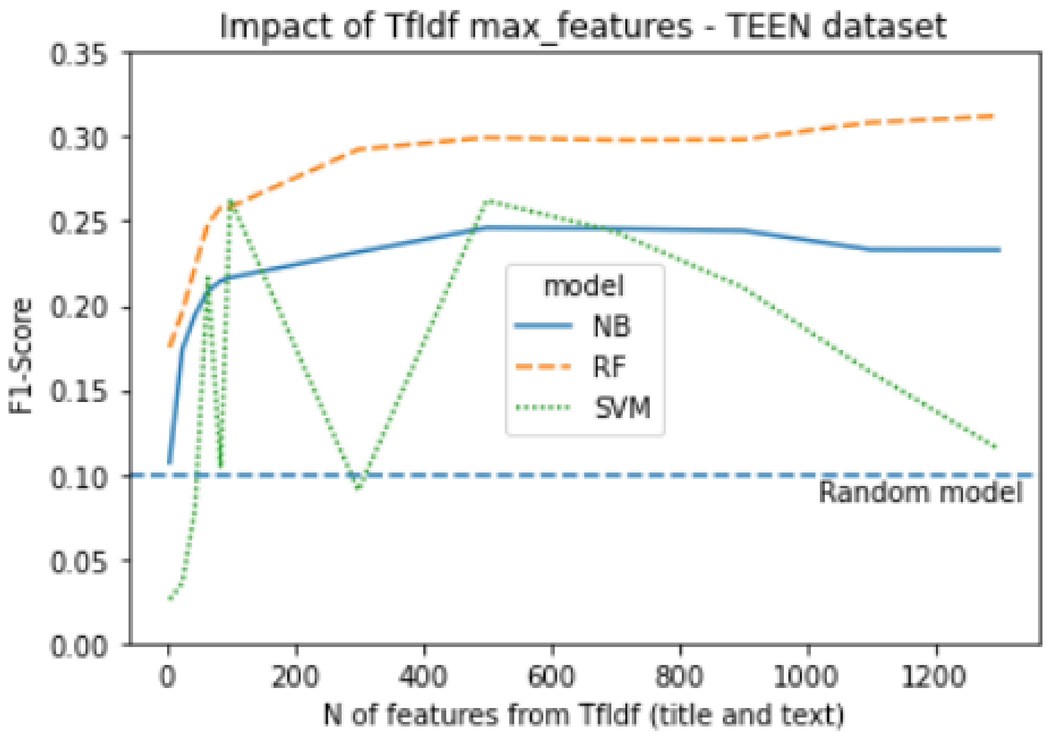

Figure 4.4: Influence of the number of features chosen by TfIdf vectorizer on obtained f1 scores, categorized by baseline model – balanced teenagers subreddit

As we can see from all figures 4.1 4.2 4.3 4.4, the *Random Forest* model was the best one for this problem. *Naive Bayes Classifier* obtained worse results than random model in case of unbalanced *AmItheAsshole* subreddit data, but usually all 3 models were capable of obtaining better f1 scores than the completely random model. In 4.1 4.2, *Linear Support Vector Classifier* obtained slightly worse results than than *Random Forest model*, whereas in 4.3 4.4, *Linear Support Vector Classifier* obtained low f1 scores, which sometimes were comparable to random model performance.

The best obtained models for each dataset were as follows:

— AITA unbalanced – **LinearSVC**, weighted F1 Score = 0.508764
— AITA balanced – **Random Forest**, weighted F1 Score = 0.426899
— teenagers unbalanced – **Random Forest**, weighted F1 Score = 0.291583
— teenagers balanced – **Random Forest**, weighted F1 Score = 0.312309

4.2. BERT results

We used also one transformer based language model - BERT. Python `transformers` library was used to download pre-trained model. We chose base uncased BERT for sequence classification. Data was first tokenized using specific BERT tokenizer. Then the model was fine-tuned for 4 epochs. Model was evaluated after each epoch on corresponding test datasets. Although BERT creators suggest 2 - 4 fine-tuning epochs in our case there were no significant differences in final results between epochs, for the tables below we chose final 4th epoch. Final weighted f-scores for each dataset are presented in the following tables:

teenagers	dataset		**balanced**	**unbalanced**
	weighted f1-score		0.41	0.38
	for each flair	meme	0.46	0.49
		other	0.56	0.4
		discuss	0.36	0.35
		social	0.46	0.25
		rant	0.37	0.38
		media	0.22	0.23
		advice	0.41	0.42
		serious	0.54	0.33
		relationship	0.4	0.33
		art	0.35	0.39

Am I The Asshole	dataset		**balanced**
	weighted f1-score		0.47
	for each flair	everyone sucks	0.29
		not enough info	0.39
		tl;dr	0.98
		no a-holes here	0.37
		asshole	0.24
		not the asshole	0.3
		update	0.22
		meta	0.99

5. Conclusions

The results obtained by BERT model outperformed baseline models for which *TfIdf* analysis was performed. This behaviour was expected as BERT model was pretrained and used transformer architecture.

As far as the baseline models are concerned, the *LinearSVC* model did not performed really well – that would mean that the classes are not linearly separable. The nonlinear model's (*RandomForestClassifier*) f1 scores were (except one surprising case of *LinearSVC*'s superiority) the best among all other 2 models.

Final BERT results were disappointing. We had not enough computational power to fit the model on whole dataset. That is why analyzed data was largely cut to permit us the training and evaluating of BERT model.

In the last while we made an error in data files' paths in Google Disk, as well as Google Colab removed our access to GPU and TPU units. That is why the results are not very satisfactory and we needed to repea the tokenization process for BERT.

As far as takeaways are concerned, it would be good idea to try learning BERT model on cased data. Lowercasing all texts and titles as a part of preprocessing step was not necessarily a good idea (especially in BERT model).

Bibliography

[1] Reddit Submissions dataset https://files.pushshift.io/reddit/submissions/