Capstone Project — Augustas Žaltauskas

Machine Learning Engineer nanodegree — April 19th, 2020

Toxic comment classifier

Warning: Due to the nature of this assignment, shown data examples might include toxic language.

Definition

Project Overview

Discussion in online community has a lot of benefits - it allows to share opinions freely and anonymously, express yourself and exchange ideas with people all over the world. However, this also gives a chance for offencive behaviour like abusement or harassment to arise. It discourages people from using online platforms to share their ideas and forces communities to limit of turn off commenting option overall [1]. To help fight this problem, the fast, automatic way of identifying toxic comments is needed.
In this capstone project I will present the model that can tell if provided comment is toxic and which of toxic behaviours (toxic, severe toxic, obscene, threat, insult, or identity hate) is expressed in it. This project uses data provided in kaggle competition [2]. it is collected from wikipedia page comments and comments' toxicity manually labeled by humans.

Problem Statement

This is a multilabel classification problem. For each comment in training dataset the algorithm should automatically predict which of the 6 toxicity classes the comment should be assigned to. Comment can be assigned to none, one or more than one class, depending on what behaviour it expresses. If comment is not assigned to any of the classes, that means that the comment is predicted as not toxic. The output for each of the comments should be a 1x6 vector with predicted score for each class. Predictions should be normalized to be between 0 and 1.

Metrics

To evaluate the model and compare to benchmark models I will use the same metrics that were used in the competition: "the average of the individual AUCs of each predicted column (class)". AUC (Area Under Curve) - is a measure of class separability, how well classifier can

separate classes. It is calculated from TPR (True Positive Rate) and FPR (False Positive Rate) by changing decision threshold. AUC metric will be calculated for each class separately and then averaged to get a single value which will then be compared against benchmark.

Analysis

Data Exploration

Provided data is available in kaggle competition page[2] where it is provided in csv files. Data is already split in training and testing datasets, as is common in Kaggle competitions.

Training data consists of ~140k records. Each record consist of comment and labels for each class, see Figure 1. If the label value is 1 - that means that such behaviour was expressed in the comment.

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0

Figure 1. Example of given data

Since this competition is already over, labels for test dataset are also provided. I will use them for model evaluation and comparison with benchmark. However, not all comments from test dataset were used for evaluation, more explanation on this topic can be found in the competition page[2]. I would like to compare my results to the performances of other competitors, therefore I will filter out the unused test data. That leaves me with ~64k records, from ~153k original records, which is plenty enough to test and evaluate the model.

Exploratory Visualization

It is always a good idea to take a look at the raw data. First look at the data always gives some insights on how we should deal with it. In Figure 2 we can see that we are dealing with highly unstructured language. Comments are highly variant in length, they have emojis, nonsensical words like usernames, irregular grammar, strange syntax, typos etc. This problem will be addressed by normalizing comments in data preprocessing step.

":::Here you go again with your condescension. I know what it means. Are you really this out of touch or are you pla
ying a game? It is called a fool's errand for a reason. Do you seriously not understand this? Do you seriously not un
derstand why that is an insult? Again, you have failed to do your job and simply tell Tribal why what she's doing in
inappropriate. Instead, you've mistakenly turned this into a behavior issue by lying about me making ""threats"" and
being a ""bully"". I've done no such things. I explained to Tribal that if she continued putting non-encylopedic, poo
rly sourced (or unsourced) content into hundreds of articles, it would be reported and she could get blocked. THAT is
not making threats or being a bully, in the context you are trying to have others believe. Not even close. If anythin
g, the bully is use for threatening me with bans, when I am trying hard to improve this project by stopping what I'm
confident is very inappropriate editing by Tribal (because of the huge number of articles it affects). Yet all you wa
nt to focus on is these completely out-of-context behavior claims against me, instead of using your admin knowledge f
or an important and useful purpose - to educate Tribal on why what she is doing is wrong. Non-encylopedic content, no
sourcing or unreliable sourcing, and adding content to leads that are very clearly not lead-worthy. You see what's ha
ppening and you are choosing to ignore it instead of saying directly to Tribal what the problems are with what she's
doing. 76.189.121.57

"
"

Happy Halloween!! And what do you think about Sand Diver - is she ready for DYK?

I did a quickie DYK check, and it meets the basic criteria...yes, no? ☯Consult "

Figure 2. Example of two comments from wikipedia page.

From sample comments above (Figure 2.) we can see that length of comments can be highly different as well. This can be visualized by showing comment length histogram, see Figure 3. Most of the comments are short quite short and only a few reaches the limit of 5000 characters.

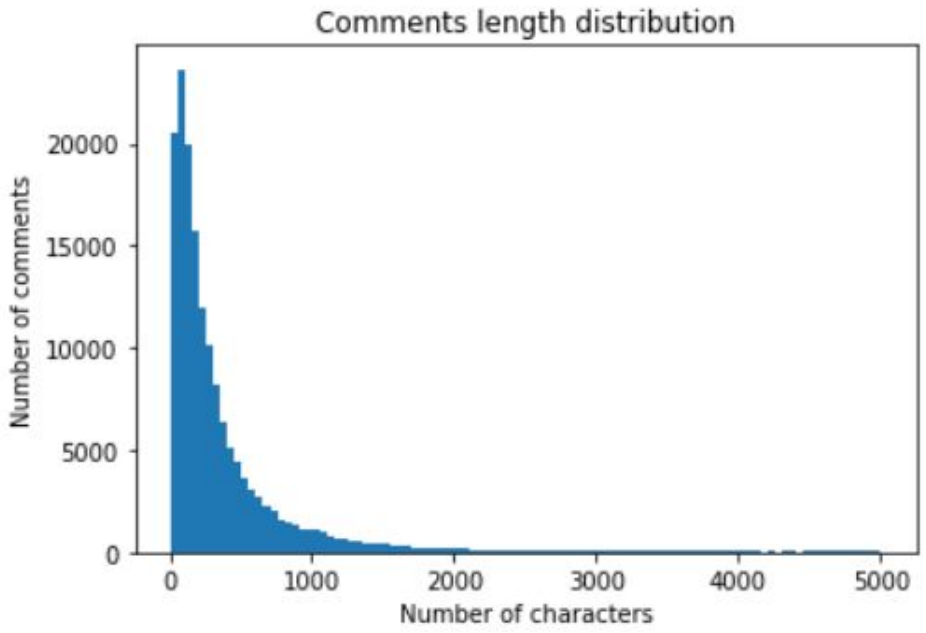

Figure 3. Histogram of comment length, measured by the amount of characters in the comments.

We can also take a look at labels in training data. Provided labels are encoded as 1 or 0 - if comment belongs to the class or not. First, let's take a look at labels histogram - how many comments are labeled with each class, see Figure 4.

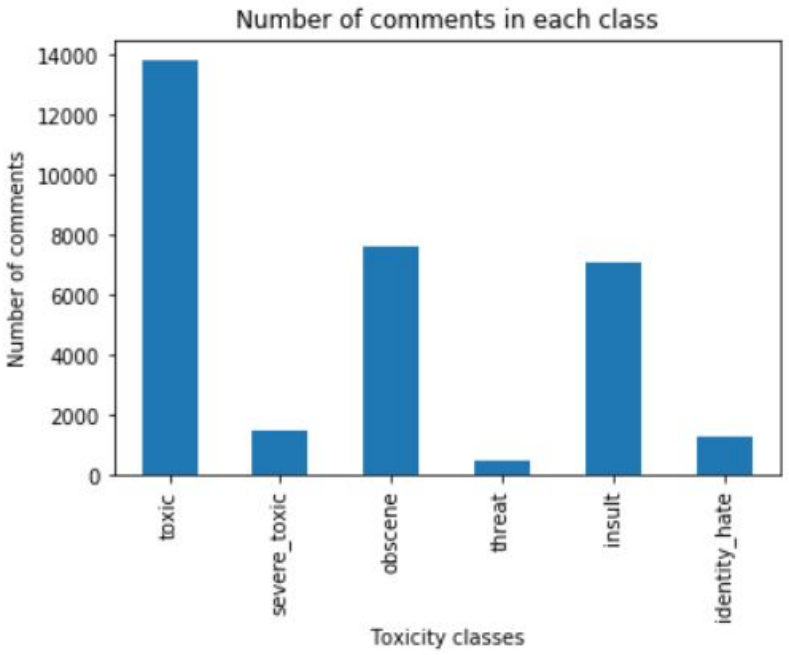

Figure 4. Number of comments assigned to each class

We can see that classes are highly unbalanced. Comments that belong to the most popular “toxic” class sums up to be less than 10% of all comments.

Algorithms and Techniques

Algorithms that I will use for this project are TFIDF, LSA and Shallow NN for multilabel classification.

TFIDF (Term Frequency – Inverse Document Frequency) - is an algorithm defined to find the most important words for a given document based on full corpus (document collection). First part of algorithm - Term frequency is calculated on a document level. It counts the number of times each word was present in a specific document. Second part “inverse document frequency” takes a look at how often the words appear in the whole corpus, in this case - in all the comments. TFIDF score for each word increases proportionally by term frequency and offset by inverse document frequency.

This method is one of the most popular text vectorization algorithms. It lowers the score for very common words, for example articles, which do not give much insights. Score is higher for words that are common in fewer texts, that can help us group and classify them.

LSA (Latent Semantic Analysis) - It is the technique used in natural language processing field. It is used for dimensionality reduction and works great with TFIDF. TFIDF gives values for every word in the corpus, therefore it becomes a really long vector that is hard to use in downstream tasks. LSA using Singular Value Decomposition lowers the amount of features, while retaining the similarity structure between documents. Dimensionality reduction is necessary if we want to use vectorized data for training the neural network model for classification task.

NN (Neural Network) - neural networks are highly flexible and are usually used for multilabel classification tasks. It is always better to start from simple algorithms, therefore shallow fully connected (dense) network will be used for this classification task. To solve multilabel classification task, Network will have same amount of neurons as classes in the output layer and sigmoid activations will be used, to produce the normalized final score for each class.

Benchmark

As this data was provided for Kaggle competition, the benchmark will be based on the evaluation metric that was used in the mentioned competition. Model should achieve the AUC score equal or higher than 0.95. The top approaches provided by the participants achieved higher scores, however this benchmark should be enough to get a reasonably well performing model, that could be further used in other applications. Models submitted to the competition and their performance can be viewed in leaderboards of the challenge[2].

Methodology

Data Preprocessing

In data preprocessing step I will normalize the comments. Code for each of the following spets can be seen in Figure 5.

```
def clean_comments(df):

    # Lowercase
    df_clean = df.str.lower()

    # Change empty space characters to spaces
    df_clean = df_clean.str.replace(r'\s', ' ', regex=True)

    # Remove non ASCII letters
    df_clean = df_clean.str.replace(r"[^a-z ']+", '', regex=True)

    # Normalize spaces (for readability)
    df_clean = pd.Series([" ".join(x.split()) for x in df_clean])

    return df_clean
```

Figure 5. Function for data cleaning.

First I lowercase all letters, to normalize each word. By doing this we make sure that it would not matter if the word was written in the beginning of the sentence, in all caps or lowercased and the model would not treat these cases as separate words.

Next, I need to normalize empty spaces between words. There can be multiple spaces, new lines, tabs, etc. We can use regex (regular expressions) to normalize spaces quickly.

Comments are usually highly unstructured. They can contain various letters, emojis, pictograms, bunch of punctuation marks etc. Once again we tackle this problem with regex. By removing all characters that are not ASCII letters, spaces or apostrophe.

Removing characters can leave extra spaces. They will not impact the performance of TFIDF, but for visualization purposes once again we normalize spaces.

""Perhaps""? Why ""perhaps""? And why did you revert the edit? 201.215.60.232 " ---------- Clearly a partisan gesture. I'll discuss this further when I am able to edit freely. —aco ---------- Question Do you have any interests outside of lamely editing Wikipedia? Even a small one? ---------- " A kitten for you! just becuz ur pussy 4 lying. " ---------- Hello, and welcome to Wikipedia! We welcome and appreciate your contributions, such as Hilltop Reservation, but we regretfully cannot accept copyrighted text or images borrowed from either web sites or printed material. For more information about Wikipedia's policies and guidelines, take a look at our Five Pillars. Happy editing! .:..:. babelfish	perhaps why perhaps and why did you revert the edit ---------- clearly a partisan gesture i'll discuss this further when i am able to edit freely aco ---------- question do you have any interests outside of lamely editing wikipedia even a small one ---------- a kitten for you just becuz ur pussy lying ---------- hello and welcome to wikipedia we welcome and appreciate your contributions such as hilltop reservation but we regretfully cannot accept copyrighted text or images borrowed from either web sites or printed material for more information about wikipedia's policies and guidelines take a look at our five pillars happy editing babelfish

Figure 6. Example of comments before normalization (left) and after normalization (right)

Implementation

implementation steps:

1) Load data

Loaded training data as pandas dataframe. Split input data (comments) and class labels to separate dataframes for easier use.

2) Create validation dataset

Separated 10% of training data to be used as validation data, so I could evaluate model performance without using test data.

3) Vectorization

Converted words from training dataset into numerical representation by using sklearn implementation of TF-IDF algorithm.

4) Dimentionality reduction

For initial model I left 10k features (represent 10k words) from vectorization step. To use these features in following steps I need to lower their dimensions. This is achieved with LSA algorithm. I used sklearn implementation of it - also called as TruncatedSVD. For initial model I have used LSA with 10 components, that gave me 10 value feature from each comment

5) Process validation dataset

Transform validation data (comments) using TFIDF and LSA models from previous steps. For the next step - training neural network, I need processed validation dataset, so I could see the classification AUC score for validation data during training and understand if model is not overfitting.

6) Train NN multi-label classifier

For initial model I defined single hidden layer neural network with 10 nodes and relu activation in hidden layer and 6 nodes (same as the number of classes) and sigmoid activation in output layer.

7) Train the model

Model trained quite well from the first time. This can be explained by the small input feature space. During training model already reached high AUC. Learning and AUC curves can be seen in Figure 7.

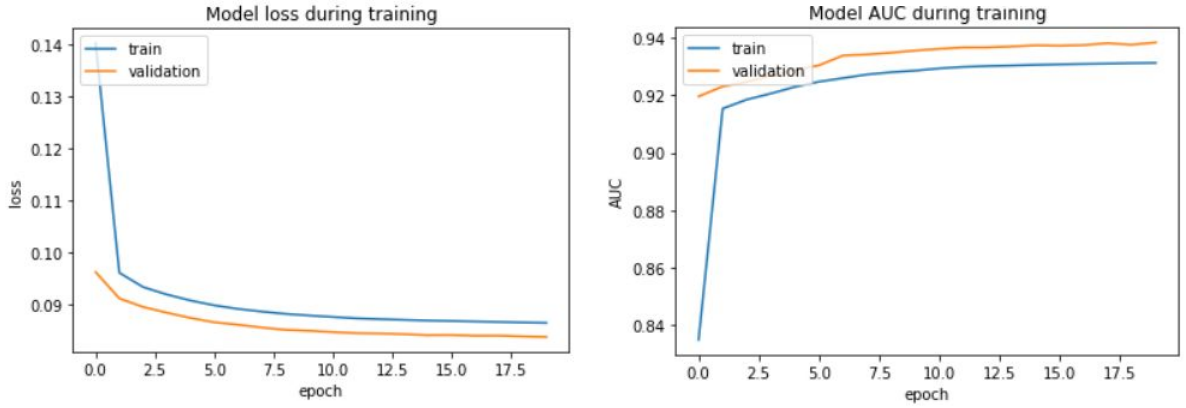

Figure 7. Loss and AUC curves during initial model training

8) Evaluation

Since Tensorflow implementation of AUC metric calculates AUC score on batches, I have used sklearn implementation to calculate the AUC score for each class and find average between them. Initial model reached mean AUC value of 0.9157.

Refinement

To refine model and improve model's performance to reach the decided benchmark - 0.95 AUC score, I have implemented hyperparameter search. For easier implementation I have user sklearn pipeline and CV grid search. Pipeline allows to integrate all models (TFIDF, LSA, and NN) into a single pipeline and CV grid search let's us define the hyper parameters that we want to explore and search them by using k-fold cross validation.

Hyperparameter search is computationally and time expensive task. This search had 81 parameter combination, was evaluated by taking mean score of 5-fold cross validation. This accumulated up to 405 network trainings and took ~6 hours.
Explored parameters space:

TFIDF max_features: [10000, 30000, 100000],
LSA n_components: [30, 50, 100],
NN hidden layer dimentions: [20, 30, 50],
NN dropout rate: [0.5, 0.3, 0.]

In Figure 8 we can see the top 10 best performing models and their parameters.

	lsa__n_components	nn__dropout	nn__l1_dim	nn__lsa_dim	tfidf__max_features	mean_auc_score	std
71	100	0.3	50	100	100000	0.960859	0.000559
59	100	0.0	30	100	100000	0.960762	0.001034
70	100	0.3	50	100	30000	0.960663	0.001552
67	100	0.3	30	100	30000	0.960483	0.000802
61	100	0.0	50	100	30000	0.960444	0.000997
79	100	0.5	50	100	30000	0.960422	0.001058
69	100	0.3	50	100	10000	0.960348	0.000794
62	100	0.0	50	100	100000	0.960343	0.000793
80	100	0.5	50	100	100000	0.960294	0.001099
68	100	0.3	30	100	100000	0.960274	0.001350

Figure 8. Top 10 best performing models (from explored subset) based on on AUC metric, their parameters and scores.

Results

Model Evaluation and Validation

Final model was evaluated on test data and achieved mean AUC of 0.9524. It have reached the required target benchmark value of 0.95 AUC.
Final model's parameters were chosen based on hyper parameter tuning and explored values:

- TFIDF stopwords: "English"
- TFIDF maximum features: 100 000
- LSA number of components: 100
- NN hidden layer size: 50
- NN dropout rate: 0.3
- NN hidden layer activation function: "relu"
- NN output layer activation function: "sigmoid"
- NN loss function: "binary_crossentropy"
- NN optimizer: "Adam"
- NN training epochs: 20

Other algorithm values are same as the default values in sklearn and keras implementations.

Model's robustness was verified by checking at the predicted toxic comments in test dataset, see Figure 9 and Figure 10. For more examples please view notebook files.

	comments	toxic	toxic_p
1	:Dear god this site is horrible.	0.0	1.0
8	== Arabs are committing genocide in Iraq, but ...	1.0	1.0
17	" \n\n ==balance== \n This page has one senten...	0.0	1.0
24	DJ Robinson is gay as hell! he sucks his dick ...	1.0	1.0
27	I WILL BURN YOU TO HELL IF YOU REVOKE MY TALK ...	0.0	1.0

Figure 9. First 5 comments predicted as "toxic" by the model. "Toxic" column stands for ground truth label, "toxic_p" - for prediction.

	comments	severe_toxic	severe_toxic_p
5124	fuck all you fucking nigers	0.0	1.0
5144	you ya fucking dirty little mother fucking coc...	0.0	1.0
6490	==Son of a bitch== \n\n Hey you fucking neo-Na...	1.0	1.0
7430	== f.uck you == \n\n you are an ass.hole and a...	1.0	1.0
7583	== for fucking israeal == \n\n fuck uuuuuuu	0.0	1.0

Figure 10. First 5 comments predicted as "severe toxic" by the model. "severe_toxic" column stands for ground truth label, "severe_toxic_p" - for prediction.

The following observations were made based on the Figure 9 and Figure 10:

- From Figure 9 we can see that algorithm mistakes classifying negative review as toxic - row 1
- From Figure 9 we can see that algorithm mistakes non toxic comment as toxic - row 17
- Model catches some comments that were (in my opinion) labeled incorrectly - Figure 9, row 27.
- Figure 10 - all retrieved comments seem to be toxic, however their toxicity level is difficult to evaluate.

Justification

Model's results reached the decided benchmark value and should perform reasonably well in toxic comment classification task as seen from Figure 9 and Figure 10. However, if compared to other top submissions to the kaggle competition, it would be within top 4000 models and would fall short by 0.036 AUC score value from the first place.

References

[1] https://www.wired.com/story/twitter-let-users-hide-replies-fight-toxic-comments/
[2] https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/overview