Machine Learning Engineer Nanodegree

Capstone Proposal

Ishan Soni
December 16th, 2018.

Proposal

Toxic Comment Classification on social media platforms

Domain Background

My project draws upon the domain of document classification with an emphasis on the natural language processing component of document classification. Machine learning has been successfully used to classify documents by topic for several decades. However, machine learning techniques do not perform as well when performing sentiment analysis which requires the parsing of more complex language structures. This is where ideas from natural language processing must be applied. My aim with this project is to make progress towards solving a complex natural language processing problem using machine learning. I think that it would be very beneficial if computers could interpret and produce the same kind of natural language of which even young children are capable. This would allow machine learning to be applied to a wider variety of tasks than it is currently capable of solving. In particular, machine learning could be applied to many problems that are not well structured and for which there is not much training data.

This capstone project is based on the Toxic Comment Classification Challenge

Discussing things you care about can be difficult. The threat of abuse and harassment online means that many people stop expressing themselves and give up on seeking different opinions. Platforms struggle to effectively facilitate conversations, leading many communities to limit or completely shut down user comments.

The Conversation AI team, a research initiative founded by Jigsaw and Google (both a part of Alphabet) are working on tools to help improve online conversation. One area of focus is the study of negative online behaviors, like toxic comments (i.e. comments that are rude, disrespectful or otherwise likely to make someone leave a discussion). So far they've built a range of publicly available models served through the Perspective API, including toxicity. But the current models still make errors, and they don't allow users to select which types of toxicity they're interested in finding (e.g. some platforms may be fine with profanity, but not with other types of toxic content).

With more people joining social media than ever before, it becomes imperative that this problem is solved. Classifying toxic comments (obscene, threat, insult, identity-based hate) will be the core of this project.

Problem Statement

In this project, I will build a multi-headed model that will be capable of detecting different types of toxicity like threats, obscenity, insults, and identity-based hate from a given comment. I'll be using a dataset of comments from Wikipedia's talk page edits. Improvements to the current models will hopefully help online discussion become more productive and respectful.

This is a supervised multi-class classification problem and as such different measurable evaluation metrics could be applied. These will be further elaborated in Section 6 - Evaluation Metrics. Given a comment, the solution will be a machine learning model that receives as input the comment and outputs either a class prediction or a class probabilty for every toxicity type.

Datasets and Inputs

I'll be using a dataset of comments from Wikipedia's talk page edits which have been labeled by human raters for toxic behavior.

The types of toxicity are:

- toxic
- severe_toxic
- obscene
- threat
- insult
- identity_hate

The data files are uploaded in the data folder. The following files are present :

1. train.csv - the training set, contains comments with their binary labels
2. test.csv - the test set, we will predict the toxicity probabilities for these comments.

First two rows from the train dataset :

id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0000997932d777bf	You, sir, are my hero. Any chance ...	$1600	0	0	0	0	0
000103f0d9cfb60f	COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK	$12	1	1	0	1	0

Given the scope of the problem, it is very appropriate to use this data set, as it conforms exactly to the problem statement from Section 2.

Solution Statement

The solution to this problem will be an algorithm that takes as input a list of comments and outputs a probability for each of the six possible types of comment toxicity (`toxicity_type`) : toxic, severe_toxic, obscene, threat, insult, identity_hate for each comment. This confidence score could be expressed as y ˆ ϵ [1,0] where a score of 1 indicates that the algorithm has perfect confidence that the observation deserves the label of that `toxicity_type` and a score of 0 indicates that the algorithm is perfectly confident that the observations should be not labeled with that `toxicity_type`. These confidence scores could help diagnose the algorithm and allow it to be applied in different settings.

Example 1: Input Text : `You, sir, are my hero` Output will be : toxic : 0.1 , obscence : 0.0, .. The output states that the algorithm thinks that there is a 0.1 probability or 10% chance that the given text is toxic. Similarly the algorithm thinks that there is 0 probability or 0% chance that this text is obscene.

Example 2: Input Text : `Oh my, you are so fuckable.` Output will be : toxic : 0.1 , obscence : 0.8, .. The output states that the algorithm thinks that there is a 0.1 probability or 10% chance that the given text is toxic. Similarly the algorithm thinks that there is 0.8 probability or 80% chance that this text is obscene. These class probabilities will enable us to make many smart decisions. Such a comment should be fine on a platform that tolerates obscenity. Such platforms can use this class probability and mark this post as `NSFW`. Other platforms that don't tolerate obscenity can chose to delete such comments automatically.

Generally, if the class probability for a particular type of toxicity is > 0.5, we will label it as a toxic comment with that toxicity type.

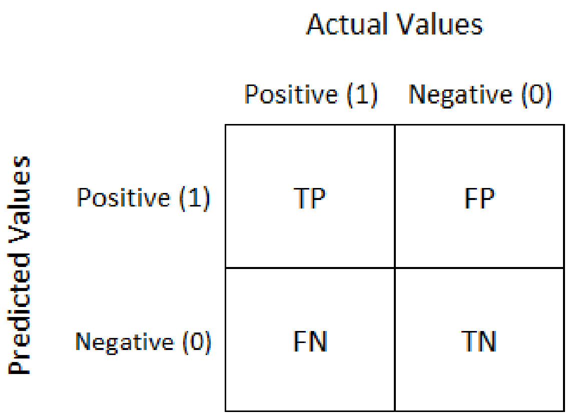
Benchmark Model

The simplest benchmark model would be one that always predicts the most commonly occurring class. Such a model would output the same result for every submission without considering the textual context. This is a very simple benchmark, but it is imperative that the final model should be able to beat this benchmark. Due to the simplicity of the above benchmark, it is important to have a second, more advanced benchmark. A slightly more complicated benchmark would be a logistic model that takes as input a bag of words structured into a document term matrix. As stated in Joulin et al. (2016) : linear classifiers that use bag of words features are a strong baseline.

Evaluation Metrics

There are many different metrics that could be used to evaluate the solution, given that this is a multi-class classification problem. The most common metric is accuracy. We can create a confusion matrix for each toxicity type and study the algorithm's accuracy for that toxicity type.

Generally, if the class probability for a particular type of toxicity is > 0.5, we will label it as a toxic comment with that toxicity type. Therefore we can easily build a confusion matrix for each toxicity class.



		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

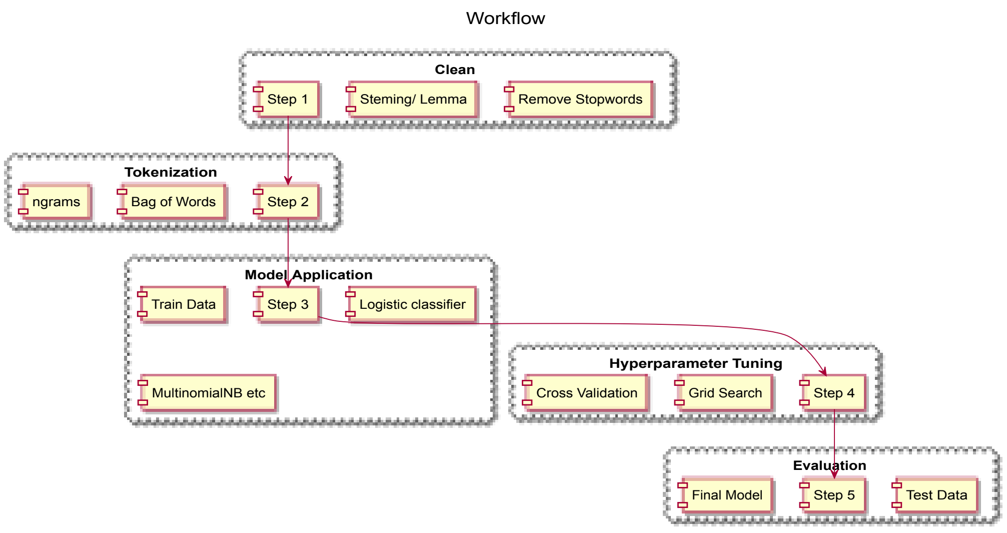
Project Design

I plan to build my models in python using scikit-learn (sklearn). Sklearn is a comprehensive machine learning library that includes many models and tools that are useful in building a machine learning pipeline. I will use sklearn to develop and test several different models that use bag of words features, ngrams etc including the baseline logistic classifier.

We will use the following workflow :

1. We will first start with cleaning the train and test data. In this step we will try to reduce the matrix sparsity of the bag of words and ngrams representations of our data so that our algorithms can train faster and perform better. Cleaning will include :
    1. Removing stopwords. These are commonly occuring words like a, the, in etc that will not contribute anything towards our analysis.
    2. Performing stemming and/or lemma. Stemming is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form. Eg love, loved and loving mean the same thing and performing a stem operation on them will reduce all of them into the same root form 'lov'. This will further reduce our vocab and reduce the matrix sparsity of our bag of words/ ngrams representations.
2. We will then tokenize our train and test data into representations that can be used by machine learning algorithms. These representations include bag of words, ngrams etc.
3. We will then apply several machine learning models on the vectorized representations of our train data. We will use models like our baseline logistic classifier, MultinomialNB etc

4. We will then perform hyperparamter tuning (eg Grid Search) to find the best model. We will combine Grid search with cross validation so that we do not end up with an overfitted model.
5. We will then test the accuracy of our final models using the evaluation metrics specified above and pick the best one.



Before submitting your proposal, ask yourself. . .

- Does the proposal you have written follow a well-organized structure similar to that of the project template?
- Is each section (particularly **Solution Statement** and **Project Design**) written in a clear, concise and specific fashion? Are there any ambiguous terms or phrases that need clarification?
- Would the intended audience of your project be able to understand your proposal?
- Have you properly proofread your proposal to assure there are minimal grammatical and spelling mistakes?
- Are all the resources used for this project correctly cited and referenced?