Word Embedding for Toxic Comment Classification Challenge

Guangyi ZHANG

April 20, 2018

1 Introduction

The threat of abuse and harassment online means that many people stop expressing themselves and give up on seeking different opinions. This Kaggle competition[1] tries to help improve online conversation by developing models that can detect negative online behaviors, like toxic comments (i.e. comments that are rude, disrespectful or otherwise likely to make someone leave a discussion). In its nature, this is a problem of multi-label classification on text documents, where Word2Vec (Mikolov et al. [2013b], Mikolov et al. [2013a]), Glove (Pennington et al. [2014]), FastText (Bojanowski et al. [2016]) and other word embedding techniques can leverage their power for a better performance.

In this report, the effect of Word2Vec for this particular task is examined under different hyperparameter settings and modeling methods. To our surprise, pre-trained embeddings from a large-scale general corpus fail to meet the need of toxic documents classification. Embeddings built on small corpus in related domain achieve superior performance. In regard to the embedding of a document, Doc2Vec (Le and Mikolov [2014]) is unsatisfactory, compared to a weighted mean of word embeddings. However, without the help of the powerful and complex neural networks such as Recurrent neural network (RNN), none of them is able to beat a traditional weighted N-gram model.

The structure of this report is organized as follows. First a brief introduction to the dataset is given. Then a general classification framework together with three embedding schemes are explained in details, followed by the section of result and evaluation. Finally the report ends with a conclusion and criticism.

2 Problem Settings

Kaggle provides a large number of Wikipedia comments which have been labeled by human raters for toxic behavior. The types of toxicity are:

[1] `https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge`

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
6	0002bcb3da6cb337	COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK	1	1	1	0	1	0
12	0005c987bdfc9d4b	Hey... what is it..\n@ | talk .\nWhat is it.....	1	0	0	0	0	0
16	0007e25b2121310b	Bye! \n\nDon't look, come or think of comming ...	1	0	0	0	0	0
42	001810bf8c45bf5f	You are gay or antisemmitian? \n\nArchangel WH...	1	0	1	0	1	1
43	00190820581d90ce	FUCK YOUR FILTHY MOTHER IN THE ASS, DRY!	1	0	1	0	1	0

Figure 1: Samples of Toxic Comments

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

In total, there are 159571 training comments and 153164 testing comments. Figure 1 shows samples of toxic comments. The task is to create a model which predicts a probability of each type of toxicity for each comment.

3 Models

Before performing any serious manipulations on the data, a preprocessing routine is needed, to clean special symbols, lower the case of the text, split a comment into sentences, tokenize a sentence into words and etc.

Then every word is embedded into a dense vector of real values, which forms a comment vector by either taking a mean or a TF-IDF[2] weighted mean of all its words vectors. Besides, a Doc2Vec model is also trained to parameterize the comment vectors directly. Later a comment vector is fed into a classifier.

As the focus of this report is the word embedding, instead of classification methods, a simple Logistic Regression classifier with default parameters has been adopted for estimating probabilities of different toxicity types for a comment.

Apart from the embedding-based methods, a baseline method using TF-IDF on unigrams is also created to testify the performance of models of our interest.

4 Embedding Schemes

Three different embedding schemes are introduced in the following subsections.

4.1 Pre-trained Embeddings

In this report, pre-trained GoogleNews-vectors-negative300[3] trained on part of Google News dataset (about 100 billion words) has been used, which contains

[2] https://en.wikipedia.org/wiki/Tf%E2%80%93idf
[3] https://code.google.com/archive/p/word2vec/

300-dimensional vectors for 3 million words and phrases. If a word does not appear in the vocabulary of embedding, it will be discarded.

4.2 Embeddings Trained from Toxic Corpus

Word2Vec embedding has been trained on the all comments of the training set, hereafter called *ToxicW2V*. In this case, all words in the training set will be preserved and every word that only appears in the testing set will be discarded.

4.3 Doc2Vec

Dov2Vec are directly trained on the all comments of the training set, and then a embedding vector will be inferred from a comment in the testing set.

5 Experimental Result

5.1 Word2Vec Hyperparameters Tuning

Multiple hyperparameters have been selected for the tuning, and the results of a grid search are shown in Table 1, where all scores are given by the Kaggle online judge system based on the predictions on the testing set with a TFIDF weighted mean comment vector. Due to the tremendous number of possible combinations of hyperparameters, a greedy grid search that fixes all hyperparameters but the target one has been employed here.

Some important discoveries are listed here.

- *iter* has a great impact in the performance of the resulting embedding, probably attributed to the small size of the training corpus.
- Embedding *size* is another hyperparameter that exerts a considerable influence.
- Negative Sampling, a simplified version of Noise Contrastive Divergence (NCE), outperforms the traditional Hierarchical Softmax.
- Down-sampling on frequent words still remains powerful even under such a small corpus.
- Surprisingly, CBOW surpasses Skip-Gram (SG) in this task even though the latter is shown in the literature to be better in almost every aspect. As we will see in the next subsection, SG beats CBOW when a mean value is used for a comment vector.

The combination of hyperparameters that produces the best performance are used in the following experiments. In addition, a Skip-Gram version is also added as a counterpart of CBOW.

sg	size	window	min_count	hs	neg	iter	sample	score
1	100	5	1	0	5	25	0.1	0.9346
1	100	5	1	0	5	25	0.01	0.9368
1	100	5	1	0	5	25	**0.001**	0.9403
1	100	5	1	0	5	5	0.001	0.9367
1	100	5	1	0	5	**25**	0.001	0.9403
1	100	5	1	0	**5**	25	0.001	0.9403
1	100	5	1	0	10	25	0.001	0.9386
1	100	**5**	1	0	5	25	0.001	0.9403
1	100	10	1	0	5	25	0.001	0.938
1	100	5	1	**0**	5	25	0.001	0.9403
1	100	5	1	1	NA	25	0.001	0.9346
1	**300**	5	1	0	5	25	0.001	0.945
1	100	5	1	0	5	25	0.001	0.9403
1	300	5	1	0	5	25	0.001	0.945
0	300	5	1	0	5	25	0.001	0.9478

Table 1: Greedy Grid Search on Hyperparameters

5.2 Classification Performance

The results are shown in Table 2. The poor performance of the pre-trained embedding is probably contributed to by the missing vocabularies, in which 75% of words are not found in the pre-trained embedding. After all, toxic words are rarely seen in Google News. To our surprise, the novel idea of Doc2Vec of parameterizing the comment vector directly does not live up to its expectation, giving the worst performance for this task.

As expected, SG gives the best result among embedding methods, in which every token is required to predict its context and has the chance to adapt its value for more times. Therefore SG benefits those infrequent words in particular, which are exactly the case for tocix words, because the majority of comments are clean. Another interesting discovery is that TF-IDF does not perform as well as a simple mean in regard to the formulation of the comment vector. In general, TF-IDF is capable of emphasizing meaningful words within a sentence, leading to a comment vector more like the vectors of toxic words in this case, which makes more sense. The reason why this happens is unknown, and one guess is that some infrequent toxic words are not trained sufficiently while holding a large weight.

However, none of the embedding-based methods are able to defeat the traditional unigram model. It seems the embedding can not fulfill all its potential without the help of a complex neural network.

Method	Score
Pre-trained + Mean	0.9487
Pre-trained + TFIDF	0.9410
ToxicW2V (SG) + Mean	**0.9541**
ToxicW2V (SG) + TFIDF	0.9450
ToxicW2V (CBOW) + Mean	0.9477
ToxicW2V (CBOW) + TFIDF	0.9478
Doc2Vec	0.8332
Unigram + TFIDF (baseline)	**0.9695**

Table 2: Classification Results

Word1	Word2	P-O-S Tag	Score	Type
take	remove	V	6.81	SYNONYMS
shine	polish	V	7.80	SYNONYMS

Table 3: SYNONYM examples in SimVerb-3500

6 Evaluation

6.1 Simverb

SimVerb-3500[4] is a gold standard evaluation resource for semantic similarity of verbs. Here only the *SYNONYM* pairs is being used for the evaluation, whose examples are shown in Table 3. The measure is defined as the percentage of the synonym pairs such that Word2 falls into the most similar words of Word1, which is displayed in Table 4. Only pairs that appear in the embedding are considered. For the sake of comparison, the pre-trained embedding is also tested. The performance of ToxicW2V is left behind by a wide margin, mainly due to the defficiency of corpus.

6.2 Sanity Check

Here we use the a selected word list and check the nearest neighbors for each word in the list, shown in Table 5. This check is very insightful and reveals interesting information about our embedding. The first one is the lack of a better preprocessing to handle special symbols, and many words are not tokenized properly. The second is our embedding does encode both syntactic and semantic information between toxic words.

[4] `http://people.ds.cam.ac.uk/dsg40/simverb.html`

Embedding	#Hits	#Pairs	Accuracy
ToxicW2V	14	298	4.698%
Pre-trained	74	306	24.183%

Table 4: SimVerb-3500 Evalution

Central Word	motherfucker	shit	bullshit
2	dawnseeker	sht	bullshi
3	meathead	82.132.244.134	99.184.231.227
3	motherfuck	shioty	willyjeeves
4	dolescum	ashit	hahahahahahahaa
5	sucker	gradnman	cliquey
6	vegetasaiyan	86.134.181.35	continues.—
7	chink==	206.207.175.162	nonsese
8	fagget	12.176.20.2	pigshit
9	googdbye	shgit	bible.
10	endian	14.136.219.161	arse94.168.210.205

Table 5: Sanity Check

6.3 PCA Visualization

Two groups of manually selected words, a group of good words and the other of toxic words, are visualized in two dimension via PCA. As can be seen in Figure 2, two groups are clearly separated in the new coordinate system, indicating their original embedding has encoded meaningful semantic information.

7 Conclusion

To sum up all the experimental results, we have investigated the effect of different hyperparameters for Word2Vec embedding, and attempted to develop and compare a meaningful document vector in multiple potential approaches. Besides, we point out the limitation of pre-trained embedding for text in a specific domain. At the end, the embedding is evaluated by various methods, which prove that both syntactic and semantic relations between toxic words are successfully encoded.

8 Criticism

Due to the limited resources, there are many possible improvements that can be made in this report. Here we list a few of them. First, phrase-level embedding has not been exploited. Second, a better preprocessing is needed. Third, The comparison between embedding models is far from complete, excluding many popular methods such as Glove, FastText and etc.

9 Appendixes

The code for this report can be found on the github repository[5].

[5] `https://github.com/JanFan/Toxic-Comment-Classification-Challenge`

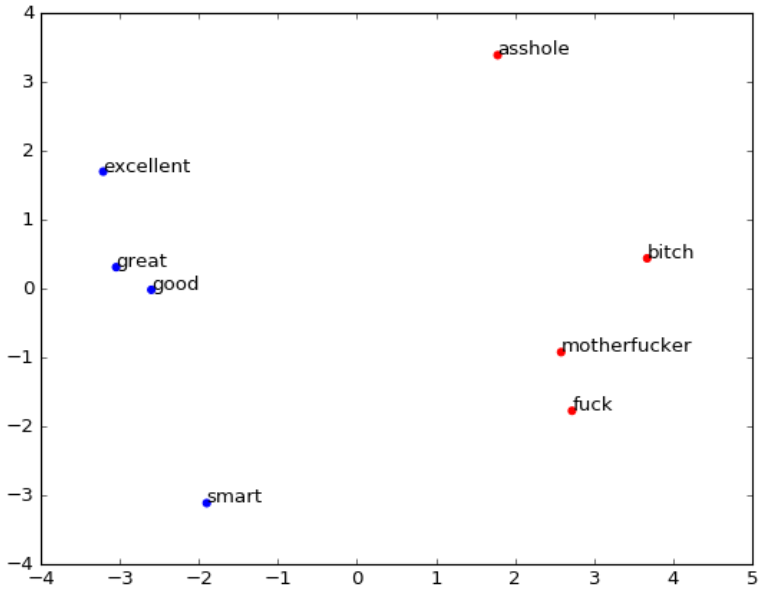

Figure 2: PCA Visualization for Good and Tocix Words

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196, 2014.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013b.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.