Sentiment Analysis based Language Model Evaluation

Aashish Yadavally
Institute of Artificial Intelligence
University of Georgia
ayadavally@uga.edu

Anirudh K. M. Kakarlapudi
Institute of Artificial Intelligence
University of Georgia
kakmaurya@uga.edu

Sumer Singh
Institute of Artificial Intelligence
University of Georgia
sumer.singh@uga.edu

Abstract

Language models assign probabilities to word sequences, and help guide and constrain the search among alternative word hypotheses. In this work, we performed an extrinsic evaluation of different language models ranging from the basic n-grams to the recently proposed language representation model BERT (Devlin et al., 2018). We embedded these models to the downstream task of sentiment analysis on the Toxic Comment Classification Challenge Dataset[1] presented by Google Jigsaw. Furthermore, we analyzed the performance of these models by comparing the predictions of the comments which contain the f-word, in cases where it was used in a toxic/non-toxic context, and how well the models have been able to interpret the nature of the word. Decoding the syntactic/semantic interpretations of the models helps investigate the importance of the incorporation of linguistic elements in language representations, and the need for future research to improve on such incorporation (Botha and Blunsom, 2014).

1 Introduction

Language modeling (LM) is a sub-task of a majority of Natural Language Processing (NLP) problems, which depends on predicting the probability of the next word, given a sequence of words. Currently, however, the field of NLP is moving from simple statistical techniques to more complex deep-learning based approaches. In such a scenario, contrasting the performance of LMs becomes interesting.

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

Traditionally, language model performance is measured by metrics such as perplexity, cross-entropy, and bits-per-character (BPC). However, metrics such as perplexity do not have a lower bound, thus making it difficult to judge the language model in the absence of an optimal value (Chen et al., 1998). Additionally, the quality of the syntactic predictions made by the LM is difficult to measure using perplexity, as perplexity rewards LMs primarily for collocational and semantic predictions (Shen et al., 2018).

An extrinsic mode of evaluation of the LMs includes embedding the model in a downstream application and comparing its performance on the downstream tasks. Sentiment analysis is one such downstream task, which refers to the use of NLP techniques to extract an opinion or view of a piece of text. For this purpose, current techniques fall broadly into two categories: knowledge-based techniques, that depend on manually created rules, and statistical techniques. As languages are highly complex, creating a rule-based system that covers every situation is virtually impossible. On the other hand, statistical techniques attempt to automatically create their own rules by processing large amounts of data.

The performance of statistical systems is contingent upon large amounts of good quality data. With the advent of the internet and high-performance computer hardware, statistical techniques have become highly effective. In this work, we evaluated the performance of several statistical systems on the application of sentiment analysis, for the Toxic Comment Dataset. The Toxic Comment dataset consists of Wikipedia comments and the task is to classify each comment into one or more of the following six toxic categories: ‘toxic’, ‘obscene’, ‘insult’, ‘identity hate’, ‘severe toxic’, ‘threat’.

We compared the performance of LMs ranging from the statistical n-grams to neural LM and BERT. In addition, we analyzed the comments in the dataset that contained the f-word, to analyze the subsentential structure learned by these models. In one study, we compared the predictions of the models for the comments containing the f-word against the ground truth labels for these comments, motivated to capture the toxic/non-toxic interpretability of the models. In another study, we analyzed the parts of speech of the f-word in different comments in the dataset, motivated to explore the effects of particular syntactic accordance of the word on the toxic/non-toxic nature.

2 Related Work

Failure of intrinsic methods of evaluation. The most widely-used intrinsic method for evaluation of language models is *perplexity*. It is defined as the probability of the test set normalized by the number of words in the test set. Intuitively, it evaluates the best language model as the one that best predicts an unseen test set. However, it is difficult to make the syntactic predictions using this metric. The failure to do so results in the inability of capturing structure by the models, which in turn results in the ambiguity of the sentence, which is a problem when extended to language understanding based downstream tasks.

Multi-label classification approaches. The distinction between multi-class classification and multi-label classification arises from the fact that the labels in the former are mutually exclusive, which isn't the case in the other. To tackle such a problem, approaches like one-vs-all (*Binary Relevance*) and *Label Powerset* are standard approaches (Cherman et al., 2011). If not, with the prediction of probabilities for each of the labels, metrics such as AUC-ROC can be used for comparing the performance of the models. This is a useful evaluation metric to distinguish between the classes, as it is independent of thresholding (Bradley, 1997). By analogy, an AUC-ROC score of 0.5 is the worst, i.e, there is no class separation.

Targeted Evaluation. LM evaluation datasets using classification tasks have been proposed in the context of semantics and discourse comprehension. Though specific datasets have been constructed for tasks such as parser evaluation (Rimell et al., 2009);(Nivre et al., 2010); (Bender et al., 2011), and machine translation systems (Sennrich, 2016), no general datasets exist that target a range of syntactic constructions for language model evaluation.

Syntax in LMs. There have been several proposals over the years to incorporate explicit syntax into LMs to overcome the inability of n-gram LMs to model long-distance dependencies in a sentence. While temporal models such as recurrent neural networks (RNN) model longer dependencies, they suffer from the vanishing gradient problem which, results in the failure of the models in capturing dependencies efficiently.

3 Toxic Comment Classification

The "Jigsaw Toxic Comment Classification Challenge" dataset consists of Wikipedia comments that are labeled by human raters for toxic behavior belonging to the following categories: 'toxic', 'severe toxic', 'obscene', 'threat', 'insult', 'identity hate'.

3.1 Dataset Statistics

The dataset includes a total of 312,735 instances, which have been annotated by the human raters. Of these instances, 159,571 belong to the train set and 153,164 to the test set. Each of the instances can simultaneously belong to multiple toxic behavior categories mentioned above. However, some comments also belong to none of the toxic categories, and can be labeled as 'clean comments'. As the contest is over, test labels are also provided. Note that all of our models are trained only on the training set and tested only on the test set. Test set labels are only used for analysis. The distribution of the classes in the train set is described in Table 1.

3.2 Common Challenges

- Each comment can belong to one or more of the categories, while each of them are highly correlated.

Classification	# of instances
clean	143,346
toxic	15,294
obscene	8,449
insult	7,877
identity hate	1,405
severe toxic	1,595
threat	478

Table 1: Distribution of labels in the train set

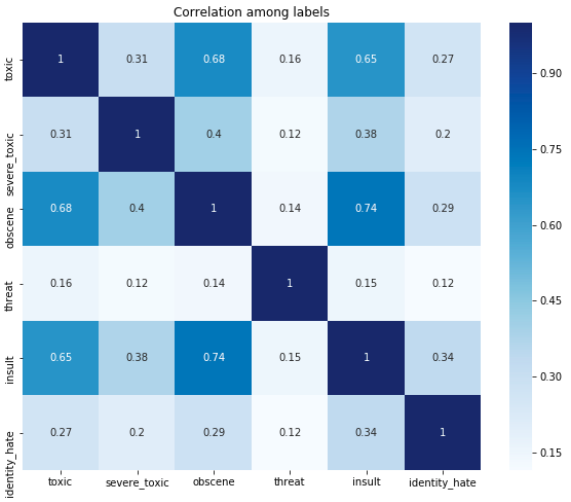


Figure 1: Correlation between labels in the dataset

- Presence of words that are not part of the training data, i.e, out-of-vocabulary words.

In Figure 2, the distribution of the number of sentences per comment is plotted, with the majority of the comments having 2 and 3 sentences per comment. However, ∼7.3% of the comments have greater than 10 sentences per comment and ∼29.8% of the comments have greater than 5 sentences per comment. This adds to the challenge in the dataset, considering the toxicity of the comment often depends on expressions made in the initial parts of the comment (van Aken et al., 2018).

3.3 Exploratory Analysis

Important features were computed for all six toxic categories[2] based on the TF-IDF scores for each of the categories, and the top-9 words with the

[2]Disclaimer: This report contains words and expletives which may be considered offensive, but the contents do not reflect the views of the authors and exclusively serve to explain challenges in linguistics.

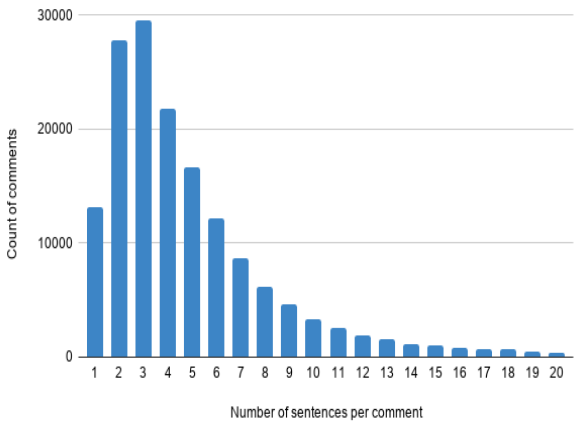


Figure 2: Number of sentences per comment

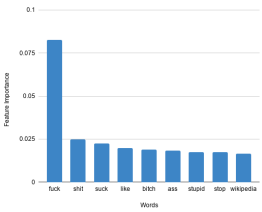
highest feature importance values were retrieved for the categories, as is shown in Figure 3. As can also be seen in Figure 1, the toxic categories are highly correlated with each other - in particular, obscene-toxic, insult-toxic, insult-obscene with correlation values of 0.74, 0.68, 0.65 respectively. This translates to common words having high feature importance across all the labels.

Furthermore, a similar analysis was conducted for the comments which had a negative ground truth for each of the toxic categories, tagged as "clean comments", so as to retrieve "clean words". These can be seen in Figure 4.

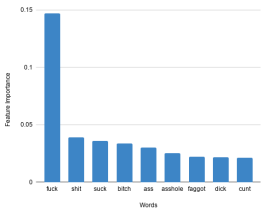
4 Methods

In this section, we study the baseline technique that was used to classify the comments. Furthermore, we describe a few statistical/machine learning models which were used to help improve the performance of the AUC-ROC metric, motivated to improve the capturing of the structural dependencies.

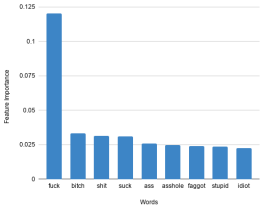
Baseline. To set up a baseline for our task, we computed the TF-IDF scores of the words for each class. Next, we created a list of top-n important words for each class, based on their TF-IDF scores. At prediction time, for each class - the model checks if any of the words in the comment are present in the top-n list for that class. If this is true, the model assigns 1 to that class, else zero.



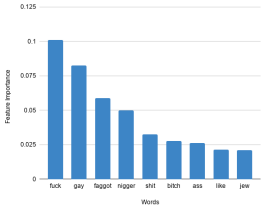
(a) “toxic” Word List



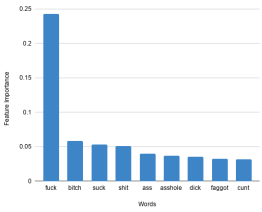
(b) “obscene” Word List



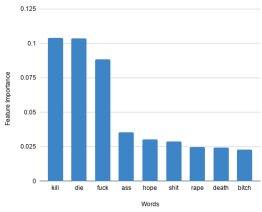
(c) “insult” Word List



(d) “identity hate” Word List



(e) “severe toxic” Word List



(f) “threat” Word List

Figure 3: Top-9 words list of toxic categories based on TF-IDF scores

We test for values of n starting from 5.

n-gram Models. An n-gram model models sequences using the statistical properties of n-grams, under the assumption that the words in the text are independent of each other. In this work, we used unigrams, bigrams and trigrams (i.e, when n= 1, 2, 3 respectively) with the Naive Bayes classifier to predict the probabilities of a comment belonging to each of the toxic categories.

Recurrent Neural Networks. The underlying assumption in the n-gram models is that the words in the text sequence are independent of each other, which is not the case with language. Temporal models such as recurrent neural networks are known to capture long-term dependencies efficiently (Sherstinsky, 2018). However, for words to be able to be used in machine learning frameworks, they need to be represented as a sequence of numbers. Thus, vector representation techniques such as word2vec (Mikolov et al., 2013) were developed, which is a group of related models that are used to build word embeddings.

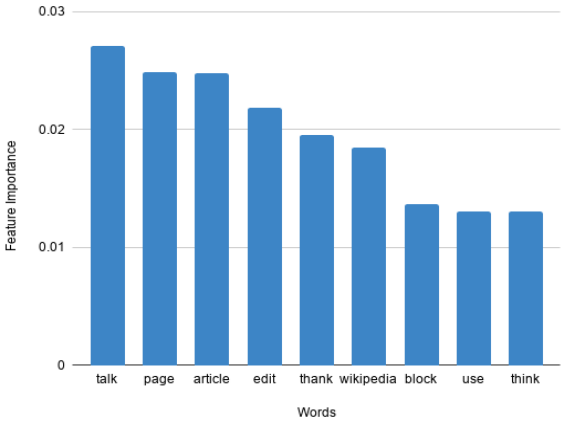


Figure 4: Top-9 clean words list based on TF-IDF scores

In this work, we used pre-trained word2vec word embeddings with the long-short term memory (LSTM) networks. In the specific case of language models, LSTMs are known to perform well, considering, for example, if it encounters a subject, it might want to output information relevant to a verb, in case that’s what follows next.

Bidirectional Encoder Representations from Transformers. Popularly refered to as BERT, these models make use of the attention mechanism in the Transformers architecture to learn the contextual relations between words in a text. Since BERT’s primary goal is to generate a language model, it is sufficient to use the encoders in the attention mechanism. While the directional models read the text in a particular direction (left-to-right or right-to-left, or a combination of both), BERT inputs the whole text as a sequence, because of which it can be considered bidirectional. In a nutshell, the BERT architecture replaces a certain percentage of the tokens in the text sequence with masks, intending to predict the words replaced by the masks in the context of the remaining unmasked words. We built on the code sample provided by google-research[3] to convert it for the task of multi-label classification. While there are two variants of the BERT model as proposed by the authors, *bert-base*, and *bert-large*, in this work, the former was used owing to the high computational requirements of the latter.

[3]https://github.com/google-research/bert

5 Experimentation

5.1 Comparison of the Models

For the baseline models, we observed that the best performance of such a technique was recorded while using the Top-10 words to predict the labels. One other observation was that the performance was constant upon increasing the number of such words beyond 20.

We observed that the Naive Bayes classifier performed better with unigrams, and the performance degraded with bigrams, and even more with trigrams. This is against the general trends where the performance usually improves by increasing the number of neighboring words. From the word lists containing the top toxic words contributing to the toxic categories, it can be observed that these words are majorly nouns. Nouns usually don't occur alongside each other, and even when they do, the first word remains a noun but gains characteristics of a modifier, modifying the second noun. Thus, by increasing the number of consequent words in the n-gram models, the performance might have dropped.

Method	**AUC-ROC (0-100)**
Presence of Top-5 words	60.23
Presence of Top-10 words	61.78
Presence of Top-20+ words	60.22
Naive Bayes using Unigrams	88.19
Naive Bayes using Bigrams	84.24
Naive Bayes using Trigrams	82.88
LSTM + Word2Vec	97.77
BERT	98.24

Table 2: Comparison of model performances

It is usually considered that word embeddings are ineffective in NLP tasks which rely heavily on contextual information, as each word can have different meanings when used in different contexts, and using the same vector representation for the word in every context wouldn't capture as much contextual information. However, this wasn't the case for this downstream task, as is justified by the significant improvement in performance ($\sim$9.58%) from the unigram model to the LSTM network trained with pre-trained *word2vec* word embeddings. This improvement in performance also acts as evidence to our previous assertion about the presence of stand-alone words in the dataset as a major contributing factor to be classified as one or more of the toxic category labels.

In general, LSTM networks can capture context only in the backward direction. In order to capture the contextual information in either direction, BERT is a useful model, as it inputs the whole text as a sequence. In this work, we experimented with the *bert-base* model, finetuning it with hyperparameters such as *maximum sequence length* and *learning rate*. The former describes the maximum length a sentence can take. Those sentences which have a length longer than this parameter are truncated and for those with the length shorter than this parameter, zeros are padded. Our best performance as recorded in Table 2 was achieved with a learning rate of $1 * 10^{-5}$ and a max sequence length of 120. However, taking into consideration the marginal difference in performance improvement from the LSTM networks to BERT, which is the state-of-the-art for most NLP tasks today, it would be interesting to note the performance of improved or toxicity-specific word-embeddings trained with LSTM networks against the performance of BERT achieved, for this particular task.

5.2 f-word Analysis

Based on the exploratory data analysis, it was observed that the f-word was most important in all the Top-9 word importance lists among each of the toxic categories. Additionally, the feature importance of the f-word was high in the clean comments as well. This intrigued us to compare the performance of the models on comments which contained the f-word in any form. Thus, we retrieved all such comments, and there were 7150 such instances in all. From the test labels, as shown in Table 2, we further segregated them into toxic and non-toxic categories, wherein the former refers to the comment falling into at least one toxic category; and the latter refers to the comment falling into none of the toxic categories.

5.2.1 Comparison of Model Predictions

By converting the predictions of each of the models for the comments containing the f-word into

Nature of comment	# of instances
toxic	6526
non-toxic	624

Table 3: Distribution of comments from the test set containing the f-word

"toxic" and "non-toxic" based on the segregation as described above, we compared the performances of the models on the test set by computing classification metrics such as f-score, precision, and recall, as is shown in Table 4. The general trend of toxic comments which contain the f-word is in line with model performance found in Table 2. What's interesting is that the Unigram model has a higher f-score compared to the LSTM and BERT models on the non-toxic f-word comments. This is reasonable and can be attributed to the low number (624) of training samples of non-toxic f-word comments. More complex models require large amounts of training samples to perform optimally. Therefore, in this case, the simpler model (Unigram) is outperforming the complex (LSTM, BERT) models on this specific class of samples. This result leads us to believe that complex models should perform better after the use of certain data augmentation techniques.

Language Model	Precision	Recall	F-Score	Nature of f-word
Unigram	0.97	0.74	0.84	toxic
	0.23	0.78	0.35	non-toxic
Bigram	0.99	0.52	0.68	toxic
	0.16	0.94	0.27	non-toxic
Trigram	0.99	0.44	0.61	toxic
	0.14	0.96	0.25	non-toxic
Neural Network	0.93	0.94	0.93	toxic
	0.31	0.30	0.30	non-toxic
BERT	0.95	0.95	0.94	toxic
	0.29	0.30	0.31	non-toxic

Table 4: Distribution of labels in the dataset

5.2.2 Investigating the effect of negative connotation words in f-word comments

To predict the effect of negative connotation words in comments containing the f-word, we compute the feature importance (TF-IDF scores) on such samples. On non-toxic f-word comments, negative words such as "neither", "nor" and "not" have high feature importance values of 0.18, 0.15 and 0.11 respectively. On performing a similar analysis on the toxic comments containing the f-word, we find that the feature importance of the "neither" and "nor" is negligible, and that of "not" is 0.05.

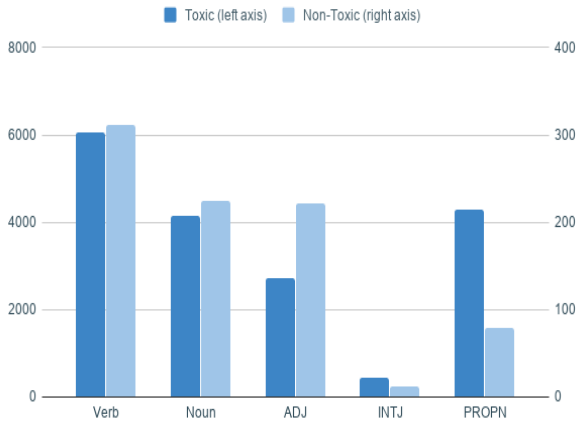


Figure 5: Frequency of POS tags of the f-word sentences in the test set in both toxic and non-toxic context

These values indicate that these negative connotation words have a prominent effect in neutralizing the toxic effect of the f-word, but not the other way around. However, the low f-scores of the models on non-toxic f-word samples indicate the failure of the models in capturing the neutralizing effect of such negative words. In the case of neural networks, it would be interesting to observe the change in information capture of these models by increasing the number of non-toxic f-word samples.

5.2.3 Analyzing the POS of f-words and their toxicity

In this section, we wanted to analyze the parts of speech of different f-words in each of these comments, so as to be able to compare the tags of the f-word in both the toxic and non-toxic context. To retrieve the POS tags of these sentences, we used the spacy[4] POS Tagger (en_core_web_sm), which is a multi-task CNN trained on OntoNotes in the English language and has an accuracy of 97.15% on the original set. From these tags, we weren't able to realize any conclusive evidence in terms of the POS tags' effect on the toxicity nature of the f-word. One observation was that the f-word was tagged as "PROPN", which refers to a proper noun, several times. Naturally, this is incorrect. Thus, we believe that there is scope for improvement by using a better POS tagger, which can result in analyzing the POS tags conclusively.

[4]https://spacy.io/

6 Conclusion

In this work, we presented multiple models for the extrinsic evaluation of the language models on the downstream task of toxic comment classification. We showed that the powerful neural network models such as LSTMs embedded with pre-trained *word2vec* vectors and language representation models such as BERT learn the toxic nature of the comments better than the statistical n-gram models, though the improvement from the LSTM-based models to the BERT model was only marginal. We performed an analysis on the comments containing the f-words, to investigate the toxic/non-toxic usage of the word in the comments. We were able to specifically show that the neural network models were able to capture the toxic nature of the f-word better than the other models, while the Unigram model captured the non-toxic nature of the f-word better, owing to the lower number of comments containing the f-word in a non-toxic context. Furthermore, we also analyzed the important features of the comments containing the f-words to show the effect of negative connotation words such as "neither", "not" and "nor", which resulted in the non-toxic nature of the f-word in that particular comment. Finally, we suggest further research in representing linguistic elements in the feature representations to improve the context understanding of the toxic words, and also of the words/structures used in a negative connotation, so as to improve the precision-recall values of the models.

Acknowledgments

We would like to thank Dr. John T. Hale for his valuable inputs and guidance throughout the course of the project. We are also grateful to the *Institute of Artificial Intelligence* for providing the computational resources for the successful execution of the project.

References

Emily M Bender, Dan Flickinger, Stephan Oepen, and Yi Zhang. 2011. Parser evaluation over local and non-local deep dependencies in a large corpus. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 397–408. Association for Computational Linguistics.

Jan Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *International Conference on Machine Learning*, pages 1899–1907.

Andrew P Bradley. 1997. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159.

Stanley F Chen, Douglas Beeferman, and Ronald Rosenfeld. 1998. Evaluation metrics for language models. Citeseer.

Everton Alvares Cherman, Maria Carolina Monard, and Jean Metz. 2011. Multi-label problem transformation methods: a case study. *CLEI Electronic Journal*, 14(1):4–4.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

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

Joakim Nivre, Laura Rimell, Ryan McDonald, and Carlos Gomez-Rodriguez. 2010. Evaluation of dependency parsers on unbounded dependencies. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 833–841. Association for Computational Linguistics.

Laura Rimell, Stephen Clark, and Mark Steedman. 2009. Unbounded dependency recovery for parser evaluation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 813–821. Association for Computational Linguistics.

Rico Sennrich. 2016. How grammatical is character-level neural machine translation? assessing mt quality with contrastive translation pairs. *arXiv preprint arXiv:1612.04629*.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. 2018. Ordered neurons: Integrating tree structures into recurrent neural networks. *arXiv preprint arXiv:1810.09536*.

Alex Sherstinsky. 2018. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *arXiv preprint arXiv:1808.03314*.

Betty van Aken, Julian Risch, Ralf Krestel, and Alexander Löser. 2018. Challenges for toxic comment classification: An in-depth error analysis. *arXiv preprint arXiv:1809.07572*.