EasyChair Preprint

№ 8512

Detection of Online Toxic Comments Using Deep Learning

K Jagadeesh, G Himabindu, P Bhargav, B Imran and G Indiravathi

July 20, 2022

Detection of Online Toxic Comments

Using Deep Learning

Jagadeesh K, Siddharth Institute of Engineering & Technology, jgdshkovi@gmail.com
Himabindu G, Siddharth Institute of Engineering & Technology, himabindhu403@gmail.com
Bhargav P, Siddharth Institute of Engineering & Technology, payyalabhargav20@gmail.com
Imran B, Siddharth Institute of Engineering & Technology, imranbodanapu786@gmail.com
Mrs. G Indiravathi, Assistant Professor, Siddharth Institute of Engineering & Technology, remala.indu@gmail.com

1. Abstract

Toxic comments are disrespectful, abusive. Unreasonable online comments that usually make other users leave a discussion. The danger of online bullying and harassment affects the free flow of thoughts by restricting the dissenting opinions of people. Sites struggle to promote discussions effectively, leading many communities to limit or close down user comments altogether.

We will systematically examine the extent of online harassment and classify the content into labels to examine the toxicity as correctly as possible. We will aim at examining the toxicity with high accuracy to limit down its adverse effects which will be an incentive for organizations to take the necessary steps like reporting the user or blocking the user.

Keywords: Online hate, Toxicity, Social Media, Deep Learning

2. Introduction

Online hate, described as abusive language, insults, personal attacks, threats or toxicity, has been identified as a major threat on online social media platforms.

There are many billions of text data that's being generated every day by in-apps messages, social media platforms, forums, blogs etc. All these channels are constantly generating large amounts of text data every second. Because of the large volumes of text data as well as unstructured data sources, we can no longer use the common approach to understand the text and this is where NLP comes in. With the increasing amount of text data being generated every day, NLP will only become more and more important to make sense of the data and used in many applications.

Social Media Platforms (SMPs) are the most prominent grounds of toxic behaviour. Even though they provide ways to flag offensive and toxic content, only 17% of all the adults have flagged harassment conversations, and only 12% have reported someone of such acts.

Manual techniques like flagging are neither effective nor easily scalable and have a risk of discrimination under subjective judgments by human annotators. Since an automated system can be much faster than human footnotes, machine learning and deep learning models to automatically detect online hate have been gaining popularity and bringing researchers from different fields together.

To address these concerns, we propose to develop an online hate classifier using state-of-the-art NLP models like Bidirectional Encoder Representations from Transformers (BERT), GPT (Generative Pre-trained Transformer) etc,. We perform transfer learning, making use of pretrained models, which reduces the cost and time for training.

3. Literature Review

The use of statistics in NLP started in the 1980s and acclaimed the birth of what we called Statistical NLP or Computational Linguistics. Since then, many machine learning techniques like Naïve Bayes, k-nearest neighbours, hidden Markov models, decision trees, random forests, and SVMs have been applied to NLP.

The use of neutral networks for Natural Language Processing didn't start until the early 2000s. Neural networks transformed NLP, enhancing or even replacing earlier techniques by the end of 2010. This has been made possible because of two reasons: (a) we

now have more data to train neural network models and (b) more powerful computing systems to do so. In traditional NLP, features were hand-crafted, and thus time consuming to create. Neural networks can learn multilevel features automatically. They also give better results.

The main innovations that have enabled us to use neural networks in NLP are :

i. Word embeddings and
ii. NN architectures.

Early language models used a feedforward NN or CNN architectures, but these didn't capture context. A Context is how a word occurs in relation to surrounding words in the sentence. To capture context, RNNs were applied. LSTM, a variant of RNN, was then used to capture long-distance context. Bidirectional LSTM(BiLSTM) improves upon LSTM models by looking at word sequences in forward and backward directions.

Sequence models like LSTM, GRUs could also be used but their model architecture prevents parallelization.

These challenges are addressed by using state of the art Transformers like BERT, GPT, T5 etc. Taking the context into consideration and handling long-range dependencies with ease.

4. Approaches & Models

For a long time, majority of methods used to study NLP problems included shallow machine learning models and time-consuming, hand-crafted features. This lead to problems such as the curse of dimensionality since linguistic information was represented with sparse representations (high-dimensional features). However, with the success of word embeddings, neural-based models have achieved superior results on various language-related tasks as compared to traditional machine learning models like SVMs or logistic regression.

4.1 Probabilistic Models

Determines the probability of occurrence of a word in the sentence based on the frequency of other words.

4.1.1 Naïve Bayes

Bayes' Theorem is useful when we are working with conditional probability, because it provides us with a way to reverse them:

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

This model assumes that every word in the sentence is independent of the other ones. This means that the model is no longer looking at entire sentences, but rather at individual words.

The Naïve Bayes model just counts the frequency of words in the given sentence.

Techniques to enhance Naïve Bayes:

- **Using n-grams**: Instead of counting single words as we did here, we could count sequences of words, like “clean match” and “close election”.
- **Using TF-IDF**: Instead of just counting frequency we could do something more advanced like also penalizing words that appear frequently in most of the texts.

4.1.2 Drawbacks of Naïve Bayes :

i. assumption of independent predictor features
ii. Zero Frequency problem, when encountering a word not present in the corpus, which can be handled by Laplacian smoothing.

4.2 Sequence Models

Recurrent Neural Networks (RNNs) are a form of machine learning algorithm that are suited for sequential data such as text, time series data, financial data, speech, audio & video etc. RNNs are ideal for solving problems where the sequence (ordering of words) is more important than the individual items themselves.

An RNNs is essentially a fully connected neural network that contains a refactoring of some of its layers into a loop.

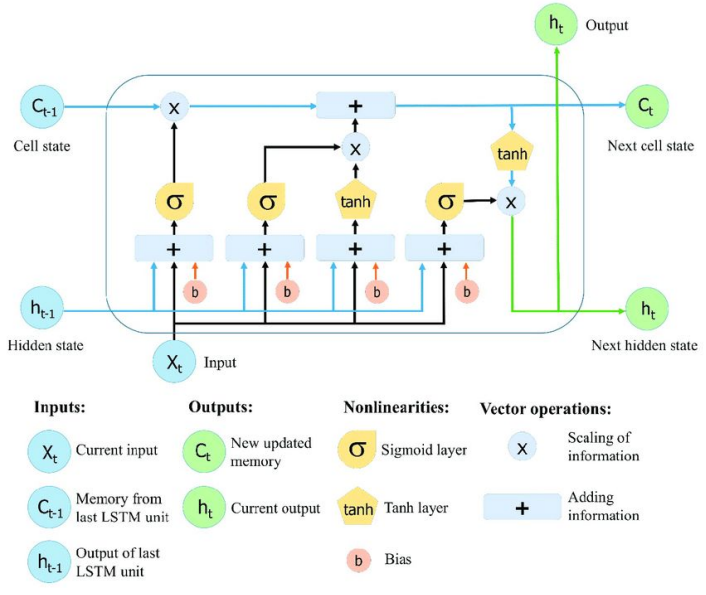
The loss landscape increases with the increase in the number of layers of RNNs and can become impossible to train, this is the vanishing gradient problem. To solve this problem a Gated Recurrent Unit (GRU) and Long Term Short Term Memory (LSTM) networks are used.

4.2.1 LSTM

An RNN has short term memory. When used in combination with LSTM Gates, the network can have long term memory.

An LSTM also has a cell state as well, alongside the hidden state. This cell state is the long

term memory. Rather than just returning the hidden state at each iteration, a tuple of hidden states are returned consisting of the cell state and hidden state.



Long Short Term Memory (LSTM) has three gates:

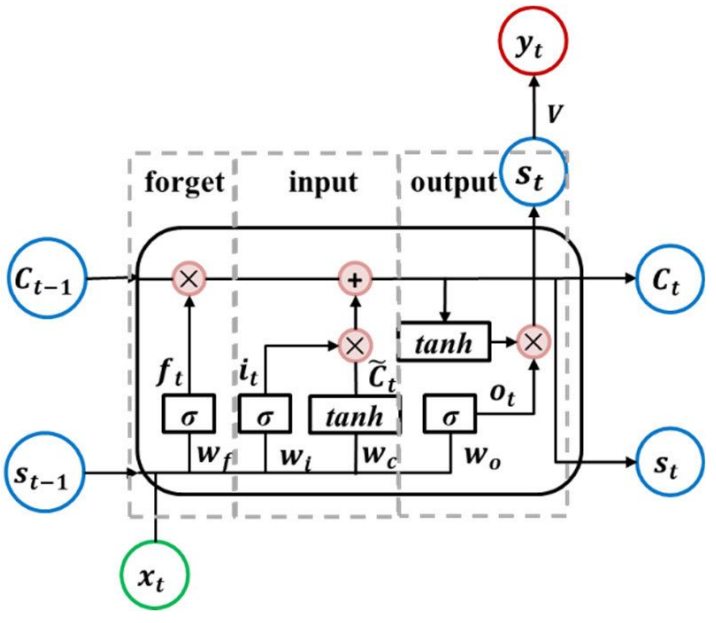
1. An Input gate, controls the information input at each time step.
2. An Output gate, controls how much information is outputted to the next cell.
3. A Forget gate, controls how much data to lose at each time step.

4.2.2 GRU

A gated recurrent unit is also called a gated recurrent network.

At the output of each iteration there is a small NN with three neural networks layers implemented, consisting of the recurring layer from the RNN, a reset gate and an update gate. The update gate acts as a forget and input gate. The coupling of these two gates performs a similar function as the three gates forget, input and output in an LSTM cell.

Compared to LSTM, a GRU has a merged cell state and hidden state, whereas in LSTM these cell states are separate.

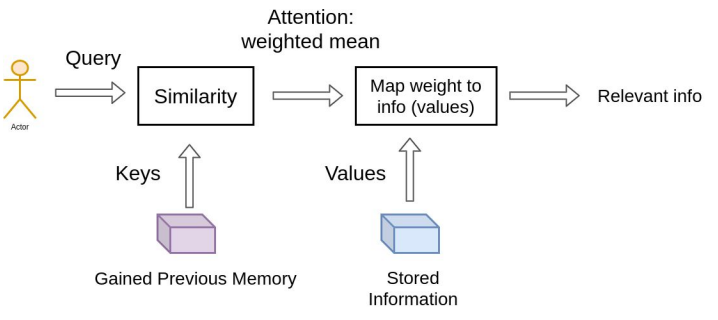


Gated Recurrent Unit (GRU) has the following gates:

1. The reset gate, takes the input activations from the previous layer, and multiplied by a reset factor between 0 and 1. This reset factor is calculated by a NN with no hidden layer.
2. The update gate controls how much of new input to take and how much of the hidden state input to take.

4.3 Attention Models

All problems of *RNNs* and *LSTMs* can be traced back to the context vector, which is a bottleneck of the whole system. One solution was proposed by Minh-Thang Luong back in 2015 called *attention*, which allows the model to focus on the relevant and important parts of the input sequence as needed.



The Transformer is a model that uses attention to boost the speed with which the models can be trained. The biggest benefit of the Transformer is how it lends itself to parallelization.
From a high-level, the transformer model is comprised of two sub-models: an encoder and a decoder.

- **Encoder**: The encoder is responsible for stepping through the input and encoding the entire sequence into a fixed length context vector.

- **Decoder**: The decoder is responsible for stepping through the output while reading from the context vector.

The encoder and decoder blocks are actually multiple identical encoders and decoders stacked on top of each other. Both the encoder and the decoder stacks have the same number of units. The number of encoder and decoder units in the transformer is a hyperparameter (the authors used 6 units).

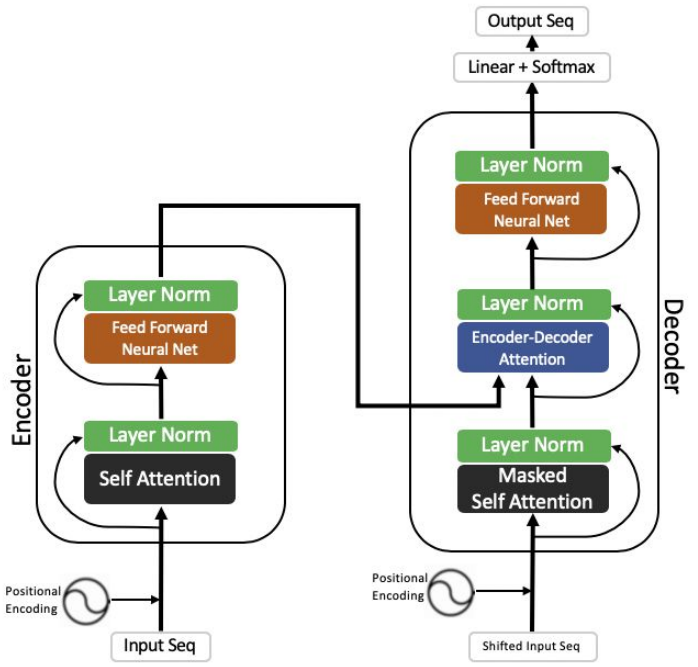


Fig. Transformer

The setup of the encoder and the decoder stack works as follows :

- The word embeddings of the input sequence are passed to the first encoder
- These are then transformed and passed on to the next encoder
- The output from the last encoder in the encoder-stack is passed to all the decoders in the decoder-stack as shown.

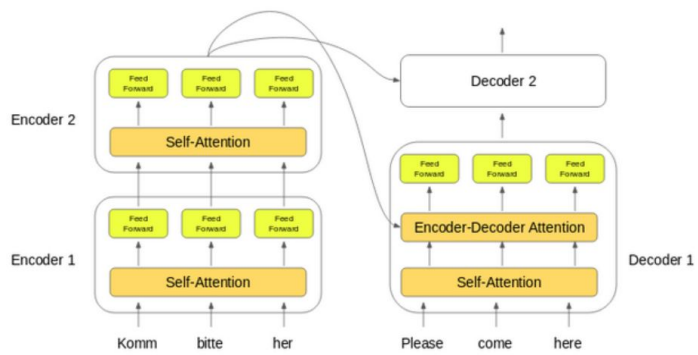


Fig. Propagation of Word Embeddings and Attention in Transformer

One major advantage of the transformer model architecture, is that at each step we have direct access to all the other steps (called self-attention), which leaves no room for information loss, as far as message passing is concerned. On top of that, we can look at both future and past elements at the same time, which also brings the benefit of bidirectional RNNs, without the 2x computation needed. All of this happens in parallel (in a non-recurrent manner), which makes both the training/ inference much faster.

The self-attention with every other token in the input means that the processing will be in the order of $O(N^2)$. It is going to be costly to apply transformers on long sequences, compared to RNNs. That's probably one area that RNNs still have an advantage over the transformer models.

Some of the leading Language models include:

- BERT (Bidirectional Encoder Representations from Transformers)
- GPT (Generative Pre-trained Transformer)
- XLNet (Generalized Auto-Regressive model for NLU)
- ALBERT (A Lite BERT)

5. Datasets

We applied the following criteria to select the datasets for this research: (a) the language is English, and (b) the dataset and available details on the annotation procedure passed a manual evaluation.

Wikipedia dataset (KAGGLE‑18)

The Wikipedia talk page Corpus includes datasets for three different categories: personal attack, toxicity, and aggression. The corpus is extracted from approximately 63M comments processed from the public dump of English Wikipedia dated till 2015. We refer to this dataset as “KAGGLE-18”, because it was published by Jigsaw (a subsidiary of Alphabet, Google’s parent company) as part of a Kaggle data science competition called “Toxic Comment Classification Challenge”.

For this study, we used only the toxicity dataset consisting of 159,571 annotated com-ments, publicly available for download. As is the case with the Reddit dataset, the Wikipedia dataset uses the term toxicity in the same sense as we use hatefulness in this work. For the toxicity task, the workers annotated comments based on the perceived toxicity and likelihood of making others leave the discussion.

Distribution of Hate Comments in the Dataset:

Toxic	Non-Toxic	Total
19,571	140,000	159,571
12.2%	87.8%	100%

Examples of hateful comments :

- All of my edits are good. Cunts like you who revert good edits because you're too stupid to understand how to write well, and then revert other edits just because you've decided to bear a playground grudge, are the problem. Maybe one day you'll realise the damage you did to a noble project.

- You should do something nice for yourself, maybe go grab a couple of Horny Goat Weeds from your local convenience store and jack off for a little longer than 3 min tonight.

- I'm sorry I screwed around with someone's talk page. It was very bad to do. I know how having templates on their talk page helps you assert your dominance over them. I know I should bow down to almighty administrators. But again, I'm going to go play outside....with your mom.

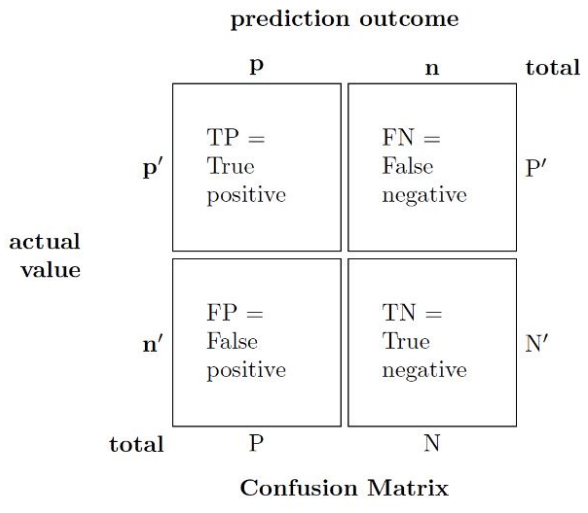
- Would you both shut up please, you donot run wikipedia, especially a stupid kid.

---

6. Metrics and Comparison

6.1. Confusion Matrix

It is a very informative performance measure for classification tasks. Ci,j an element of matrix tells how many of items with label i are classified as label j. Ideally we are looking for a diagonal Confusion matrix where no item is miss-classified.



Confusion Matrix

The matrix above is a good representation for our binary classification. Positive (p) represents toxic label and n(negative) represents non-toxic label.

6.2. BLEU score

The Bilingual Evaluation Understudy Score (BLEU), is a metric for evaluating a generated sentence to a reference sentence.

In this approach, we count the matching n-grams in the candidate translation to n-grams in the reference text, where 1-gram or unigram would be each individual token and a bigram comparison would be each word pair. The comparison is made regardless of word order. BLEU scores are based on an average of unigram, bigram, trigram and 4-gram precision.

Problems with BLEU:

BLEU score doesn't consider meaning of word, sentence structure or handle morphologically-rich languages well.

Two popular methods to help address some shortcomings of BLEU are:

- NIST, weights n-grams based on their rareness. Correctly matching a rare n-gram improves your score more than correctly matching a common n-gram.
- ROUGE, a modification of BLEU that focuses on recall rather than precision. It looks at how many n-grams in the reference translation show up in the output, rather than the reverse.

6.3. GLUE Benchmark

The General Language Understanding Evaluation (GLUE) benchmark is the collection of resources for training, evaluating, and analyzing natural language understanding systems.

GLUE consists of:

- A benchmark of nine sentence- or sentence-pair language understanding tasks built on established existing datasets and selected to cover a diverse range of dataset sizes, text genres, and degrees of difficulty,
- A diagnostic dataset designed to evaluate and analyze model performance with respect to a wide range of linguistic phenomena found in natural language, and
- A public leaderboard for tracking performance on the benchmark and a dashboard for visualizing the performance of models on the diagnostic set.

The format of GLUE benchmark is model-agnostic, so any system capable of processing sentences and producing corresponding predictions is eligible to participate. The benchmark tasks are selected so as to favor models that share information across tasks using parameter sharing or other transfer learning techniques. The ultimate goal of GLUE benchmark is to drive research in the development of general and robust natural language understanding systems.

6.4. Complexity Comparison*

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

n: is the length of the input sequence.
d: is the dimension of the representation (256, 512, 1024 in general).
k: is the size of the kernel.
r: is the size of the neighborhood in attention.

6.5. Quality Comparison*

Experiments on two tasks of translation (a large number of sentence pairs are translated (36 million sentences for English-French), and the quality of translation score is calculated.) showed that these models generate better translations, while performing parallel calculations and requiring smaller training time than other models. The quality evaluation is done using the BLEU score.

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

*[1] Attention Is all You need.

6.6. Scalable Architecture

From a processing standpoint, any data science solution must also be considerate of the processing load capacity of the platform being used.

Even with a relatively small training data set, one struggles to process intensive text vectorization workload, on a single CPU instance. Using GPU instances one can easily have access to powerful computation provided for affordable prices by cloud platforms such as AWS, Azure and GCP.

7. Conclusion

The ability to readily and accurately identify comments as toxic could provide many benefits while mitigating the harm. Also, our research has shown the capability of readily available algorithms to be employed in such a way to address this challenge.

This research work focuses on developing a model that would automatically classify a comment as either toxic or non-toxic using attention models.

We saw how powerful the Transformers are compared to the RNNs for translation tasks. We are excited about the future of attention-based models.

8. References

[1] Attention Is All You Need , Google Brain and Google Research. https://arxiv.org/pdf/1706.03762.pdf

[2] Classification of Online Toxic Comments Using Machine Learning Algorithms, Rahul, Harsh Kajla, Jatin Hooda, and Gajanand Saini, DOI: 10.1109/ICICCS48265.2020.9120939

[2] "Developing an online hate classifier for multiple social media platforms", Joni Salminen, Maximilian Hopf, Shammur A. Chowdhury, Soon-gyo Jung, Hind Almerekhi & Bernard J. Jansen https://doi.org/10.1186/s13673-019-0205-6

[4] Zaheri, Sara; Leath, Jeff; and Stroud, David (2020) "Toxic Comment Classification," SMU Data Science Review: Vol. 3 : No. 1 , Article 13. Available at: https://scholar.smu.edu/datasciencereview/vol3/iss1/13

[5] "Identification and Classification of Toxic Comments on Social Media using Machine Learning Techniques", P. A. Ozoh, A. A. Adigun, M. O. Olayiwola, IJRIAS, Volume IV, ISSN 2454-6194

[6] "A Web of Hate: Tackling Hateful Speech in Online Social Spaces", H. M. Saleem, K. P. Dillon, S. Benesch, and D.Ruths, http://arxiv.org/abs/1709.10159.