Sentiment Analysis of CyberBullying data

Pradeep Pujari
pradeep.pujari@gmail.com

Abstract

Social media is here to stay. As a result, cyberbulling has became a terrible menance of modern times. Cyber bullying is a kind of harassament that takes place in social networks. The problem has been growing since the outset of social networks. Sentiment Analysis has the potential not only to detect bullying phrases but also victims who pose high risk to themselves or others. My work focuses on using deep learning and natural language understanding methods to detect bullying traces in social media posts. I designed the netwrok using Recurrent Neural Network (RNN), LSTM cell with BERT embedings and second, replaced embedings layer with recently released embedings API from OpenAI and finally compared their performance.

1 Introduction

Bullying, also called peer vicimization has been recognised as a serius health issue by The While House (2011), the American Academy of Pediatrics (2009) and the American Psychological Association (2004). Cyber bullying is a kind of harassament that takes place in social networks. Also, Cyber Bullying percentage is higher than in person Bullying because it is easier to hide behind a screen name. Although it has been an issue for many years, the recognition of its impact on young people has recently increased. Bullying at school, college and university is increasing worldwide. Given its importance in society and work places, to detect and prevent in a timely manner, I designed a Recurrent Neural Network. However, a key challenge is dataset acquisition and analysis.Cyber Bullying text often employ that are charged with positive and negative connotations. The task here is to detect the sentences that contains bullying tokens and to the polarity and the classes. Development of Natural Language Understanding (NLU) algorithms aimed at the detection of cyberbullying is one of the ways to achieve that goal. Such algorithms require annotated data at least to measure their performance. Moreover, the most popular Machine Learning (ML) algorithms, Deep Neural Networks (DNN) in particular, require large annotated corpora in order to obtain high quality classification results. Unfortunately, there are only a few publicly available dataset for cyberbullying detection:

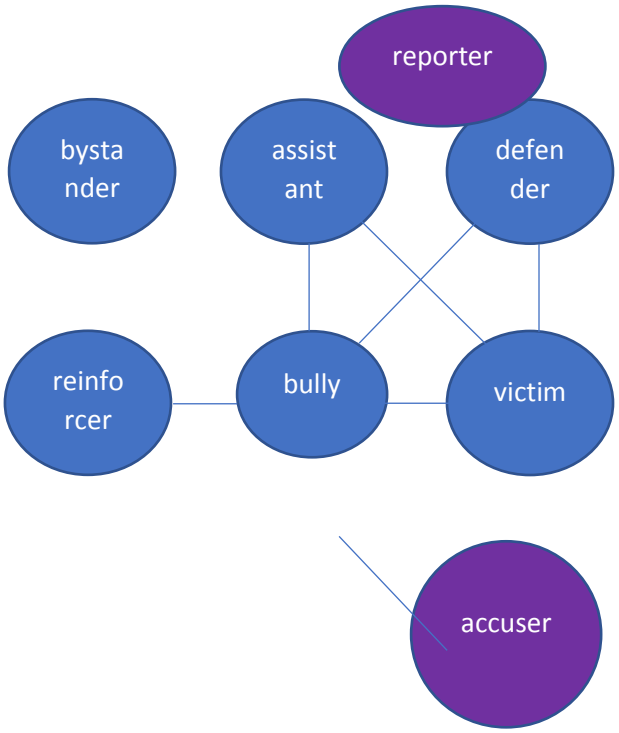


Figure 1: Roles is a bullying episode blue circles repreent roles from social science while purple are new roles added for social media analysis

- Kaggle Formspring data for Cyberbullying Detection
- MySpace Group Data Labeled for Cyberbullying

2 Related Work

The Embedings are numerical representations of concepts converted to number sequences, which make it is easy for a computer to understand the relationships between those concepts. Embedings that are numerically similar are also symantically similar. Embeddings are useful for working with natural language and code, because they can be readily consumed and compared by other machine learning models and algorithms. Embedding models are explicitly optimized to learn a low dimensional representation that captures symantic meaning of the input data. Mainly, we use unsupervised learning like word2vec, Glove, BERT etc. Text embeding is used in many applications like classification, semantic search, clustering to name a few. Now, OpenAI provides a new end point for embeddings based on GPT-3 model. The new embeddings endpoint in the OpenAI API provides text and code embeddings with a few lines of code as below:

```
import openai
response = openai.Embedding.create(
input="canine companions say",
engine="text-similarity-davinci-001")
print(response
{
"data": [
{
"embedding": [
0.000108064,
0.005860855,
-0.012656143,
...
-0.006642727,
0.002583989,
-0.012567150
],
"index": 0,
"object": "embedding"
}
],
"model": "text-similarity-babbage:001",
"object": "list"
  }
```

Grokking is a phenomenon when a neural network suddenly learns a pattern in the dataset and jumps from random chance generalization to perfect generalization very suddenly. Training accuracy goes high and validation accuracy close to zero for a long time and suddenly meets the training accuracy for a small data set. The Grokking paper is a result of research at the OPEN AI labs, and it talks about something very important, generalization and over-fitting. The goal of the paper is to know about data efficiency, memorization, generalization, and speed of learning.

3 Data

Nandhini and Subha [2] had proposed a technique for detecting cyber bullying using data from MySpace. They reported an accuracy of 91%. Sentiment analysis is widely studied in natural language processing and has been applied in many business and social domains (Liu and Zhang, 2012) but very rare to detect bullying. So availabilty of large and clean data set is is difficult to find.

I collected cyberbullying data from Formspring. Formspring is a site where people can comment annonymously about each other. Formspring.me, a question-and-answer formatted website that contains a high percentage of bullying content. The data was labelled using Mechanical Turk. The Kaggle Formspring dataset was originally annotated using the Mechanical Turk service. The methodology behind the annotation process was pretty simplistic [1]. Namely, a post was marked as containing cyberbullying if two of the annotators indicated that it contains that phenomenon. Moreover, the annotators did not have any training towards the detection of cyberbullying. As a result, the annotation has a moderate quality (which we discuss in Section 2.1). Formspring data for Cyberbullying Detection (a large unlabeled Formspring dataset, from a Summer 2010 crawl [1]), available on Kaggle and prepared by Kelly Reynolds was chosen as a train dataset. The main reasons for choosing this dataset were:

- Its comparatively large size: 12772 data samples.
- The fact that it is fairly well-known dataset with an initial release in October, 2016, and the last update in January, 2017.
- The fact that the number of sentences with bully contents reflects the real-world proportions of cyberbullying and

nocyberbullying content (more than 84% of samples were labeled as "no cyberbullying").

- The option of anonymity that encourages cyberbullying and other harmful behaviors (Formspring allowed users to post questions anonymously to any other user's page).

3.1 Insights about the data set

Each sample was labeled by three Amazon's Mechanical Turk workers with "yes" and "no" answers for a question if it contains cyberbullying. The cyberbullying was also tagged for severity from 0 (no bullying) to 10. A post was considered harmful if at least two out of three annotators answered "yes" for the primary question. As a result 802 samples out of 12772 (6.3%) were classified as "cyberbullying"

An analysis of the annotated samples showed that at least 2.5% of posts classified as "no cyberbullying" should be labeled as "cyberbullying" due to their possible harmful impact. This is a noticeable amount compared to the percentage of the samples labeled as "cyberbullying". Similarly, about 15-20% of the samples labeled as "cyberbullying" could be labeled as "no cyberbullying". Therefore, it is better to re-annotate the whole Formspring dataset. But due to lack of time I used the dataset as it is.

Table 1: The number of correctly annotated samples (out of 30) for the people that performed the annotation.

ID	Correct Labels	Percentage
01	**24**	**80.00%**
02	**24**	**80.00%**
03	**23**	**76.70%**
04	**22**	**73.30%**
05	**22**	**73.30%**
06	**21**	**70.00%**
07	**21**	**70.00%**
08	**20**	**66.70%**

The dataset contains approximately 300 thousand of tokens, making it rather small regarding current ML trends. There are no big differences in length between the posted questions and answers (approx. 12 words). On the other hand, the harmful samples are usually a bit shorter than the non-harmful samples (approx. 23 vs. 25 words). The number of harmful examples is small, amounting to only 7%, yet it seems to be rather high compared to the average content of social networks.

Userid – user id of a person giving the answer to a post
Post – The post and it's reply. Separated by Q: and A.
Ques and ans – The question and answer.
The same information is available in post, so we drop post
Asker – id of the person asking the question
Ans#, severity#, bully# - answer by Mechanical Turk, severity score assigned, bully word/phrase

4 Models

SpacyTextBlob python package is a general purpose sentiment analysis tool popular among high school students. Sentiment analysis is widely studied in natural language processing and has been applied in many business and social domains (Liu and Zhang, 2012) but very rare to detect bullying.

Bag of words text representation works well for longer documents by relying on a few words with strong sentiments like "awesome" or "harassment" etc. However, sentiment accuracies even for binary i.e. positive and negative classification for single sentences has not exceeded 80% for many years. For the multiclass case we include "neutral" class, accuracy is often below 60% for short messages on Twitter (Wang et el. 2012)
First we count the frequency of words in positive, negative and neutral classes. Common words like "the" appear very often in both positive and negative reviews. Instead of finding the most common words in positive or negative reviews,

what we really want are the words found in positive reviews more often than in negative reviews, and vice versa. To accomplish this, we need to calculate the **ratios** of word usage between positive and negative reviews.

As you can see, common words like "the" appear very often in both positive and negative reviews. Instead of finding the most common words in positive or negative reviews, what you really want are the words found in positive reviews more often than in negative reviews, and vice versa. To accomplish this, you'll need to calculate the **ratios** of word usage between positive and negative reviews.

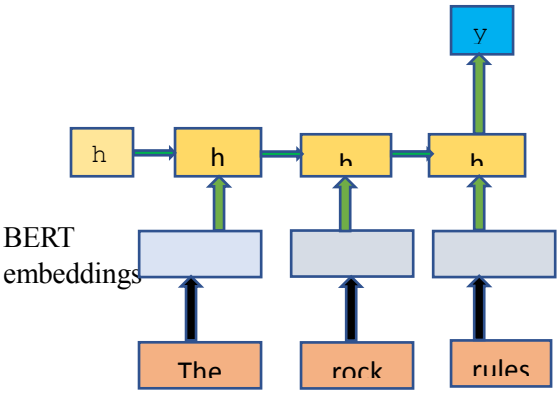


figure 2: RNN Network with BERT embeddings

I will be using a new kind of cross entropy loss, which is designed to work with a single sigmoid output. Binary Cross Entropy Loss or BCELoss in short, applies cross entropy loss to a signle value between 0 and 1. I will use the preporcessed data, although it is a very small data set.

Incorporating embeddings will improve performance of any machine learning model. The BERT embedding layer is replaced with OpenAI embeddings. Retrain and remeasured the performance. I copmared both the performances. I used BERT to extract features, namely word and sentence embedding vectors from text data.

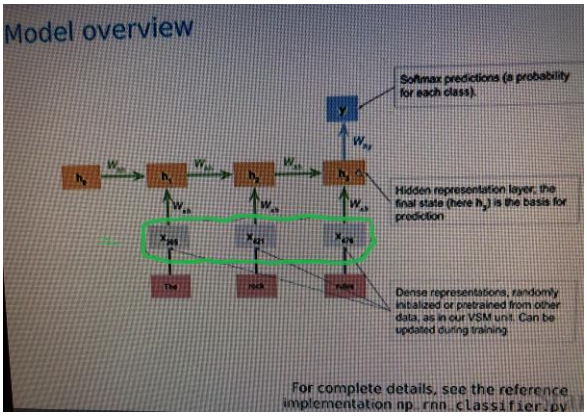


Figure 3: Embeddings from OpenAI

The classifier (y) in the above diagram is again a deep neural network as shown below.

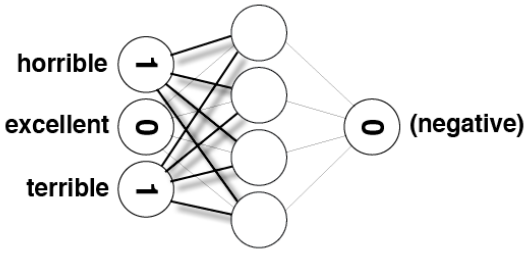


Figure 4: Deep neural network

4.1 BERT Embeddings

BERT model sizes have a large number of encoder layers (which the paper calls Transformer Blocks). I remember BERT comes with a base model and a large model. They also have larger feedforward-networks (768 and 1024 hidden units respectively), and more attention heads (12 and 16 respectively) than the default configuration in the reference implementation of the Transformer in the initial paper (6 encoder layers, 512 hidden units, and 8 attention heads). NLP models such as LSTMs or CNNs require inputs in the form of numerical vectors, and this typically means translating features like the vocabulary and parts of speech into numerical representations. In the past, words have been represented either as uniquely indexed values (one-hot encoding), or more helpfully as neural word embeddings where vocabulary words are matched against the fixed-length feature embeddings that result from models like Word2Vec or Fasttext. BERT offers an advantage over models like Word2Vec, because while each word has a fixed representation under Word2Vec

regardless of the context within which the word appears, BERT produces word representations that are dynamically informed by the words around them.

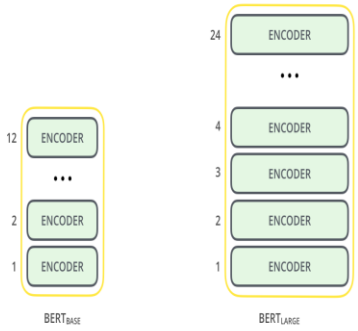


Figure 5: BERT base and large model

Next we need to convert our data to torch tensors and call the BERT model. The BERT PyTorch interface requires that the data be in torch tensors rather than Python lists, so we convert the lists here - this does not change the shape or the data.

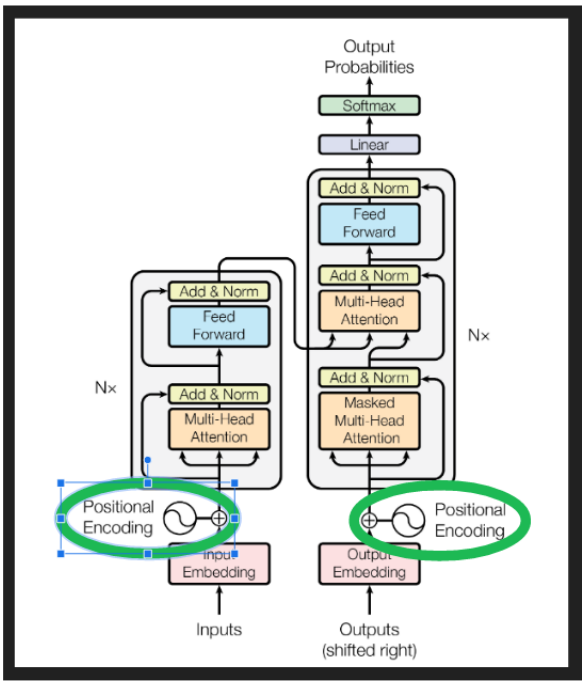


Figure 6: BERT architecture

5 Experiment

Many sentiment analysis algorithms first learn a sentiment lexicon with weights, which are words and phrases commonly used to express positive or negative sentiments, from a coded list and/or existing corpus. The sentiment of a new document is then detected by aggregating the weights of the sentiment lexicon that appears in the document.

I found that formspring bullying data set has many jargons. Data proprocessing is really took time. I found that it is difficult to distinigush between bullying and teasing. In addition to that target (class) is not annotated. I wrote a logic to derive target. If any two answers are same and severity is greater than zero, as Positive or Negative depending on the value of the answer. I converted to numeric value as follows:

No Bullying (Positive) = 1
Bullying (Negative) = -1
Neutral = 0

I split the dataset into training set and a testing set in the ratio of 80:20 I observed that generally the embedding representation is very rich and information dense. For example, reducing the dimensionality of inputs using SVD or PCA, even by 10% resulted in worse downstream perfromance.

Bert Embedding	~59%
OpenAI Embedding	~61%

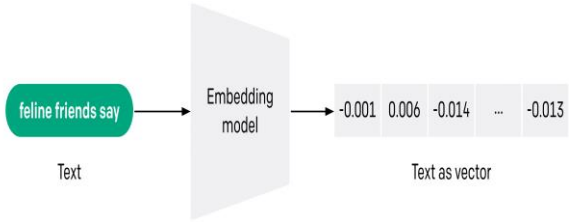
6 Analysis

I used pre-trained and contextual embeddings, such as BERT. I believed that using more powerful models that capture linguistic and semantic information would lead to better performance. Then I saw a new paper and blog post from OpenAI, which claims their embeddings improves performance 64%. So, I compared performance of both in the below table. OpenAI based embeding network model establishes a new single-model state-of-the-art BLEU score of 40.8.

6.1 OpenAI Embedding API

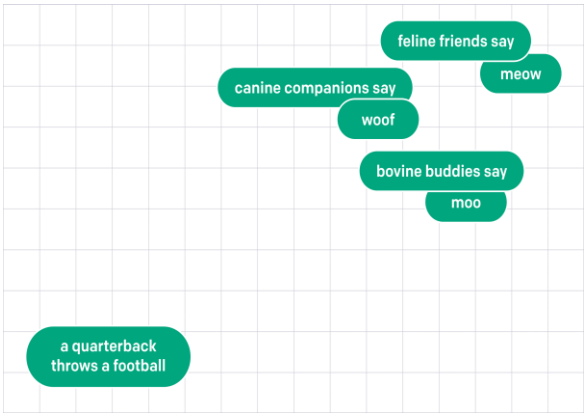
OpenAI introduced embeddings, a new endpoint API that makes it easy to perform natural language and code tasks like semantic search, clustering, topic modeling, and classification. Embeddings are numerical representations of concepts converted to number sequences, which make it easy for computers to understand the relationships between those concepts. This embeddings algorithm outperform top models in 3 standard benchmarks, including a 20% relative improvement in code search.

Embeddings are useful for working with natural language and code, because they can be readily consumed and compared by other machine learning models and algorithms like clustering or search.



Source: OpenAI blog

Embeddings that are numerically similar are also semantically similar. For example, the embedding vector of "canine companions say" will be more similar to the embedding vector of "woof" than that of "meow."



Source: OpenAI blog

The new endpoint uses neural network models, which are descendants of GPT-3, to map text and code to a vector representation—"embedding" them in a high-dimensional space. Each dimension captures some aspect of the input. The new /embeddings endpoint in the OpenAI API provides text and code embeddings with a few lines of code:

```
import openai
response = openai.Embedding.create(
    input="canine companions say",
    engine="text-similarity-davinci-001")

print(response)
```

To obtain an embedding vector for a piece of text I make a request to the Embeddings endpoint. It requires API key which can be obtained from their website. Since, dataset is very small, next step is to try zero-shot learning or few-shot-learning.

6.2 Zero-Shot Classification

I experimented embeddings for zero shot classification without any labeled training data. For each class, I embed the class name or a short description of the class. To classify some new text in a zero-shot manner, I compared its embedding to all class embeddings and predict the class with the highest similarity.

6.3 Few-Shot Classification

In recent times, there is a subential benefit on many NLP tasksby using large pretrained transformer based models and fine tuning it for a specific task. Although this approach is task agnostic, still it requires thousands of examples to fine tune for a specific task. In contrast, humans perform new language task with very less examples. This paper provides a solution to reduce this gap. Few-Shot Learning refers to the practice of feeding a machine learning model with a very small amount of training data to guide its predictions, like a few examples at inference time, as opposed to standard fine-tuning techniques which require a relatively large amount of training data for the pre-trained model to adapt to the desired task with accuracy. The ability to efficiently learn from little data is critical to applying NLP to tasks where data collection is costly or otherwise difficult. This is a challenging setting both academically and practically, particularly because training neutral models typically require large amount of labeled data.

More recently, advances in pretraining on unlabelled data have brought up the potential of better few-shot learning. In many Machine Learning applications, the amount of available labeled data is a barrier to producing a high-performing model. The latest developments in NLP show that you can overcome this limitation by providing a few examples at inference time with a large language model - a technique known as **Few-Shot Learning**. Few-Shot Learning refers to the practice of feeding a machine learning model with a very small amount of training data to guide its predictions, like a few examples at inference time, as opposed to

standard fine-tuning techniques which require a relatively large amount of training data for the pre-trained model to adapt to the desired task with accuracy. I tried this technology to see if it helps in improving performance of the model.

6.3 Hyperparameters Tuning:

Used the following as hyperparameter and obtained their optimal value by early stopping method.
lr – Learning rate for the optimizer
epochs: number of times to iterate thru the training data
clip: The maximum gradient value to clip at to prevent exploding gradients
Drop out value
Optimiser: Adam
Number of epochs = 100

7 Conclusion

The in-depth analysis of the Formspring dataset performed. . In case of any NLP task it is hard to say that any annotation is perfect. This system demonstrates that its novel approach can comprise an effective way for cyberbullying detection. In that case, high recall goes side by side with high precision, which indicates the possibility of using the system for real-time automatic cyberbullying blocking and content moderation. Furthermore, both the annotation process and the error analysis show how much depends on the adopted criteria, and therefore how important it is for a cyberbullying detection system to be effectively adjustable to the given criteria.

Acknowledgments

I thank Prof. Christopher Potts, our course facilitator Petra Parikova and the staff of XCS224U for their guidance throughout this project.

Authorship

I wrote code and cleaned the cyberbullying dataset, designed and implemented the network.

References

Arvind Neelakantan, Tao Xu et al. 2022 Text and Code Embeddings by Contrastive Pre-Training
https://doi.org/10.48550/arXiv.2201.10005

Ashish Vaswani et al. "Attention Is All You Need" 2017
https://doi.org/10.48550/arXiv.1706.03762

Hanxiao Liu, Zihang Dai, David R. So, Quoc V. Le et al. 2021 Pay Attention to MLPs
https://doi.org/10.48550/arXiv.2105.08050

Kelly Reynolds, April Kontostathi et al. "Using Machine Learning to Detect Cyberbullying" in Machine learning and Applications and workshops (ICMLA), 2011 10th International conference, vol. 2. IEE, 2011, pp.241-244.
https://ieeexplore.ieee.org/document/6147681

http://www.chatcoder.com/Data/BayzickBullyingData.rar

openAI blog
https://beta.openai.com/docs/guides/embeddings/use-cases

Bert Tutorials
https://mccormickml.com/2019/05/14/BERT-word-embeddings-tutorial/

Jay Alammar's Tutorials
https://jalammar.github.io/