International Journal of Research
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 07 Issue 05
May 2020

Toxic Message Classification Using Convolution And Gru Network

D. Ramana kumar

P.Manideep Pasulad

Afreen Shaik

Nimish Reddy

Email: ramanacse@cvsr.ac.in,16h61a0593@cvsr.ac.in , 16h61a05b3@cvsr.ac.in, 16h61a05a5@cvsr.ac.in .

Department of Computer Science and Engineering, ANURAG GROUP OF INSTITUTIONS

Abstract: Now-a-days, derogatory comments are often made by one another, not only in offline environments but also immensely in social networking websites and online communities. Thus an identification model which reads any piece of text appearing in any platform is classified and detects the type of toxicity like obscenity, threats, insults and identity-based hatred. Existing system works on the LSTM type of RNN model. Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. In order to improve the efficiency and accuracy of the existing system, we propose Concurrent GRU (ensemble of CNN and GRU models). The motivation for our project is to build a model that can detect toxic comments and find bias with respect to the mention of select identities. The data pre-processing consists of text tokenization and normalization which varies slightly when processing features for the model.

Keywords: CNN (Convolutional Neural Network), RNN (Recurrent Neural Network), LSTM (Long Short Term Memory), GRU (Gated Recurrent Unit).

1. Introduction

In recent years, the exponential growth of social media such as Twitter and community forums has been increasingly exploited for the propagation of hate speech and the organization of hate based activities. The anonymity and mobility afforded by such media has made the breeding and spread of hate speech - eventually leading to hate crime - effortless in a virtual landscape beyond the realms of traditional law enforcement. In the UK, there has been significant increase of hate speech towards the migrant and Muslim communities following recent events including leaving the EU, and the Manchester and the London attacks. This correlates to record spikes of hate crimes, and cases of threats to public safety due to its nature of inciting hate crimes, such as that followed the Finsbury van attack. Surveys and reports also show the rise of hate speech and related crime in the many states of India where 80% of young people have encountered hate speech online and 40% felt attacked or threatened.

Social media companies (e.g., Twitter, Facebook) have come under pressure to address this issue, and it has been estimated that hundreds of millions of euros are invested every year. However, they are still being criticized for not doing enough. This is largely because standard measures involve manually reviewing online contents to identify and delete offensive materials. The process is labour intensive, time consuming, and not sustainable in reality.

The pressing need for scalable, automated methods of hate speech detection has attracted significant research using semantic content analysis technologies based on Natural Language Processing (NLP) and Machine Learning (ML) . Despite this large amount of work, it remains difficult to compare their performance, largely due to the use of different datasets by each work and the lack of comparative evaluations. This work makes three contributions to this area of work. First, we use a Concurrent GRU (ensemble model of convolutional neural network and gated recurrent unit network) neural network model optimised with dropout and pooling layers, and elastic net regularisation for better learning accuracy. Compared to existing deep learning models that use only CNNs, the GRU layer also captures sequence orders that are useful for this task. Second, we create a public dataset of hate speech by collecting thousands of tweets on the subjects of religion and refugees, which extends currently available datasets by both quantity and subject coverage. Third, we test our model against several baselines and also previously reported results on all existing public datasets, and set new benchmark as we show that our model outperforms on 6 out of 7 datasets by as much as 13% in F1. We also undertake error analysis to identify the challenges in hate speech detection on social media.

2. Literature Survey

In today’s era of Smartphones and computers the internet has changed the idea of communication via messages. Due to lack of security, various cyber-crimes have emerged in the past decade. Cyber security plays a significant role in the current development of information technology and services. Cyber security is thus an attempt by users to keep their personal and professional information intact from the attacks on the social media over the internet. The main function of cyber security is to protect the privacy of users, networks, computers, programs from unauthorized access and loss. Maximum number of users are not aware of the risks and share their information unknowingly and their lack of knowledge makes them vulnerable to cyber-attacks and cyberbullying.

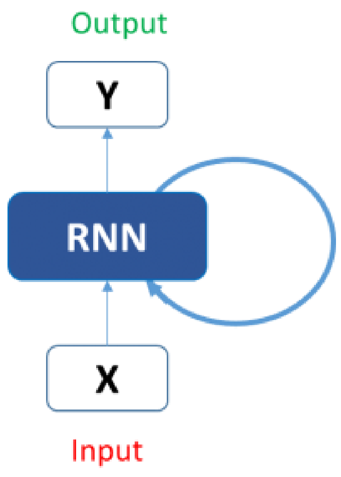
Cyberbullying is bullying that takes place over digital devices like cell phones, computers, and tablets. Cyberbullying can occur through SMS, Text, and apps, or online in social media, forums, or gaming where people can view, participate in, or share content. Cyberbullying includes sending, posting, or sharing negative, harmful, false, or mean content about someone else. It can include sharing personal or private information about someone else causing embarrassment or humiliation.

2.1 Introduction to LSTM

Sequence prediction problems have been around for a long time. They are considered as one of the hardest problems to solve in the data science industry. These include a wide range of problems; from predicting sales to finding patterns in stock markets' data, from understanding movie plots to recognizing your way of speech, from language translations to predict your next word on your iPhone's keyboard.
With the recent breakthroughs that have been happening in data science, it is found that for almost all of these sequence prediction problems, Long short Term Memory networks, a.k.a LSTMs have been observed as the most effective solution. LSTMs have an edge over conventional feed-forward neural networks and RNN in many ways. This is because of their property of selectively remembering patterns for long durations of time. The purpose of this article is to explain LSTM and enable you to use it in real life problems.

2.2 Recurrent Neural Network (RNN)

Take an example of sequential data, which can be the stock market's data for a particular stock. A simple machine learning model or an Artificial Neural Network may learn to predict the stock prices based on a number of features: the volume of the stock, the opening value etc. While the price of the stock depends on these features, it is also largely dependent on the stock values in the previous days. In fact for a trader, these values in the previous days (or the trend) is one major deciding factor for predictions. In the conventional feed-forward neural networks, all test cases are considered to be independent. That is when fitting the model for a particular day, there is no consideration for the stock prices on the previous days. This dependency on time is achieved via Recurrent Neural Networks. A typical RNN looks like:



Now it is easier for us to visualize how these networks are considering the trend of stock prices, before predicting the stock prices for today. Here every prediction at time t (h_t) is dependent on all previous predictions and the information learned from them. RNNs can solve our purpose of sequence handling to a great extent but not entirely. We want our computers to be good enough to write Shakespearean sonnets. Now RNNs are great when it comes to short contexts, but in order to be able to build a story and remember it, we need our models to be able to understand and remember the context behind the sequences, just like a human brain. This is not possible with a simple RNN.

2.3 Improvement over RNN: LSTM

When we arrange our calendar for the day, we prioritize our appointments right? If in case we need to make some space for anything important we know which meeting could be cancelled to accommodate a possible meeting. Turns out that an RNN doesn't do so. In order to add new information, it transforms the existing information completely by applying a function. Because of this, the entire information is modified, on the whole, i. e. there is no consideration for 'important' information and 'not so important' information. LSTMs on the other hand, make small modifications to the information by multiplications and additions. With LSTMs, the information flows through a mechanism known as cell states. This way, LSTMs can selectively remember or forget things. The information at a particular cell state has three different dependencies.

We'll visualize this with an example. Let's take the example of predicting stock prices for a particular stock. The stock price of today will depend upon:

1. The trend that the stock has been following in the previous days, maybe a downtrend or an uptrend.
2. The price of the stock on the previous day, because many traders compare the stock's previous day price before buying it.
3. The factors that can affect the price of the stock for today. This can be a new company policy that is being criticized widely, or a drop in the company's profit, or maybe an unexpected change in the senior leadership of the company.

These dependencies can be generalized to any problem as:

1. The previous cell state (i.e. the information that was present in the memory after the previous time step)
2. The previous hidden state (i.e. this is the same as the output of the previous cell)
3. The input at the current time step (i.e. the new information that is being fed in at that moment)

.

Another important feature of LSTM is its analogy with conveyor belts! That's right! Industries use them to move products around for different processes. LSTMs use this mechanism to move

International Journal of Research
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 07 Issue 05
May 2020

information around. We may have some addition, modification or removal of information as it flows through the different layers, just like a product may be molded, painted or packed while it is on a conveyor belt.

2.4 Architecture of LSTM

The functioning of LSTM can be visualized by understanding the functioning of a news channel's team covering a murder story. Now, a news story is built around facts, evidence and statements of many people. Whenever a new event occurs you take either of the three steps.
Let's say, we were assuming that the murder was done by 'poisoning' the victim, but the autopsy report that just came in said that the cause of death was 'an impact on the head'. Being a part of this news team what do you do? You immediately forget the previous cause of death and all stories that were woven around this fact.
What, if an entirely new suspect is introduced into the picture. A person who had grudges with the victim and could be the murderer? You input this information into your news feed, right?
Now all these broken pieces of information cannot be served on mainstream media. So, after a certain time interval, you need to summarize this information and output the relevant things to your audience. Maybe in the form of "XYZ turns out to be the prime suspect.".
Now let's get into the details of the architecture of LSTM network:

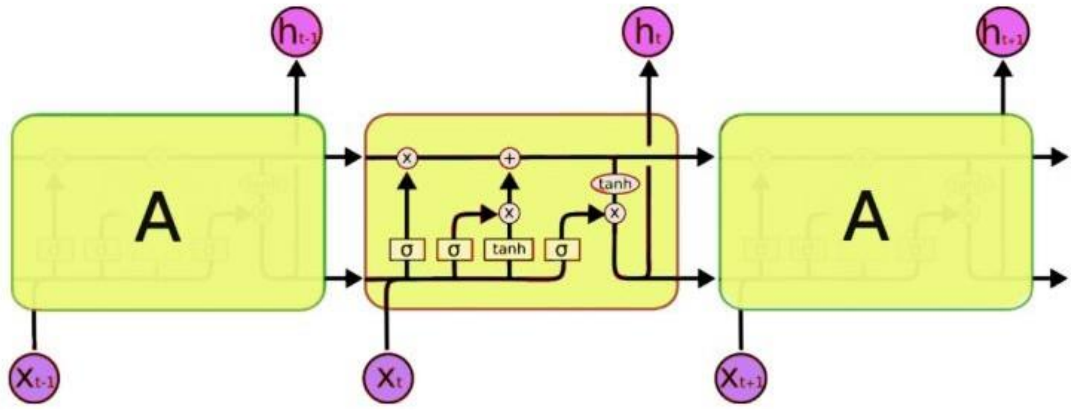


Fig. 2.3 – LSTM Architecture

Now, this is nowhere close to the simplified version which we saw before, but let me walk you through it. A typical LSTM network is comprised of different memory blocks called cells
(the rectangles that we see in the image). There are two states that are being transferred to the next cell; the cell state and the hidden state. The memory blocks are responsible for remembering things and manipulations to this memory is done through three major mechanisms, called gates.

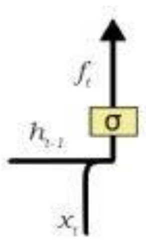
Each of them is being discussed below

Forget Gate

Taking the example of a text prediction problem. Let's assume an LSTM is fed in, the following sentence:

Bob is a nice person. Dan on the other hand is evil.

As soon as the first full stop after "person" is encountered, the forget gate realizes that there may be a change of context in the next sentence. As a result of this, the subject of the sentence is forgotten and the place for the subject is vacated. And when we start speaking about "Dan" this position of the subject is allocated to "Dan". This process of forgetting the subject is brought about by the forget gate.



A forget gate is responsible for removing information from the cell state. The information that is no longer required for the LSTM to understand things or the information that is of less importance is removed via multiplication of a filter. This is required for optimizing the performance of the LSTM network.

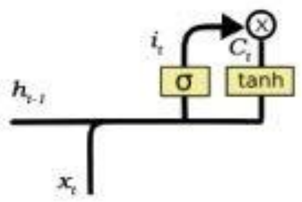
Input Gate

Okay, let's take another example where the LSTM is analysing a sentence:

Bob knows swimming. He told me over the phone that he had served the navy for 4 long years.

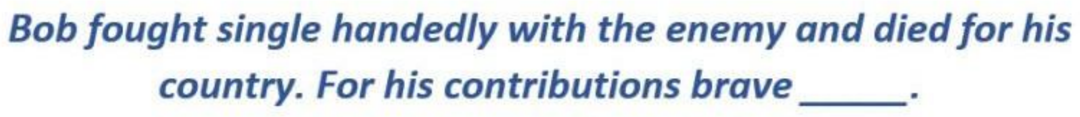
International Journal of Research
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 07 Issue 05
May 2020

Now the important information here is that “Bob” knows swimming and that he has served the Navy for four years. This can be added to the cell state, however, the fact that he told all this over the phone is a less important fact and can be ignored. This process of adding some new information can be done via the input gate.
Here is its structure:

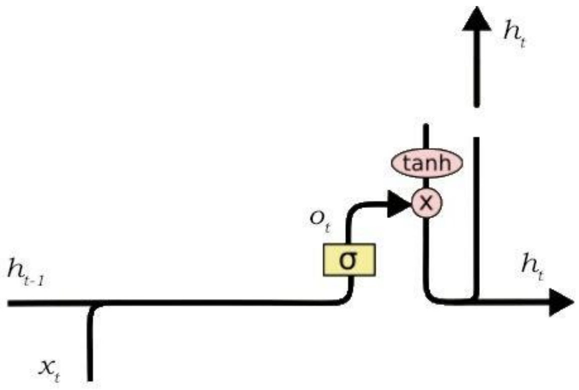


Output Gate

Not all information that runs along the cell state, is fit for being output at a certain time. We’ll visualize this with an example:


Bob fought single handedly with the enemy and died for his country. For his contributions brave ______.

In this phrase, there could be a number of options for the empty space. But we know that the current input of ‘brave’, is an adjective that is used to describe a noun. Thus, whatever word follows, has a strong tendency of being a noun. And thus, Bob could be an apt output. This job of selecting useful information from the current cell state and showing it out as an output is done via the output gate. Here is its structure:



3. Analysis

3.1 Existing System

Classifying the toxicity levels of a comment has been difficult and time consuming in the earlier systems. These levels of threats to the classification can impact the accuracy of the model. Long Short-Term Memory (LSTM) networks are a type of Recurrent Neural Network (RNN), and are capable of learning order dependence in sequence prediction problems. This model is used because of the individual cell states in the model, that is, long-term information has to sequentially travel through all cells before getting to the present processing cell. This means it can be easily corrupted by being multiplied many times by small numbers. Another issue of RNN is that they are not hardware friendly. It takes a lot of resources we do not have to train these networks fast.

3.2 Proposed System

Due to the delimiter of the existing system, we need to deliver the best accurate results by covering the problems faced in the existing system. We are proposing a system using Concurrent GRU Architecture delivering better accuracy and efficiency. The Preliminary model which we will be using is, the Naive Bayes- multinomial Bernoulli event model with the n-gram bag-of-words count features. We use Bag-of-words methods which vectorizes each comment by creating a ”vector space” of the entire vocabulary. This proposed model, we use Keras with a TensorFlow backend to implement a one-dimensional convolutional neural network. We first fed the comment into the word embeddings from the GloVe 50, 100, and 200 dimension vectors. We then fed these word embeddings into a three-layer CNN. The final output from the dense layer was a toxicity score given from 0.0 to 1.0 with the goal of predicting the toxicity scores provided in the dataset.

International Journal of Research
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 07 Issue 05
May 2020

3.3 Software Requirement Specification

3.3.1 Purpose

One of the widely used Natural Language Processing and Supervised Machine Learning (ML) task in different business problems is “Text Classification”, which is an example of Supervised Machine Learning task since a labelled dataset containing text documents and their labels is used for training a classifier. CNN is a class of deep, feed-forward artificial neural networks where connections between nodes do not form a cycle and use a variation of multilayer perceptrons designed to require minimal preprocessing. Applications include image captioning, language modeling and machine translation. CNN’s are good at extracting local and position-invariant features.

3.3.2 Scope

By working with different types of Neural Network models with word embedding initializations, we could conclude which models may be better suited for the task of toxic comment classification. We found that the best model in our case was the Concurrent GRU model with word embedding. Although its performance accuracy is marginally better than the LSTM model, it could gain a better categorization of toxic comment accuracy. This was noted by passing an input threat comment to the app function developed to find the toxicity levels of the comment for CNN model with GRU model.

4. Implementation

4.1 Data Pre-processing

Given a tweet, we start by applying a light pre-processing procedure described below based on that reported in, to normalize its content. This includes: 1) removing the characters | : , ; & ! ? \; 2) applying lowercase and stemming, to reduce word infections; and 3) removing any tokens with a document frequency less than 5, which reduces sparse features that are not informative for learning. Empirically, this was found to lead to better classification accuracy. Further, we also normalize hashtags into words, so ‘#refugeesnotwelcome’ becomes ‘refugees not welcome’. This is because such hashtags are often used to compose sentences. We use dictionary based look up to split such hashtags.

Stages of pre-processing

4.1.1 Remove numeric and empty texts

This stage involves in removing the numbers and unwanted white spaces in a comment. This helps us to keep the text free from numeric and in a clean format.

4.1.2 Cleaning unnecessary text

This is a necessary stage when dealing with NLP (Natural Language Processing). In this stage we remove the stopwords from each and every comment in the data set. A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query. We would not want these words taking up space in our database, or taking up valuable processing time. For this, we can remove them easily, by storing a list of words that you consider to be stop words. NLTK (Natural Language Toolkit) in python has a list of stopwords stored in 16 different languages.

4.1.3 Tokenizing

Tokenization is the process of dividing text into a set of meaningful pieces. These pieces are called tokens. Depending on the task at hand, we can define our own conditions to divide the input text into meaningful tokens. In this stage a comment is divided into individual pieces as tokens and groups them together.

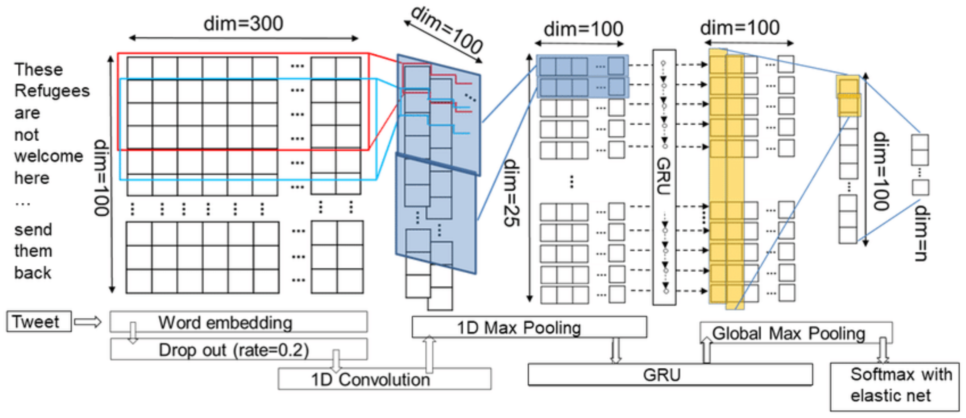
4.1.4 Lemmatization

Lemmatization, unlike Stemming, reduces the inflected words properly ensuring that the root word belongs to the language. In Lemmatization root word is called Lemma. A lemma is the canonical form, dictionary form, or citation form of a set of words.
In this stage, we take a word and replace it with its root word. This helps the model to get exact word each time when a comment is passed.

4.1.5 Sequence Creation

This stages gathers the data and extracts the features and creates sequences that are required for the model to train. These set of features helps the model to refer through these sequences to classify the level toxic level. This is a final and crucial stage for a model before training.

4.2 Concurrent GRU Architecture and Model Training



In our Concurrent GRU Architecture (i.e; ensemble of CNN and GRU models) as shown in Fig.5.1, the first layer is a word embedding layer, which maps each text message (in generic

terms, a ‘sequence’) into a real vector domain. To do so, we map each word onto a fixed dimensional real valued vector, where each element is the weight for that dimension for that word. In this work, we use the word embedding with 300 dimensions pre-trained on the 3billion-word Google News corpus with a skip-gram model3 to set the weights of our embedding layer. We also constrain each sequence to be 100 words which is long enough to encode tweets of any length, truncating long messages and pad the shorter messages with zero values. The embedding layer passes an input feature space with a shape of 100×300 to a drop-out layer with a rate of 0.2, the purpose of which is to regularize learning to avoid over-fitting. Intuitively, this can be thought of as randomly removing a word in sentences and forcing the classification not to rely on any individual words. The output feeds into a 1D convolutional layer with 100 filters with a window size of 4, padding the input such that the output has the same length as the original input. The rectified linear unit function is used for activation.
This convolves the input feature space into a 100×100 representation, which is then further down-sampled by a 1D max pooling layer with a pool size of 4 along the word dimension, producing an output of shape 25×100. Each of the 25 dimensions can be considered an ‘extracted feature’. These extracted features then feed into the GRU layer, which treats the feature dimension as timesteps and outputs 100 hidden units per timestep. The key difference between a GRU and an LSTM is that a GRU has two gates (reset and update gates) whereas an LSTM has three gates (namely input, output and forget gates). Thus GRU is a simpler structure with fewer parameters to train. In theory, this makes it faster to train and generalize better on small data; while empirically it is shown to achieve comparable results to LSTM. Next, a global max pooling layer ‘flattens’ the output space by taking the highest value in each time step dimension, producing a 1×100 vector. Intuitively, this selects the highest scoring features to represent a tweet. Finally, a softmax layer takes this vector as input to predict probability distribution over all possible classes (n), which will depend on individual datasets. The softmax activation is also regularized using the elastic net regularization that linearly combines the L1 and L2 norms, but is designed to solve the limitations of both. We use the categorical cross entropy loss function and the Adam optimizer to train the model. The first is empirically found to be more effective on classification tasks than other loss functions including classification error and mean squared error, and the second is designed to improve the classic stochastic gradient descent (SGD) optimizer and in theory combines the advantages of two other common extensions of SGD .
Model parameters. As described above, many of our model parameters are based on empirical findings reported previously, default values or anecdotal evidence, except the batch size and epoches for training which we derive from developmental data (to be detailed later). Arguably, these may not be the best settings for optimal results, which are
always data-dependent. However, we show later in experiments that the model already obtains promising results even without extensive data-driven parameter tuning.

4.3 Predicting Outputs and UI

Python's Flask, it is simple to integrate machine learning models with a user-friendly HTML interface. This framework can be applied to any example where a user provides data and receives a prediction from a machine learning model. A RESTful API creates a link between the user and the server where the pre-trained model is hosted. We use Python Flask to interact with what the user enters and receives on the webpage. Python Flask acts as a bridge between the pre-trained model and the HTML page.
We load the pre-trained model and tokenizer for pre-processing. we create a helper function that applies pre-processing. Thus, the identifiers are attached to the words entered by the user are misaligned with the identifiers assigned during the training process. As a result, the Word CNN would interpret the input as a sentence entirely different from what the user inputs. Loading the pickled tokenizer ensures consistency with the model training.

Now that we have the pre-processing and pre-trained model loaded, we interact Flask with our HTML page. When the user enters text and clicks the form's submit button, as a result, text_entered from the user in the HTML form becomes textData, which is converted to an array — named Features — containing numeric identifiers for each word. This array is fed through the pre-trained Word CNN to generate a prediction representing the probability of being spam. Thus the given input comment/message is categorized and classified into six different toxicity categories.

Categories:

1. Toxic
2. Severe toxic
3. Obscene
4. Threatening
5. Insult
6. Identity hate.

5. Conclusion

Both the industrial and the research community in the last few years have made several tries to identify an efficient model for online toxic comment prediction due to its importance in online interactive communications among users, but these steps are still in their infancy. This work is devoted to the study of a recent approach for text classification involving word representations and Convolutional Neural Networks. We investigate its performance in comparison to more widespread text mining methodologies for the task of toxic comment classification. As shown

Concurrent GRU can outperform well established methodologies providing enough evidence that their use is appropriate for toxic comment classification. The promising results are motivating for further development of CNN based methodologies for text mining in the near future, in our interest, employing methods for adaptive learning and providing further comparisons with n-gram based approaches.

6. Bibliography

[1] https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge
[2] https://www.researchgate.net/publication/323723283_Detecting_hate_speech_on_Twitter_using_a_convolution-GRU_based_deep_neural_network
[3] https://www.researchgate.net/figure/The-CNN-GRU-network-architecture-This-diagram-is-best-viewed-in-colour_fig1_323723283
[4] https://medium.com/@zake7749/top-1-solution-to-toxic-comment-classification-challenge-ea28dbe75054
[5] http://cs229.stanford.edu/proj2019spr/report/69.pdf
[6] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In NIPS 2014 Deep Learning and Representation Learning Workshop, 2014
[7] T. Davidson, D. Warmsley, M. Macy, and I. Weber. Automated hate speech detection and the problem of offensive language. In Proceedings of the 11th Conference on Web and Social Media. AAAI, 2017
[8] K. Dinakar, B. Jones, C. Havasi, H. Lieberman, and R. Picard. Common sense reasoning for detection, prevention, and mitigation of cyberbullying. ACM Trans. Interact. Intell. Syst., 2(3):18:1–18:30, Sept. 2012.
[9] B. Gamb¨ack and U. K. Sikdar. Using convolutional neural networks to classify hate-speech. In Proceedings of the First Workshop on Abusive Language Online, pages 85–90. Association for Computational Linguistics, 2017
[10] BBCNews. Countering hate speech online, Last accessed: July 2017, http://eeagrants.org/News/2012/.
[11] BBCNews. Finsbury park attack: Son of hire boss held over facebook post, Last accessed: May 2017, http://www.bbc.co.uk/news/uk-wales-40347813.
[12] P. Burnap and M. L. Williams. Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making. Policy and Internet, 7(2):223–242, 2015.
[13] J. P. Chiu and E. Nichols. Named entity recognition with bidirectional lstm-cnns. Transactions of the Association for Computational Linguistics, 4:357–370, 2016.