Hate speech detection related to League of Legends community comments

Giovanna Ferraro
MS in Computer Engineering / 1st year
giovanf@stud.ntnu.no

Abstract

This paper is about the hate speech classification of League of Legends comments on the Twitch community. This topic is compelling as the League of Legends community is known to be aggressive and negative towards streamers and other people from the community. The challenging part of the project was handling Twitch emotes, a sort of Twitch-specific dictionary. The well-known lexica as VADER have no knowledge about the semantics of emotes so various approaches were used to obtain hate speech detection. Both a rule-based model classification and an LSTM NN were helpful in addressing positive, negative, and neutral comments. The topic might have a large interest and can be improved with further effort towards training on new sets of emotes and fine-tuning classification.

1 Introduction

Twitch is an American video live streaming service that focuses on video game live streaming, including broadcasts of esports competitions. Throughout the year it gained more and more success becoming the leading streaming platform on the market.

As the popularity increased, also the community did and a new form of language was created. That is the case for emotes, textual and graphical emoji (like the ones we commonly use on Facebook or Twitter) that gained their specific meaning over time.

Each Twitch community has its way of using emotes and each community addresses a specific meaning to each emote. Given the interesting aspect of this phenomenon, it seemed interesting to study the semantics of emotes and try to create a neural network that was able to learn the sentiment of emotes. Luckily this is not a new field of study and many papers have been written on this topic.

The reason is that people on the Internet tend to say whatever they have in mind, not paying attention to the tone or the rudeness of their comments. To moderate those kinds of comments Twitch allowed streamers to ban a specific set of words or phrases known as negative. However, people can easily find loopholes and write comments that appear normal but it's clear from the context that the comment is not positive nor neutral. For this reason, automatically detecting the sentiment of a chat is a compelling aspect of all streaming platforms and gained interest over time.

The initial project goal was to draw the most negative community out of the most 10 active League of Legends streamers on Twitch; however, the task of preprocessing the dataset was too challenging and took a considerable amount of time. Eventually, the goal is to create a neural network trained on League of Legends comments, able to make sentiment analysis on Twitch comments. The approach used is cross-cutting, as rule-based methods and neural networks were used to achieve satisfactory results.

Even if this sounds counterintuitive, Kim Jaeheon et al. (1) studied that the most used emotes are the same across all the Twitch streamers, making the project extendable to any other Twitch community and open to further works.

In Section 2 there will be a brief but precise description of the main theoretical concepts used for the project; Section 3 discusses the related work made prior on this topic; Section 4 will discuss the setup of the dataset, the models used for the project and structure of the project; Section 5 contains the result of the experimentation; Section 6 discusses the obtained results; Section 7 draws the conclusion and the related future work for this paper.

2 Background

This section contains the main theoretical topics used to produce this project. the discussion will be brief but precise.

2.1 Sentiment Analysis

Sentiment analysis is a natural language processing technique used to determine the emotional tone or attitude expressed in a text. The main goal is to identify and extract subjective information from text data and classify it as positive, negative, or neutral. Those results can be achieved via different methods, both statistical or using machine learning algorithms.

2.2 Recurrent Neural Network

Recurrent neural networks (RNNs) are a type of neural network designed to process sequential data by allowing information to be passed from one time step to the other. Unlike traditional feedforward neural networks, RNNs can process inputs of variable length and produce output at each time step.

The key feature of RNNs is that they have recurrent connections, which allow the network to maintain a state or memory of past inputs as it processes new inputs. This makes them particularly useful for tasks such as natural language processing and speech recognition. (2)

2.3 Long Short-Term Memory

LSTM networks are a type of RNN that are specifically designed to handle long-term dependencies in sequential data by using a memory cell that can selectively forget or update information over multiple time steps. They overcome the issue of the vanishing gradients that often occurs in traditional RNNs. LSTM networks are well-suited to classifying, processing and making predictions based on series data such as text. (3)

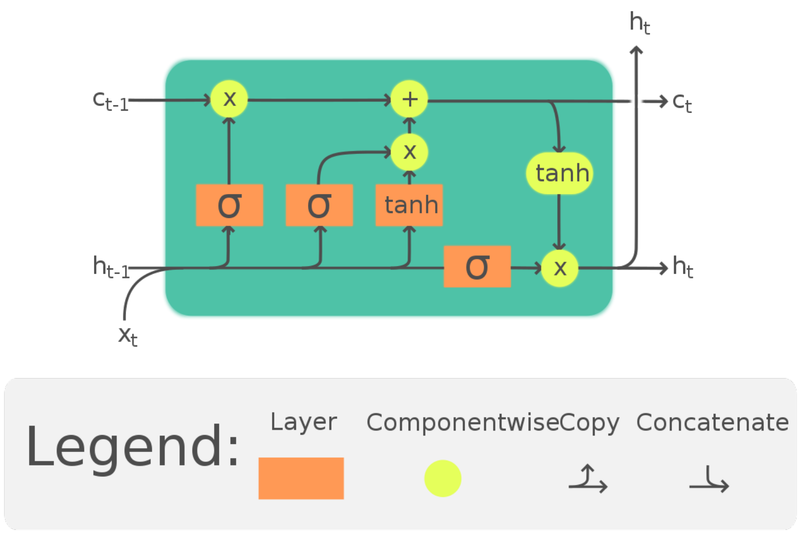


Figure 1: Long short-term memory cell

2.4 Bidirectional Long Short-Term Memory

A Bidirectional LSTM is a sequence processing model that consists of two LSTMs: one taking the input in a forward direction, and the other in a backwards direction. BiLSTMs effectively increase the amount of information available to the network, improving the context available to the algorithm. (4)

3 Related Work

Hate speech classification has always been a hot topic in the research community. It's worth citing the work made by Björn Gambäck and Utpal Kumar Sikdar (5) that first used CNN for hate speech on Twitter corpora.

One of the first works ever made on Twitch datasets dates back to 2018 (1). At the time, neural networks were a fairly new topic and most of the classification methods were rule-based, involving a loss of hand annotating work. However hiring people with proper semantic knowledge, in this case, Twitch comments, helped produce a high-quality labeled dataset. This paper uses a hand-labeled dataset dated back to 2018: reason is that many Twitch users helped label the dataset giving insights on the context of a comment, community-specific meaning related to the peculiar use of some emotes, and so on.

Another valuable study was made by Konstan Tin Kobs et al (6), where they stressed the importance of emotes as a way of delivering sentiment on Twitch comments. They notice the lack of specific lexica for Twitch comments so they created their lexica combining VADER lexica (7), a specific lexicon for emoji (8), and a new lexicon for emotes that they built from scratch. All together were useful in creating a newly distribution-based lexicon able to classify comments with an accuracy of 62%. As those labels were quite weak, they chose to build a SentenceCNN to be trained with the new labeled data, however, the improvement was quite small. This study has been quite effective in addressing emotes as a very powerful means for delivering sentiment and a new form that can indeed carry a semantic component while highlighting the importance of the context while doing sentiment analysis.

4 Architecture

This section has the scope of illustrate how the dataset was obtained, the methodology used to obtain a working project and the model for the neural network used.

4.1 Dataset

This section is about how the dataset is collected.

4.1.1 Retriving the data

By the date of this project, there aren't many labeled datasets for Twitch comments. The one that was found (8) is a CSV file with [sentiment, comment] column fields and only 1922 entries.

As the dimension is small, it was necessary to enhance the dimension. To do so, it was used a rule-based classification (8) can produce weak labels for an unlabeled dataset. It was proved that the accuracy of this classification is approximately 63%, enough for coarse labeling.

The number of comments from the ten most active League of Legends streamers was different, so to even out the dimensions the biggest datasets were pruned and the pruned part was used to augment the former labeled dataset, obtaining a total of 300.000 labeled comments as a training set.

4.1.2 Preprocessing of the data

The rule-based classifier used in the previous step is also responsible for the preprocessing of the data. It was of major importance to keep every emote and every emoji and the VADER lexicon itself is not enough. On top of that, was used an emoji lexicon and a custom-made emotes lexicon in charge of handling the emotes and making sure not to trim or prune them I n the preprocessing step.

4.2 Neural Networks

LSTM and BiLSTM were used as models for this project.

LSTM goes only forward and has lower training time; BiLSTM also goes backward and took twice as long as LSTM.

Both LSTM and BiLSTM have the same model sequence and it is drawn here below.

First was used an embedding layer. By using an embedding layer in a deep learning model, the dataset is transformed into dense low-dimensional embeddings that capture the semantic meaning of the words. It's a trainable parameter and it's updated through backpropagation. Dense and low-dimensional mapping is the state-of-the-art in learning semantics with neural networks.

The Adam optimizer was used as it does not require fine-tuning for the learning rate. The categorical cross-entropy (9) is the loss function used. The activation function used is the softmax.

To regularize the model it was used dropout with a probability of 0.5 and 0.6 for LSTM and BiLSTM, respectively.

Both neural network architectures' are shown below.

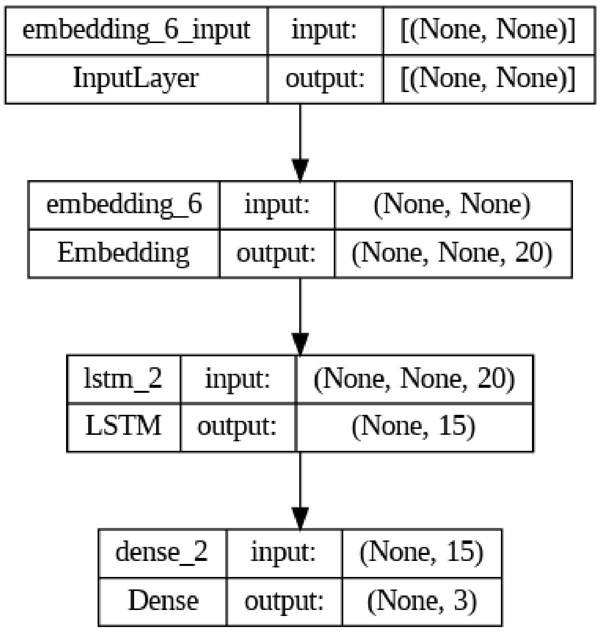


Figure 2: Model architecture for LSTM

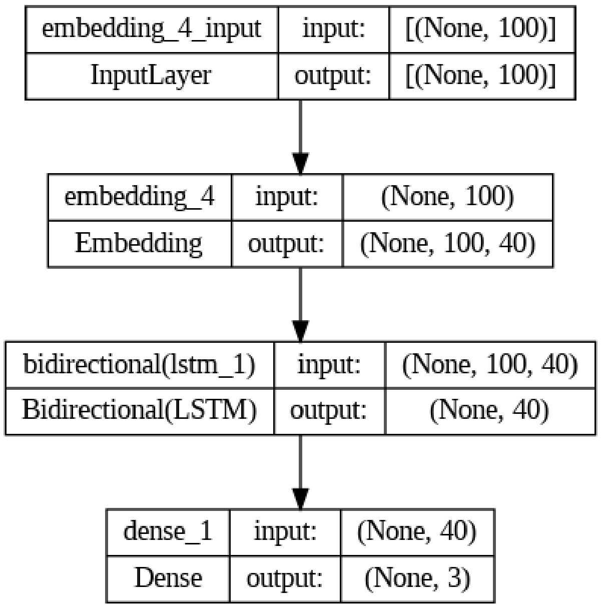


Figure 3: Model architecture for BiLSTM

The model takes inspiration from: https://github.com/sergiovirahonda/TweetsSentimentAnalysis

5 Experiments and Results

This section follows the trials and errors made while producing this project.

5.1 Experimental Setup

The model architecture used is the one explained in the upper section. The dataset used is approximately 300000 entries, split into train (75%) and test set (25%).

As the optimizer used is Adam, hence automatic, there was no need to fine-tune the learning rate.

First, the entire dataset was used, with only two epochs for the neural network, both for the LSTM and BiLSTM models.

Then, one-third of the dataset was used, increasing the number of epochs to ten for the LSTM model and five for the BiLSTM model.

For both setups, the training time was 30 minutes and 45 minutes for LSTM and BiLSTM respectively.

5.2 Experimental Results

This subsection draws the results obtained with the experimental setups, first analyzing the training and testing on the entire dataset and then on the pruned dataset.

5.2.1 Training the entire dataset

For the LSTM model.

During the training, the accuracy reached is 0.936, the loss is 0.21.

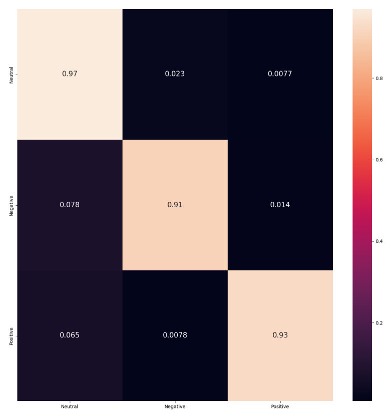
During the validation, the accuracy reached is 0.95 and the loss is 0.17.

For the BiLSTM model.

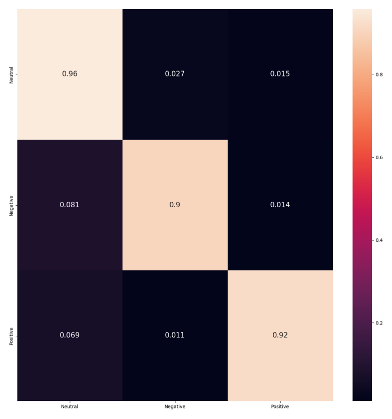
During training the accuracy reached is 0.94, the loss is 0.20.

During validation the accuracy reached is 0.943 and the loss is 0.19.

Here follows the confusion matrixes of the model after the validation.



(a) LSTM



(b) BiLSTM

Figure 4: Confusion matrixes made with seaborn

5.2.2 Training the pruned dataset

For the LSTM model.

During the training, the accuracy reached is 0.953, the loss is 0.15.

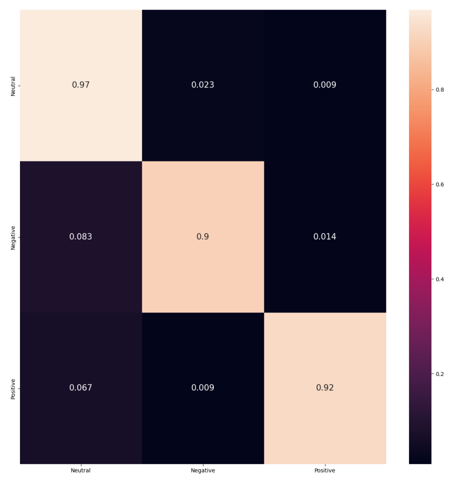
During the validation, the accuracy reached is 0.9504, the loss is 0.1751.

For the BiLSTM model.

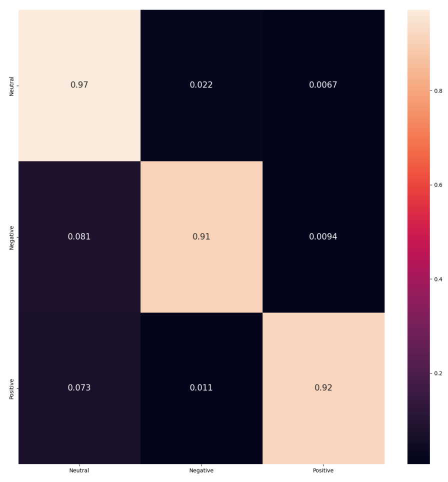
During the training, the accuracy reached is 0.9508, the loss is 0.1631.

During the validation, the accuracy reached is 0.9485, the loss is 0.1736.

Here follows the confusion matrixes of the model after the validation.

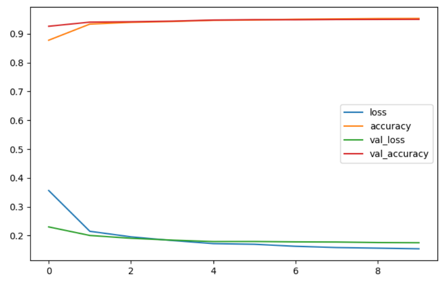


(a) LSTM

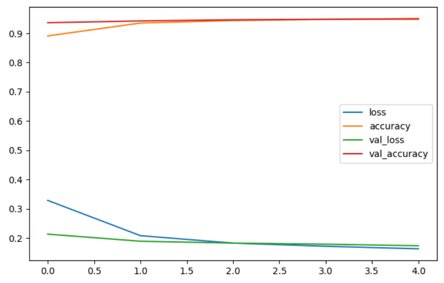


(b) BiLSTM

Figure 5: Confusion matrixes made with seaborn



(a) LSTM



(b) BiLSTM

Figure 6: Accuracy and loss over the training - using matplotlib

6 Evaluation and Discussion

The models have both shown to perform extremely well in the sentiment analysis on League of Legends comments.

Both LSTM and BiLSTM reached high level of accuracy and thanks to the confusion matrix it is claer how the model is well trained in classifying the correct label.

Surprisingly, the differences in the accuracy and the loss levels are so small that it is difficult to tell which model performed better.

It's clear, however, how important is to keep a decent number of epoch to allow the network to truly be deep and improve the loss.

7 Conclusion and Future Work

Regardless the kind of model used, this project showed how important is to deeply understand the context while producing a sentiment analyzer and how important is to preserve it.

The challenges face while handling the emote highlighted how these new forms of language are crucial in community life on social media and live-streaming platforms.

The project was trained and tested on League of Legends comments but as shown in past studies, the most used set of emotes are the same across the streamers and the communities, making this project easily extended to other communities and other games.

Can be worth putting some further effort into drawing the sentiment in live stream chats, making the moderation of chats automated and easier for streamers.

References

[1] Jaeheon Kim, Donghee Yvette Wohn, and Meeyoung Cha. Understanding and identifying the use of emotes in toxic chat on Twitch. *Online Social Networks and Media*, 27:100180, January 2022.

[2] Recurrent neural network, April 2023. Page Version ID: 1148729712.

[3] Long short-term memory, April 2023. Page Version ID: 1148032239.

[4] Papers with Code - BiLSTM Explained.

[5] Bjrn Gamb§ck and Utpal Kumar Sikdar. Using Convolutional Neural Networks to Classify Hate-Speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90, Vancouver, BC, Canada, August 2017. Association for Computational Linguistics.

[6] Konstantin Kobs, Albin Zehe, Armin Bernstetter, Julian Chibane, Jan Pfister, Julian Tritscher, and Andreas Hotho. Emote-Controlled: Obtaining Implicit Viewer Feedback Through Emote-Based Sentiment Analysis on Comments of Popular Twitch.tv Channels. *ACM Transactions on Social Computing*, 3:1–34, May 2020.

[7] C. Hutto and Eric Gilbert. VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text. *Proceedings of the International AAAI Conference on Web and Social Media*, 8(1):216–225, May 2014. Number: 1.

[8] Konstantin Kobs. Emote-Controlled, January 2023. original-date: 2019-10-08T07:24:49Z.

[9] tf.keras.losses.CategoricalCrossentropy | TensorFlow v2.12.0.

the model used: https://github.com/sergiovirahonda/TweetsSentimentAnalysis
support for tokenization and lexicon: https://github.com/konstantinkobs/emote-controlled