FeelsGoodMan: Inferring Semantics of Twitch Neologisms

Anonymous ACL submission

Abstract

Twitch chat messages pose a unique problem in natural language understanding due to a large presence of neologisms, specifically emotes. There are a total of 8.06 million emotes, over 400k of which were observed during the study period. There is virtually no information on the meaning or sentiment of emotes, and with a constant influx of new emotes and drift in both their frequencies and their perceived meanings, it becomes impossible to maintain an updated manually-labeled dataset. Our paper makes a two-fold contribution. First, we establish a new baseline for sentiment analysis on Twitch data, outperforming the previous benchmark by 7.36 percentage points. Secondly, we introduce a simple but powerful unsupervised framework based on word embeddings and k-NN to enrich existing models with out-of-vocabulary knowledge. This framework allows us to auto-generate an emote pseudo-dictionary, and we show that we can nearly match the supervised benchmark above, even when injecting such emote knowledge into sentiment classifiers trained on extraneous datasets such as movie reviews or Twitter.

1 Introduction

Live streaming platforms such as Amazon Twitch or YouTube Live have become increasingly popular in the last decade and have seen an even faster growth in the last couple of years due to the COVID-19 pandemic and the rise of esports. Users on these platforms watch videogame players livestream their gameplay, and comment live on the stream to share their opinions with the streamer and the rest of the audience. Given the instantaneous and idiosyncratic nature of the chat room culture, the language used is very different from a formal conversation. It is riddled with grammatical errors, abbreviations, game-specific lingo, as well as emoji and emoji-like icons. In particular, Twitch users make heavy usage of *emotes*, which are Twitch-specific icons or animations used to express a particular emotion, feeling, or inside-joke[1].

Emotes on Twitch have become a language of their own and have both changed and enriched how people communicate with each other on the platform. They can be interspersed within text to change the meaning of a message, or constitute an entire message on their own. They are rendered when users type a predefined string, e.g. "Kappa" → and "LUL" → . There are over 8 million emotes—over 400,000 were observed in the week surveyed, constituting one third of all unique tokens on Twitch. Like memes, emotes are generated by the community, causing a constant change in their frequency and meaning.

One emote which whose meaning has changed over time is "FeelsGoodMan" → , based on a cartoon frog from a 2005 comic by the artist Matt Furie. Furie's cartoon frog was adopted by right wing posters on various online forums like 4chan in the early 2010s. Since then, Furie has campaigned to reclaim the meaning of his character, and the emote has seen an upsurge in more mainstream non hate usage (Glitsos and Hall, 2019) and positive usage on Twitch. Our results on Twitch agree, showing that "FeelsGoodMan" and its counterpart "FeelsBadMan" are mainly being used literally.

Continuous introduction of new emotes and their cryptic origins makes it unfeasible to maintain curated dictionaries documenting their meaning and semantics. With the exception of the recent work by (Kobs et al., 2020), which classified 100 top emotes and labeled 2000 Twitch chat messages, there is a lack of analytical studies focusing on understanding Twitch data and the enigmatic language of emotes. In this paper, we aim to fill this gap.

[1] Even though there is a visual component to emotes (as well as emojis), we focus on the NLP understanding of Twitch messages, interpreting emotes as words. Extracting signals from their visual counterparts is outside of the scope of this manuscript.

2 Our contribution

In this paper we set to address two core tasks:

(A) Perform sentiment analysis on Twitch data better than previous baselines set by (Kobs et al., 2020). In addition, we introduce a framework that can handle emote drift without additional major data labeling effort.

(B) Provide a broad insight into emote semantics and their sentiment. This is to address the lack of a broad understanding of thousands of emotes.

To address Task (A):

- We conducted a thorough set of experiments comparing standard traditional machine learning methods for supervised sentiment analysis on Twitch data. To the best of our knowledge, no such foundational analyses have been performed; only a lexicon-based approach and a deep learning approach with noisy labels have been tried by (Kobs et al., 2020).
- We show that our best model outperforms the previous benchmark set by (Kobs et al., 2020) by 7.36 percentage points on accuracy.
- We break down the performance of our base classifiers and demonstrate that features with emotes constitute more than 50% of feature importance, while comprising only 20% of features.
- We introduce a drift-resilient framework to Learn Out Of Vocabulary Emotions (LOOVE). Requiring little to no additional data labeling, LOOVE is able to incorporate new emote knowledge into existing models without relying on emotes as explicit features.

As for Task (B):

- We create an emote pseudo-dictionary based on word embedding neighborhoods.
- We automatically infer a corresponding sentiment for thousands of emotes from the emote pseudo-dictionary.

The remainder of the paper is organized as follows: Section 3 provides an overview of the related literature. Section 4 presents our experiments to establish a new set of supervised baselines on the Twitch dataset. In Section 5, we introduce our framework LOOVE to augment external classifiers with emote knowledge. Finally, Section 6 discusses the construction and properties of the emote pseudo-dictionary.

Additionally, in the Appendix A we present the Emote Case study. In Appendix B we study of trends in the Twitch Unlabeled Dataset that we collected for the study. In Appendix C we showcase additional applications of Twitch w2v embeddings.

3 Related Work

There are few studies on Twitch and emotes. We relied on all existing relevant Twitch studies as well as the relevant literature found on other neologisms such as emoticons, emoji and slang.

3.1 Emotes and Twitch

Labeled Twitch emote data is virtually non-existent. Kobs et al. (2020) conducted a study with the Twitch community and semantically labeled 100 frequently occurring emotes in 2018. Although the top 100 emotes account for 35.1% of the tokens and the top 1000 account for 52.1% of tokens, there are over 8 million total emotes, with over 400,000 emotes observed in the week studied, and the number growing every day. The primary contribution of that work was providing the sentiment analysis baseline for Twitch data, as well as a labeled dataset of 2000 chat messages. For the baseline, they used an Average Based Lexicon approach, which represents a comment as a sequence of tokens with an assigned sentiment from a look up table. To come up with the sentiment score for the entire comment, they averaged the sentiment scores of the tokens. This approach along with its variation achieved a 61.8% and 62.8% accuracy, respectively. They also employed a Convolutional Neural Network (CNN) approach, which was weakly-trained on the data generated by the Average Based Lexicon approach. This resulted in 63.8% accuracy.

Another noteworthy study of emotes was done by (Barbieri et al., 2017), who tried to address emote prediction, i.e. which emote the user is more likely to use, and trolling detection, i.e. "a specific set of emotes which are broadly used by Twitch users in troll messages." The authors were only predicting the top 30 most frequently used emotes. The highest F1 score for emote prediction was 0.39. The highest F1 score for trolling detection was 0.81.

Other emote and Twitch related studies include "Classification of viewers by their consumption behavior and analysis of subscribers' emote usage" (Oh et al., 2020); prediction of subscription status of a user in a channel based on user's comments (Loures et al., 2020); and a master's thesis on language variety on Twitch entitled "The present text is a research into the language usage in Computer-Mediated Communication, specifically on the online streaming platform Twitch.tv." (Hope, 2019).

3.2 Emoji and Emoticons

The sentiment analysis involving emojis and emoticons have been addressed with both deep learning (DL) and traditional machine learning (ML) approaches using various datasets. A notable DL approach is a emotion-semantic-enhanced bidirectional long short-term memory (BiLSTM) network with the multi-head attention mechanism model (EBILSTM-MH) (Wang et al., 2020). This achieved 71.70% accuracy on microblog text data involving emojis. Additionally authors showed that an SVM model achieved 66.81 accuracy on the same dataset. Another traditional ML approach for sentiment analysis, in this case using Twitter data, was carried out by (Illendula and Yedulla, 2018). It is based on emoji embeddings from 147 million tweets, trained on Random Forest (RF) and Support Vector Machine (SVM) with an overall accuracy of 62.1% and 65.2% respectively. This approach outperformed the then current state of the art.

Studies have been done to understand the semantics of emoji and emoticons. One notable work is EmojiNet by (Wijeratne et al., 2016). This is the first machine readable sense inventory for emojis. The researcher created a centralized table of emoji definitions, incorporated from multiple resources. Additionally, through word sense disambiguation techniques they assigned senses to emojis. Additionally, (Illendula and Yedulla, 2018) presented a thorough study of emoji semantics and their use in social media posts.

3.3 Slang

(Wilson et al., 2020) used Urban Dictionary [2] as a corpus to create slang word embeddings. For sentiment prediction (64.4% accuracy) and sarcasm prediction (80.2% accuracy) they achieved marginally better scores than other standard pre-trained embeddings such as word2vec-GoogleNews when initializing classifiers with UD embeddings. Other notable approaches include: a framework that combines probabilistic inference with neural contrastive learning that models the speaker's word choice in a slang context (Sun et al., 2021). In addition a BiLSTM based model was utilized for slang detection and identification at the sentence level with an F1-score of 0.80.

4 Supervised Sentiment on Twitch: new SoTA

In this section we establish a new set of baselines for Twitch chat sentiment outperforming the past state of the art method (Kobs et al., 2020) by 7.36 percentage points on accuracy. We also investigate the driving features behind the method, showing that emotes contribute significantly to the performance of the model, constituting on average over half of the Gini feature importance (using a Random Forest classifier). This is despite being only a fifth of the classifier's features.

4.1 Dataset

For our training and testing corpus we used the Twitch sentiment dataset by (Kobs et al., 2020) which we will refer to as the Emote Controlled dataset or EC. This dataset is composed of $1,880$ examples with 40.6%/38.0%/21.4% positive/neutral/negative class split. Data was split in a stratified fashion; designating 80% of the data, 1502 examples for training and 20%, 378 examples for testing [3].

4.2 Features & Models

We focused on Twitch chat sentiment analysis using traditional ML approaches, because, to the best of our knowledge, it had not previously been investigated. We trained Naive Bayes (NB), Logistic Regression or Maximum Entropy (ME), Random Forest (RF) and Support Vector Machines with linear kernel (SVM) models as they have been the most popular traditional ML algorithms for sentiment detection to date (Yadav and Vishwakarma, 2020; Zimbra et al., 2018).

The input features to the models were constructed using a well established simple sentiment analysis approach based on a bag-of-features (Pang

[2] https://www.urbandictionary.com/

[3] A 5-fold cross validation evaluation is consistent with the results obtained with a fixed split.

et al., 2002). We tested both unigrams and unigrams plus bigrams as input features. We additionally tried three text processing methods. In our study we call minimal text processing Processing 1 (P1). It involves punctuation removal, lowercasing tokens and removing like characters that occur more than three consecutive times [4]. Processing 2 (P2) refers to P1 processing plus stop word removal. Processing 3 (P3) is P2 plus lemmatization of tokens.

4.3 Results

To our surprise, the "textbook" ML approaches with minimal processing, which we call P1, outperforms the previous Twitch sentiment baselines (Kobs et al., 2020) of 63.8% by 7.36 percentage points on accuracy in the best case on the same EC dataset. The accuracy results are summarized in Table 1.

The first column of Table 1 describes the classifier and the type of input features, the rest of the columns are processing type. The integer following the classifier's name refers to the number of ngrams generated from the corpus [5].

Clf	P1	P2	P3
NB.1	61.9	59.73	58.63
NB.2	60.58	59.45	58.9
ME.1	68.52	68.22	67.12
ME.2	69.31	69.59	68.49
SVM.1	67.72	69.59	69.32
SVM.2	68.78	69.59	68.22
RF.1	70.37	66.85	67.4
RF.2	**71.16**	68.49	67.95

Table 1: EC Test Dataset Accuracy for various models

From Table 1 it is evident that RF.2 with P1 processing, P1.RF.2, outperforms all benchmarks in accuracy delivering 71.16%.

We break down the performance of P1.RF.2 to demonstrate the driving features behind the classifier. Table 2 shows the cumulative sum of the Gini RF feature importance. Because we trained a ternary one-versus-rest classifier, results in the table are displayed for each class: positive, negative and neutral. Additionally, because the features consist of unigrams and bigrams, emotes can occur as unigrams and as a part of bigram. We differentiate it as follows: the column "emotes only" refers to unigram emote features and the "emotes+" column refers to bigrams that have at least one emote. Across the three classes "emote only" contributes on average 0.4493, "emote+" on average contributes 0.0938, while constituting 0.104 and 0.1036 respectively of the total features. Combined emote features constitute 0.5431 [6] of the Gini feature importance to the performance of the model (using a RF classifier). This is despite being only 20.76% of the features. This is summarized in Table 2.

	emote only	emote+	other
Importance Positive	0.5556	0.1092	0.3789
Importance Neutral	0.4191	0.0764	0.5318
Importance Negative	0.3733	0.0958	0.5557
Features Fraction	0.104	0.1036	0.7924

Table 2: Feature Importance by Label and Features Fraction for each token type

We further examine the top 100 features of positive, negative and neutral classifiers. First we arrange features of each classifier by Gini index. Then we split the features into 2 categories: emote features [7] and "other" features. For each feature set we generate a histogram from the feature position index. Results are presented in Figure 1. From the figure it is evident that emote histogram is biased toward "0" implying higher importance, while the "other" histogram is biased towards "100". The mean/median of the emote feature indices is 42.16/38 and the mean/median of "other" feature indices is 59.35/61.

To conclude, it is important to note that the performance difference between P1.RF.1 and P1.RF.2 is marginal. This implies that the introduction of bigrams is not that significant to the overall performance of the classifier. In fact the difference between a significant number of classifier combinations listed in Table 1 are marginal, implying that the choice of a classifier with these features is not significant, perhaps with the exception of NB. However, the presence of emote features is significant.

[4] For instance: *loooove -> looove, haaaaaate -> hate.*

[5] For example, NB.1 is a Naive Bayes classifier trained on unigrams, while RF.2 is a Random Forest classifier trained on unigrams and bigrams.

[6] The average is computed across positive, negative and neutral classifiers including emote features and emote+ features.

[7] "emote only" and "emote+"

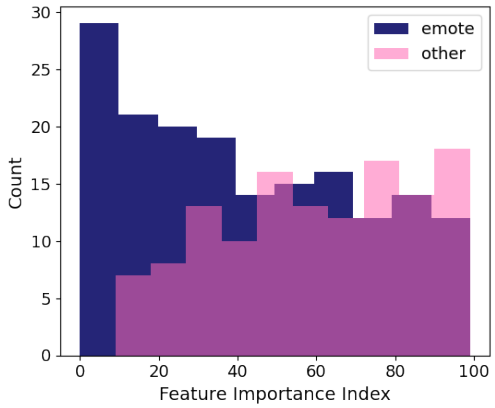


Figure 1: Histograms of top 100 Gini feature importance of P1.RF.2 (for negative, positive, neutral classifiers) for emote and "other" features

5 LOOVE - Learning Out Of Vocabulary Emotions

Now that we have established solid baselines for the fully supervised case, we consider the task of nearing these benchmarks with a solution that resists drift and requires minimal to no supervision. This is necessary because new emotes are constantly introduced, and their usage distribution frequently changes.

Our baseline models study from Section 4 demonstrated the critical importance of including emote information in the model. In that case, that information was explicitly encoded in per-emote features. We now want to abstract away from this requirement in order to further resist drift and ensure generalization to new emotes.

5.1 LOOVE

We introduce a simple but powerful framework that successfully meets the requirements above. In particular, our framework—which we call LOOVE—is able to Learn Out Of Vocabulary Emotions, and enrich existing models with this knowledge.

The framework is depicted in Figure 2. We start with an existing sentiment classifier: this could be a Twitch sentiment classifier that needs to be enriched with the knowledge of newly-introduced emotes; or could be a sentiment classifier trained on a completely separate dataset such as Twitter. The output of this classifier is then concatenated with emote sentiment stats obtained without needing labeled data. Specifically, for each unseen emote in the text being evaluated, its sentiment is auto-generated, averaging out the sentiment of known words in the word embedding neighborhood of the emote. Rather than introducing per-emote features, we perform "pooling" by keeping only a few statistics, such as the mean, max, min of the sentiment of the emotes in the text.

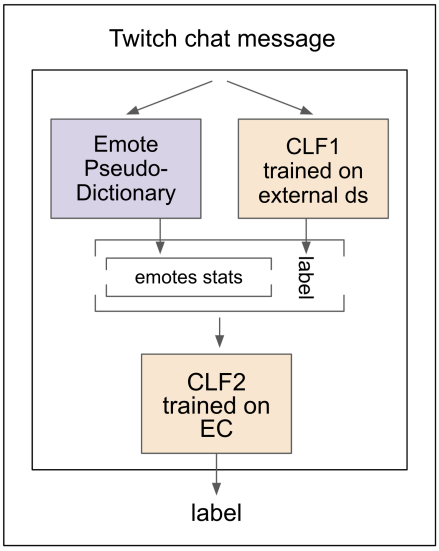


Figure 2: LOOVE framework.

The LOOVE framework has several amenable properties. First, word embeddings can be trained in an entirely unsupervised manner. A periodic retraining or fine-tuning of this space removes the need to maintain a labeled dataset or a manual lexicon as new emotes are introduced. Second, our framework decouples the existing classifier from the new OOV knowledge. In practice this is very important since companies are wary of completely changing or retraining their production classifiers given they might be used across different applications. Third, while we could have encoded the emote knowledge simply by concatenating the actual emote embedding vector (with some pooling across emotes), the decision to encode just a few stats (possibly even just the average inferred sentiment) is a much more robust choice: it results in just a handful of parameters which makes tuning of the final classification extremely simple and customizable (which can be done manually or learned with very few examples); these stats are also resilient to word embedding space rotations or shifts happening upon retraining or fine-tuning. Finally, we point out that our framework is not limited to sentiment analysis, nor emotes, and can be applied to other scenarios suffering from out-of-vocabulary issues.

5.2 Emote Pseudo-Dictionary

We trained a w2v model (Mikolov et al., 2013) on a Twitch Unlabeled Dataset [8], with minimal word occurrence set at 30 and a context window of size 5. This generated 444,714 embeddings comprised of words, emotes, emojis and emoticons. Emotes represented 33.3% of the vocabulary, words 66.2%, and the last 0.5% was spread across emojis and emoticons. Token statistics are displayed in 3.

	word	emote	emoji	emoti
Unique Tokens	286795	144339	1899	240
Unique Fraction	0.6619	0.3331	0.0044	0.0006
Occurences Fraction	0.922	0.0548	0.0086	0.0022

Table 3: The Twitch Unlabeled Dataset token statistics

In addition to the w2v model we compiled a reference sentiment table. We used the VADER lexicon (Hutto and Gilbert, 2014) augmented with an emoji/emoticon lexicon (Novak et al., 2015) [9]. For each emote, we generated a sentiment value by finding the top 5 neighboring words in the embedded space with an existing sentiment value in the reference sentiment table and took their mean[10].

We observed 0.353 RMSE when tested against Vader's vocabulary and 0.275 RMSE accuracy when we tested against 100 manually labeled emotes provided by (Kobs et al., 2020). We want to point out that this method is limited, as not every emote has neighbors that are in the reference sentiment table. Due to this limitation, we are able to generate sentiment for 22,507 emotes, even though we have embedding for over 144,000 emotes. Despite this limitation, automatically labeling over 22,000 emotes is a tremendous leap forward, as only 100 emotes have been classified before in the literature.

5.3 External Datasets

We used the EC dataset described in Section 4 and three publicly available datasets for ternary sentiment classification, Rotten Tomatoes (RT) (Pang and Lee, 2005), Twitter Dataset (T) of manually labeled tweets (Eisner et al., 2016) and sampling Yelp Dataset [11] (Y). All datasets were split in a stratified fashion with 80% designated for training and 20% for testing. RT has 8,528 with 42%/20%/38% positive/neutral/negative split; Twitter has 64,596 examples with 29%/46%/24% class split. For Yelp we used 150,000 examples with balanced classes.

5.4 Final Classification

For the second stage of the model, we incorporated abstracted emote information in the form of their sentiment stats and combined it with the prediction of the classifier trained on external data. The resulted feature vector is shown in column 1 of Table 5. Using these features we trained a secondary classifier using the EC dataset. Despite using EC for training, we are only effectively using this data for statistical information about emotes.

	None	ME	SVM	RF
None		61.11	61.64	65.61
EC	71.16	71.16	71.16	71.16
RT	36.77	61.64	61.11	65.08
T	43.92	64.29	64.55	69.31
Y	41.8	61.38	62.17	65.61

Table 4: Accuracy results for LOOVE variants tested on EC dataset. Each row is an external dataset that CLF1 is trained on. Each column is a CLF1 variant.

5.5 Results

Accuracy results for LOOVE variants [12] are presented in Table 4. As depicted in Figure 2 LOOVE is composed of 2 classifiers: CLF1 and CLF2. CLF1 is trained on an external dataset. CLF2 is trained on EC dataset using the outputs of emote pseudo-dictionary stats and CLF1 output. The idea is to maximize the use of external datasets, while minimizing the reliance on EC. In our experiment we trained CLF1 on 4 external datasets: EC, RT, T and Y using three algorithms: ME, SVM and RF. Additionally we tested 2 "edge cases". For the first edge case we removed CLF1 so the model only predicts using emote pseudo-dictionary. For the second edge case we removed emote pseudo-dictionary and CLF2 testing the performance of CLF1.

In Table 4 each numerical column represents an algorithm choice for CLF1. Each numerical row

[8] For a detailed dataset construction refer to Appendix B.

[9] While VADER provides a large lookup table of 7500 words and 100 emoticons, and the Emoji lookup table contains 750 emojis.

[10] To avoid outliers, we limited the search of sentiment-tagged up to the 1,000th nearest neighbor. Other methods such as weighting by distance, using the median, and various outlier removal techniques were explored, but a simple average worked best.

[11] https://www.yelp.com/dataset

[12] Trained using P1 processing and tested against the EC test set

refers to the dataset CLF1 is trained on. The first numerical column depicts the first edge case. The first numerical row of represents the second edge case.

As expected, CLF1 trained on EC gives the best performance irrespective of CLF2. However, The best performance independent of EC training for CLF1 is obtained when using RF for CLF1 trained on Twitter data (RF.T). This combination achieves 69.31% accuracy which is on par with the fully supervised SoTA baselines obtained in Section 4. We want to note that CLF1 trained only on Twitter without any emote information performs worse than a coin flip when applied to Twitch data. However, when enriched by LOOVE, its performance shoots up and nearly matches our best supervised benchmark.

In Table 5 we examine RF.T's features for CLF2. We again see that the driving features are emotes. Only 0.0708 Gini importance comes from the predicted label of CLF1, the rest 0.9292 are driven by emotes stats.

Feature Name	Importance
mean emote sentiment	0.2853
min emote sentiment	0.2568
max emote sentiment	0.2546
number of emotes	0.0968
source ds predicted label	0.0708
std emote sentiment	0.0357

Table 5: LOOVE framework: feature importances in descending order.

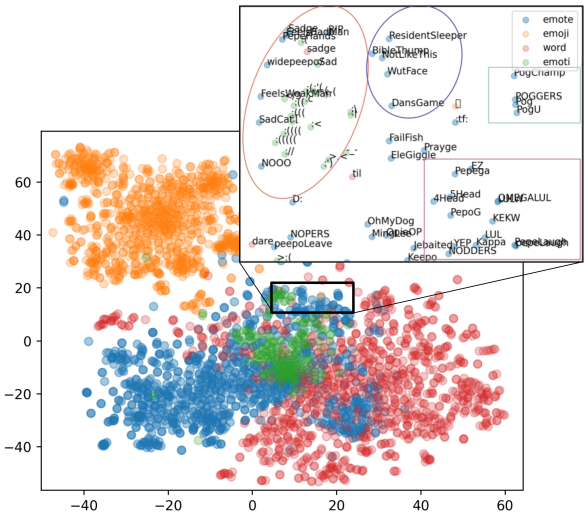


Figure 3: Top emotes, emojis, words, and emoticons (t-SNE with $perplexity = 50$, $n_{iters} = 3000$.). The orange oval is sadness, the purple oval is annoyance/disappointment, the pink square represents laughing/trolling, and the blue square excitement.

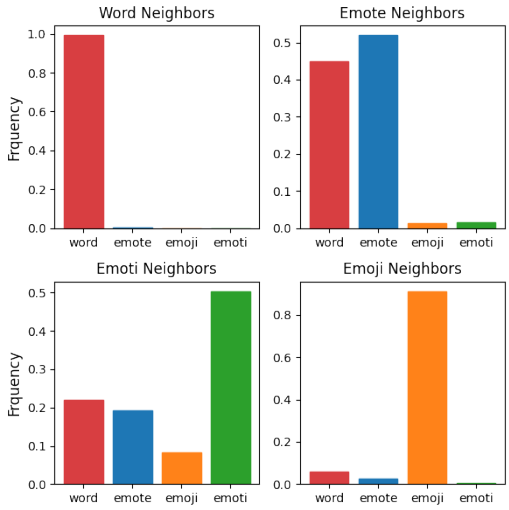


Figure 4: Distribution of neighbors for each type.

6 Word Embedding Space Analysis

Given the vital importance of emotes and the success of the LOOVE we want to examine the w2v embedding space that LOOVE is based on and take a closer look at the structure of the emote pseudo-dictionary.

We used t-SNE to visualize embeddings in 2D of the top 1000 emotes, 1000 words, 1000 emojis, and 240 emoticons, for a total of 3240 tokens (Figure 3). Visually, one can see that words, emotes, and emoticons overlap while the emoji cluster is more isolated. However, it is also visually evident that tokens cluster by their corresponding type. In Figure 4, we show the distributions of the 100 closest neighbors for each token type [13]. We can see that emojis indeed tend to cluster around their own type with very little exceptions, just like words do. Emotes and emoticons also like to cluster around their own type, but have neighbors from other types as well. A partial success of the emote pseudo-dictionary can be attributed to the fact that emotes tend to cluster around words with 0.45 frequency. Since the labeled tokens from VADER are predominantly words, these are used as neighbors in k-NN to learn emote sentiment. This is why it is possible to construct the emote pseudo-dictionary without

[13] looking at the same 3,240 tokens from the t-SNE visualization

relying on the sentiment of the nearest neighbor emotes themselves.

The box overlaying Figure 3 zooms into a particular area of the space where we can find four distinct clusters representing the emotions of sadness, annoyance/disappointment, laughing/trolling, and excitement ("PogChamp"-like emotes). We examine this trend—clustering by sentiment—to see if it remains true across the embedded space. Using tokens from the reference sentiment table from Section 5, we looked at the sentiment of the top 1,000 token neighbors and plotted sentiment histograms by label type (Figure 5, top row). Since the sentiment in the lookup table is a float between -1 (negative sentiment) and 1 (positive sentiment), we define each label type using an even 0.66 partitioning. In addition to tokens in the sentiment lookup table, we also plotted the derived emote sentiment (Figure 5, bottom row).

The top row shows that all distributions are bimodal, with only the negative distribution being significantly biased towards the negative sentiment. On the other hand, the distributions from the bottom row are more pronounced. We can see that the positive distribution now has a clearly defined shift towards 1. The neutral distribution is a lot less bimodal. However, the negative distribution is not as pronounced as before, though it is still heavily biased towards -1. Overall, the newly generated distributions have a more pronounced bias towards the expected sentiment class. When we look at the derived emote sentiment, the bottom row of (Figure 5, we see a more prominent distribution for the positive and neutral class, with a slight bias towards negative sentiment for negative classes. This serves as an indicator that on average local neighbors tend to have the same sentiment, further illustrating the strength of our emote pseudo-dictionary.

In Appendix A we present a case study involving "FeelsGoodMan" and "FeelsBadMan" emotes as an example of emote pseudo-dictionary use for emote interpretation. In Appendix C we propose several applications of the Twitch w2v embeddings in other fields.

There are many problems that still need to be addressed. Despite establishing a new Twitch sentiment baseline, which performs sentiment analysis on par with other methods and datasets in terms of accuracy (Zimbra et al., 2018), overall performance can still be improved. A potential improvement to our transfer learning model as well as emote emote pseudo-dictionary could be in the construction of an actual synonym space, rather than directly using w2v space. A notable approach to create a synonym-antonym space has been proposed in the literature by (Samenko et al., 2020). Using this kind of vector space to find both synonym and antonym emotes could be more successful.

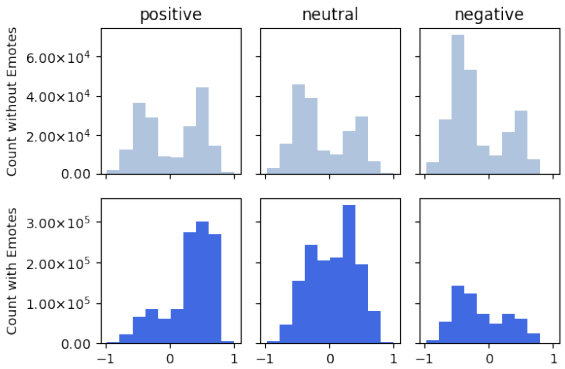


Figure 5: Top row depicts the sentiment of 1000 neighbors for each token of the original lookup sentiment table. Bottom row is for the derived emote sentiment.

7 Conclusion

We created multiple baselines for sentiment analysis that in the best case outperformed the previous metric by 7.36 percentage points. We established the importance of emotes in sentiment analysis of Twitch data by examining the features of the baseline models, showcasing the importance of emote features. We then introduced our LOOVE unsupervised framework, which abstracts away from the explicit use of emotes as features and uses emote stats along with a classifier trained on non Twitch data to predict sentiment. This model performs nearly on par with fully supervised baselines. LOOVE is based on w2v embeddings trained on over 313 million Twitch chat messages in conjunction with k-NN. A driving feature behind the framework is a emote pseudo-dictionary which can be used to derive sentiment for unknown emotes. Using this emote pseudo-dictionary, we created a sentiment table for $22,507$ emotes. This is the first case of emote understanding on this scale.

References

Francesco Barbieri, Luis Espinosa-Anke, Miguel Ballesteros, Juan Soler-Company, and Horacio Saggion. 2017. Towards the understanding of gaming audiences by modeling twitch emotes. In *Proceedings*

of the 3rd Workshop on Noisy User-generated Text, pages 11–20, Copenhagen, Denmark. Association for Computational Linguistics.

Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bošnjak, and Sebastian Riedel. 2016. emoji2vec: Learning emoji representations from their description. *Computation and Language, arXiv:1609.08359. Version 2*.

Laura Glitsos and James Hall. 2019. The pepe the frog meme: an examination of social, political, and cultural implications through the tradition of the darwinian absurd. *Journal for Cultural Research*, 23(4):381–395.

Henrik Hope. 2019. "hello [streamer] PogChamp": The language variety on twitch. Master's thesis in literacy studies, University of Stavanger, Norway.

Clayton J. Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *ICWSM*. The AAAI Press.

Anurag Illendula and Manish Reddy Yedulla. 2018. Learning emoji embeddings using emoji co-occurrence network graph. *Social and Information Networks, arXiv:1806.07785*.

Konstantin Kobs, Albin Zehe, Armin Bernstetter, Julian Chibane, Jan Pfister, Julian Tritscher, and Andreas Hotho. 2020. Emote-Controlled: Obtaining Implicit Viewer Feedback Through Emote-Based Sentiment Analysis on Comments of Popular Twitch.tv Channels. *ACM Transactions on Social Computing*, 3(2):1–34.

Túlio Corrêa Loures, G. L. Fernandes, Fernanda G. Araújo, Karen S. Martins, and Pedro O. S. Vaz de Melo. 2020. Stinkycheese: Chat-based model for subscription classification. In *ChAT@PKDD/ECML*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Petra Kralj Novak, Jasmina Smailović, Borut Sluban, and Igor Mozetič. 2015. Sentiment of emojis. *PLOS ONE*, 10(12):e0144296.

Soyoung Oh, Jina Kim, Honggeun Ji, Eunil Park, Jinyoung Han, Minsam Ko, and Munyoung Lee. 2020. Cross-cultural comparison of interactive streaming services: Evidence from twitch. *Telematics and Informatics*, 55:101434.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 115–124, Ann Arbor, Michigan. Association for Computational Linguistics.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - EMNLP '02*, volume 10, pages 79–86. Association for Computational Linguistics.

Igor Samenko, Alexey Tikhonov, and Ivan P. Yamshchikov. 2020. Synonyms and Antonyms: Embedded Conflict. *Computation and Language, arXiv:2004.12835. Version 2*.

Zhewei Sun, Richard Zemel, and Yang Xu. 2021. A computational framework for slang generation. *Transactions of the Association for Computational Linguistics*, 9:462–478.

Shaoxiu Wang, Yonghua Zhu, Wenjing Gao, Meng Cao, and Mengyao Li. 2020. Emotion-semantic-enhanced bidirectional LSTM with multi-head attention mechanism for microblog sentiment analysis. *Information*, 11(5):280.

Sanjaya Wijeratne, Lakshika Balasuriya, Amit Sheth, and Derek Doran. 2016. EmojiNet: Building a machine readable sense inventory for emoji. *Computation and Language, arXiv:1610.07710*.

Steven Wilson, Walid Magdy, Barbara McGillivray, Kiran Garimella, and Gareth Tyson. 2020. Urban dictionary embeddings for slang NLP applications. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4764–4773.

Ashima Yadav and Dinesh Kumar Vishwakarma. 2020. Sentiment analysis using deep learning architectures: a review. *Artificial Intelligence Review*, 53(6):4335–4385.

David Zimbra, Ahmed Abbasi, Daniel Zeng, and Hsinchun Chen. 2018. The state-of-the-art in twitter sentiment analysis: A review and benchmark evaluation. *ACM Transactions on Management Information Systems*, 9(2):5:1–5:29.

A Emote Case study

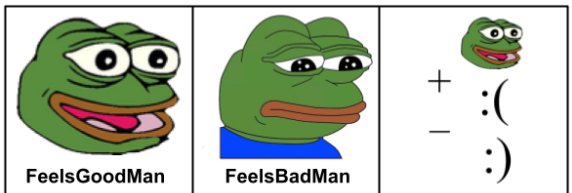
In Figure 3 we showed a zoomed-in t-SNE plot of four distinct clusters representing the emotions of sadness, annoyance/disappointment, laughing/trolling, and excitement ("PogChamp"-like emotes). Here we present a case study, looking at two contrasting emote representatives, "FeelsGoodMan" and "FeelsBadMan". We aggregate the top 10 neighbors for each emote and display them in Figure 6.

From the figure it is evident that the top neighbors for each are semantically similar. The top neighbors for "FeelsGoodMan" are "EZY", "EZ", "FeelsAmazingMan", "FeelsOkayMan", "widepeepoHappy", "pepeW" and "Okayge". Each of these can be considered a different emote "flavor"

of “FeelsGoodMan”, depicting “Pepe the Frog” with a positive connotation. Other neighbors are used with a positive sentiment as well.

“FeelsBadMan” tells a similar story, but with a polar opposite sentiment. “Sadge”, “PepeHands”, “Smoge”, “sadge”, “peepoSad” again depict “Pepe” just like “FeelsBadMan”, but with a sad, negative connotation. Other neighbors, do not feature “Pepe” but represent sadness as well. It is quite remarkable that the neighbors in both cases not only contain the same semantics as the original emote, but several feature “Pepe” as well.

To further strengthen the case we show that it is possible to find “FeelsBadMan” using vector additions starting from “FeelsGoodMan” (and vice-versa) . As demonstrated by (Mikolov et al., 2013) with the $Woman + King - Man = Queen$ example, we observed that by adding the frown emoticon “:(” to “FeelsGoodMan” and subtracting the smile emoticon “:)”, we obtain “FeelsBadMan” in the top 3 closest embeddings Figure 6.



Token	Similarity	Token	Similarity	Token	Similarity
EZY	0.65	Sadge	0.896	Sadge	0.721
FeelsAmazingMan	0.647	PepeHands	0.896	PepeHands	0.681
Clap	0.634	Smoge	0.806	FeelsBadMan	0.68
EZ	0.624	:(	0.806	:'(	0.643
FeelsOkayMan	0.616	:'(	0.78	sadge	0.638
widepeepoHappy	0.589	sadge	0.758	widepeepoSad	0.638
pepeW	0.58	peepoSad	0.758	peepoSad	0.631
=D	0.568	=(	0.736	t_t	0.603
Okayge	0.567	:(	0.73	;(	0.582
OkayChamp	0.561	:/	0.724	😭	0.579

Figure 6: Neighbors of “FeelsGoodMan” vs. “FeelsBadMan” (all strings besides the emoticons above are emotes). Adding “:(” to “FeelsGoodMan”, subtracting “:)”

B Twitch Unlabeled Dataset

Our unlabeled dataset consisted of 313M chat messages from 521 thousand streams over the course of 1 week in April (06/07/21 - 06/13/21). There was an average of roughly 45K unique streamers per day. The number of messages per stream varies wildly, depending on the popularity of the streamer and the game they are playing. Up to 30% of streams on any given day receive no messages. Looking at the data for a randomly selected day (06/08/21), the median number of messages for a stream is 117, the mean 744, with a STD of 13,585. The top 1200 streams account for 50% of all messages, while representing 1.7% of all streams (71,917).

B.1 Emotes Dataset

We fetched 8.06M emotes from three sources: the Twitch official API, FrankerFaceZ (FFZ), and BetterTTV (BTTV). FFZ consists of 253,335 emotes, BTTV consists of 381,389 emotes, with the remaining Twitch official emotes. Of these, 41k emotes appear in more than one group. While the number of Twitch official emotes dwarf the other two, FFZ and BTTV emotes are incredibly popular, representing 44 of the top 100 most used emotes.

B.2 Trends

Considering that emotes account for 33% of unique tokens in the Twitch Unlabeled Dataset (as depicted in 3), we wanted to understand their usage frequency. By generating a rank-frequency distribution, we showed that emotes follow a power law (Figure 7). In fact, emote rank-frequency distribution is quasi-Zipfian with a power of 0.97. Similarly, we observe that words follow a power law, as expected. However, emojis and emoticons behave somewhat differently.

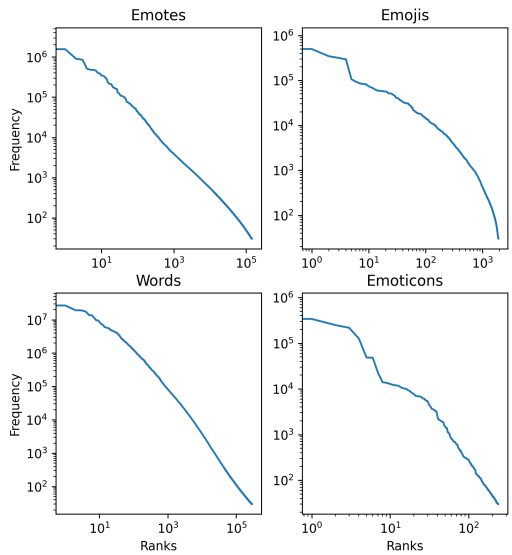


Figure 7: Frequency vs Rank by token type (log-log)

C Embeddings Applications

In addition to using embeddings for sentiment analysis, there are other useful ways to apply these

new-found embeddings. They can be used to learn slang immediately as it proliferates in the wild, improve brand safety classifiers, quickly extend a knowledge graph or add dimension to existing nodes within it, and build improved classifiers for games, genres, and industries.

Today, slang evolves and proliferates extremely quickly due to the ubiquity of the Internet and peer-to-peer communication platforms. A platform like Twitch can be used to learn synonyms and replacements for common words and slang. The huge volume of messages and the culture of the Twitch user base makes it an ideal place to learn about new memes and slang in an unsupervised way, since they will often be some of the first users.

Brand safety is a related application. Many moderation tools rely on keywords and regular expressions to detect profane, racist, toxic, and sexual content. While these are precise, they are likely to miss clever and new misspellings, and will certainly miss entirely new strings which are being used to "safely" convey the same meaning as their known counterparts. These could be words, emojis, or even emotes. This w2v model provides a way to organically learn which alternate constructions are being used to circumvent existing moderation filters.

Another application is in expanding a Knowledge Graph to incorporate related entities, or to add variations/nicknames of known entities. This works best in the gaming-space since that is the community's focus, but also for television shows, cryptocurrencies, sports, etc. Given the string "hikaru", the closest embeddings are names of dozens of other chess players. In 6 Given the string "morde" (short for "Mordekaiser", a champion in League of Legends), the model returns other champions from the game and their nicknames. This was also tested for a few other words such as "vaxx" (short for vaccine), "grau", a popular gun in the game Call of Duty: Warzone, and other words.

	hikaru	morde	grau	vaxx	troll
1	danya	darius	ffar	vax	bully
2	levy	vayne	kar98	vacc	tryhard
3	kasparov	aatrox	mac10	vacine	bait
4	anish	panth	spr	vaccin	jebait
5	naroditsky	leona	bruen	vaccination	grief
6	samay	voli	bullfrog	phizer	ban
7	lefong	trundle	m13	vaccine	simp
8	fabi	garen	ram7	astrazeneca	toxic
9	nihal	heimer	dmr	moderna	bm
10	nili	sion	fara	covid-19	trolling

Table 6: Five words and their most similar tokens.