FeelsGoodMan: Inferring Semantics of Twitch Neologisms

Pavel Dolin
Spiketrap
San Francisco, CA, USA
pavel@spiketrap.io

Luc d'Hauthuille
Spiketrap
San Francisco, CA, USA
luc@spiketrap.io

Andrea Vattani
Spiketrap
San Francisco, CA, USA
andrea@spiketrap.io

Abstract

Twitch chats 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 used in the week studied. There is virtually no information on the meaning or sentiment of emotes, and with a constant influx of new emotes and drift in their frequencies, 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 supervised benchmark by 7.9% 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 a pseudo-dictionary of emotes 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 outbreak and the rise of esports. Users on these platforms watch videogame players live-streaming their gameplay, and comment on the stream in realtime to share their opinions with the streamer and the rest of the audience. Given the realtime nature of the chat room and the subject matter, the language is very different from common English, being riddled with grammatical errors, abbreviations, game-specific lingo, alongside 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 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 with over 400k of which were used in the week surveyed, constituting 1/3 of all unique tokens on Twitch. Similarly to memes, emotes are user- (or community-)generated causing a constant change in their distribution and semantic.

An example of an emote whose meaning has evolved drastically over time is "FeelsGoodMan". "FeelsGoodMan" is based on Pepe the Frog, a character in the 2005 comic "Boy's Club" by artist Matt Furie. Pepe memes were spread widely in 2008 on Myspace and 4chan, and Pepe was crowned "biggest meme of the year" on Tumblr in 2015. The character then became heavily politicized, first co-opted by the alt-right in 2015[2], and later used in the 2019-2020 Hong Kong protests. While the Anti-Defamation League included it in its database of hate-symbols in 2016, it found that most instances of Pepe were not actually used for hate (Glitsos and Hall, 2019). Our results on Twitch agree, showing that "FeelsGoodMan" and its counterpart "FeelsBadMan" are mainly being used literally. A 2020 documentary called "Feels Good Man" explores the origins and development of Pepe as a meme, and the creator's struggle to reclaim control of the character.

Continuous introduction of new emotes and their cryptic origins makes it unfeasible to maintain curated dictionaries documenting their meaning and

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

[2] They also co-opted "kek" from World of Warcraft.

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.

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 pose the question whether this can be done while also handling emote drift without additional major data labeling effort.

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

To address Task (A):

- We conduct 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 analysis have been performed; only a lexicon-based approach and a deep learning approach with noisy labels have been tried by (Kobs et al., 2020).
- Our best model outperforms the previous benchmark set by (Kobs et al., 2020) by 7.89% points.
- We break down the performance of our base classifiers and demonstrate that features with emotes constitute 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-none 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 lookup table for thousands of emotes from the emote "pseudo dictionary".
- Showcase how global conversations can be extracted by looking at the local neighborhoods of Twitch word embeddings.

The remainder of the paper is organized as follows: Section 2 describes our contribution and the methods we used. 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 statistical properties of emotes and their behavior in the embedded space.

3 Related Work

Sentiment Analysis Opinion mining or sentiment analysis aims to infer opinions from texts, images, videos and other media expressed toward an entity and to quantitatively assign a sentiment polarity (e.g., positive, negative, neutral) to each instance. Some of the applications include social media analytics, user reviews analysis, financial market prediction, crime prediction, disaster assessment and politics.

The recent survey by (Yadav and Vishwakarma, 2020) divides sentiment classification techniques into three categories: lexicon-based, traditional ML based and Deep Learning (DL) based. Lexicon-based methods are handcrafted and depend on calculation rules and a collection of lexical units specific to their corresponding sentiments. Traditional ML predominantly uses supervised techniques such as Support Vector Machine (SVM), Maximum Entropy (ME), Random Forest (RF) and others. Deep learning techniques also predominantly use supervised approaches including Convolution Neural Networks, Recursive Neural Networks, Recurrent Neural Networks (LSTM, GRU), Deep Belief Networks, Attention-based networks and others.

Each technique has its flaws and advantages. A lexicon based approach takes advantage of the feature control and intuition behind the model. It does not require training samples and is easy to implement. This comes at a tradeoff due to the constraints of the manual selection of calculation rules, feature selection and aggregation of lexical units. Similarly, traditional ML requires feature

engineering but takes advantage of small, labeled datasets and general optimization algorithms. In many cases this delivers more accurate results. DL techniques, like traditional ML, rely on data and the optimization of algorithms. However, the downside of the method is the requirement of large datasets to satisfy the optimization of millions and in some cases billions of algorithm parameters. Data labeling and the model effectively serving as a blackbox presents a major challenge for many researchers. However in some cases the performance of DL can outweigh these tradeoffs by outperforming other approaches by a large margin. (TODO citation)

Transfer learning Transfer learning is a method of applying knowledge from a source domain to a target domain using the similarity of data, data related features, models, tasks, etc. (Liu et al., 2019) presented a survey of sentiment analysis based on transfer learning. They split transfer learning methods into three categories: parameter-transfer methods, instance-transfer methods and feature-representation-transfer methods. Parameter-transfer methods, as the name suggests, use parameters of a model for transfer learning. Instance-transfer methods use data sharing of the source and target domain. Feature-representation-transfer methods utilize crossover features shared between the source and the target domain.

Emotes and Twitch Labeled Twitch emotes 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 400k emotes used in the week surveyed, and the number grows 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 average 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 CNN approach, which 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 "The present text is a research into the language usage in Computer-Mediated Communication, specifically on the online streaming platform Twitch.tv." (Hope, 2019).

Emoti and Emoji (Wang et al., 2020) introduced Emotion-Semantic-Enhanced Bidirectional LSTM (BiLSTM) and a multi-head attention mechanism combined model applied to microblog sentiment analysis. Their model separated emojis and words at the input level. They used Word2Vec model approach for word embeddings. The overall accuracy was 71.7% for the best model variation.

(Chen et al., 2018) presented bi-sense emoji embedding in order to capture complex sentiment information. Embeddings were taken from Twitter data. Data labeling was done automatically, generating weak labels using a rule-based sentiment analysis algorithm VADER (Hutto and Gilbert, 2014a) and trained using LSTM. Their model only predicts positive and negative sentiments. The method performs marginally better than the state of the art. Their accuracy was 90% on the auto-annotated testing set and 83.9% for the human-annotated testing set.

(Illendula and Yedulla, 2018) learned and analyzed emoji embeddings from 147 million tweets (Wijeratne et al., 2017). New embeddings outperformed some of the state-of-the-art results for sentiment analysis tasks at that time. Namely a 1.4% point increase for Random Forest and 1.6% point increase for SVM compared to the state of the art methods, for an overall 62.1% and 65.2% accuracy respectively.

(Wijeratne et al., 2016) introduced EmojiNet, the

first machine readable sense inventory for emojis. They created a centralized table of emoji definitions, incorporated from multiple resources. Additionally, through word sense disambiguation techniques they assigned senses to emojis.

(Eisner et al., 2016) pre-trained embeddings for emojis and showed that they outperformed a skip-gram model trained on a large collection of tweets by a small margin, with an overall accuracy of 60.6% using Linear SVM on the entire dataset with Google News + emoji2vec.

Slang (Wilson et al., 2020) used Urban Dictionary [3] 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 GloVe and word2vec-GoogleNews when initializing classifiers with UD embeddings.

(Sun et al., 2021) created a framework that modeled the speaker's word choice in a slang context. The framework ultimately produced machine generated slang. Their "framework encode[d] novel slang meaning by relating the conventional and slang senses of a word while incorporating syntactic and contextual knowledge in slang usage," which "combine[d] probabilistic inference with neural contrastive learning to generate novel slang word usages."

(Pei et al., 2019) used deep learning methods for slang detection and identification. "We found that a prominent feature of slang is the surprising use of words across syntactic categories or syntactic shift (e.g., verb -> noun). Our best models detect the presence of slang at the sentence level with an F1-score of 0.80 and identify its exact position at the token level with an F1-Score of 0.50."

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 7.89% points. We also showcase the driving features behind the method, showing that emotes contribute significantly, constituting up to 50% of the Gini feature importance to the performance of the model (using a Random Forest classifier). This is despite being only 20% of the features.

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

Dataset For our training and testing corpus we used the Twitch sentiment dataset provided 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. [4].

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 actively investigated.

We tried the simplest approaches for creating features based on unigram and bigram token counts. To our surprise the "textbook" ML approaches outperformed the previous Twitch sentiment baselines by 7.89% points on accuracy in the best case. These results are summarized in Table 1 and Table 2.

In those tables, P1 refers to minimal text processing - punctuation removal, lowering of tokens and removal of like characters that occur consecutively more three times, for instance: *loooove* -> *looove, gooooood* -> *goood*. P2 refers to P1 processing plus stop word removal. P3 is P2 plus lemmatization of tokens.

The baselines were constructed using a well established simple sentiment analysis approach based on a bag-of-features (Pang et al., 2002). We tested unigrams and unigrams plus bigrams as input features.

We trained Naive Bayes (NB), Logistic Regression or Maximum Entropy (ME), Stochastic Gradient Descent (SGD), 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 in the past (Yadav and Vishwakarma, 2020; Zimbra et al., 2018). The integer following the classifier's name refers to the number of ngrams generated from the corpus: for example, ME.1 is a logistic regression classifier trained on unigrams, while RF.2 is a random forest classifier trained on unigrams and bigrams.

Results Tables 1-2 show that RF.2 with P1 processing (P1.RF.2) outperforms all benchmarks in both accuracy and macro average f1-score, delivering 0.7169 and 0.6835 respectively.

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

	P1	P2	P3
NB.1	0.619	0.5973	0.5863
NB.2	0.6058	0.5945	0.589
ME.1	0.6852	0.6822	0.6712
ME.2	0.6931	0.6959	0.6849
SGD.1	0.6772	0.6877	0.6767
SGD.2	0.6958	0.6904	0.6767
RF.1	0.7037	0.6685	0.674
RF.2	0.7169	0.6849	0.6795
SVM.1	0.6772	0.6959	0.6932
SVM.2	0.6878	0.6959	0.6822

Table 1: EC Test Dataset Accuracy

	P1	P2	P3
NB.1	0.586	0.5602	0.5485
NB.2	0.569	0.5573	0.5503
ME.1	0.6451	0.6473	0.64
ME.2	0.6523	0.6582	0.6497
SGD.1	0.6413	0.6533	0.6454
SGD.2	0.6547	0.6506	0.6408
RF.1	0.6691	0.6294	0.6421
RF.2	0.6835	0.6419	0.6432
SVM.1	0.6366	0.6578	0.6593
SVM.2	0.6504	0.6582	0.6447

Table 2: EC Test Dataset Avg F1-score

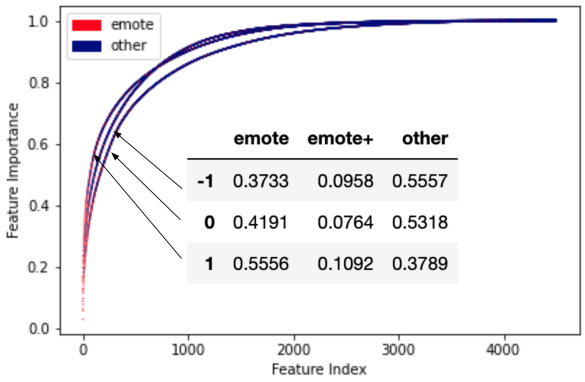


Figure 1: CUSUM of GINI feature importance of P1.RF.2 for -1, 0, 1 classes

We break down the performance of P1.RF.2 to demonstrate the driving features behind the classifier. Figure 1 shows the cumulative sum of Gini RF feature importance. Because we trained a one-versus-rest classifier, we depict scatter lines for each class.

To highlight the vital importance of emotes in the classification we depict them in red. Because features are ordered by importance we see a high concentration of features that include emotes closer to zero - signifying that they are the most important features. We also display the overall sum of the Gini importance of emotes vs other tokens, displayed in the table of Figure1. In the table columns "emotes" refers to unigram emote features and "emotes+" refers to bigrams that have at least one emote. On average, across 3 classifiers, emotes and emotes+ contribute a cumulative average of 0.5112 Gini importance over other features. Overall emotes and emotes+ make up 934 input features of the classifier which is 20.76% of the total input features.

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 Tables 1 and 2 are marginal, implying that the choice of a classifier with these features is not significant, perhaps with the exception of NB.

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 is resistance to drift and requires minimal-to-none supervision. This is necessary because new emotes are constantly introduced and their usage distribution is subject to frequent changes.

Our baseline models study from Section 4 demonstrated the importance of including emote information into 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 be resilient to Twitch emote drift.

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 enriches any existing model with this knowledge.

The framework is depicted in Figure 3. 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, unseen emotes; or could be a sentiment classifier trained on a completely separate dataset such as Twitter. The output of this classifier is then concate-

nated with emote sentiment stats obtained without the need of 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 only keeping a few statistics, such as the mean, max, min of the sentiment of the emotes in the text.

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 of maintaining 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; for example, in the case of deep learning models, catastrophic forgetting would likely occur upon periodical fine-tuning on new emotes (assuming new labeled data was available). 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 (could 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.

Experiments To simulate the case of an existing classifier, we trained a Logistic Regression sentiment classifier on extraneous datasets (that do not include emotes). Then we enrich it with the emote sentiment stats obtained from a word embedding and test the final classification against the Emote Controlled test set from the previous section.

External Datasets We used EC dataset described in Section 4 and three publicly available datasets for ternary sentiment classification (positive, negative, neutral), Rotten Tomatoes (RT) (Pang and Lee, 2005), Twitter Dataset (T) of manually labeled tweets (Eisner et al., 2016) and sampling Yelp Dataset [5] (Y). All datasets were split in a stratified fashion with 80% designated for training and 20% testing. RT has $8,528$ with 42%/20%/38% positive/neutral/negative split; T has $64,596$ examples with 29%/46%/24% class split. For Y we used $150,000$ examples with balanced classes.

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 are devoid of 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).

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.

Emote Sentiment Dictionary First we trained a w2v model on the Twitch Unlabeled Dataset, capping minimal occurrence at 30, with context window = 5. We 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 (Figure 2).

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

	Unique tokens	fraction of unique	fraction of occurences
word	286795	0.6619	0.9336
emote	144339	0.3331	0.0555
emoji	1899	0.0044	0.0087
emoti	240	0.0006	0.0022

Figure 2: Number of unique tokens, fraction of vocabulary, and number of occurrences by type

In addition to the w2v model we aggregated a reference sentiment table. We used the VADER lexicon (Hutto and Gilbert, 2014b) augmented with an emoji/emoticon lexicon (Novak et al., 2015). .

For each emote we generated a sentiment value by finding the top 5 neighboring words in the embedding space with an existing sentiment value in the reference sentiment table and took their mean[6]. 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 thousand emotes. Despite this limitation, automatically labeling $22,507$ emotes is still a tremendous leap forward as only 100 emotes have been classified before in the literature.

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 first stage, resulting in a vector with a handful of feature, listed in column 1 of Table 4. Using these features we trained and tested a secondary classifier using EC dataset. Despite using EC for training, we are only effectively using this data for statistical information about emotes - in practice these parameters could be set using a few examples or even manually for greater control. The resulting features with their learned importance are listed in Table 4.

Results The model performs on par with the baselines and does not rely explicitly on emotes. The performances for various source datasets and classifier combinations are presented in Table 3. The best performance is by LOOVE trained on Twitch with RF as the secondary classifier. It achieves a 0.6931 accuracy on the EC dataset which is on par with the fully supervised SoTA baselines obtained in Section 4.

[6]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.

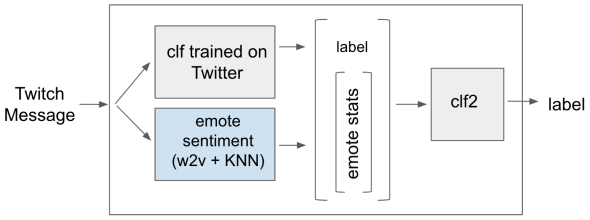


Figure 3: Transfer learning + emote stats setup

	None	ME'	SVM'	RF'
EC	0.6931	0.6931	0.6931	0.6825
RT	0.246	0.6111	0.6085	0.6746
T	0.4286	0.6243	0.6323	0.6931
Y	0.4074	0.6058	0.6032	0.672

Table 3: Results for transfer learning + emote stats

name	importance
mean emote sentiment	0.2853
min emote sentiment	0.2568
max emote sentiment	0.2546
num of emotes	0.0968
source ds pred label	0.0708
std emote sentiment	0.0357

Table 4: Transfer learning using Twitter + emote stats. RF feature importance

6 Twitch Semantic Analysis

Considering that emotes account for 33% of unique tokens in the Twitch Unlabeled Dataset Figure 2 we wanted to understand their usage frequency. By generating a rank-frequency distribution, we showed that emotes follow a power law Figure 4. 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 differently.

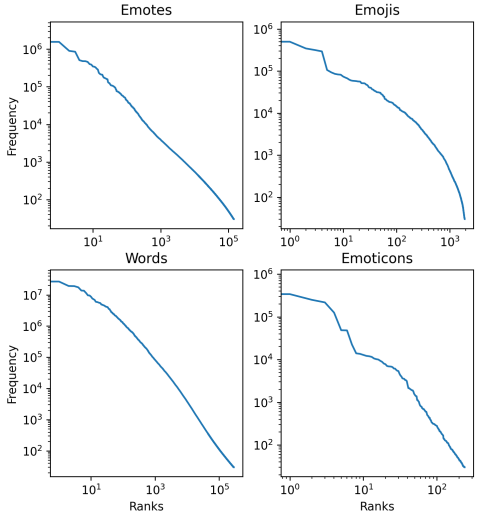


Figure 4: Frequency vs Rank (log-log)

In addition to rank-frequency distributions we wanted to understand how each token type clustered in the w2v vector space (which we used for the construction of Emote Sentiment Dictionary in Section 5). We used TSNE to visualize the token embeddings in 2D for the top 1000 emotes, words, and emojis, and 240 emoticons Figure 5. Visually one can see that word, emote, and emoticons seem to have some overlap while the emoji cluster is completely isolated. It is visually evident that tokens cluster by type, so the clusters that form in embedding space are based on the semantics and the token type.

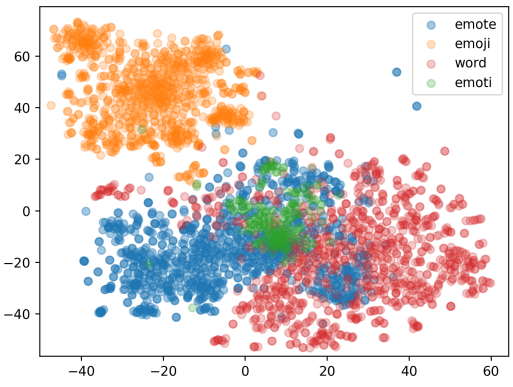


Figure 5: Top 1000 emotes, emojis and words, 240 emoticons (TSNE with $perplexity = 50$, $n_{iters} = 3000$.)

We wanted to investigate this idea further and examine the neighborhoods by token type. In Figure 7 we plotted the distributions of the 100 closest neighbors for all token types, looking at the same 3240 tokens from the TSNE visualization. We can see emoji indeed tend to prefer to cluster around their own type with very little exceptions, as do words. Emotes and emoticons also like to cluster around their own type, but tend to be neighbors with other token types. This is also an evidence why it is possible to construct an emote sentiment dictionary without relying on the sentiment of nearest neighbor emotes.

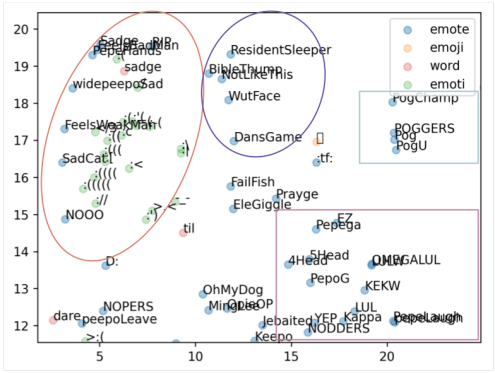


Figure 6: The orange oval is sadness, the purple oval is annoyance/disappointment, the pink square represents laughing/trolling, and the blue square excitement ("Pog").

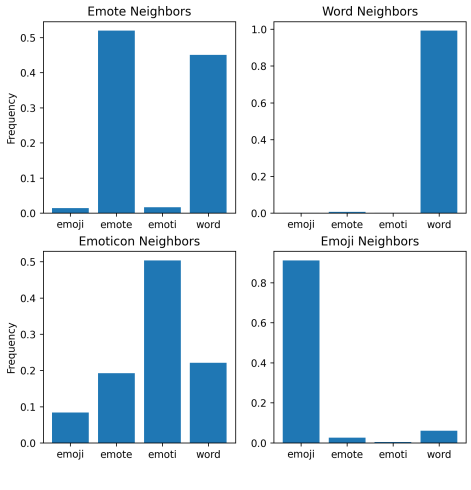
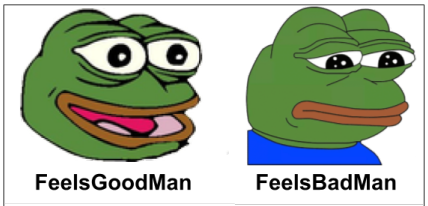


Figure 7: Distribution of the top 100 neighbors by type, averaged for 1000 emotes, emojis, and words, 240 emoticons.

Emote Clusters, a Closer Look In addition to investigating clustering by token type, we wanted

to examine semantically similar clusters of tokens in the TSNE plots. We able to identify several such regions. As an example we would like to demonstrate a zoomed-in TSNE plot of four distinct clusters representing the emotions of: sadness, annoyance/disappointment, laughing/trolling, and excitement ("PogChamp"-like emotes), depicted in Figure 6.

As an example we pick 2 emote representatives, FeelsGoodMan an emote expressing happiness and FeelsBadMan an emote expressing sadness, and take a closer look at their neighbors in Figure 8. As evident from the figure all neighbors seem to be semantically the same.



FeelsGoodMan

	Token	Distance
1	EZY	0.650
2	FeelsAmazingMan	0.647
3	Clap	0.634
4	EZ	0.624
5	FeelsOkayMan	0.616
6	widepeepoHappy	0.589
7	pepeW	0.580
8	=D	0.568
9	Okayge	0.567
10	OkayChamp	0.561

FeelsBadMan

	Token	Distance
1	Sadge	0.896
2	PepeHands	0.896
3	Smoge	0.806
4	:(	0.806
5	:'(	0.780
6	sadge	0.758
7	peepoSad	0.758
8	=(	0.736
9	:(	0.730
10	:/	0.724

Figure 8: 10 Nearest Neighbors of FeelsGoodMan vs. FeelsBadMan (all strings besides "Sadge" and the emoticons above are emotes)

The prototypical vector addition problem in word2vec, that Woman + King - Man = Queen (Mikolov et al., 2013) has an analog in emotes. If we add the frown emoticon ":(" to the emote FeelsGoodMan and subtract the smile emoticon ":)", we observe that FeelsBadMan is in the top 3 closest "sad" embeddings Figure 9, and "Sadge" is the first one.

	Token	Distance
1	Sadge	0.721
2	PepeHands	0.681
3	FeelsBadMan	0.680
4	:'(	0.643
5	sadge	0.638
6	widepeepoSad	0.638
7	peepoSad	0.631
8	t_t	0.603
9	;(	0.582
10	😭	0.579

Figure 9: Adding :(to FeelsGoodMan, subtracting :).

7 Discussion

Twitch Chat Sentiment The performance of P1.RF.2 constitutes a major improvement over the current sentiment classifier by (Kobs et al., 2020) which delivers 0.638 accuracy. Our classifier outperforms this benchmark by 7.89% points. We chose "textbook" input features and traditional ML classifiers in order to mitigate any additional complexity. Twitch chat dataset is not a standard dataset due to a large presence of neologisms, which are largely unclassified and are absent in other datasets used in previous studies.

Our introduction of LOOVE addressed the emote drift and avoided additional manual data labeling while remaining on par with other baselines. Despite slightly lower performance against P1.RF.2 by 2% points on accuracy, we believe that this tradeoff is counterbalanced by its ability to address emote drift through using emote stats rather than emotes themselves.

LOOVE can also be repurposed for other tasks, such as toxicity or profanity detection in Twitch data with a swap of datasets. Because the model relies on transfer learning for its first stage, we can easily swap the sentiment labeled dataset for a toxicity labeled one. Similarly for the second stage of the model we can easily generate the emote toxicity lookup table by replacing sentiment lookup tables with toxicity lookup tables. Further testing will be required to establish the viability of these use cases.

Emote Understanding: Improvements and Applications The fact that embeddings cluster by token type suggests that our approach will be more successful with access to more labeled data across

the different types. While VADER provides a large lookup table of 7500 words and 100 emoticons, and the Emoji lookup table contains 750 emojis, our labeled emote set is limited to 90 emotes and so are likely constrained in performance and number of predictions made for emotes. Still, emoticons, emojis, and words all contribute to sentiment predictions for emotes, since they also appear in the neighborhood.

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 after it is developed and used in the wild, improve brand safety classifiers, quickly extend a knowledge graph and learn alternate names for entities of it, and build improved classifiers for games, genres, and industries.

Slang develops and spreads extremely quickly in 2021, often in days or weeks, due to the state of the internet and the ubiquity of 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 words are being used to mean something else.

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 TV shows, cryptocurrencies, sports, etc. Given the string "hikaru", the word2vec model returns as most similar words the names of 100s of other chess players. 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.

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

Figure 10: A few example words and their similarities.

8 Conclusion

We created multiple baselines for sentiment analysis that in the best case outperformed the previous metric by 7.89% 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, that abstracts away from explicit use of emotes as features and uses emote stats along with transfer learning to predict sentiment. This model performs nearly on par with fully supervised baselines.

We trained a w2v model on over 313 million Twitch chat messages and showed several use cases for it. A "pseudo dictionary" was created emotes, which is based on local vector neighborhoods, creation of a sentiment table for $22,507$ emotes, using a deeper understanding of abstracted conversations around a given token. This is the first case of emote understanding on this scale.

There are may 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.

In this study we tried to use the simplest approaches while addressing the problem. We avoided unnecessary complexity or tailing the newest trends. We wanted to get a baseline understanding of the most important driving features behind the performance. With this accomplished, more complex methods can be explored. However Twitch chat is a very dynamic dataset with significant drift, new methods must account for this rapid change and more data must be labeled.

(Kobs et al., 2020) showed that training a CNN using weak labels generated via a fixed lexicon is not driving the performance of the model. Other labeling approaches or transfer learning could improve the performance.

A potential improvement to our transfer learning model as well as 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 have been proposed in the literature by (Samenko et al., 2020). Using this kind of vector space in order to find synonym emotes as well as antonym emotes could be more successful.

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. ZSCC: NoCitationData[s0] Accepted: 2017-11-21T10:22:31Z Publisher: ACL (Association for Computational Linguistics).

Yuxiao Chen, Jianbo Yuan, Quanzeng You, and Jiebo Luo. 2018. Twitter sentiment analysis via bi-sense emoji embedding and attention-based LSTM. In *Proceedings of the 26th ACM international conference on Multimedia*, MM '18, pages 117–125. Association for Computing Machinery. ZSCC: 0000047.

Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bošnjak, and Sebastian Riedel. 2016. emoji2vec: Learning emoji representations from their description. ZSCC: 0000189.

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. ZSCC: NoCitationData[s0] Accepted: 2019-07-01T12:53:05Z Publisher: University of Stavanger, Norway.

C. Hutto and Eric Gilbert. 2014a. VADER: A parsimonious rule-based model for sentiment analysis of social media text. 8(1):216–225. ZSCC: 0002455.

Clayton J. Hutto and Eric Gilbert. 2014b. 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. ZSCC: 0000007.

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. ZSCC: 0000005.

Ruijun Liu, Yuqian Shi, Changjiang Ji, and Ming Jia. 2019. A survey of sentiment analysis based on transfer learning. 7:85401–85412. ZSCC: 0000047 Conference Name: IEEE Access.

Tulio Correa Loures, Gustavo Lucius Fernandes, Fernanda G Araujo, Karen S Martins, and Pedro O S Vaz-de Melo. 2020. StinkyCheese: Chat-based model for subscription classification. page 10. ZSCC: 0000001.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

Petra Kralj Novak, Jasmina Smailović, Borut Sluban, and Igor Mozetič. 2015. Sentiment of emojis. 10(12):e0144296. ZSCC: 0000655.

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. 55:101434. ZSCC: 0000002.

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. ZSCC: 0010633.

Zhengqi Pei, Zhewei Sun, and Yang Xu. 2019. Slang detection and identification. pages 881–889. ZSCC: 0000007.

Igor Samenko, Alexey Tikhonov, and Ivan P. Yamshchikov. 2020. Synonyms and Antonyms: Embedded Conflict. *arXiv:2004.12835 [cs]*. ZSCC: 0000001 arXiv: 2004.12835.

Zhewei Sun, Richard Zemel, and Yang Xu. 2021. A computational framework for slang generation. 9:462–478. ZSCC: 0000000.

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. 11(5):280. ZSCC: 0000002 Number: 5 Publisher: Multidisciplinary Digital Publishing Institute.

Sanjaya Wijeratne, Lakshika Balasuriya, Amit Sheth, and Derek Doran. 2016. EmojiNet: Building a machine readable sense inventory for emoji. ZSCC: 0000053.

Sanjaya Wijeratne, Lakshika Balasuriya, Amit Sheth, and Derek Doran. 2017. EmojiNet: An open service and API for emoji sense discovery. ZSCC: 0000055.

Steven Wilson, Walid Magdy, Barbara McGillivray, Kiran Garimella, and Gareth Tyson. 2020. Urban dictionary embeddings for slang NLP applications. pages 4764–4773. ZSCC: 0000005.

Ashima Yadav and Dinesh Kumar Vishwakarma. 2020. Sentiment analysis using deep learning architectures: a review. 53(6):4335–4385. ZSCC: 0000079.

David Zimbra, Ahmed Abbasi, Daniel Zeng, and Hsinchun Chen. 2018. The state-of-the-art in twitter sentiment analysis: A review and benchmark evaluation. 9(2):5:1–5:29. ZSCC: 0000083.