Text Analysis from Scraping Reddit Comments

Matthew Li Yuen Fong (RID: 500991570)
4-6-2020

1. Introduction

With the dramatic shift and prevalence of social media in modern society, text analytics can be a powerful tool for companies to take strategic action based on valuable insights on common themes and trends found on the internet. To make text analytics the most efficient, organisations can use text analytics software, leveraging machine learning and natural language processing algorithms to find meaning in enormous amounts of text. Reddit is an American social news aggregation, web content rating, and discussion website and as of Jan 2020 according to Alexa Internet, it ranks as the fifth most visited website in the Canada and 18th in the world. For users, one advantage to using reddit is that all topics are aggregated in a sub community (known as a subreddit) where individuals may post and comment on topics of specific interest to them. Therefore, there is a wealth of commentary, opinions, and knowledge that can be mined from the submissions and comments of reddit that can have the potential to provide valuable insight to any company and/or topic of interest.

For the scope of this project, our methodology is as follows; 1) Utilise Reddit's API to scrape subreddit submission titles and comments 2) pre-process (stop words, symbols, apostrophes, lemmatize) and tokenize text 3) upload the data to a MongoDB database 4) Apply TF-IDF, words2vec, and sentiment analysis models on the pre-processed text 5) Deploy streaming version to get live updates of subreddits.

2. Literature Review

2.1 Řehůřek, R. (2019, November 1). Gensim Topic Modelling for Humans Core Concepts. Retrieved from https://radimrehurek.com/gensim/index.html

This webpage documents the free python package known as gensim and explores key concepts about its usage and required understanding of terms such as documents (text), corpus (collection of documents), vector (mathematical representation of corpus, and models (algorithms employed to transform vectors from one representation to another). It also provides a walkthrough and example for users to get started using genism as well as providing API references.

2.2 Jurafsky, D. and Martin, J.H., "Vector Semantics and Embeddings," in Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, 3rd ed. Stanford University, UK: Online, 2019, pp. 94–122.

The chapter "vector semantics and embeddings" explains 5 main topics which include (1) the various methods to represent words, (2) ways to measure similarity, (3) TF-IDF model, (4) Word2vec, and (5) the properties of embeddings. Additionally, the concept of information retrieval is introduced which is used to find documents in a collection that best matches a query from a term-document matrix. For its application, TF-IDF is a useful model to give weight to terms in a given vector, and it is dependent on the term frequency and its frequency of appearance in documents, such that words that are useful for describing documents from rest of collection are given more weight, while those that occur frequently are less. Therefore, TF-IDF is good for weighting co-occurrence matrices for IR.

In addition to representing words in a term-document matrix, word-word matrices enable the identification of n-grams around a target word and by using cosine similarity it is possible to determine if a word is similar to another pair based on its frequency. Furthermore, to identify collocations it is practical to use pointwise mutual information to calculate the probability of 2 words (x and y) occurring compared to what we would expect if they were to appear independently.

Alternatively, rather than asking how often each word occurs near a target word as with TF-IDF. It is possible to ask using the Word2vec model, how likely a word is to appear next to a target word. This model is a binary classification trained on a logistic regression classifier which uses an algorithm known as skip-gram. Skip-gram is a shallow neural network that treats the target word and each neighboring word as a positive example while randomly sampling other words to get negative samples. This method uses a logistic regression model to train classifier to distinguish the 2 cases and uses the regression weights as embeddings.

2.3 Hutto, C.J. & Gilbert, E. (2015). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Proceedings of the 8th International Conference on Weblogs and Social Media, ICWSM 2014.

Highlights the development, evaluation, and validation of VADER (for Valence Aware Dictionary for sEntiment Reasoning). Sentiment analysis was previously heavily reliant on pre-existing manually constructed lexicons where each lexical feature was labelled manually according to their semantic orientation. The article identifies 3 sentiment oriented (polarity-based) lexicons, which include LIWC, General Inquirer (GI), and Hu & Liu. The authors note that the common disadvantage for using them is that they are unable to discern sentiment intensity of words and have an inability to parse sentiment-bearing lexical items such as slang or emoticons from social text. The authors construct and empirically validate a gold standard list of lexical features (along with their associated sentiment intensity measures) which are specifically adapted to sentiment in microblog-like contexts.

2.4 TextMiner (2014, January 17). Text Mining Online: NLP. Retrieved from https://textminingonline.com/

TextMiner provides an in-depth walkthrough using python's natural language processing package known as NLTK. The website contains a series of articles that introduce key concepts in the preprocessing of text such as word tokenization, parts-of-speech-tagging, and stemming and lemmatization all within the python environment.

2.5 MonkeyLearn. (2020). Text Analysis. Retrieved from https://monkeylearn.com/text-analysis/

Text analysis plays an important role in how people and businesses gain valuable insights from an information saturated world. Monkeylearn's article introduces the topic outlining the various methods of text analysis which includes basic techniques such as word frequency, collocation, and concordance. Furthermore, the authors also describe a wide array of advanced methods such as text classification, sentiment analysis, topic analysis, intent detection, text extraction, keyword extraction, entity recognition, word sense disambiguation, text clustering.

2.6 Turney, P.D. (2002). Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, ACL 2002

The author introduces a simple unsupervised learning algorithm primarily for rating reviews as a thumbs up or down (recommended or not recommended). This algorithm takes reviews and extracts sentences that contain adjectives or adverbs, then estimates its semantic orientation through pointwise mutual information – information retrieval and classifies the review based on the average semantic orientation of the phrases.

3. Data set

3.1 PRAW API: Subreddit & Comment Data

The dataset used is an on-demand stream of subreddit submission titles and comments obtained from reddit. Using PRAW: The Python Reddit API Wrapper, we can pull submission titles and all comments of "n" hottest submissions from the front page and store the data in a mongoDB database (schema-less NoSQL document database).

Within the database, we can have multiple collections each representing a submission within the subreddit and all the comments stored as documents specific to that submission. Furthermore, we have an additional collection called "[subreddit name]_overview" which includes information about all submissions we scraped which includes their title, score, id, url, number of comments, and content body. Lastly, we have 3 collections that are similar in that they contain an aggregation of comments from all submissions and includes information such as the message, submission title, and timestamp. However they differ in that one contains text that has not been preprocessed, while another has text that has gone through the preprocessing step (refer to approaches section for details on preprocessing) but is not tokenized, and the third has both been tokenized and preprocessed.

For the scope of this project, we intend to scrape 2 subreddits as an example and take the top 20 hottest (due to computing and time constraints) submissions and its comments.

3.2 Derived Data

3.2.1 Word Frequency: Corpus & Query

This data set has 2 features: a list of unique words scraped from comments and a count of their appearance.

3.2.2 Information Retrieval Query Results

This data set is obtained from querying our TF-IDF transformed corpus, which returns the most relevant documents to our query based on their similarity score. The result is a table that has 4 attributes: Score (Similarity Score), Document Number (Index ID), message (original comment), and processed message (tokenized and preprocessed comment).

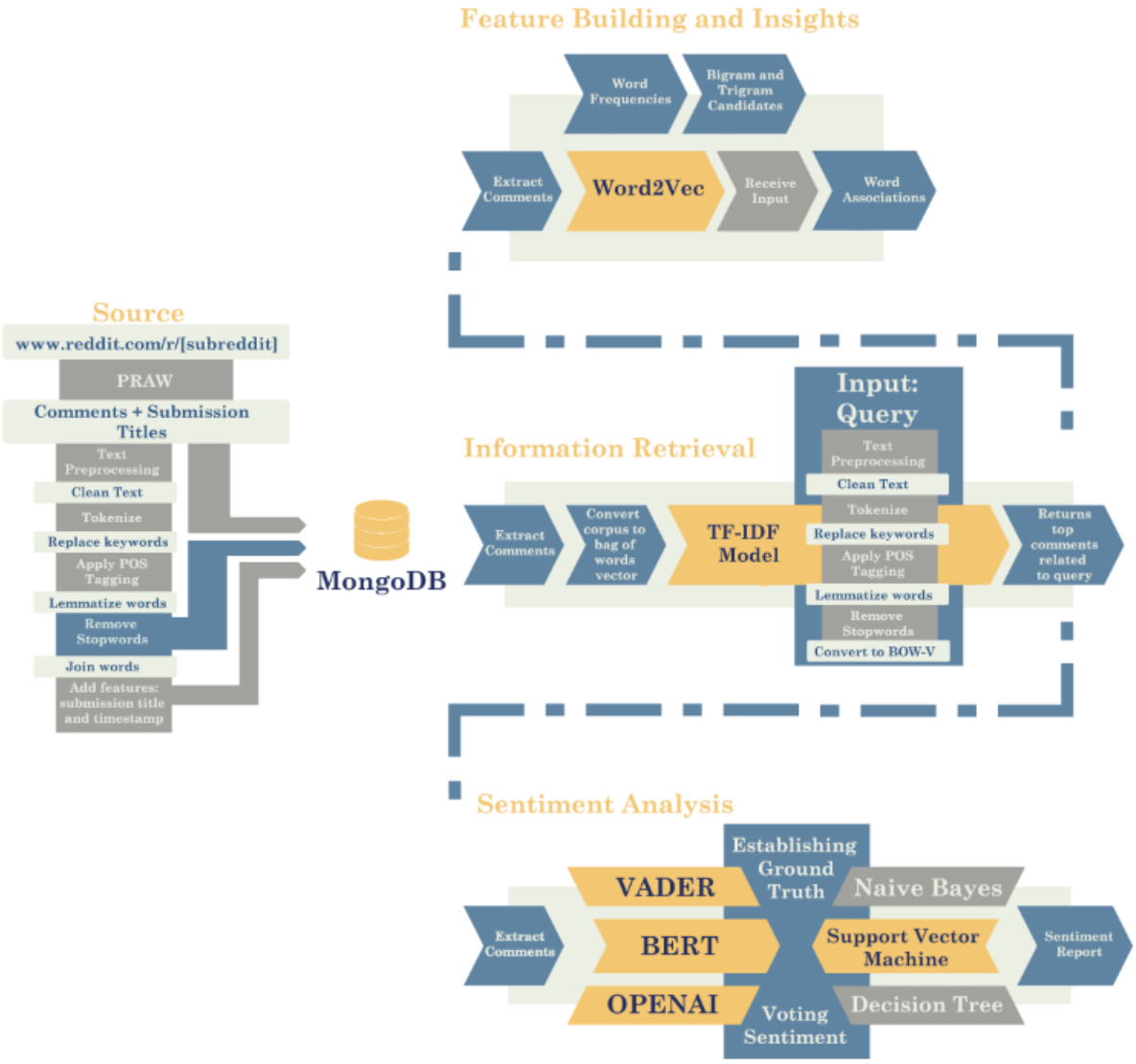
3.2.3 VADER Sentiment Analysis: Corpus & Query

This data set is created from running VADER sentiment analysis on our entire corpus or query. The resulting table has 7 features: sentiment class (positive, neutral, or negative), processed message (tokenize and preprocessed comment), message (original comment), compound score, % positive, % neutral, and % negative.

3.2.4 Bigram and Trigram Scoring and Frequency: Corpus & Query

This data set is created after searching for bigrams and trigrams for the corpus or query. For each potential bigrams or trigrams data set there are 3 tables, which include a frequency (bigram/trigram count), Chi-squared (Bigram/Trigram, Chi-squared), and PMI table (bigram/trigam, PMI score).

4. Approach



4.1 Build Corpus

1. Using PRAW, select subreddit and scrape current N hottest submissions/comments.
2. Store submissions and comments into MongoDB.

4.2 Text Preprocessing

1. Extract and aggregate all comments from MongoDB into one dataset.
2. Remove symbols, spaces, punctation, and digits (except <’>).
3. Lower case all comments.
4. Tokenize words.

5. Find and replace words in our “apostrophe” dictionary which are a set of words that can be converted from their contracted form. Additionally, any shorthand and abbreviations can be corrected here.
6. Apply parts-of-speech (POS) tagging to all words.
7. Lemmatized tokenized text based on their POS tag.
8. Remove stop words, the resulting dataset is now tokenized and preprocessed. Add features, submission title (the submission the comment was extracted from) and timestamp (date/time it was posted). Upload new dataset into MongoDB.
9. Join all words with a space where there used to be a comma, to un-tokenize the comments. Add features, submission title (the submission the comment was extracted from) and timestamp (date/time it was posted). Upload new dataset into MongoDB.

4.3 Build Features (Word Frequency)

1. Using pre-processed and tokenized dataset, we count the frequency of each word and filter words that appear less than 2 times under the assumption, words that appear once are unimportant.
2. Export unique words and word count to a .csv

4.4 Build Script to generate bigrams and trigrams from entire corpus and query subset

1. Load dataset (either full tokenized corpus, or tokenized query subset)
2. Using the NLTK package we use TrigramCollocationFinder and BigramCollocationFinder on our list of words
3. Generate frequency table based on the results of the trigram/bigram finder.
4. Find bigrams-trigrams based on pointwise mutual information to calculate the probability of 2 words (x and y) occurring.
5. Generate PMI table of top bigram/trigram candidates.
6. Find bigrams-trigrams based on Chi squared test to calculate the likelihood that words are independent.
7. Generate Chi squared table of top bigram/trigram candidates.

4.5 Information Retrieval (Word embeddings with TF-IDF)

1. Using genism, convert corpus into bag-of-words vector (BOW-V).
2. Create a TF-IDF model to transform BOW-V into a vector space where the frequency counts are weighted according to the relative rarity of each word in the corpus.
3. To prepare for similarity queries, we index the transformed TF-IDF model.
4. Receive query from user, tokenize, and convert into (BOW-V) then query the similarity of our BOW-V document against every document in the corpus.
5. Returns the top documents containing our query and we generate and export a dataframe with the document ID, similarity score, preprocessed comment, and unprocessed comment. This is now a subset of our data set that is specifically related to our query.

4.6 Build Sentiment Analysis Model from entire corpus/training dataset and query subset

1. Load dataset (either full tokenized corpus, or tokenized query subset)

2. Using vaderSentiment for python, we build a sentiment analyser
3. Run sentiment analyser on each message to determine what the breakdown of each comment is rated as and the overall sentiment of the comment.
4. Based on documentation provided by Hutto, C.J. & Gilbert, Eric, compound scores dictate whether a comment is considered positive, neutral, or negative.
5. Using these thresholds;
    a. positive sentiment: (compound score >= 0.05),
    b. neutral sentiment: (compound score > -0.05) and (compound score < 0.05)
    c. negative sentiment: (compound score <= -0.05).
6. Apply pretrained Bidirectionnal Encoder Representations for Transformers (BERT) on dataset to obtain sentiment (Positive/Negative)
7. Apply OpenAI sentiment neuron (unsupervised machine learning) on dataset and set thresholds to define our own positive, neutral, and negative sentiments.
8. Develop voting ruleset to establish ground truth.
9. Train Naïve Bayes, Support Vector Machine, and Decision tree algorithms to dataset.
10. When run on our query subset, we have identified what the sentiment for each comment related to our query is.

4.7 Word Similarities Using Word2vec.

1. Load our preprocessed and tokenized corpus and using genism, we import their Word2Vec function.
2. We apply Word2Vec on our messages and query our transformed model
3. Returns words that are most likely to return with our target word

5.0 Results

5.1 Dataset

On April 1st, 2020, our script to scrape reddit via PRAW was run and the resulting dataset from the top 20 submissions submitted this week from r/leagueoflegends yielded 13 564 comments (Figure 1). Notably, the submission called “ok” with 957 comments was dropped in subsequent analysis as it only contains comments that say “ok” or a variation of it. Subsequently text was preprocessed under the following conditions previously described in Approaches (4.2 Text Preprocessing) and the transformed text (Figure 2) was exported in .csv format. Of interest in figure 2, we note that some valuable information was stripped from the comment such as the “:')” and “!” which can add emphasis and context to the comment.

5.2 Information Retrieval

Applying the TF-IDF model we test the successfulness of our information retrieval system by attempt to query our corpus with the text “tsm” and export the results in a csv. Of interest, out of the 13.5k comments, our model was able to extract comments related to this term and rank them based on relevance (Figure 4). Our prior understanding of league of legends is the reason why we selected the term tsm to query. TSM is an esports team that compete on a weekly basis in the league of legends championships series and the team and its players have garnered a substantial fan base. Subsequent queries will use the term “tsm” for consistency.

5.3 Word Frequency, Bigrams, and Trigrams

5.3.1 Word Frequency

Following our query utilising the information retrieval system, we are able to calculate the most frequent words not only from the entire subreddit but from our query in an effort to gain insights on any topics that may be mentioned at the time of scraping. In figure 3, we have described the top 20 most frequent words found from the top 20 submissions from r/leagueoflegends and from our query of "tsm". Furthermore, through Tableau, we can generate word clouds to represent the word frequency visually (Figure 6). From our query, we note that unsurprisingly, our query is the most frequent. In contrast, the results from the top 20 submissions, many words do not have any significance but some words like game, champion, play and people are all key aspects of the multiplayer online battle arena (MOBA) league of legends.

5.3.2 Bigrams + Trigrams

When extracting bigrams and trigrams from our text, we employ 4 measures to identify top candidates and these include, word frequency, likeliness ratio, chi-squared test, and pointwise mutual information (Figure 5). We found that using word frequencies to identify provided us with the greatest number of candidates however it was not robust at filtering non-meaningful words. This is specifically evident in the generation of bigrams where we observe word pairs such as "it is", "did not", and "that is" as our most frequent pairs (Figure 7). We provide the identification of 127 183 bigrams and 173 343 trigrams using frequency as the criteria, while when using more robust statistical tests such as Chi-Squared and PMI we are only able to identify 264 bigram and 36 trigram candidates (Figure 8 + Figure 9). Our last association measure we utilise is likelihood ratio which returned 120 459 bigrams and 153 978 trigrams (Figure 10). Notably the way it ranked the bigrams and trigrams, it was much better at identifying more useful bigrams and trigrams. Overall, we observe that the majority of trigrams generated end up being website urls and difficulty we face in identifying useful bigrams suggests that better text processing is required. Furthermore, based on intuition and game knowledge, our results suggest that likelihood ratio is the best measure in quickly identifying candidate bigrams and trigrams.

We also demonstrate its use in identifying bigrams and trigrams within the return comments from our query of "tsm". Like our previous results we see more bigrams and trigrams for those identified by frequency and likelihood ratio measures however using chi-squared and PMI, we were unable to generate prospective bigrams and trigrams (Figure 8,9,10). Therefore, our data indicates that likelihood ratio performs best and can quickly identify candidate bigrams and trigrams.

5.4 Word2Vec

Utilising genism Word2Vec module, we apply this model to our text to assess its ability to produce better word embeddings compared to TF-IDF. Similar to TF-IDF, we test our model by querying "tsm" and based on our own knowledge on r/leagueoflegends we describe what words our model deems most similar. Word2Vec parameters were adjusted based on the resulting output of our query in order to obtain what we estimate is the most similar word

(Figure 11). Ultimately, we settled on using a window between 5-20, minimum frequency count of 5, sg = 1 (skip-gram), negative sampling of 15 and iterations of 3. The closest words to our query are quite similar in meaning as we see other team names appear such as “CLG” and “Liquid”. We also see words associated with the upcoming playoffs such as “championship”, “playoff”, and “final”. Furthermore, we projected the word embeddings via TSNE by reducing the dimensionality to 2 and plotted the words using a scatter plot (Figure 12). In this representation we can see how similar words are closer together. In Figure 13 (Top) the names of game characters that play similarly are grouped together as well as in a separate cluster, words that are associated with keyboard keys or actions are clustered (Figure 13 Bottom). To that end, Word2Vec appears to have performed relatively well when assigning meaning to the words in our corpus.

5.5 Sentiment Analysis – Establishing Ground Truth

5.5.1 Dataset

On March 2nd, 2020 our script was run to obtain 99 265 comments from the top 38 submissions of all time (Figure 14). Subsequently text was preprocessed under the following conditions previously described in Approaches (4.2 Text Preprocessing) and transformed text was exported in .csv format. All subsequent applications of this dataset is using the preprocessed text.

5.5.2 VADER

Running VADER on entire corpus resulted in 40 4032 positively labeled samples, 31 286 neutrally labelled samples, and 26 121 negatively labelled samples (Figure 15 Left). VADER classifies sentiment based on a polarity score (compound score), so we also visualise the distribution of the compound score against the sentiment (Figure 15 Right). We observe 3 distinct peaks with our neutral sentiment at around 0.0 compound score being the highest. While negative and positive sentiments have a flatter but wider peak covering a larger range of scores. This is unsurprising as VADER has predetermine thresholds when classifying sentiment, where a positive sentiment has a compound score >= 0.05, neutral sentiment a compound score between - 0.05 and 0.05, and a negative sentiment has compound score <= -0.05. Investigating deeply, we sample a random set of comments along with their sentiment and based on our intuition it appears VADER is getting a relatively accurate impression of sentiment (Figure 16).

5.5.3 BERT

Using a pre-trained (trained on the IMDB Large Movie Review Dataset) model provided by Google-Research (Devlin *et al.,* 2018). We applied the BERT sentiment algorithm on our dataset which classified comments as either positive or negative, resulting in a total of 52 254 negative samples and 45 585 positive samples (Figure 17 Left). BERT classifies sentiment based off a polarity score which is represented in a 2D array. Like VADER, we sample a random set of comments along with their sentiment and based on our intuition it appears BERT is getting a relatively accurate impression of sentiment (Figure 17 Right).

5.5.4 OpenAI

OpenAI sentiment neuron is a trained a multiplicative LSTM with 4,096 units on a corpus of 82 million Amazon reviews to predict the next character in a chunk of text (Ratford *et al*., 2017). We applied this trained model on our comment data and as this is an unsupervised machine learning algorithm, we are only provided with a polarity score (Figure 18). Notably, we observe a normal distribution, thus we assigned thresholds to define our positive (> 0.1), neutral (between 0.1 and -0.1), and negative classes (<-0.1). The algorithm determines sentiment on a per character basis and the final score is determined after the entire comment has been assessed (Figure 19). Our own thresholds were decided based on comparisons between BERT and VADER sentiment classification (Figure 20 + 21). When analysing BERT sentiments against OpenAI's polarity score distribution, we observe that there are 2 curves that overlap, which may suggest that the area between maybe useful in identifying neutral sentiment. When looking at VADER, it becomes less clear as the 3 distributions have more overlap, but we do see slight shifts left and right for negative and positive sentiments respectively. Based on these thresholds we classify 34 074 positive comments, 34 987 neutral comments, and 28 778 negative comments (Figure 22). Like the previous models described, we sample a random set of comments along with their sentiment and based on our intuition it appears OpenAi is getting a relatively accurate impression of sentiment (Figure 23).

5.5.5 Establishing Ground Truth

By using the results from the 3 algorithms we aim to describe the ground truth based on a set of voting rules (Figure 24). After voting, our final training dataset has 37 172 positive comments, 26 048 neutral comments, and 34 619 negative comments.

5.6 Sentiment Analysis Classification

By establishing the ground truth, we can train our own sentiment classifier. We ran 3 supervised machine learning algorithms on our training data and tested its ability to classify sentiment on our dataset. We first split the data, adjusting our testing dataset to be 33% of our total data and applied scikit learn TFIDF transformation to vectorize our text. We then trained 3 models, Naïve Bayes, Decision Tree, and Support Vector Machine using a 10-fold cross validation strategy and detail the results in Figure 26 + 27. We highlight 3 metrics to assess the performance of our models which includes accuracy, recall, and precision. Initial results reveal that SVM has overall a higher accuracy score at a 71.91% followed by naïve bays at 69.20% and lastly decision tree at 59.49% (refer to Appendix 1 for detailed output). Our output also describes the precision and recall for each of the classes (positive, negative, and neutral) as well as the F1 score. We observe that our model is unable to accurately label neutral sentiment across the board as it has the lowest precision and recall (in turn lowest f1 score)(Figure 27). SVM and Naïve Bayes are the better performing algorithms with both having comparable results with the difference that SVM has a higher recall on classifying positive statements. Interestingly decision tree has the best results (albeit still poor in the scope of the project) when classifying neutral statements.

6.0 Figures and Graphs

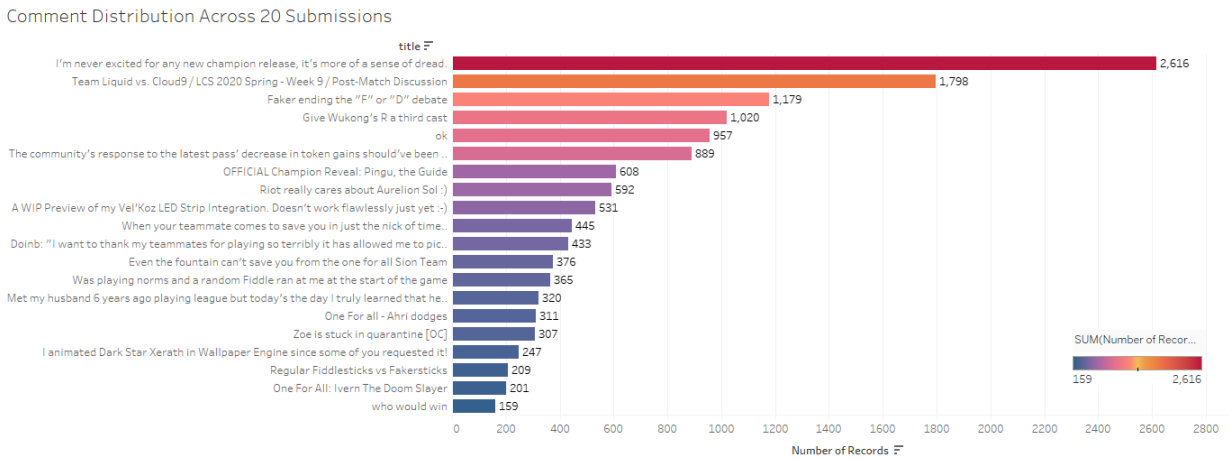


Figure 1: Distribution of comments per submission of the top 20 submissions from r/leagueoflegends

Original Text: Bringing tears to my eyes. Amazing work! Please keep me updated on progress! :')

Processed Text: bring tear eye amaze work please keep update progress

---

Figure 2: Text Preprocessing of an extracted comment

Subreddit	
word	**Count**
not	2498
is	2498
play	2236
get	2212
game	2197
like	2011
ok	1483
make	1318
would	1269
one	1237
champion	1153
people	1135
it	1133
champ	1125
think	1102
time	974
even	947
go	927
do	878
say	861

Query	
word	**Count**
tsm	67
win	32
team	27
tl	25
get	18
playoff	18
lose	15
game	14
split	14
na	14
not	13
play	11
doublelift	11
like	10
look	10
is	9
make	9
final	8
time	8
year	8

Figure 3: Top 20 most frequent words from our entire corpus and query

	doc_no	score	msg	processed_msg
0	12816	0.69746	Didn't even get to TSM	['did', 'not', 'even', 'get', 'tsm']
1	12442	0.667353	The TSM special!	['tsm', 'special']
2	12315	0.63638	The tsm classic	['tsm', 'classic']
3	12581	0.636299	They lost to TSM by getting out aggressioned but that TSM is nowhere to be found so I concur	['lose', 'tsm', 'get', 'aggressioned', 'tsm', 'nowhere', 'find', 'concur']
4	12468	0.572583	We still got TSM to lose to in finals.	['still', 'get', 'tsm', 'lose', 'final']
5	11552	0.535107	I'm disapointed about TSM , especialy coaching staff is the worst off the league BTW But TSM is in play off every time , TL 4th time champions knocked out XD	['am', 'disapointed', 'tsm', 'especialy', 'coach', 'staff', 'worst', 'league', 'btwxbbut', 'tsm', 'play', 'every', 'time', 'tl', 'th', 'time', 'champion', 'knock', 'xd']
6	12742	0.531945	I am aware. I am saying in the last 2 years both C9 and TSM has won against each other in playoffs. It is no longer the like 3 (?) years of TSM always beating them out in playoffs.	['aware', 'say', 'last', 'year', 'tsm', 'win', 'playoff', 'longer', 'like', 'year', 'tsm', 'always', 'beat', 'playoff']
7	12572	0.517227	as a TSM fan myself. Thats hilarious.	['tsm', 'fan', 'thats', 'hilarious']
8	11903	0.464625	Dont worry, they are gonna lose against TSM like always ...	['dont', 'worry', 'gonna', 'lose', 'tsm', 'like', 'always']
9	12741	0.464469	TSM lost to clutch 2018 spring C9 in 2018 Summer TL in 2019 Spring Clutch again in 2019 Summer	['tsm', 'lose', 'clutch', 'springc', 'summertl', 'springclutch', 'summer']
10	12733	0.436482	I miss the days when TSM was on point... I still find it very interesting that after all these seasons, LCS has still been won by only the big 4 teams. At this point, I'm just hoping that "clutch in BO5" TSM somehow shows up	['miss', 'day', 'tsm', 'point', 'still', 'find', 'interest', 'season', 'lcs', 'still', 'win', 'big', 'team', 'point', 'am', 'hop', 'clutch', 'bo', 'tsm', 'somehow', 'show']
11	12734	0.424334	As a TSM fan I'm obviously biased, but the 2nd TSM - C9 game was definitely my favorite of the split. It definitely felt very high level, especially in comparison to a lot of the games from this weekend...	['tsm', 'fan', 'am', 'obviously', 'bias', 'nd', 'tsm', 'game', 'definitely', 'favorite', 'split', 'definitely', 'felt', 'high', 'level', 'especially', 'comparison', 'lot', 'game', 'weekend']
12	11760	0.4211	What a relief! Now the only thing left to do is loose 2-3 to TSM in the finals!	['relief', 'thing', 'leave', 'loose', 'tsm', 'final']
13	12241	0.411014	TSM DONT HAVE RIGHT TO SPAM PERIOD. SIT	['tsm', 'dont', 'right', 'spam', 'period', 'sit']
14	12366	0.410127	Yea that mid game stunk like the TSM loss that I was for sure we were gonna lose.	['yea', 'mid', 'game', 'stunk', 'like', 'tsm', 'loss', 'sure', 'gonna', 'lose']
15	12737	0.396522	How is that even possible when IMT has a worse record than TSM	['even', 'possible', 'imt', 'worse', 'record', 'tsm']
16	12466	0.382891	Don't worry, TSM is still in there ready to beat us on some bullshit :^)	['do', 'not', 'worry', 'tsm', 'still', 'ready', 'beat', 'us', 'bullshit']
17	12738	0.378482	That's Kinda hard when tsm is up a game on 6th place bud.	['that', 'is', 'kinda', 'hard', 'tsm', 'game', 'th', 'place', 'bud']
18	12304	0.378123	TSM just went 0-2. This means they are going to 3-0 the first series and get 0-3'd before reaching the finals.	['tsm', 'go', 'mean', 'go', 'first', 'series', 'get', 'reach', 'final']
19	11909	0.373458	Tl lost with a far far superiorio draft my god it's not even clsoe to what tsm done	['tl', 'lose', 'far', 'far', 'superiorio', 'draft', 'god', 'it', 'is', 'even', 'clsoe', 'tsm']
20	12599	0.369487	C9 and TSM fans enjoy blaming him for stupid shit. Hes a god	['tsm', 'fan', 'enjoy', 'blame', 'stupid', 'shit', 'hes', 'god']

Figure 4: Top 2 returned documents from query "tsm", ranked through the TF-IDF information retrieval system.

Top 20 Submissions

	Bigram	Trigram
Likelihood ratio	120459	153978
Chi-Sq	264	36
PMI	264	36
Frequency	127183	173343

Query

	Bigram	Trigram
Likelihood ratio	1060	1095
Chi-Sq	0	0
PMI	0	0
Frequency	1138	1219

Figure 5: Number of candidate bigrams and trigrams based on 4 different measures

Figure 6: Word clouds generated by tableau using our word frequencies exported from python. Top: word cloud from the top 20 submissions, Bottom: word cloud from our query “tsm”.

Bigram: Freq Full Text	freq
('it', 'is')	1122
('do', 'not')	876
('that', 'is')	491
('you', 'are')	421
('does', 'not')	378
('did', 'not')	332
('he', 'is')	269
('is', 'not')	255
('they', 'are')	192
('there', 'is')	187
('new', 'champ')	185
('feel', 'like')	161
('was', 'not')	159
('new', 'champion')	137
('prestige', 'skin')	128
('play', 'game')	124
('not', 'even')	118
('mint', 'chocolate')	113
('reddit', 'com')	108
('not', 'know')	106

Trigram: Frequency Full Text	freq
('www', 'reddit', 'com')	105
('do', 'not', 'think')	84
('do', 'not', 'know')	80
('reddit', 'com', 'leagueoflegends')	79
('com', 'leagueoflegends', 'comment')	73
('www', 'youtube', 'com')	62
('op', 'gg', 'summoner')	54
('do', 'not', 'get')	50
('do', 'not', 'want')	49
('you', 'are', 'chase')	45
('it', 'is', 'like')	45
('do', 'not', 'even')	44
('think', 'it', 'is')	43
('people', 'do', 'not')	41
('does', 'not', 'matter')	37
('chase', 'greatnessits', 'sayits')	37
('ok', 'ok', 'ok')	36
('are', 'chase', 'greatnessits')	36
('do', 'not', 'like')	34
('say', 'it', 'is')	31

Bigram: Frequency Query	freq
('na', 'team')	7
('did', 'not')	6
('lose', 'tsm')	5
('tsm', 'get')	5
('make', 'playoff')	5
('it', 'is')	5
('tsm', 'fan')	4
('they', 'are')	4
('tsm', 'win')	4
('guy', 'mercy')	3
('put', 'dl')	3
('does', 'not')	3
('team', 'play')	3
('playoff', 'win')	3
('playoff', 'guy')	3
('best', 'team')	3
('tsm', 'kick')	3
('doublelift', 'tsm')	3
('kick', 'doublelift')	3
('zven', 'kick')	3

Trigram: Frequency Query	freq
('doublelift', 'tsm', 'kick')	3
('kick', 'playoff', 'guy')	3
('zven', 'kick', 'doublelift')	3
('kick', 'doublelift', 'tsm')	3
('playoff', 'guy', 'mercy')	3
('tsm', 'kick', 'playoff')	3
('team', 'look', 'tier')	2
('look', 'tier', 'everyone')	2
('tier', 'everyone', 'tsm')	2
('everyone', 'tsm', 'get')	2
('sad', 'corona', 'happen')	2
('top', 'na', 'team')	2
('tsm', 'get', 'stronger')	2
('get', 'stronger', 'stronger')	2
('stronger', 'stronger', 'na')	2
('stronger', 'na', 'team')	2
('na', 'team', 'look')	2
('win', 'world', 'strong')	2
('strong', 'top', 'na')	2
('world', 'strong', 'top')	2

Figure 7: Top 20 returned bigrams and trigrams based on the entire corpus and ranked documents from the query "tsm" using frequency as a measure to identify candidates

Bigram: Chi-Sq full text	chi-sq
('greatnessits', 'sayits')	2E+05
('ice', 'cream')	1E+05
('lee', 'sin')	1E+05
('april', 'fool')	98010
('chase', 'greatnessits')	95384
('you', 'are')	82573
('it', 'is')	82104
('do', 'not')	64632
('weekly', 'mission')	61363
('mint', 'chocolate')	60919
('leagueoflegends', 'comment')	52911
('noot', 'noot')	44187
('www', 'youtube')	43624
('com', 'leagueoflegends')	43504
('gg', 'summoner')	42817
('www', 'reddit')	41835
('laning', 'phase')	37978
('aurelion', 'sol')	36789
('that', 'is')	36166
('dark', 'star')	33052

Trigram: Chi-Sq Full Text	chi-sq
('chase', 'greatnessits', 'sayits')	5E+08
('reddit', 'com', 'leagueoflegends')	4E+07
('com', 'leagueoflegends', 'comment')	3E+07
('op', 'gg', 'summoner')	3E+07
('www', 'youtube', 'com')	3E+07
('www', 'reddit', 'com')	3E+07
('are', 'chase', 'greatnessits')	2E+07
('ok', 'ok', 'ok')	2E+07
('www', 'op', 'gg')	4E+06
('http', 'www', 'reddit')	3E+06
('you', 'are', 'chase')	2E+06
('does', 'not', 'matter')	341605
('do', 'not', 'know')	195026
('you', 'are', 'right')	157569
('do', 'not', 'think')	154401
('do', 'not', 'care')	128892
('do', 'not', 'want')	117561
('do', 'not', 'understand')	113093
('you', 'are', 'play')	101487
('think', 'it', 'is')	96960

Bigram: Chi-Sq query	chi-sq
()	

Trigram: Chi-Sq Query	chi-sq
()	

Figure 8: Top 20 returned bigrams and trigrams based on the entire corpus and ranked documents from the query “tsm” using Chi-Squared as a measure to identify candidates

Bigram: PMI Full Text	PMI
('greatnessits', 'sayits')	12.26
('lee', 'sin')	11.5
('ice', 'cream')	11.42
('chase', 'greatnessits')	11.29
('weekly', 'mission')	11.21
('noot', 'noot')	11.11
('april', 'fool')	10.97
('laning', 'phase')	10.82
('dark', 'star')	10.37
('aurelion', 'sol')	10.36
('zven', 'vulcan')	10.35
('solo', 'queue')	10.17
('gg', 'summoner')	9.633
('pay', 'attention')	9.524
('leagueoflegends', 'comment')	9.503
('www', 'youtube')	9.46
('imgur', 'com')	9.279
('wear', 'mask')	9.255
('com', 'leagueoflegends')	9.107
('mint', 'chocolate')	9.077

Trigram: PMI Full text	PMI
('chase', 'greatnessits', 'sayits')	23.549
('are', 'chase', 'greatnessits')	19.149
('op', 'gg', 'summoner')	19.049
('reddit', 'com', 'leagueoflegends')	18.803
('ok', 'ok', 'ok')	18.789
('com', 'leagueoflegends', 'comment')	18.782
('www', 'youtube', 'com')	18.74
('www', 'reddit', 'com')	17.921
('www', 'op', 'gg')	17.151
('http', 'www', 'reddit')	16.965
('you', 'are', 'chase')	15.688
('does', 'not', 'matter')	13.074
('you', 'are', 'right')	12.01
('get', 'prestige', 'skin')	11.558
('do', 'not', 'care')	11.352
('do', 'not', 'understand')	11.196
('does', 'not', 'mean')	11.041
('do', 'not', 'know')	10.788
('every', 'new', 'champ')	10.508
('do', 'not', 'want')	10.253

Bigram: PMI Query	PMI
()	

Trigram: PMI Query	PMI
()	

Figure 9: Top 20 returned bigrams and trigrams based on the entire corpus and ranked documents from the query "tsm" using PMI as a measure to identify candidates

Bigram: Liklihood ratio FullText	Trigram: Liklihood ratio FullText	Bigram: Liklihood ratio Query	Trigram: Liklihood ratio Query
('mint', 'chocolate')	('www', 'reddit', 'com')	('honda', 'commercial')	('honda', 'commercial', 'doublelift')
('prestige', 'skin')	('www', 'youtube', 'com')	('team', 'liquid')	('honda', 'commercial', 'sponsor')
('www', 'reddit')	('www', 'instagram', 'com')	('look', 'great')	('lol', 'honda', 'commercial')
('reddit', 'com')	('www', 'douyu', 'com')	('blow', 'face')	('face', 'honda', 'commercial')
('new', 'champ')	('www', 'leagueofgraphs', 'com')	('commercial', 'doublelift')	('team', 'liquid', 'aint')
('com', 'leagueoflegends')	('www', 'marketingeye', 'com')	('active', 'cd')	('team', 'liquid', 'kind')
('leagueoflegends', 'comment')	('www', 'amazon', 'com')	('actually', 'antihype')	('turn', 'team', 'liquid')
('ice', 'cream')	('www', 'buymeacoffee', 'com')	('always', 'appear')	('blow', 'face', 'every')
('feel', 'like')	('www', 'imdb', 'com')	('anivia', 'orianna')	('homer', 'look', 'great')
('www', 'youtube')	('www', 'learningmind', 'com')	('another', 'slow')	('look', 'great', 'dragon')
('youtube', 'com')	('www', 'linkedin', 'com')	('antihype', 'gotta')	('defend', 'team', 'liquid')
('april', 'fool')	('www', 'popularmechanics', 'com')	('atrocious', 'part')	('change', 'team', 'liquid')
('new', 'champion')	('www', 'quora', 'com')	('aurelio', 'mobility')	('team', 'liquid', 'team')
('greatnessits', 'sayits')	('www', 'uhdpaper', 'com')	('base', 'model')	('blow', 'face', 'anyone')
('gg', 'summoner')	('reddit', 'com', 'leagueoflegends')	('bbtrox', 'fix')	('extra', 'blow', 'face')
('op', 'gg')	('com', 'leagueoflegends', 'comment')	('begin', 'meme')	('look', 'great', 'since')
('lee', 'sin')	('http', 'www', 'reddit')	('bring', 'occasion')	('poppy', 'look', 'great')
('every', 'single')	('chocolate', 'mint', 'chocolate')	('business', 'truly')	('skin', 'look', 'great')
('chase', 'greatnessits')	('mint', 'chocolate', 'mint')	('call', 'bbtrox')	('second', 'blow', 'face')
('buy', 'pas')	('mint', 'chocolate', 'ice')	('cant', 'dodge')	('actually', 'antihype', 'gotta')

Figure 10: Top 20 returned bigrams and trigrams based on the entire corpus and ranked documents from the query “tsm” using likelihood ratio as a measure to identify candidates

```
In [177]: model= Word2Vec(data_token['message'], window=10, min_count = 5, workers=10, iter=3, sg=1, negative = 15)
     ...: #model= Word2Vec(data_token['message'], window=5, size = 10, min_count = 5, workers=10, iter=3, negative = 15,
alpha = 0.75)
     ...:
     ...:
     ...: w1 = "tsm"
     ...: pprint.pprint(model.wv.most_similar (positive = w1))
[('final', 0.9853488802909851),
 ('summer', 0.9791070818901062),
 ('liquid', 0.9678799510002136),
 ('playoff', 0.9570814371109009),
 ('clg', 0.9527089595794678),
 ('lcs', 0.9513887166976929),
 ('dl', 0.9470458030700684),
 ('tactical', 0.9462488293647766),
 ('hope', 0.9413915872573853),
 ('spring', 0.938597559928894)]


In [176]: model= Word2Vec(data_token['message'], window=5, min_count = 5, workers=10, iter=3, sg=1, negative = 15)
     ...: #model= Word2Vec(data_token['message'], window=5, size = 10, min_count = 5, workers=10, iter=3, negative = 15,
alpha = 0.75)
     ...:
     ...:
     ...: w1 = "tsm"
     ...: pprint.pprint(model.wv.most_similar (positive = w1))
[('final', 0.989540696144104),
 ('liquid', 0.9859861731529236),
 ('clg', 0.9832355976104736),
 ('summer', 0.9829040765762329),
 ('playoff', 0.9824739694595337),
 ('lead', 0.9805071353912354),
 ('korea', 0.980046272277832),
 ('lcs', 0.9791181087493896),
 ('backdoor', 0.9778221845626831),
 ('hope', 0.9775675535202026)]


In [178]: model= Word2Vec(data_token['message'], window=20, min_count = 5, workers=10, iter=3, sg=1, negative = 15)
     ...: #model= Word2Vec(data_token['message'], window=5, size = 10, min_count = 5, workers=10, iter=3, negative =
alpha = 0.75)
     ...:
     ...:
     ...: w1 = "tsm"
     ...: pprint.pprint(model.wv.most_similar (positive = w1))
[('final', 0.9842275977134705),
 ('summer', 0.9788450002670288),
 ('clg', 0.9685006141662598),
 ('playoff', 0.9664933681488037),
 ('spring', 0.9561878442764282),
 ('split', 0.9484572410583496),
 ('championship', 0.9483106136322021),
 ('liquid', 0.9446500539779663),
 ('tactical', 0.936021089553833),
 ('lcs', 0.934034526348114)]
```

Figure 11: Word2Vec parameters and results from searching for the word “tsm” using different window sizes.

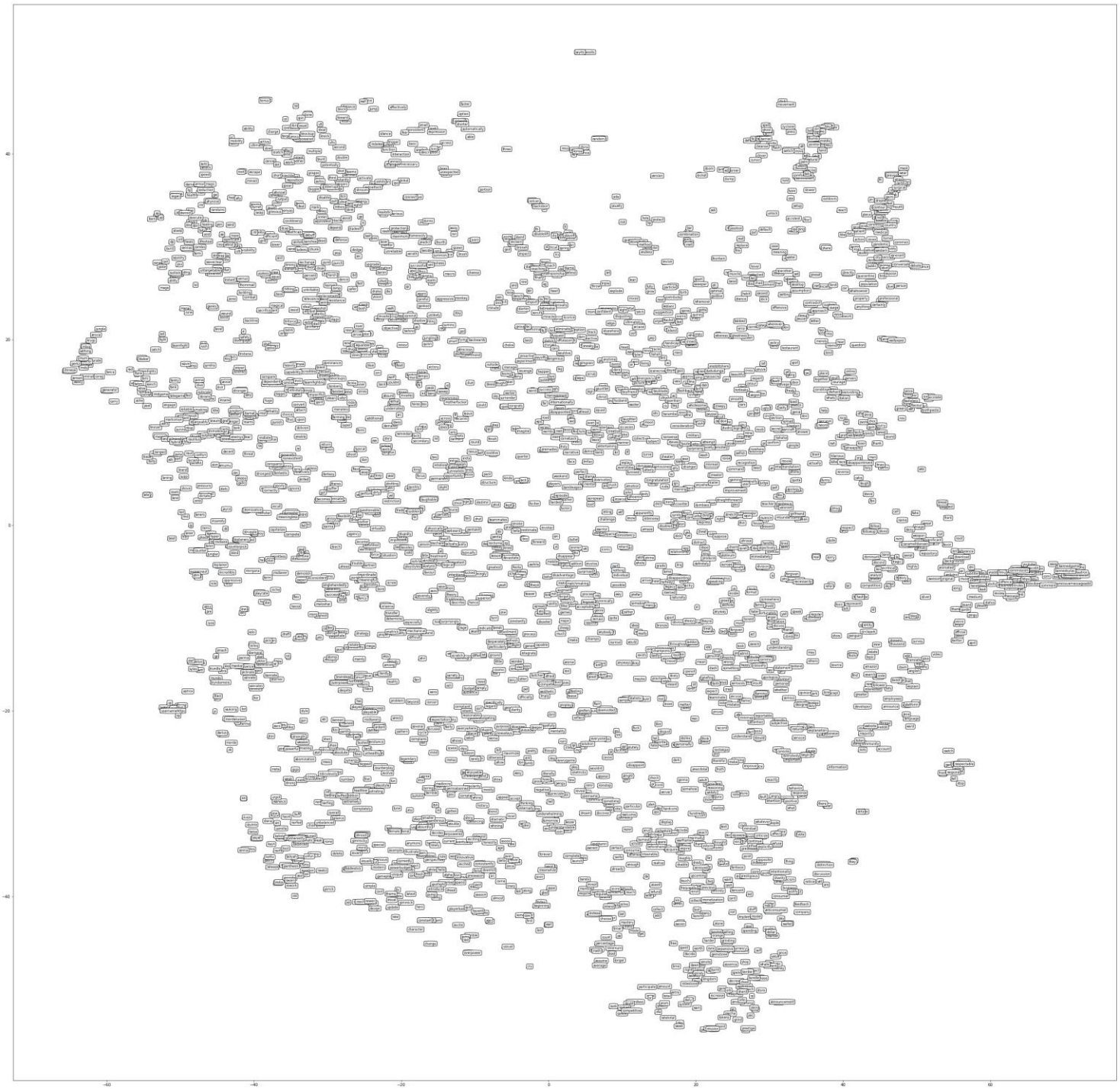


Figure 12: TSNE representation of Word2Vec Model (Window = 10, min_count = 5, iter =3, negative = 15)

Figure 13: Zoomed in areas of TSNE projection

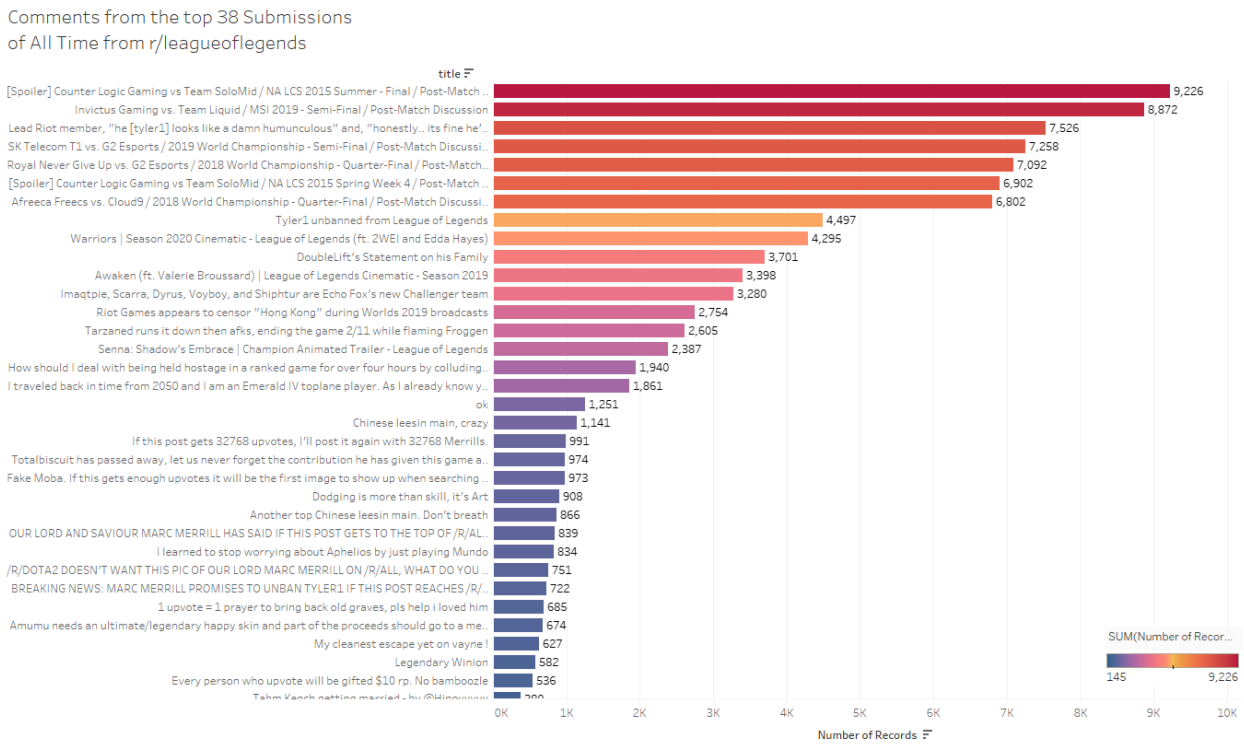


Figure 14: Distribution of comments per submission of the top 38 submissions of all time from r/leagueoflegends

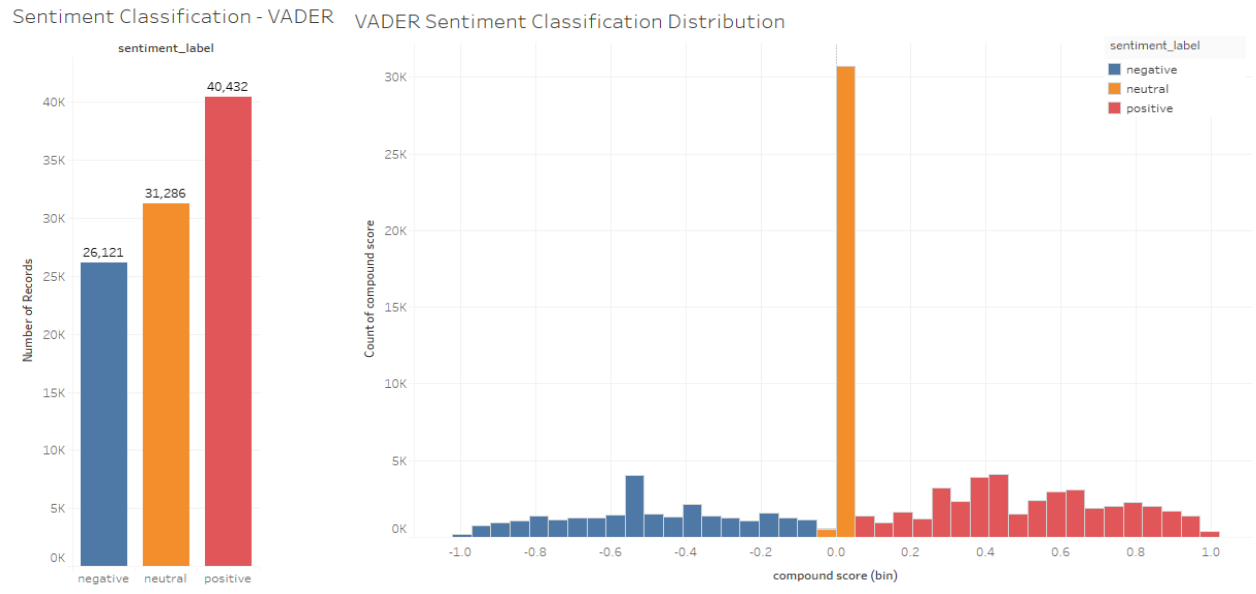
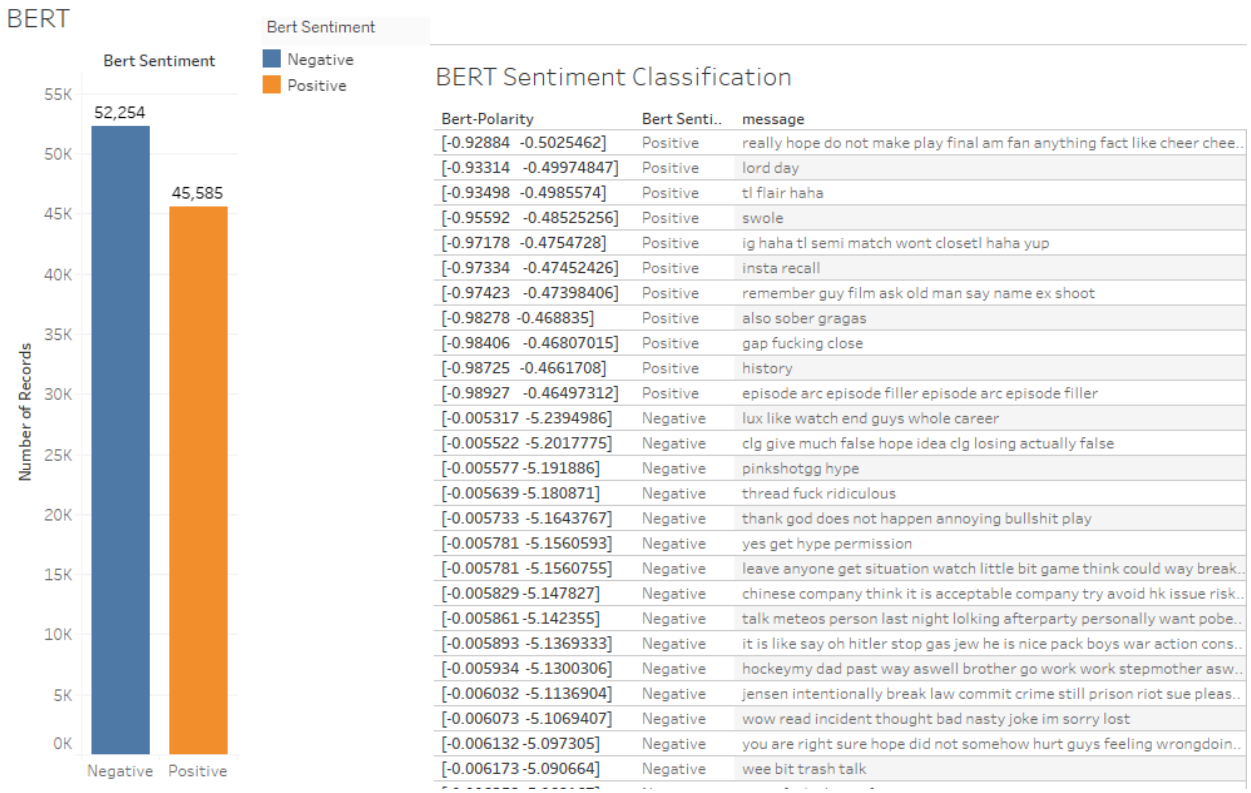


Figure 15: (Left) Distribution of sentiment after classification by VADER. (Right) Distribution of comments and sentiment based on compound scores .

VADER Sentiment Classification

message	compound score ..	sentiment_..	Unname..
it is good everyone others unexperienced player get play tea..	0.9042	positive	37,209
maybe invest team win doestn matter troll around protest	0.44040000000..	positive	37,208
like call tsm fan deffend dyrus lmao respect player does not ..	0.96250000000..	positive	37,207
physical endurance involve actually study kinesiology you ar..	0.48980000000..	positive	37,206
play competitively year make better competitive	0.71840000000..	positive	37,205
eh normal lcs day number have not increase dramatically sin..	-0.16850000000..	negative	37,204
point really get visibility	0	neutral	37,203
promotion tournament na	0	neutral	37,202
yeah riot say million dollar rick fox great ambassador league	0.40189999999..	positive	37,201
cant get relegate split franchise start	0	neutral	37,200
np np thanks time will lookout	0.77170000000..	positive	37,199
oh see have not really watch since play dig would not know	0.3947	positive	37,198
meh act like do not get investor fund ef	0.29599999999..	positive	37,197
have around amateur team year mainly oq one know fact aro..	0	neutral	37,196
move nac play online	0.34000000000..	positive	37,195
did not know it is pretty convenient org player	0.49390000000..	positive	37,194
read somewhere comment that is pretty shitty source ngl	-0.1027	negative	37,193
think want new setting bit	0.07720000000..	positive	37,191
coincidentally tune stream sound like pretty much want trav..	0.80740000000..	positive	37,190
dont reply	0	neutral	37,189
mechanical skill max take pro infact many pro player bad mec..	-0.25	negative	37,188
doesnt make sense well aware becxause only	-0.20569999999..	negative	37,187
anf probably arent good echo fox roster probably well doesn..	-0.07879999999..	negative	37,186
key difference tl offer challenger exception playoff online lea..	0.128	positive	37,185
also did not want move cali does not fo also player big enoug..	-0.20760000000..	negative	37,184
tl also challenger team is not fill people he is friend low stres..	0.65969999999..	positive	37,183
last split na prob want fun	0.55740000000..	positive	37,182
there is way has got nay week lol	0.34000000000..	positive	37,181

Figure 16: VADER sentiment classification of random set of comments pulled from the training data set



BERT Sentiment Classification

Bert-Polarity	Bert Senti..	message
[-0.92884 -0.5025462]	Positive	really hope do not make play final am fan anything fact like cheer chee..
[-0.93314 -0.49974847]	Positive	lord day
[-0.93498 -0.4985574]	Positive	tl flair haha
[-0.95592 -0.48525256]	Positive	swole
[-0.97178 -0.4754728]	Positive	ig haha tl semi match wont closetl haha yup
[-0.97334 -0.47452426]	Positive	insta recall
[-0.97423 -0.47398406]	Positive	remember guy film ask old man say name ex shoot
[-0.98278 -0.468835]	Positive	also sober gragas
[-0.98406 -0.46807015]	Positive	gap fucking close
[-0.98725 -0.4661708]	Positive	history
[-0.98927 -0.46497312]	Positive	episode arc episode filler episode arc episode filler
[-0.005317 -5.2394986]	Negative	lux like watch end guys whole career
[-0.005522 -5.2017775]	Negative	clg give much false hope idea clg losing actually false
[-0.005577 -5.191886]	Negative	pinkshotgg hype
[-0.005639 -5.180871]	Negative	thread fuck ridiculous
[-0.005733 -5.1643767]	Negative	thank god does not happen annoying bullshit play
[-0.005781 -5.1560593]	Negative	yes get hype permission
[-0.005781 -5.1560755]	Negative	leave anyone get situation watch little bit game think could way break..
[-0.005829 -5.147827]	Negative	chinese company think it is acceptable company try avoid hk issue risk..
[-0.005861 -5.142355]	Negative	talk meteos person last night lolking afterparty personally want pobe..
[-0.005893 -5.1369333]	Negative	it is like say oh hitler stop gas jew he is nice pack boys war action cons..
[-0.005934 -5.1300306]	Negative	hockeymy dad past way aswell brother go work work stepmother asw..
[-0.006032 -5.1136904]	Negative	jensen intentionally break law commit crime still prison riot sue pleas..
[-0.006073 -5.1069407]	Negative	wow read incident thought bad nasty joke im sorry lost
[-0.006132 -5.097305]	Negative	you are right sure hope did not somehow hurt guys feeling wrongdoin..
[-0.006173 -5.090664]	Negative	wee bit trash talk

Figure 17: (Left) Distribution of sentiment after classification by BERT. (Right) BERT sentiment classification of random set of comments pull from the training data set.

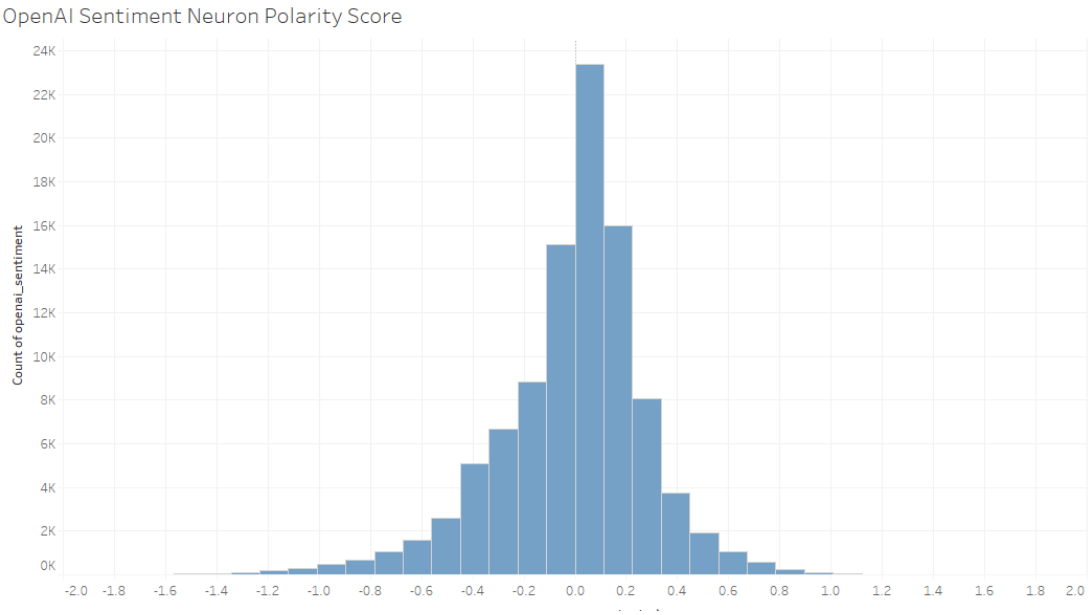


Figure 18: Polarity score distribution of OpenAI sentiment neuron on the training data set.

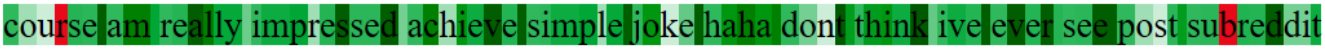


Figure 19: Heat map visualisation of OpenAI sentiment neuron assessing sentiment on a per character basis and changing sentiment as it progresses.

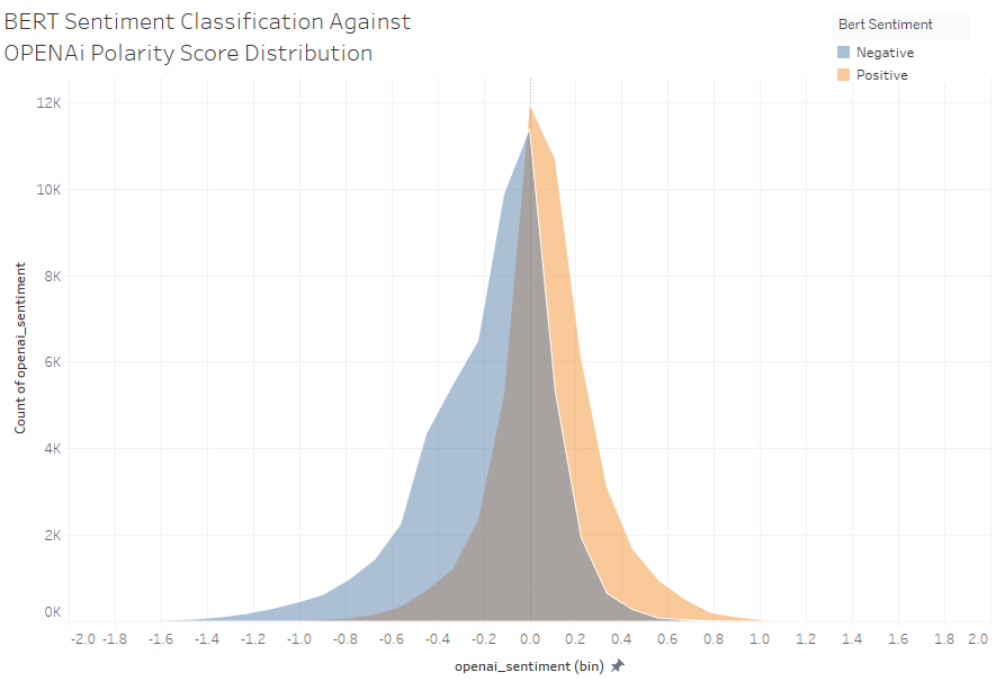


Figure 20: Graph illustrating the sentiment breakdown by BERT in comparison to OpenAI polarity score.

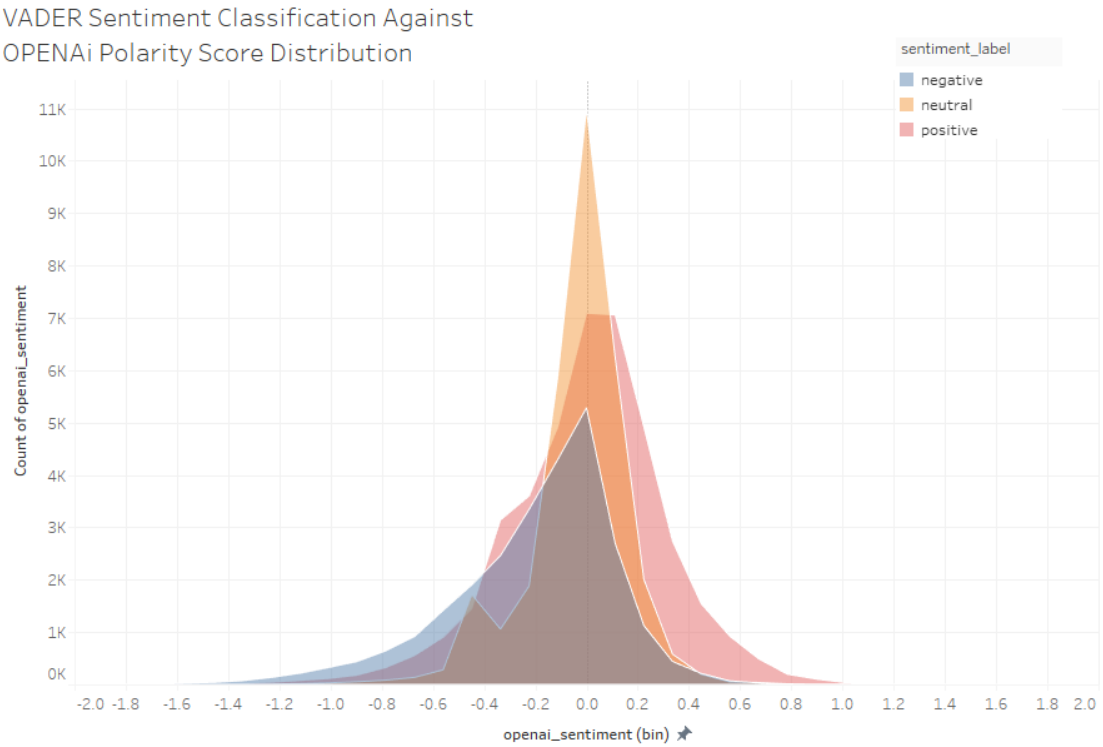


Figure 21: Graph illustrating the sentiment breakdown by VADER in comparison to OpenAI polarity score.

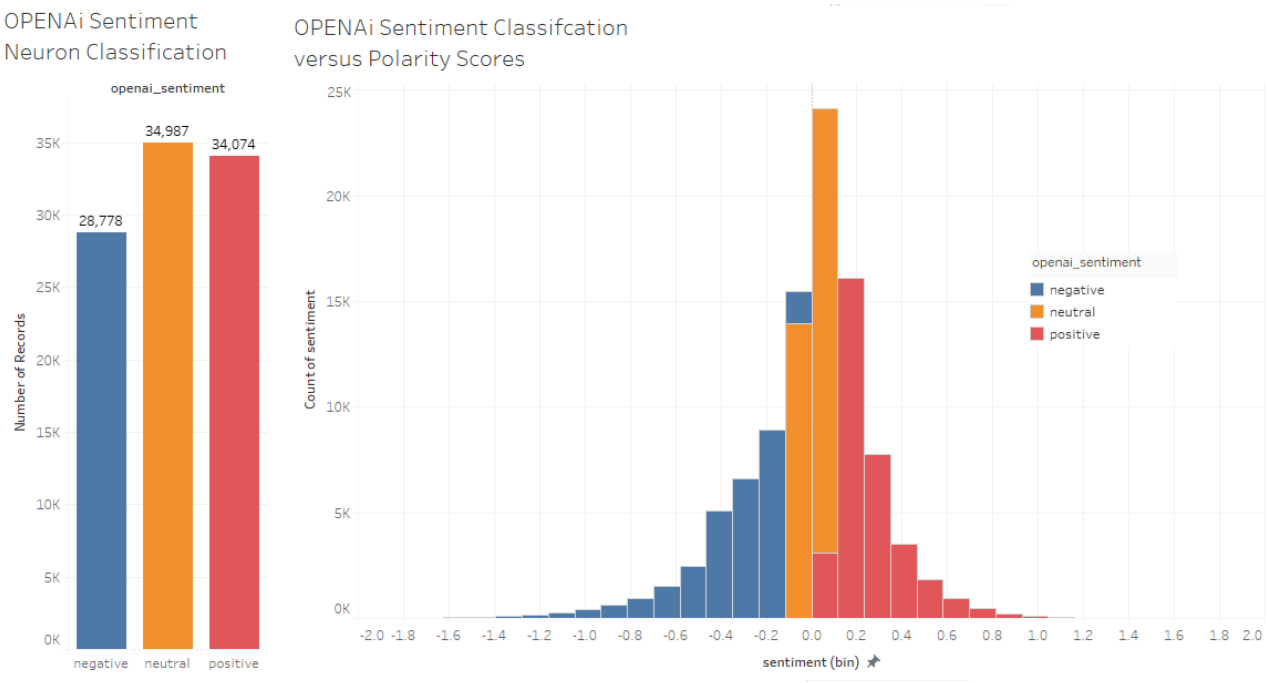


Figure 22: (Left) OpenAi sentiment neuron classification after establishing thresholds. (Right) Distribution of sentiment against OpenAi polarity scores.

OPENAi Sentiment Neuron Classification

F1 (copy)	openai_sen..	message
66924	neutral	winof course
66925	positive	thats interest ive see use correctly case
66926	positive	give birth unicorn win lulw
66927	positive	he is even downvoted tsm sub lol he is idiot
66928	positive	nice international trophy
66929	positive	changhong xiaopeng twila gala mark best player worl..
66930	negative	swear fuck god please give us rift rival deserve year d..
66931	positive	rift rival happy game
66932	positive	you are say na win
66933	neutral	yeah would not wanna challenge fun region fun tourn..
66934	neutral	want msi guy rift rival
66935	positive	take energy
66936	positive	spirit bomb power everyones energy na baby
66937	neutral	reverse gap
66938	positive	amazing timeline would msi final two western team b..
66939	neutral	im fine wirh sejuani let boisedidcalled
66940	negative	whatever take
66941	positive	am ask please tl quickly
66942	positive	yes
66943	negative	sign faker right problem solve
66944	negative	guy give faker backstage pas
66945	negative	apologise family us advance
66946	neutral	narrative perkz prove adc agaist dl
66947	positive	east west significant semis final west would cool
66948	positive	nah bro he is feel eu
66949	neutral	lpl best upset happen like might tomorrow skt unlikel..

Figure 23: Selected samples after classification by OpenAi Sentiment Neuron.

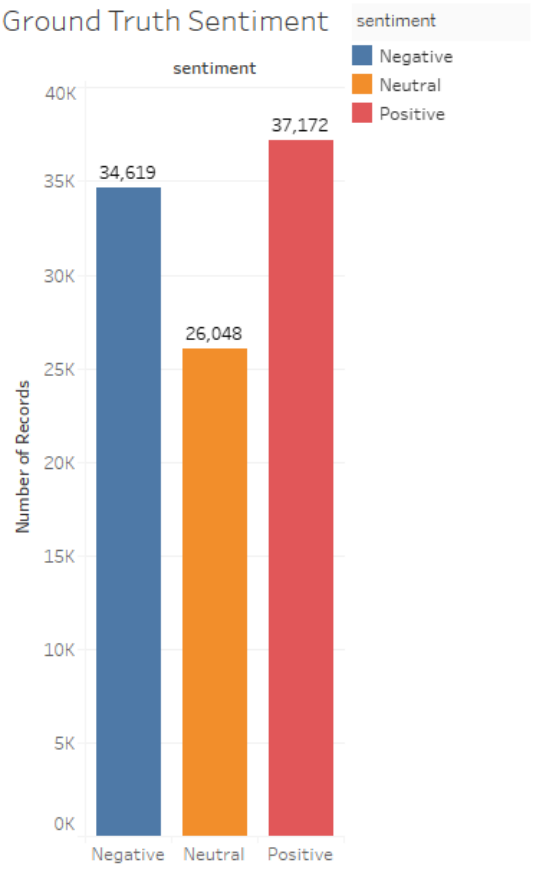
Establishing Ground Truth

Vader	Positive					
Bert	Positive			Negative		
OpenAi	Positive	Negative	Neutral	Positive	Negative	Neutral
Final outcome	Positive	Positive	Positive	Positive	Negative	Neutral

Vader	Neutral					
Bert	Positive			Negative		
OpenAi	Positive	Negative	Neutral	Positive	Negative	Neutral
Final outcome	Positive	Neutral	Neutral	Neutral	Negative	Neutral

Vader	Negative					
Bert	Positive			Negative		
OpenAi	Positive	Negative	Neutral	Positive	Negative	Neutral
Final outcome	Positive	Negative	Neutral	Negative	Negative	Negative

Figure 24: Voting rules for establish ground truth.



Ground Truth Voting Sentiment

doc_no (co..	message	sentiment
47916	remove	Neutral
47917	everytime kill rng player since game legitimately c..	Neutral
47918	exactly hilarious	Positive
47919	have not clap ward since season world	Neutral
47920	hp jhin	Positive
47921	still clue uzi die last fight blink go jump	Negative
47922	alright guy know dowadid wardid guy edit spell	Negative
47923	that is exactly pickem let go brother	Neutral
47924	get except afreeca beating ceu final	Positive
47925	exact mean something right	Neutral
47926	sion noc create chaos really hard baron pull help c..	Positive
47927	coconut	Neutral
47928	point situation different later game gold lead was ..	Negative
47929	reason did not easily close game think ahead enou..	Negative
47930	think watch	Neutral
47931	nah wouldnt lose fight could get interesting sure r..	Negative
47932	people already forget awful game early season ha..	Negative
47933	idk enjoy fight smart late game play take baron ru..	Positive
47934	yes	Positive
47935	thank mr roccat	Positive
47936	he is ok suppose	Positive
47937	eh rng isnt best chinese team tournament ig ig bet..	Positive
47938	seriously calm spam dude lol	Positive
47939	careful say fellow eu breathen might jinx	Neutral
47940	pretty sure get memed rest world	Positive
47941	amaze interview sjokz barely able breathe want g..	Negative
47942	actually send series end try smug	Negative

Figure 25: (Left) Distribution of sentiments after voting. (Right) Randomly sampled comments after establishing ground truth.

Naïve Bayes - precision

	Negative	Neutral	Positive	Average
1	0.71	0.33	0.69	0.58
2	0.7	0.67	0.7	0.69
3	0.7	0.75	0.69	0.71
4	0.69	0.67	0.69	0.68
5	0.69	0	0.67	0.45
6	0.7	0.67	0.68	0.68
7	0.7	0	0.69	0.46
8	0.71	1	0.68	0.8
9	0.7	0	0.7	0.47
10	0.68	0.5	0.69	0.63

SVM - precision

	Negative	Neutral	Positive	Average
1	0.76	0.6	0.7	0.69
2	0.76	0.63	0.71	0.70
3	0.74	0.5	0.7	0.65
4	0.75	0.62	0.7	0.69
5	0.75	0.5	0.68	0.64
6	0.74	0.5	0.69	0.64
7	0.75	0.77	0.7	0.74
8	0.76	0.83	0.69	0.76
9	0.75	0.6	0.69	0.68
10	0.73	0.45	0.69	0.62

Decision Tree - Precision

	Negative	Neutral	Positive	Average
1	0.64	0.27	0.61	0.51
2	0.63	0.31	0.62	0.52
3	0.64	0.28	0.62	0.51
4	0.62	0.29	0.61	0.51
5	0.63	0.26	0.59	0.49
6	0.63	0.28	0.6	0.50
7	0.63	0.26	0.6	0.50
8	0.64	0.32	0.6	0.52
9	0.65	0.26	0.62	0.51
10	0.62	0.27	0.63	0.51

Naïve Bayes - recall

	Negative	Neutral	Positive	Average
1	0.76	0	0.78	0.51
2	0.77	0.01	0.77	0.52
3	0.76	0.01	0.78	0.51
4	0.77	0.02	0.75	0.51
5	0.75	0	0.77	0.51
6	0.76	0.01	0.77	0.51
7	0.76	0	0.78	0.51
8	0.77	0.01	0.77	0.52
9	0.77	0	0.76	0.51
10	0.75	0.01	0.78	0.51

SVM - recall

	Negative	Neutral	Positive	Average
1	0.75	0.03	0.85	0.54
2	0.77	0.04	0.83	0.55
3	0.76	0.02	0.83	0.54
4	0.78	0.03	0.8	0.54
5	0.75	0.03	0.82	0.53
6	0.77	0.03	0.81	0.54
7	0.76	0.06	0.83	0.55
8	0.78	0.05	0.82	0.55
9	0.75	0.02	0.83	0.53
10	0.75	0.01	0.83	0.53

Decision Tree - recall

	Negative	Neutral	Positive	Average
1	0.61	0.19	0.67	0.49
2	0.62	0.23	0.66	0.50
3	0.64	0.2	0.65	0.50
4	0.65	0.21	0.62	0.49
5	0.63	0.17	0.64	0.48
6	0.6	0.2	0.66	0.49
7	0.61	0.19	0.66	0.49
8	0.64	0.21	0.64	0.50
9	0.65	0.18	0.65	0.49
10	0.64	0.18	0.66	0.49

Naïve Bayes - f1-score

	Negative	Neutral	Positive	Average
1	0.73	0.01	0.73	0.49
2	0.74	0.01	0.74	0.5
3	0.72	0.02	0.73	0.49
4	0.73	0.04	0.72	0.49
5	0.72	0	0.72	0.48
6	0.73	0.01	0.72	0.49
7	0.73	0	0.73	0.49
8	0.74	0.01	0.72	0.49
9	0.73	0	0.73	0.49
10	0.72	0.02	0.73	0.49

SVM - f1-score

	Negative	Neutral	Positive	Average
1	0.76	0.05	0.76	0.52
2	0.77	0.07	0.77	0.54
3	0.75	0.04	0.76	0.52
4	0.76	0.06	0.75	0.52
5	0.75	0.06	0.75	0.52
6	0.75	0.05	0.75	0.52
7	0.75	0.11	0.76	0.54
8	0.77	0.09	0.75	0.54
9	0.75	0.04	0.75	0.51
10	0.74	0.03	0.76	0.51

Decision Tree - f1-score

	Negative	Neutral	Positive	Average
1	0.62	0.22	0.64	0.49
2	0.62	0.26	0.64	0.51
3	0.64	0.23	0.64	0.50
4	0.64	0.24	0.62	0.50
5	0.63	0.21	0.62	0.49
6	0.62	0.23	0.63	0.49
7	0.62	0.22	0.63	0.49
8	0.64	0.25	0.62	0.50
9	0.65	0.21	0.63	0.50
10	0.63	0.21	0.64	0.49

Figure 26: Cross validation results of Naïve Bayes, SVM, and Decision trees using 10 folds. Precision, recall, and F1 Score reported for all positive, neutral, and negative classes

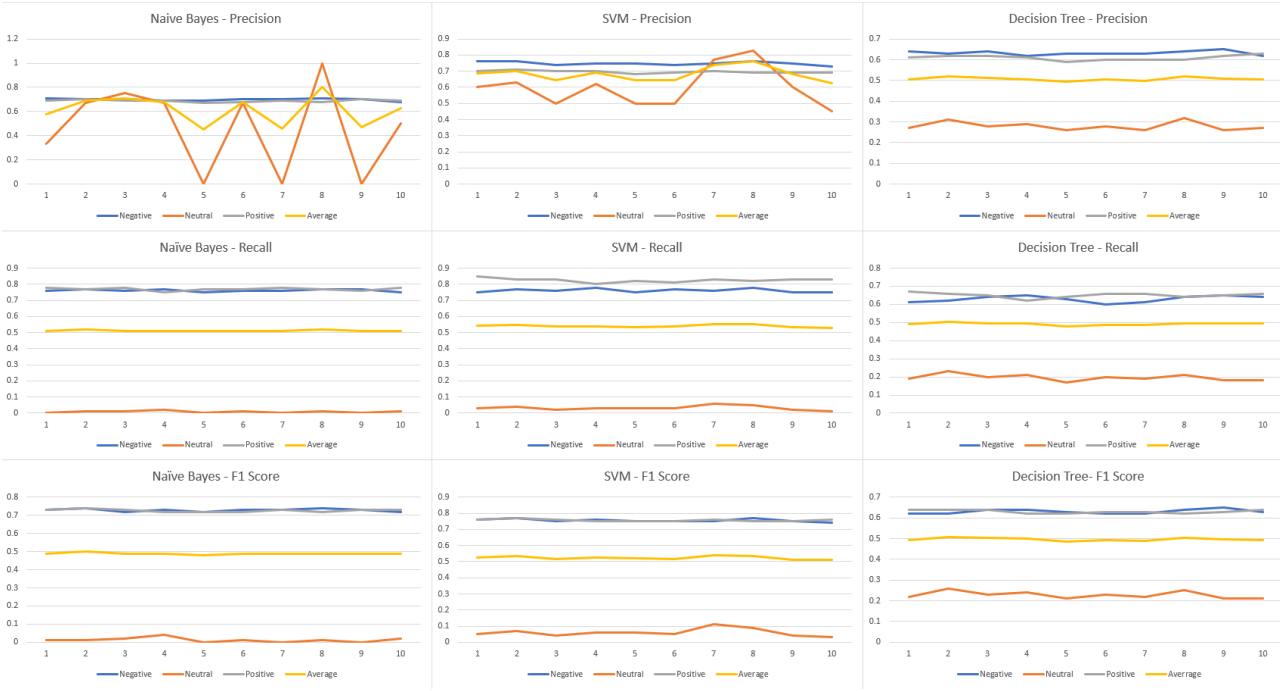


Figure 27: Graphs of machine learning models performance after 10 folds

7.0 Conclusions

We demonstrate a proof-of-concept script that enables us to scrape subreddits on demand for their comments and glean insights based on our query using an information retrieval system. We also test 2 methods for generating word embeddings, which are TF-IDF and Word2Vec and based on our intuition and knowledge of the subreddit, they are fair at determining word representations. We also illustrate that it is possible to identify the sentiment of comments using 3 different algorithms as well as generate our own model which was revealed to be moderately successful.

There are multiple steps within our approach that can be changed to improve our sentiment analysis and information retrieval. Namely, one significant limitation is that BERT and OpenAI were trained on IMDB and Amazon reviews respectively, as opposed to social media comments such as facebook or twitter posts. Thus, the preprocessing of social media text may have significant impact on our results as we strip all symbols and lower case the text which removes text cues such as emoticons (ie 😊, ☹, 😐) and EMPHASIS. Furthermore, our text filters and replacements are not robust enough to identify slang and sarcasm which all can influence how we view the sentiment of a comment. One option that is to pass the entire corpus through without any pre-processing and observe if our sentiment models have better performance based on a more accurate ground truth. Another issue is validating our ground truth, currently we use a voting system that is rule based and the output is given as is. It is unfeasible to analyse all comments to estimate whether the algorithms have labelled each sentiment correctly, therefore we must trust that our assumptions are correct when determining the ground truth.

There are several questions that remain to be answered which include: (1) how do we evaluate the effectiveness of information retrieval? (2) Can we fine tune the parameters for Word2Vec, and how do we test it? (3) How does increasing our training data affect the overall performance of our machine learning algorithms? (4) Can we train BERT on our own dataset rather than use the pre-trained model? (5) Is VADER, BERT, or OpenAI sufficient to establish ground truth? (6) Can we fine tune the parameters of SVM, Naïve Bayes, and Decision tree to improve our accuracy? (7) Have we considered the correct statistics when evaluating our pipeline?

5. References

1. Řehůřek, R. (2019, November 1). Gensim Topic Modelling for Humans Core Concepts. Retrieved from https://radimrehurek.com/gensim/index.html

2. Jurafsky, D. and Martin, J.H., "Vector Semantics and Embeddings," in Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, 3rd ed. Stanford University, UK: Online, 2019, pp. 94–122.

3. Hutto, C.J. & Gilbert, E. (2015). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Proceedings of the 8th International Conference on Weblogs and Social Media, ICWSM 2014.

4. TextMiner (2014, January 17). Text Mining Online: NLP. Retrieved from https://textminingonline.com/

5. MonkeyLearn. (2020). Text Analysis. Retrieved from https://monkeylearn.com/text-analysis/

6. Turney, P.D. (2002). Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, ACL 2002

7. Radford, A., Jozefowicz, R., & Sutskever, I. (2017). Learning to Generate Reviews and Discovering Sentiment. Retrieved 6 April 2020, from https://arxiv.org/abs/1704.01444

8. Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Retrieved 6 April 2020, from https://arxiv.org/abs/1810.04805