# GIF Prediction from Text Messages

**Javan Rajpopat** jkrajpop@ncsu.edu
**Prakshat Shah** pmshah@ncsu.edu
**Shreya More** ssmore@ncsu.edu
**Vasvi Desai** vcdesai@ncsu.edu

## 1 Introduction and Background

GIF prediction exists on various applications but for that, the user needs to explicitly search for the GIF based on certain keywords. The resulting GIFs are also a little too exasperating to choose from, with an apparently unlimited scope of scrolling, bombarding the user with GIFs. The current models openly available online do not predict a GIF's relevance based on the likelihood of being chosen by the user, which we aim to change.

Our proposed method deals with a situation not currently incorporated by GIF predicting platforms. Our project suggests GIFs to a user based on the text message they are typing dynamically, doing away with the hassle of searching for a relevant GIF. The novelty here lies in being able to suggest GIFs in real time i.e. by taking text strings from the chat, eliminating the need for explicit search for keywords, and in narrowing down the search space by limiting the number of GIFs to suggest to a user, instead of the current behaviour of platforms that provide us with a massive number of suggestions. An important original feature this proposed method leverages is a score that associates the likelihood of choosing a GIF customized based on a user's GIF history.

**Related work:** GIFs are highly prevalent on social media and have infiltrated our lives to a great extent. Despite this, the field of GIF prediction hasn't seen much formal research. However, there has been a tremendous amount of research in NLP approaches relevant to this project, one of those being Doc2Vec introduced by Le and Mikolov in [1]. They suggest an unsupervised framework that learns continuous distributed vector representations for pieces of texts of variable length ranging from sentences to documents and referred to as paragraph vectors. Trained using Stochastic Gradient Descent and Backpropagation, their work was inspired by research on word level text representation. With the lowest positive/negative and fine grain error rates, the performance of paragraph vectors was the best amongst all vectorization approaches until the time their research was published in 2014. Their findings are backed by a more recent evaluation [2] of Doc2Vec, in which Lau and Baldwin empirically evaluate the quality of document embeddings generated by Doc2Vec with two baseline methods - word2vec and an n-gram model - as well as two competitor methodologies - skip-thought and paragram phrase. They found that doc2vec performs rather well and generates robust document embeddings even when trained on large external corpora, and benefits from pre-trained word embeddings. A paper [3] on collecting emotional animated GIFs from the Internet used clustering, an approach our project also employs, on the structure of emotions in GIFs. Their assessment clusters 17 emotions based on visual features, tags and emotion scores and generates a dataset based on that.

## 2 Methods

We have decided to carry out overall GIF prediction in 3 stages:

1. Data Gathering
2. Data Cleaning
3. Exploratory Data Analysis [contains ML and NLP technique]

Submitted for CSC 522 Automated Learning and Data Analysis (Spring 2019).

The third part in Exploratory Data Analysis is the core and majorly involves ML and NLP techniques performed in a series of four functions the details of which are explained below:

### 2.1 Sentiment Analysis [NLP Technique]

The basic idea behind this at the initial stage is to collect the user's tone for input, which can easily help our algorithm narrow down GIFs. We are using the TextBlob library to perform this step.

- *Polarity Score*: for each word in the user input we will calculate polarity score using TextBlob. It ranges from [-1, 1].
- After taking polarity score of each word, we will average them to find the overall tone of the sentence whether it is in positive tone or negative.
- If sentence turns out to be a positive one, our algorithm will only have GIFs having positive sentiment or Neutral.
- For example, sentence polarity score of 'I am happy' > polarity score of ' I am sad'.

### 2.2 Word Embedding: [NLP Technique]

**Library: Gensim in Python**

- We are representing each GIF title as a vector to make it easy to find cosine distances between GIF titles. This operation is done only once at the beginning.
- Similarly, we are implementing this method on the user's input also.
- This helps us to find proximity between GIF descriptions which is used in the next two stages of clustering and KNN.
- Sentences containing similar words and structure will most likely be closely aligned.

### 2.3 K Means ++ Clustering: [ML Technique]

**Library: Scikit-learn in Python**

- Using the document vectors we found in the previous step, we generate document clusters of various sizes to determine the right number of clusters for our dataset.
- The metric used to calculate distances between all document vectors was cosine distance. Once the model has been trained for a number of clusters, we do not need to train it again for every new user input.
- The clusters' sum of squared distances (SSD) is plotted against the number of clusters to obtain the following plot:
- "SSD" of all clusters was found to wane with the increasing number of clusters and its saturation was achieved at about 1400. We decided to generate 600 clusters in our model to maintain the purity as well as the significance of each cluster.
- As we had expected, the silhouette coefficient of the same increased with an increase in the number of clusters, depicted in the results section of this report.

### 2.4 K-Nearest Neighbour: [ML Technique]

**Library: Scikit-learn in Python**

- The previous stage of clustering significantly reduces the search space by mapping an input text to a cluster of GIF descriptions.
- We now compare the cosine distances of all the GIF descriptions within the chosen cluster with the input text.
- List of K nearest GIFs is passed on to the next stage that inculcates a user's GIF preference, an integral part of this project. A weightage of 70% is given to the cosine similarity and 30% to the score when generating the final list of GIFs to display to the user.

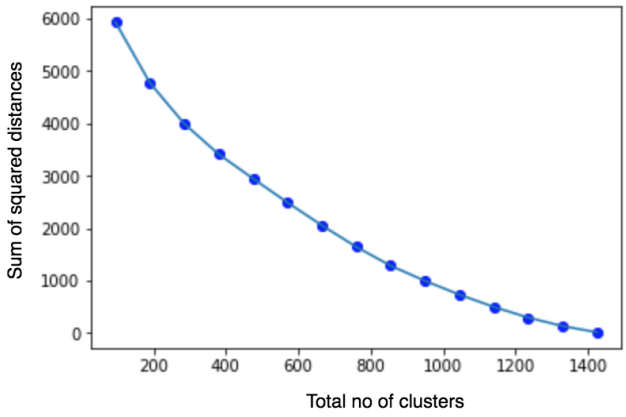


Figure 1: SSD vs Clusters

### 2.5 Score Analysis:

**Method inspired from Adaboost to maintain and update scores**

The GIFs are also filtered based on user preference. A total of n (where n is a hyper-parameter) GIFs are recommended to the user. This is done after sentiment analysis and filtering based on synonyms and k-Nearest Neighbors of Word2Vec (K is set to 20 here). The similarity is combined with a score, which together have to score a certain threshold T to be displayed to the user. Initially, si is set to 0.5 for all GIFs and is later updated based on user selections. Working on the lines inspired by Adaboost,

If selected,

$$si = si + sin(\alpha) \tag{1}$$

Otherwise for the remaining 9, the GIFs that were displayed but not selected,

$$si = si - sin(\alpha) \tag{2}$$

Where, range of $\alpha = [0, 90]$

**Rationale:**

- *Sentiment Analysis* - A key feature of data when texting people is the sentiment behind sending a message. A basic sentiment analysis determining whether a text is positive, negative or neutral can be incredibly helpful in filtering the GIFs before final selection. Texts can easily be generalized and associated with GIFs that are similar in sentiment to the intended message being conveyed, making it a suitable choice as a feature in the pipeline of GIF prediction.
- *Doc2Vec* - The results found in [1] and [3] establish doc2vec as a promising approach towards generating word embeddings. It has been proven to be successful in preserving the syntactic and semantic meaning of sentences as well as entire documents. Doc2vec delivers the best results when compared to the other competitive approaches as found in [3].
- *Clustering* - This acts as a precursor to KNN to help reduce the search space for the most relevant GIFs. Once clusters have been generated, a massive amount of calculation can be avoided by simply ignoring GIFs from the clusters a user's input has not been assigned to. Using K means ++ reduces the convergence time of the algorithm, making it more efficient.
- *K Nearest Neighbour* - This is used to find the word embeddings of GIF titles that are the most similar to user inputs, through doc2vec. The representation being a vector space necessitates the use of KNN as we are dealing with independent sentences that need to be mapped to subsets of the corpus in a many to many mapping.

- *Score (Data Analysis)* - This is used to keep track of a user's history of GIF selection. It is a technique we developed independently and chose to base loosely on Adaboost. Incrementing scores of GIFs that were selected by the user and decrementing scores of the ones not chosen allows us to quantify said history without actually having to keep track of all the GIFs selected by the user. This will not differentiate us from the existing products, but also make our product more user-centric.

## 3 Plan and Experiment

**Datasets:** GIFs prediction for our project is done using the proposed method explained above. This requires the use of a dataset. But, we could not find any dataset that is already available online that suited our purpose based on our designed algorithms. Therefore, we chose to create our own dataset, the details of the same are described below:

*GIPHY Dataset*: **9246** GIFs

- Source - https://giphy.com/ and its API.
- Method - Used requests library in python to fetch GIFs data from giphy-api in JSON format.
- The data was accessed based on certain words/tags that we tend to commonly used in chat conversations.
- The effort to be put in the Data Cleaning task was reduced here since we created our own dataset. We already filtered out the attributes by extracting only the ones which are useful for our algorithm.
- But we performed Data Pre-processing on the description of each of the GIFs. We applied various NLP techniques like Tokenization, Lemmatization on the GIF metadata (explained in detail later).
- The JSON stub for each of the objects for every individual GIFs contains the following five attributes -
  "id": this is the unique ID of each and every GIF (unique identifier).
  "link": this is the actual link or the url to the GIF on giphy.com which has been stored via web archiving.
  "description": meaningful textual description about the GIF (metadata). This is one of the main attributes that is used to train our model and access relevant GIFs to the user's text message.
  "Score": is an additional custom attribute, which is updated based on user preference depending on constraints defined. This helps our model learn from the user's history of GIF selections.
  "Sentiment": one moreadditional attribute, which denotes the sentiment of the GIF (positive, negative or neutral).

**Hypotheses:** The questions that spiked our interest in GIF prediction are:

1. Are the user preferences pertaining to GIF selection that might or might not change over time, related to the GIFs recommended by the algorithm?
2. Is it possible to predict and recommend GIFs considering dynamically typed text by the user based on the underlying emotional tone of the text while considering the entire sentence structure?

These form the hypotheses, which lay the foundation for our project, and is implemented using the workflow described below.

**Experimental Design:** To get the answers of the hypotheses mentioned above we decided to carry out 3 below processes with the help of GIF data. The completed steps so far fall in the 3 stages mentioned below:

1. Data Gathering: At the initial stage, we collected data containing various GIFs. These datasets contain a proper meaning/title for each GIF in the dataset. Overall, we are working

with 9246 GIFs and their title. A detailed description of the dataset is provided in section 3 of the report.

2. Data Cleaning:

   - After gathering we performed various data cleaning operations on our dataset which can provide deep insights into the data to perform ML operations in later steps.
   - This cleaning is done only once at the initial stage.
   - Other than that, we will perform data cleaning on the user's text input whenever it's given. This can help us to more accurately perform our techniques.

   *Library used: NLTK in python*
   For this we have carried out these methods to bring data in proper format:

   - Consider user's text: 'Let's go "partying" after the presentation!'
   - Removal of punctuations & convert to lower-case [on GIF dataset and user's input]: We removed all punctuation based on a string of punctuations that we defined specific to all punctuation and special characters that we wanted to remove from the original string. And for consistency during similarity measure, we also converted the string to lower case.
     After lowercase and punctuation removal: 'lets go partying after the presentation'
   - Tokenizing words [on GIF dataset and user's input]: We also tokenized each word in titles and stored it back.
     After Word Tokenization: ['lets'. 'go', 'partying', 'after', 'the', 'presentation']
   - Removal of stopwords: [on GIF dataset and user's input]: We removed all stop-words using NLTK's default corpus for stopwords and our custom words which give us no information in further steps.
     After stopwords removal: ['lets'. 'go', 'partying', 'presentation']
   - Lemmatizing words [on GIF dataset and user's input]: this step was eminent for our proposal as it can easily convert different tenses of a word in base word.
     After lemmatization: ['let'. 'go', 'party', 'presentation']

3. Exploratory Data Analysis
   This will use 5 techniques. Two of which (Sentiment Analysis and Doc2Vec) have been implemented which give us an idea of the emotional tone of the user and correlated words. Using a combination of above 3 techniques with Clustering, KNN Score Analysis lead us to answers to the questions asked in hypotheses. The details of the entire flow are explained in section 2 of the report. So, the basic flow is:

   - Once we get the pre-preprocessed user text message and dataset with preprocessed description (along with Sentiment Analysis key-value for each GIF object)
   - Sentiment Analysis is done for the text message
   - Then the description of GIFs is vectorized using Doc2Vec
   - K-means++ Clustering is performed on the vectorized dataset
   - k-NN classification is performed based on Cosine similarity
   - After which score analysis is performed in combination with above found similarity in order to clear the threshold (T) to give the final output to user,

$$n\ outputs = \begin{cases} GIF(i), & if\ T < s_{i^{th}}\ and\ i < n - r \\ GIF(random(i)), & if\ n - r < i < n \\ reject, & otherwise \end{cases}$$

   - After user selection, we will update scores and T to make the model learn from user preference and to make it more accurate
     Now, updating the threshold (T),

$$s_{i\,(new)} = \begin{cases} s_{i\,(old)} + \sin(\alpha), & \text{if GIF } i \text{ is selected} \\ s_{i\,(old)} - (\frac{\sin(\alpha)}{n}), & \text{if GIF } i \text{ is not slected} \end{cases}$$

$$T_{new} = T_{old} + t, \text{where } t = \beta * \#iteration$$

## 4 Results

As GIF preferences are subjective to the user using it and can vary from user to user, we decided to bring out the overall analysis of the system into 3 analyses which can help us understand the working and overall accuracy of the system in a better way:

**Analysis 1: Sentiment Analysis of user's input and GIFs predicted**

- The sentiment class predicted for the User's text message is compared to the majority of predicted sentiment class for the 10 recommended GIFs, so as to measure the accuracy. In case of a tie, we accept the same class as the user's text message if it is a part of the tie, otherwise we generalize it to the neutral class.
- For the analysis part, 300 common user text messages were used over 9246 GIFs, and these are the results we got:

| *Emotional Tone* | **GIF Predicted** | | |
|---|---|---|---|
| **Actual User Text** | Positive | Negative | Neutral |
| Positive | 89 | 10 | 26 |
| Negative | 17 | 81 | 27 |
| Neutral | 12 | 5 | 33 |

We got an **Accuracy: 0.6767 = 68% (approx.)** here.

- This analysis answers our hypothesis 2 stated in section 3 to a certain extent.

**Analysis 2: Silhouette Analysis**

- Another factor we considered to know our system's performance is how far away is each GIFs context from its neighboring clusters and is thus well located and matched to its own cluster.

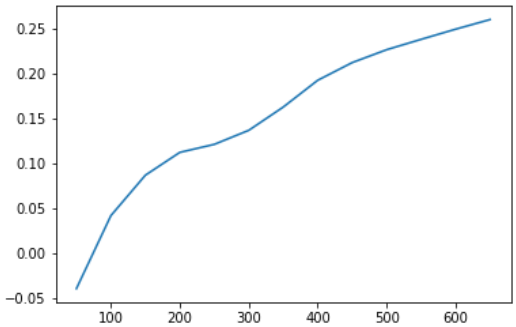


Figure 2: Silhouette Coefficient

- Since we had decided to generate 600 clusters based on the elbow plot in step 3 of the proposed method, we only considered silhouette coefficients until this value. This finding aligns with our previous understanding that the greater the number of clusters we generate,

the better since we see the approximately concave down plot increasing with the number of clusters.

**Analysis 3: A survey of our system amongst different users**

- As the system is subjective to the user using it, we decided to conduct a survey amongst people by asking them to evaluate the system after using it based on their common chatting vocabulary. The results we got from the user's experience:

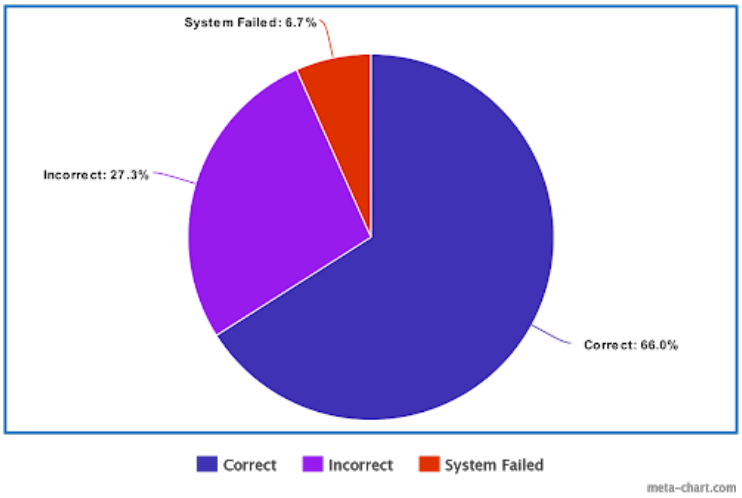


Figure 3: User satisfaction

- Here, we observed the user survey's correctness **= 66% approximately** matches the accuracy we got in Analysis 1.
- Also, **93.8%** out of 32 said that system was successfully able to store their **GIF preferences** over time which successfully answers our hypothesis 1 stated in section 3 of the report.
- Survey link is available here: Survey Link

## 5 Conclusion

1. We understood the importance of an accurate dataset, and how much it can affect further computation cost. Gathering a new database was initially difficult but paid off later, since we needed minimal Data Cleaning and relatively less Data Pre-processing.
2. We learned how and when to apply NLP techniques. NLP didn't just help us to find better cosine similarities by reducing computation and preserving context but also helped us to maintain uniformity between data collected and user input.
3. Various different embedding techniques (word2vec, doc2vec, GloVe) were compared and one was used to embed vast metadata for faster computation. We combined clustering in our model to minimize time complexity and get more accurate results.
4. Research was also done to compare techniques like kmeans, kmeans++ to combine with knn. Elbow plot was fairly helpful to find and experiment with different values of k.
5. We invented a technique to make the model learn and adapt to user inputs. The technique tries to remember user's preferences and through this, our model learned to take user's history of GIF selection into consideration
6. As a future scope, we wish to implement the same idea using RNN to take advantage of faster and accurate learning for our model. RNN would also be helpful to compute a much larger dataset.
7. Demo Link: Link to demo video

8. All codes are available on GitHub. [1]

## References

[1] Le, Q. & Mikolov, T.. (2014). Distributed Representations of Sentences and Documents. Proceedings of the 31st International Conference on Machine Learning, in PMLR 32(2):1188-1196

[2] Lau, J.H., & Baldwin, T. (2016). An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation. Rep4NLP@ACL.

[3] Chen, Weixuan & Rudovic, Ognjen & Picard, Rosalind. (2017). GIFGIF+: Collecting emotional animated GIFs with clustered multi-task learning. 410-417. 10.1109/ACII.2017.8273647.

[1] `https://github.ncsu.edu/vcdesai/CSC-522_P03_GIF-Prediction`