**CS 6200: Information Retrieval**

**Final Project Report**

**Team Members: Meet Nandu, Praguna Singh, Yash Kothari , Sanyam Harne**

The task –

The digital space gives us an opportunity to explore and experiment without any limitations – literally anything we dream of can manifest right in front of us with just a few clicks. GIFs are one way we’re pushing these limits, altering the way we communicate with one another in cyberspace.

Even if you can’t define the word “GIF,” you’ve definitely seen one before. They helped define the early internet, and they’re more popular now than ever before. But what exactly is a GIF, and how do you use them? In its simplest form, a GIF (pronounced “gif” or “jiff”) is just an image file. Like the JPEG or PNG file formats, the GIF format can be used to make still images. But the GIF format has a special feature—it can also be used to create animated images like the one below.

A picture containing text

Description automatically generated

GIFs really started infiltrating pop culture when netizens began slicing up iconic moments from TV shows and films and sharing it with their friends. It quickly gained popularity as people can relate to that singular moment and sharing it with their friends is like sharing an inside joke. Just like emoticons, it has slowly evolved into a way to express yourself on the internet.

With the rise in popularity, companies like GIPHY and Tenor are taking advantage of them while using a searchable database like Google. GIPHY is one of the largest search databases and search engines for GIFs. And the second largest after Google. It serves over 10 billion GIFs a day through social media and messaging apps. Another popular GIF search engine is Tenor, acquired by Google in March of 2018.

The main task of this project would be to create a search engine that would retrieve GIFs with queries on description of the GIF, dialogues, captions, and so on. A user would type in a query and would expect a GIF related to that query. That is what the search engine would aim to do.

**Proposed Modeling Approach** –

We have a corpus of around 129000 GIFs with their descriptive tags that can be used for the search engine. We will use the Vector Space Model approach for the query search as follows:

* **Stopping**: We plan to first remove all the stop words from the corpus to stop them from being indexed. Stop words are the most frequently used words in a language. These words being very frequently used do not provide any significant context to the document. Thus, we can ignore them while creating tokens for the document. Hence, we will remove all the stop words from the set of gif documents first.
* Next, we will remove all the punctuations from the documents as they do not provide any information about the context of the gifs. Hence, they may cause the same words in different documents to be considered different. For example, two documents having “Dwayne Johnson” and “Dwayne Johnson.” Tags respectively might be similar, and may both be the result of the query but due to the punctuation they might provide incorrect results and only one of these might show up based on which is closer to the provided query.
* **Stemming:** Next, we will do stemming on the descriptive tags of the documents to get the root words of different forms of the same words, for example having and have, etc. This will allow us to remove redundant tags which might have the same meaning and provide a similar context to the documents or GIFs.
* **Tokenize**: Next, we will tokenize the documents and get all the tokens that provide meaningful context to the documents and can be used for our search engine.
* **Tfidf**: Next we will calculate the tfidf of all the terms/tokens in the documents and store it in the dictionary as the value of the tokens which can be further used for our modelling approach.
* **Vectorizing Documents**: Next we will create a vector representation of each document or GIF and store it for query search using cosine similarity. The vector of the documents seen in a tabular fashion will look like the following figure:

Table

Description automatically generated

Figure : Term-Document matrix for a collection of 4 documents

As seen in figure 1, we will create a dictionary of the vectors of each document which will have tfidf values of the tokens present in the document as their values and 0 values for tokens absent in the documents. Each document token will look like figure 2.

Text

Description automatically generated

Figure : Vector for Document

* **Query Processing:** We will take the query and remove any punctuations from the query and implement stemming as before. This will provide us with an optimized query for the search.
* **Vectorizing Query:** Next we will vectorize the query similar to the documents. Hence, creating a vector representation of the query as well.

Text

Description automatically generated with medium confidence

Figure : Vector for Query

* **Cosine Similarity**: Next we will find out the cosine similarity between the query and each document, to find which document matches closely to the query. Then we will sort the cosine similarities for the query with each document in a descending order and the top 10 results will be shown to the user as query search results. We wll use the following formula for the getting the query similarity:

Schematic

Description automatically generated with medium confidence

Figure : Cosine Similarity between Query and Documents

**In General, what do Queries Look Like?**

The proposed search engine will expect queries like:

* Man woman
* Dog cat
* Happy man
* Dancing woman
* Man woman dancing

**Query Results –**

The search engine will provide with a list of 10 GIFs that match closely or are potential search results for the query. These will be sorted in the order of how likely the GIF is the result the user is looking for.

For example, if the query is: “man woman dancing” the result will show a GIF with a man and woman dancing on top of a GIF which might only have a man dancing or a only a woman dancing.

The query result for “happy woman man” will be something like this:

Graphical user interface, application

Description automatically generated Graphical user interface, website

Description automatically generated

A picture containing text, monitor, screen, screenshot

Description automatically generated A screen shot of a group of people

Description automatically generated with low confidence

Graphical user interface, application

Description automatically generated

**Evaluation Metrics –**

We will be using the following metrics to decide the correctness of our proposed search engine:

* **Precision**: This will show us how relevant are the search engine’s retrieved results. It is given as a fraction of the number of relevant documents retrieved by the number of retrieved documents.

Text

Description automatically generated with low confidence

Figure : Precision

* **Recall**: This will show us how good is the search engine in retrieving the relevant results. It is given as a fraction of the number of relevant documents retrieved by the total number of all the relevant documents. It is often called the sensitivity of the search engine.

**Text

Description automatically generated with low confidence**

Figure : Recall

* **Weighted precision**:We also plan to use our own evaluation metric based on how satisfying the retrieved results are. We will first rate the results as completely satisfying (2), partially satisfying (1) or unsatisfactory(0). Sum these weights for each retrieved result and then divide it by the 3\*(Number of retrieved documents.)