CSDS 234 - Project Report

Analysis of Different Representations of Twitter Data

https://github.com/jexiao8211/Analysis-of-Different-Representations-of-Tweet-Data

Jerry Xiao, Akansh Devendra, Quoc Trung Nguyen

1 PROBLEM DEFINITION

The main objective of this paper is to analyze the space and time complexity of storing and querying different text-data representations. Twitter data is particularly interesting as it has a large user base of 450 million monthly users [1], leading to around 500 million daily tweets [1]. This data comprises information that can play an important role in diagnostic and descriptive analysis, such as sentiment analysis about political leaders, sports teams, and stocks. It could also detect crimes, bots, or world events.
Since each tweet is limited to 280 characters, processing tweets is much more optimized, making it more efficient to search and query a dataset.[2] Our results will illustrate the efficiency of each representation compared to each other, allowing us to make conclusions about the risks/rewards of each implementation. Our tests would generally apply to any vectorized text representation of data. While accuracy is essential, so is the ease of storage and access of this data. Therefore, because twitter data is massive, it is of great importance for researchers to use it for further research. Thus, we need to be able to store and query such large amounts of information efficiently.[2]

2 RELATED WORK

Prior research has shown different ways of using NLP keyword extraction to test the accuracy of varying keyword extraction algorithms (Slobodan Beluga, 2014). Unsupervised learning algorithms for keyword searches are the norm for social media-related data. Still, TF-IDF has always been considered the common baseline for keyword extraction, compared to the newly proposed methods like TextRank or PageRank. In 2014, Five researchers from Carnegie Mellon University wrote a paper regarding automatic keyword extraction from Twitter. The paper also provides an in-depth analysis and accuracy of different twitter data storage methods. The paper also addresses frequent usage, certain keywords, linguistic variants, and the high variance of the cardinality of keywords in each tweet. The methods proposed in this paper address these concerns and significantly improve this task's data set.[3][4]. In January 2021, English researchers collected tweets that mentioned specific keywords related to COVID-19. Individuals exposed to COVID-19 were able to be identified by researchers by writing additional expressions. The researchers then made the individual's tweet timestamp and geolocation available to the public, which became a complementary resource for tracking the spread of COVID-19.[6]

3 METHOD

3.1 Overview

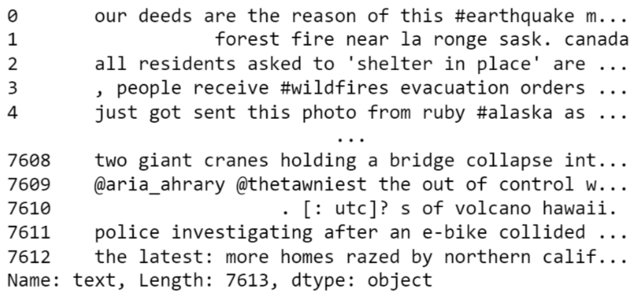
We utilized two text-data vector representation methods: CountVectorizer and Term-Frequency Inverse Document Frequency (TF-IDF). These methods are based on a concept called Bag of Words (BoW), a natural language processing and information retrieval tool. BoW is often used in various applications, including information retrieval, text mining, keyword extraction, search engine, etc. In addition to CountVectorizer and TF-IDF, we tried a newer representation method called Brown Clustering - which is extremely useful in Clustering lexical variants. We are primarily testing for the following metrics:

1) Time is taken to query each representation with varying query sizes

2) Storage size for each representation with varying dataset sizes, and
3) Query time for each representation varying the dataset sizes.

3.2 Raw Data

We imported data from Kaggle's Natural Language Processing with Disaster Tweets. The data consisted of 7612 Tweets talking about natural disasters and five columns, namely "id," "text," "location," "keyword," "and target." We then deleted all columns except for text data. We removed all non-printable characters, character references, URLs, and HTML tags from the dataset.[7]



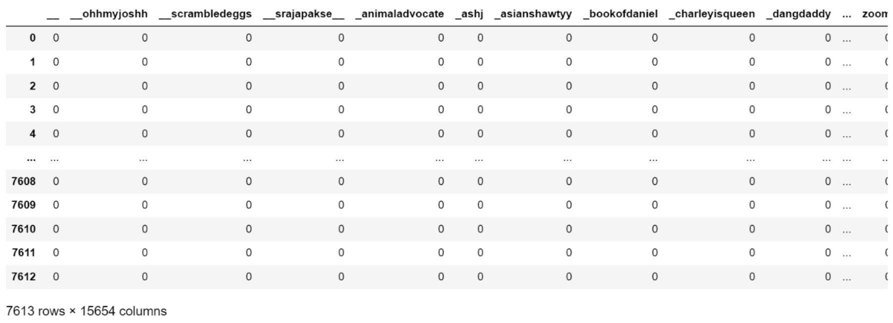
```
0       our deeds are the reason of this #earthquake m...
1                    forest fire near la ronge sask. canada
2       all residents asked to 'shelter in place' are ...
3       , people receive #wildfires evacuation orders ...
4       just got sent this photo from ruby #alaska as ...
                              ...
7608    two giant cranes holding a bridge collapse int...
7609    @aria_ahrary @thetawniest the out of control w...
7610                          . [: utc]? s of volcano hawaii.
7611    police investigating after an e-bike collided ...
7612    the latest: more homes razed by northern calif...
Name: text, Length: 7613, dtype: object
```

Figure 1: Text data used to create vectorized representations

We then applied the different text-data representations on the cleaned data.

3.3 CountVectorizer

CountVectorizer is a scikit-learn library tool that can convert text data into a vector depending on the frequency of each word in the text. It tokenizes the documents (Tweets) to make a vocabulary dictionary of all the words represented in every record in the corpus. A column represents each distinct word, and each row represents a tweet. As a result, each document is represented as a vector whose size is the vocabulary list size, and each value is the frequency of the words in that tweet.[8][9]



	__	__ohhmyjoshh	__scrambledeggs	__srajapakse__	_animaladvocate	_ashj	_asianshawtyy	_bookofdaniel	_charleyisqueen	_dangdaddy	...	zoom
0	0	0	0	0	0	0	0	0	0	0	...	(
1	0	0	0	0	0	0	0	0	0	0	...	(
2	0	0	0	0	0	0	0	0	0	0	...	(
3	0	0	0	0	0	0	0	0	0	0	...	(
4	0	0	0	0	0	0	0	0	0	0	...	(
...	...	...	...	...	...	...	...	...	...	...	...	..
7608	0	0	0	0	0	0	0	0	0	0	...	(
7609	0	0	0	0	0	0	0	0	0	0	...	(
7610	0	0	0	0	0	0	0	0	0	0	...	(
7611	0	0	0	0	0	0	0	0	0	0	...	(
7612	0	0	0	0	0	0	0	0	0	0	...	(

7613 rows × 15654 columns

Figure 2: CountVect vectorized text

Pros of CountVectorizer:

- Easy to understand and implement
- Helpful in understanding text types by word frequency

Cons of CountVectorizer:

- Unable to identify which word or concept is important
- Unable to extract the context of the corpus
- Does not identify the linguistic similarity of words

3.4 TF_IDF (term frequency-inverse document frequency)

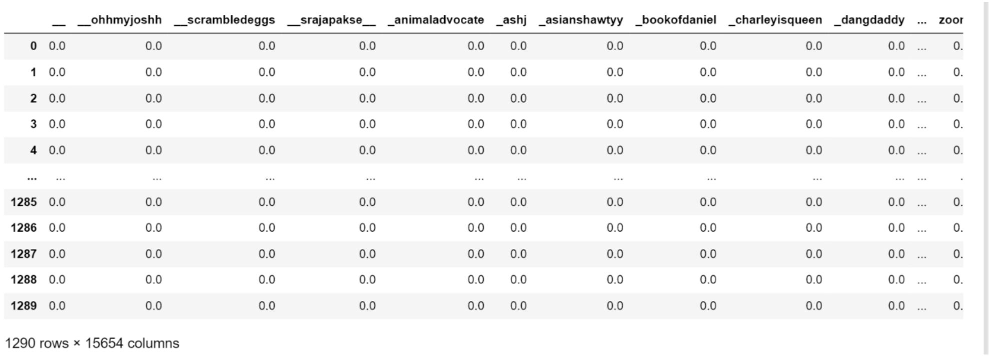
TF-IDF is a scikit-learn library tool that changes text data into a vector based on the frequency of a word in the text. TF-IDF includes the statistical measure of how a keyword is relevant to the text collection. TF-IDF consists of two measurements:

- Number of times a word appears in the document (term frequency)

  TF(t,d) = number of times t appears in d / total number of terms in d
- The inverse document frequency of a keyword across the documents

$$idf(t, D) = log\left(\frac{N}{count\ (d \in D{:}t \in d)} \right)$$

Multiplying tf(t,d) and idf(t,d) will result in the tf-idf(t,d) relevancy of the word in the whole set of documents. A low TF-IDF(t,d) closer to 0 means that t is a common term across the corpus of documents. A TF-IDF(t,d) closer to 1 standard such than t is a rare term across the corpus. Like CountVectorizer, TF-IDF does not carry the semantic meaning of words. The statistical measurement only shows the "importance" of the word in the tweet. In addition, compound terms would not be considered a single unit, thus breaking the semantic meaning of the sentence.[8][9]



	__	__ohhmyjoshh	__scrambledeggs	__srajapakse__	_animaladvocate	_ashj	_asianshawtyy	_bookofdaniel	_charleyisqueen	_dangdaddy	...	zoor
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.
...	...	...	...	...	...	...	...	...	...	...	...	.
1285	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.
1286	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.
1287	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.
1288	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.
1289	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.

1290 rows × 15654 columns

Figure 3: TFIDF vectorized text

Pros of TF-IDF:

- Easy to compute and implement
- Compared to CountVectorizer, TF-IDF considers the importance of the word
- Can make the model less complex by reducing input dimensions

Cons of TF-IDF:

- Unable to deal with linguistic similarities of words in a corpus
- Useful only as a linguistic-level feature

3.5 Brown Clustering

Brown Clustering is a hierarchical clustering algorithm based on the context of the words if similar words have similar distributions of words to their left and right. All keywords in the document are assigned a bit string. To use Brown Clustering to predict the keyword, we must compare the surrounding words of the different keywords in bit-string-form. If there is a good similarity between those surrounding words of keywords, they will be under the same cluster of keywords.[10][11]

The algorithm is based on a cluster n-gram model. The algorithm divides items into classes, and the optimization function average mutual information (AMI) is utilized to merge keywords such that there is an almost negligible loss in global mutual information. The output of the algorithm is a sequence of merges. [13]

Mutual information (MI) = $$\mathrm{MI}(\mathrm{C}_i, \mathrm{C}_j) = p(\langle \mathrm{C}_i, \mathrm{C}_j \rangle)\ \log_2 \frac{p(\langle \mathrm{C}_i, \mathrm{C}_j \rangle)}{p(\langle \mathrm{C}_i, * \rangle)\, p(\langle *, \mathrm{C}_j \rangle)}$$

Average mutual information (AMI) = $$\mathrm{AMI}(\mathrm{C}) = \sum_{\mathrm{C}_i, \mathrm{C}_j \in \mathrm{C}} \mathrm{MI}(\mathrm{C}_i, \mathrm{C}_j)$$

We chose 800 clusters because setting 800 clusters showed a logarithmic relationship between accuracy and the number of tweets for a dataset size of 7613 tuples (Leon, 2015).
Ideally, a cluster will look something like: [yes, yep, yup, nope, yess, yesss, yessss, ofcourse, yeap, likewise, yepp, yesh, yw, yuup, yus], where all words in the cluster generally mean the same thing (Owoputi et al., 2013). Our implementation of Brown was not as successful and consisted of many clusters of a single word, several clusters with 1-10 words, and a handful of clusters with hundreds of words. For each data representation. Improving our brown model has proven difficult, but it is still far from ideal.

Pros of Brown Clustering:
- Able to handle large datasets because it is based on a statistical model that allows the Clustering to approximate the clustering process.
- Suitable for document classification and semantic process.

Cons of Brown Clustering:
- Computationally intensive, making it difficult to use on large datasets.
- Based on a statistical model may not produce the most accurate or interpretable result.

3.6 Query Function
The purpose of the query function is to return all tweets relevant to the given keyword(s). The Function has the same input and output for all vectorized data representations.
Input: list of keywords
Output: all tweets are about those keywords and the runtime of the query function.
- **Implementation for TF-IDF and CountVect:**

Traverse all columns of the vectorized tweets representations. Then, columns that match the input keywords are searched row-wise for all tweets where the keyword exists (value > 0). We then return the tweet index values alongside their respective runtime values.
- **Implementation for Brown:**

Given a set of keywords, linearly search the brown clusters for keyword matches, then return the cluster associated with each match. The clusters are then concatenated into a single list and passed into the TF-IDF query function.[13]

3.7 TEST IMPLEMENTATIONS
3.7a Test runtimes of each representation with different query sizes
As discussed later, any query of more than eight keywords took extraordinarily long and was unnecessary for this demonstration.
For each query size, s (where s is 1, 2, 3, 4, 5, 6, 7, or 8):
1. A random sample of s keywords was selected as the query words.
2. We query each data representation using those selected query words.
3. The time it took to run the query was recorded to reach different data representations.

4. We repeat Steps 1-3 for ten iterations and obtain the average time, for a total of 80 queries.

3.7b Test the relationship between the number of tweets and the size of the data representation

We create an instance for each data representation of the following sizes: 1500 tweets, 3000 tweets, 4500 tweets, 6000 tweets, and 7613 tweets (whole dataset). These were all stored as .csv files with the same formatting. The size of each CSV representation was retrieved. It is important to note that the Brown representation utilizes both the brown clusters and the TF-IDF representation, so we calculate the size by adding the two sizes.

3.7c Test the relationship between the number of tweets and query time

For each data representation, each of the different data sizes from step 2 was tested with a query size of 3 keywords. Twenty test iterations were performed for every data representation/size combination, and the runtimes were recorded and averaged for each data size.

4 RESULTS AND ANALYSIS

4.1 Average query time for different number of keywords

Query Size (Number of Keywords)

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
CountVect	0.01724226474761963	0.02783757448196411	0.03952515125274658	0.051900172233581544	0.06796498298645019	0.08448317050933837	0.09484838247299195	0.1126623272895813
TFIDF	0.014464783668518066	0.02550452947616577	0.0376893401145935	0.04739259481430054	0.06234642267227173	0.08207361698150635	0.09202570915222168	0.10178455114364623
Brown	7.319691252708435	24.610164976119997	33.989655470848085	55.077748489379886	77.4521006822586	110.89618942737579	129.91393275260924	145.31892223358153

Table 1: Average query time of varying query sizes on all different data representations, measured in seconds

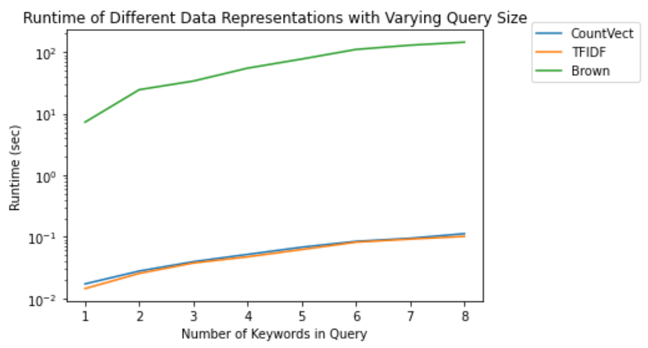


Figure 4a: all representations with a log scale y-axis

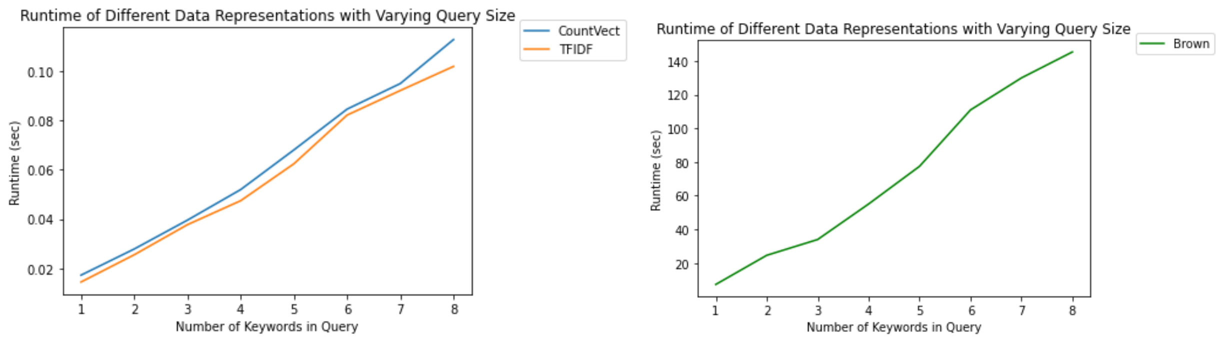


Figure 4b: TFIDF and Countvect plotted together, Brown plotted by itself with different scales

TFIDF and CountVect are significantly faster to query than the Brown representation, which is valid for all query sizes 1-8. This is likely due, in large part, to the uneven clustering phenomenon that we ran into when trying to implement Brown. With an ideal Brown representation where the clusters are mostly the same size, we expect querying brown to be about n times slower than TFIDF, where n is the average size of the clusters. The growth in query time as query size increases is linear for all representations, with TFIDF and CountVect having a near-identical growth rate.

4.2 Space taken to store data of different sizes with different representations

Size of Dataset (Number of Tweets)

	1500	**3000**	**4500**	**6000**	**7613(full dataset)**
CountVect	14.8 MB	50.3 MB	100 MB	160 MB	239 MB
TFIDF	29.8 MB	101 MB	201 MB	321 MB	478 MB
Brown	30.15546875 MB	102.05 MB	202.83 MB	323.73 MB	481.6 MB

Table 2: Size of data stored as .csv file for varying dataset sizes for each data representation

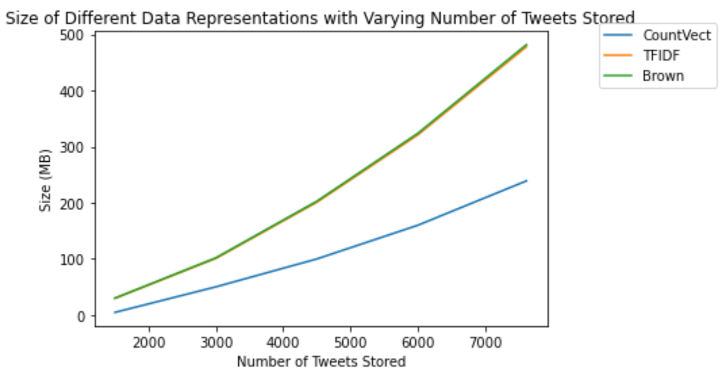


Figure 5

Figure 5 shows that the TF-IDF representation of the data is overall larger than the CountVect representation for every dataset size- this is no surprise as TF-IDF stores each value as a float, while CountVect stores each value as an int. It is interesting to note that TFIDF also grows in size faster than CountVect as dataset size increases. Again, this could likely be attributed to TF-IDF storing values as floats rather than integers. The Brown curve follows the TF-IDF curve very closely, which makes sense considering the relationship between the two: Brown cluster is only composed of every unique word in the entire dataset, which is asymptotically insignificant compared to TF-IDF.

4.3. Query time for different dataset sizes.

Size of Dataset (Number of Tweets)

	1500	**3000**	**4500**	**6000**	**7613(full dataset)**
CountVect	0.013947844505310059	0.024823665618896484	0.028082501888275147	0.0350399613 3804321	0.04536161422729492
TFIDF	0.011231791973114014	0.0216595292 0913696	0.026416730880737305	0.0321657896 0418701	0.04337368011474609
Brown	31.487934696674348	32.487717390060425	29.90742540359497	61.54061765670777	46.681753146648404

Table 3: Average query time (with query size of 3 keywords) of varying dataset sizes for each different data representation

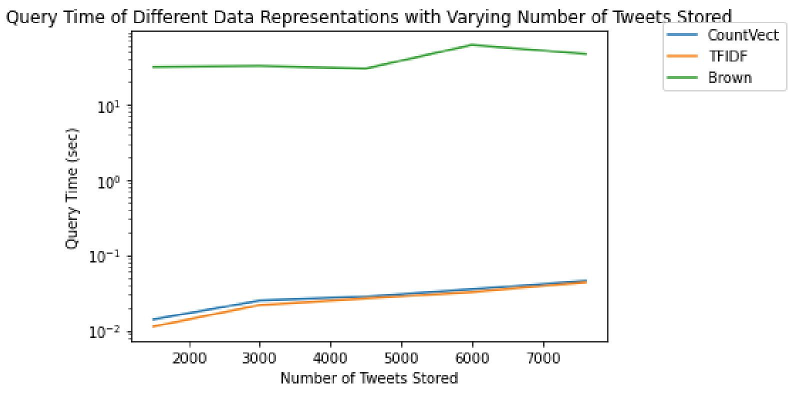


Figure 6a: all representations with a log scale y-axis

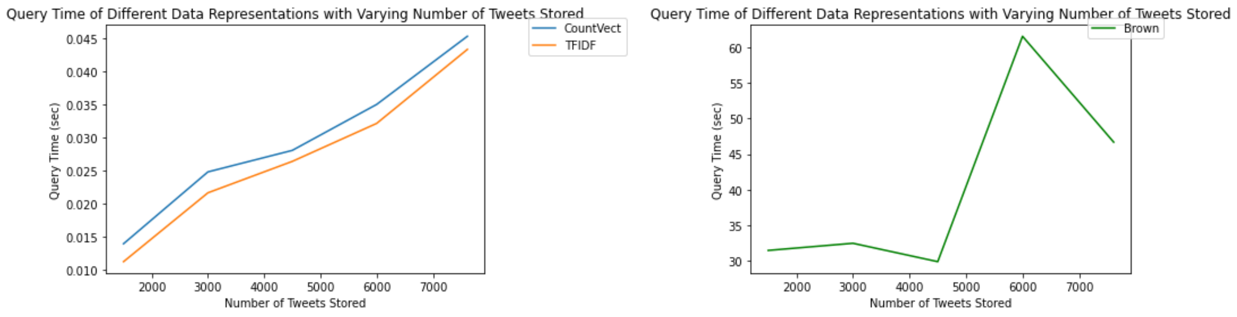


Figure 6b: TFIDF and Countvect plotted together, Brown plotted by itself with different scales

As the dataset increases, with a fixed query size, CountVect and TFIDF again grew at near identical rates, with TF-IDF just taking slightly longer. We could not get any meaningful data from the growth of the dataset and query time for Brown because of the clustering issue - the average query time for brown was likely dependent on the frequency of a large cluster found. However, the Brown representation seems to have a generally increasing trend.

5 CONCLUSIONS

CountVect is the most efficient in query time and storage space, especially as the size of the dataset increases. TF-IDF is very close in query time to Countvect, but as the number of tweets increases, the storage space increases much more than for Countvect. Therefore, a tradeoff exists for TF-IDF, as it includes information about the importance of each word in each tweet. Brown is like TF-IDF in the space it takes up, though it is significantly slower in querying. That is the tradeoff for being able to query using semantic meaning. Based on these results, each representation is useful in different scenarios. CountVect would be best if one wants to query for every instance of a specific word, as importance or semantics of the word is irrelevant. If a researcher wants to analyze usage of a particular word, they could use CountVect to extract all tweets that contain that word. TFIDF is best used if the importance of a word is relevant, like if researchers want to only extract tweets where a word is above a certain importance threshold. Brown is best used if semantics are relevant, like if researchers want to retrieve all tweets that contain a word or any word that is a synonym/is closely related to that word.

6 FUTURE WORKS

We would ideally prefer to run these tests in a more controlled environment with more extensive computing power. The system used for these tests fluctuated throughout the testing period and could not effectively handle larger queries. A larger dataset for testing would be preferable to mirror potential research environments more accurately, as Twitter datasets could be massive. Also, more testing on different vectorizations/query methods could be done, for example, for skip-gram word associations. Our research could aid researchers using large datasets of short text documents, helping them choose the most efficient one for their purpose.

Individual Contributions:

Jerry

- Coding tests and results
- Writing sections 4-6

Akansh

- Poster Presentation
- Writing sections 1-3.1 and editing whole document

Derrick

- Writing section 3
- Coding each text representation

Bibliography

[1]Ruby, D., & About The Author Daniel Ruby Content writer with 10+ years of experience. I write across a range of subjects. (2022, December 3). 57+ eye-opening Twitter statistics for 2023 (Infographics and data). demandsage. Retrieved December 10, 2022, from https://www.demandsage.com/twitter-statistics/#:~:text=Twitter%20Statistics%20(Top%20Picks)&text=Twitter%20has%20around%20450%20million,users%20in%20the%20United%20States

[2] Obar, Jonathan A.; Wildman, Steve (2015). "Social media definition and the governance challenge: An introduction to the special issue". Telecommunications Policy. 39 (9): 745–750. [Online] [Accessed 04-Dec-2-22]

[3]Anonymous et al,"GLOBAL SOCIAL MEDIA STATS." DataReportal – Global Digital Insights, 2022, datareportal.com/social-media-users. [Online] [Accessed 04-Dec-2-22]

[4] Michael Busch. "Ealrybird: Real-Time search at Twitter" "https://ieeexplore.ieee.org/abstract/document/6228205" [Online] [Accessed 04-Dec-2-22]

[5]Marina Litvak and Mark Last. 2008. Graph-based keyword extraction for single-document summarization. In Proceedings of the Workshop on Multisource Multilingual Information Extraction and Summarization, MMIES '08, pages 17–24, Stroudsburg, PA, USA. Association for Computational Linguistics.

[6]Costas R, Van Honk J, Franssen T. Scholars on Twitter: who and how many are they? Int Conf Sci Inf. 2017; 15: 224- 235. [Online] [Accessed 04-Dec-2-22]

[7] Natural language processing with disaster tweets. Kaggle. (n.d.). Retrieved December 10, 2022, from https://www.kaggle.com/competitions/nlp-getting-started/data

[8]Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In Advances in Neural Information Processing Systems, pages 3111–3119. "https://papers.nips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html" [Online] [Accessed 04-Dec-2-22]

[9]David Sayce. (2022, August 16). The number of tweets per day in 2022. David Sayce. Retrieved December 10, 2022, from https://www.dsayce.com/social-media/tweets-day/#:~:text=Every%20second%2C%20on%20average%2C%20around%206%2C000%20tweets%20are%20tweeted%20on,200%20billion%20tweets%20per%20year.

[10] Kietzmann, Jan H.; Hermkens, Kristopher (2011). "Social media? Get serious! Understanding the functional building blocks of social media". Business Horizons [Online] [Accessed 04-Dec-2-22]

[11]Zhenhui Li, Ding Zhou, Yun-Fang Juan, and Jiawei Han. 2010. Keyword extraction for social snippets. In Proceedings of the 19th International Conference on World Wide Web, WWW '10, pages 1143–1144, New York, NY, USA. ACM.

[12] Lu´ıs Marujo, Wang Ling, Isabel Trancoso, Chris Dyer, Alan W. Black, Anatole Gershman, David Martins de Matos, Jo˜ao P. Neto, and Jaime Carbonell. "Automatic Keyword Extraction on Twitter" "https://aclanthology.org/P15-2105.pdf" [Online] [Accessed 04-Dec-2-22]

[13] Derczynski, L., Chester, S., & Bøgh, K. S. (1970, January 1). *[PDF] tune your brown clustering, please: Semantic scholar*. undefined. Retrieved December 10, 2022, from https://www.semanticscholar.org/paper/Tune-Your-Brown-Clustering%2C-Please-Derczynski-Chester/472f19f49cb40acfe34f79eb784154edca79e0a8