Semantic Search Algorithm

Daniel Stern

April 4, 2022

Introduction

The New World Order problem indicates that trends and online discourse have become increasingly important since the onset of Covid-19 for social, political, and economic activity. As a solution, I have implemented a semantic search engine in Python 3.7, 'semantic.py', which searches keywords contained in Twitter and Reddit posts.

The final solution is the function, combined_search(), which takes a query as its input and returns a list of related words, in order of semantic similarity to the search term. The solution is a combination of two search algorithms, each with a different method of constructing word embeddings.

Each of the two algorithms incorporates a measurement of recency for the semantic associations in the construction of the word embeddings. Both methods ultimately compute the cosine similarity of the query with all other unique words in the corpus, scoring and ranking the search results.

Many words in the corpus are not unique. Stop words appear frequently and should be avoided as search engine output. A solution to the uniqueness problem is the use of the cosine similarity function, which finds commonalities between word meanings. It is more robust than a crude solution, such as a search engine which counts the frequency of the query's co-occurrence with all other words and outputs a sorted list based on that count. In the Results section, the cosine similarity function shows a considerable improvement over the crude method in removing stop words.

Twitter and Reddit posts are referred to here as "documents". Words, keywords, and terms, all refer to the words contained in the documents and are also referred to as "trends". The search engine output can be viewed as a ranked list of recently occurring trends, related to the query.

As the qualitative evidence shows, the combined search engine is effective in identifying trends related to an assortment of queries. In the Conclusions section, I discuss several opportunities for further research.

Data

The dataset used in this analysis is constructed from the two sources, nwo-sample.graph.tweets and nwo-sample.graph.reddit in GCP. The data are accessed in Python using the credentials in the provided JSON file.

As indicated previously, this analysis treats documents from Twitter and Reddit as identical. The dataset consists of two variables: document text and time stamp. The two variables are extracted for a random selection of 2000 Twitter documents and 2000 Reddit documents.

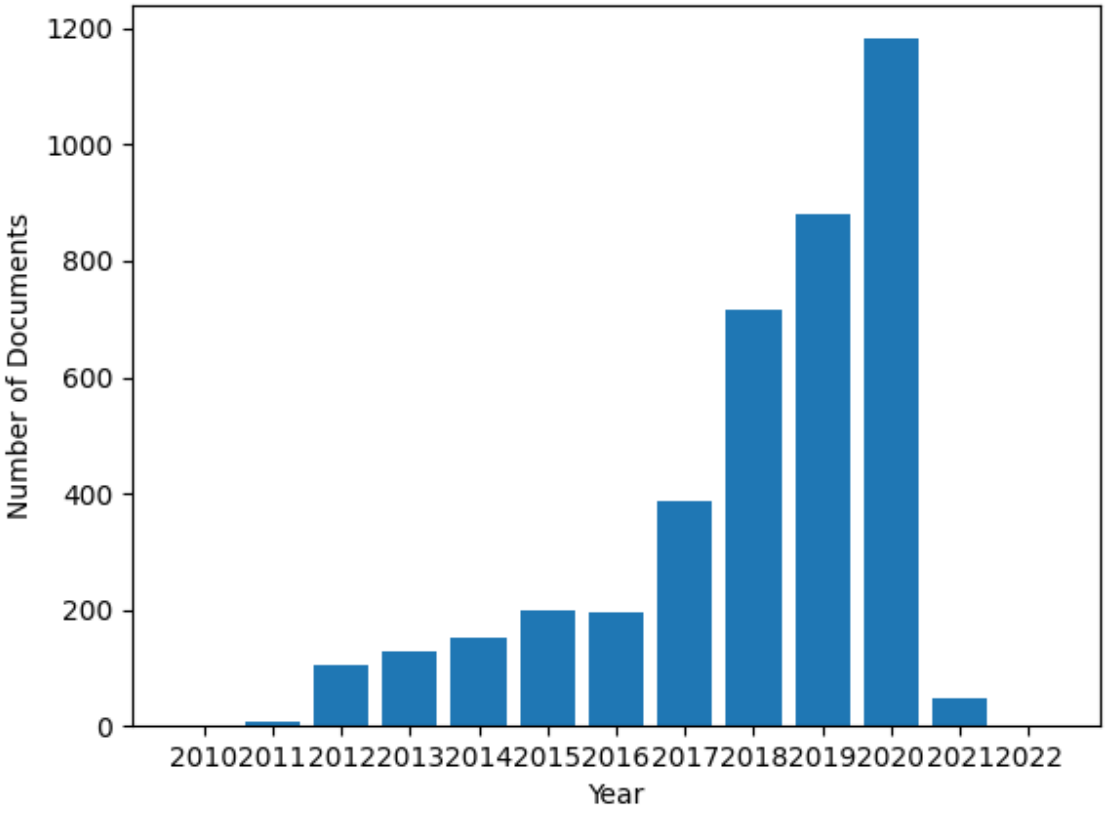
The document text is cleaned by removing all non-alphabetic characters. The words are then separated by spaces. The time data are stored as UTC timestamps.

Exploratory Analysis

The first document in the dataset occurred on 10/4/09, and the most recent document occurred on 1/18/21. There were 17,205 unique words across all 4000 documents.

Figure 1 shows the distribution of documents between 2010 and 2022, with the greatest number of documents occurring in 2020.

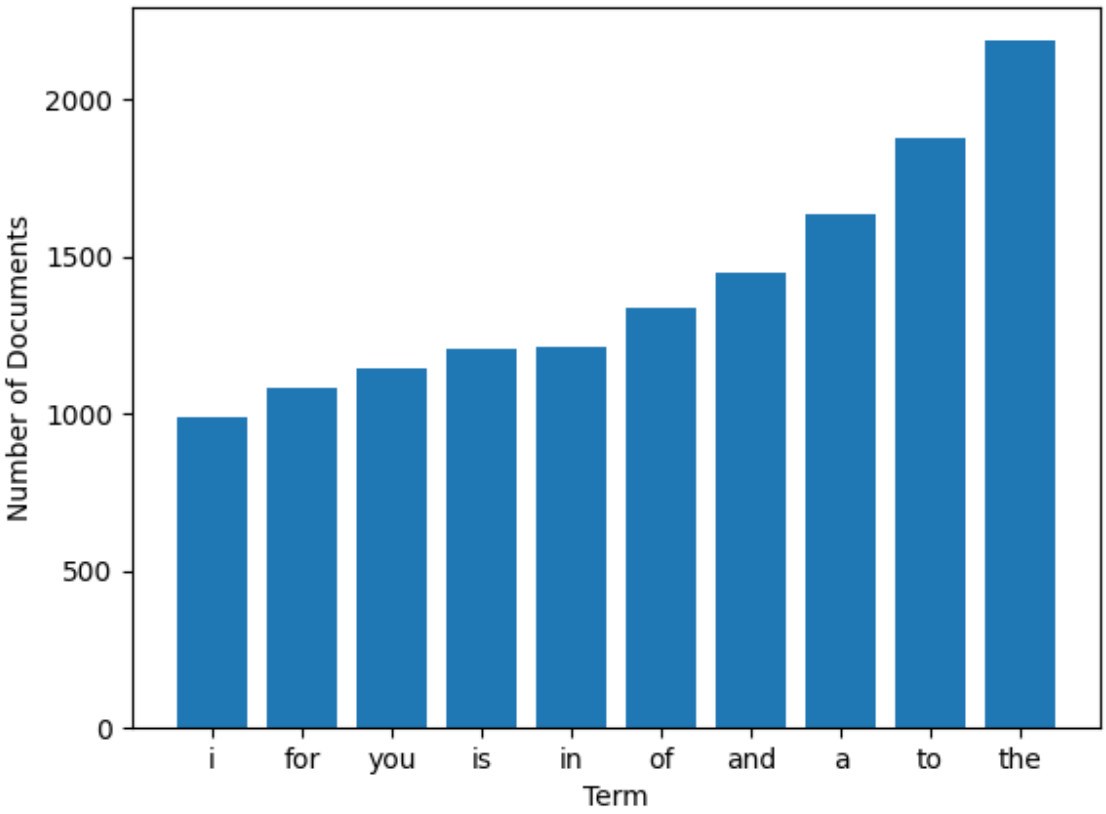
Figure 1:
Number of Documents by Year (N = 4000)



The recency adjustments to the data are calculated based on the most recent document date, as further described in Results.

Figure 2 shows the top words in the corpus ranked by appearance in the greatest number of documents. The word, “the”, occurs in the greatest number of documents, followed by other common stop words.

Figure 2:
Number of Document Occurrences (N = 4000)



Results

A baseline search algorithm is constructed simply by ranking the co-occurrences of trends with the query. The baseline result does not solve the uniqueness problem, as it contains several stop words. It is not a very effective solution for semantic similarity.

Query: ‘iphone’
Baseline Output (cut off at top 10):
['to', 'the', 'in', 'for', 'that', 'from', 'years', 'of', 'a', 'or']

The first of the two algorithms comprising the combined search solution constructs word embeddings by storing an indicator variable for each of the 4000 documents, for each unique word in the corpus. For example, the embedding for ‘iphone’ stores 4000 indicator variables, equaling 1 (+ recency boost) if the document contains ‘iphone’ and 0 if not.

The indicator variables are modified to incorporate recency. A recency boost is constructed by the rule: 2 if the document occurs within three months of the most recent document in the data, 1 if it occurs within six months, and 0 if it occurs beyond six months. The boost is then added to the original indicator variable.

Query: ‘iphone’
Occurrence Indicators Output (cut off at top 10):
['unveils', 'obscure', 'fastcompany', 'innovationfastcompanycommo', 'att', 'stops', 'hourly', 'moments', 'charging', 'steve']

The second algorithm constructs word embeddings by storing a count of how often each word co-occurs with each other unique word in the corpus. The lengths of the embedded vectors are equal to the number of unique words in the corpus. The same recency boost (2 if three months, 1 if six months, 0 if beyond) is added to the frequency counts.

Query: ‘iphone’
Co-occurrence Count Output (cut off at top 10):
['phones', 'affordthey', 'alltheres', 'taxi', 'motorbike', 'everythinganother', 'argos', 'sells', 'gigantic', 'stuffits']

The combined search result adds the cosine similarities across all results for the two algorithms, with equal weights. The combined output is similar to the output of the two previous methods.

Query: ‘iphone’
Combined Output (cut off at top 10):
['unveils', 'obscure', 'fastcompany', 'innovationfastcompanycommo', 'affordthey', 'alltheres', 'taxi', 'motorbike', 'everythinganother', 'argos']

Unrelated terms rank very low. In the Combined Output for ‘iphone’, the term ‘biden’ ranked 688, and ‘perfume’ ranked 16,511.

The top search result is ‘unveils’. The most recent document containing the terms ‘iphone’ as well as ‘unveils’ is the following:

“
From the obvious (Steve Jobs unveils the iPhone) to the obscure (AT&T stops charging an hourly rate for internet access).

25 moments in #tech that defined the past 25 years. via @fastcompany #innovation fastcompany.com/90565059/25-mo…
”

The tweet occurred on 10-22-2020, giving it the highest recency boost. It appears that Steve Jobs’ original iPhone unveiling was trending.

Additional results from the Combined Search are as follows:

apple: ['fang', 'suspiciously', 'weekaapl', 'baba', 'bidu', 'googl', 'nflx', 'tsla', 'twtr', 'httpswwwmicrosectorscom']

biden: ['happen', 'wrote', 'abuse', 'unemployed', 'responsibility', 'lifestyle', 'dough', 'claimshey', 'desired', 'conspiring']

china: ['chinese', 'chinas', 'remains', 'sentiment', 'discussion', 'downvoted', 'anywhere', 'numbers', 'stands', 'tracing']

tesla: ['commandeered', 'weekaapl', 'baba', 'bidu', 'googl', 'nflx', 'tsla', 'twtr', 'httpswwwmicrosectorscom', 'pictwittercomnqenisiub']

tv: ['schtick', 'loudmouth', 'manner', 'freedman', 'elliot', 'affordthey', 'alltheres', 'taxi', 'motorbike', 'everythinganother']

Conclusions

The Occurrence Indicator embeddings associate words that appear in the same documents. This strategy is useful for identifying trends where the associated words do not have a similar semantic meaning.

The Co-occurrence Count embeddings associate words that appear alongside a similar set of other words. This is a useful strategy for associating words with a similar semantic meaning.

The Combined Search gives equal weight to each strategy, accounting for trends with similar semantic meaning, as well as trends that do not share linguistic traits. In Results, the top result for ‘iphone’ can be compared across the three methods. In the first, ‘unveils’ suggests that an unveiled iPhone is trending. In the second, the result ‘phones’ indicates that iPhone and phones have a similar semantic meaning. The combined search keeps ‘unveils’ as the top result.

Further Research

Areas of improvement for further research can be found in the data construction process. The SQL Rand() function does not take a seed value when used with GCP. As a result, this exact analysis is not replicable. If accessing the data directly with Python, rather than through SQL, the np.random.seed() function can be used. Otherwise, an alternative method should be found in SQL.

To research the efficacy of various search algorithms, a test data set should be extracted and sequestered from the start of the research process. Even though the machine learning task is entirely unsupervised, human decisions about which models work best and which model parameters should be used, are all influenced by performance on the training data.

The recency boosts used can be tuned for desired output. The variable constructed (2 if three months, 1 if six months, 0 if beyond) is arbitrary and should be explored further. The training data can be used to provide qualitative results with a search engine, by modifying both the recency reward as well as time frames considered for recency. The training data can even be split further into training and validation data, so as not to contaminate the test data for the final research conclusions.

Weights used for the combination of the two search algorithms can similarly be explored, using the training data. In this analysis, the two algorithms are weighted equally in the combined solution. Further research should first examine the benefits and drawbacks to each embedding and provide a detailed explanation of the circumstances under which each embedding should be used. Weights for the two algorithms can then be tested based on those criteria.

The “spaCy” Python library can be incorporated into the combined search algorithm. Spacy embeddings provide semantic information that has been trained on data outside of the documents considered here. The spaCy embeddings can be weighted, along with the two algorithms used in the combined search, to supplement the association of trends with additional linguistic information.

This analysis considers queries and results that only consist of one word. Since the goal is to find trends, it would be useful to analyze terms that consist of multiple words. Tools such as spaCy can readily analyze trends with multiple words and provide insight into other linguistic characteristics. Other areas of application include removing non-words in the word cleaning process and identifying multiple forms of the same trend, such as China or China’s.