Tom Halloin

Springboard data science career track

Capstone Project 2: Analysis of Berkshire Hathaway Shareholder Letters Using Natural Language Processing (NLP) Techniques

“Read 500 pages every day. That's how knowledge works. It builds up, like compound interest. All of you can do it, but I guarantee not many of you will do it.”

- Warren Buffett, when asked how to prepare for an investing career.

# The Problem

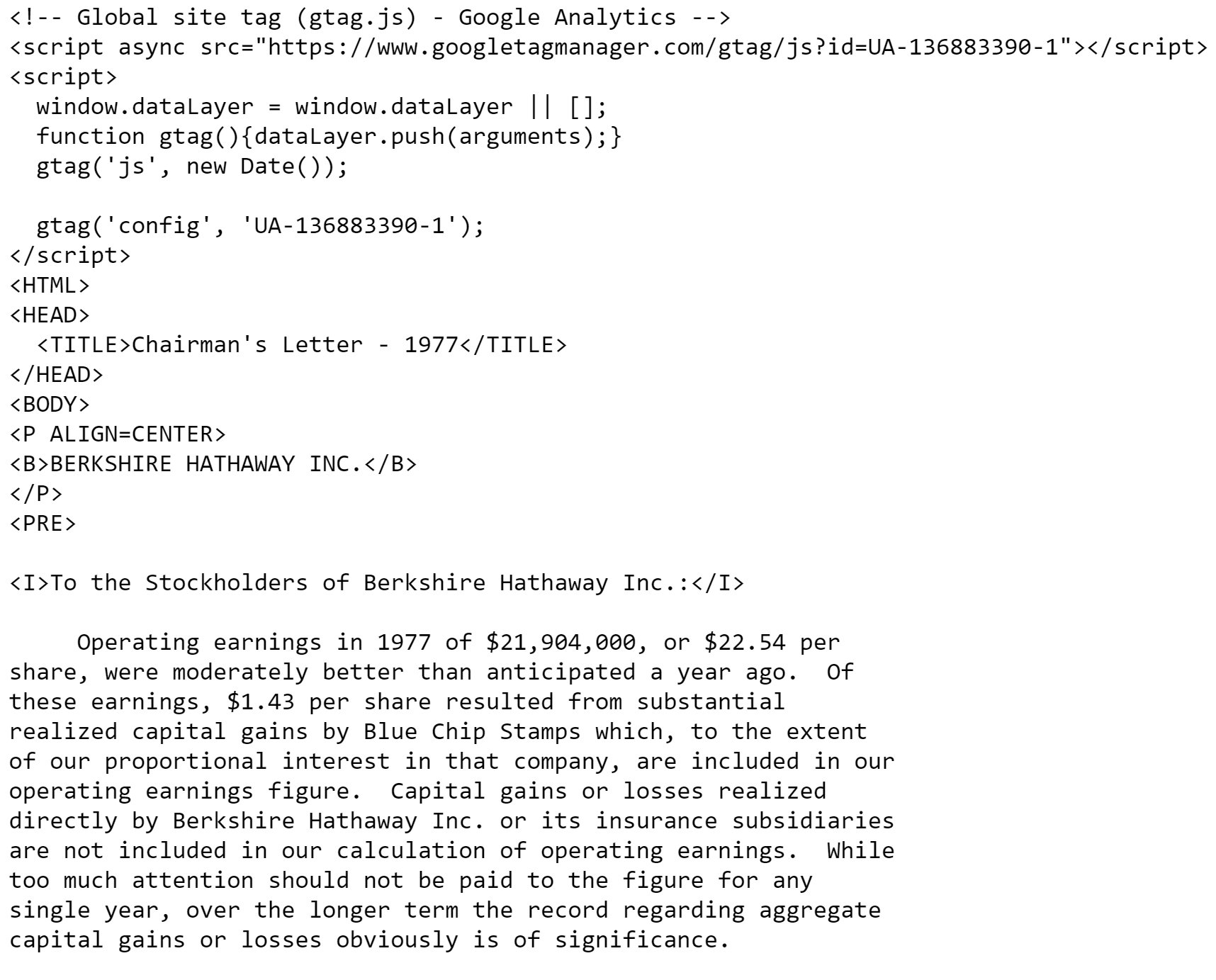
Warren Buffett became one of the most successful investors of all time because of his ability to comprehend thousands of pages of data. Buffett is the exception to the rule. Most people don’t have the patience to read 500 pages a day, let alone comprehend it (and I am not the exception). Instead, I am going to use machine learning to summarize the equivalent of reading 500 pages, Warren Buffett’s annual letters to shareholders since 1977. This capstone will be an exploration of Berkshire Hathaway’s annual shareholder letters. Using Natural Language Processing, I will try to summarize the letters and create a topic model to discover hidden themes that are present across the letters. This model will try to extract meaning from these letters by identifying recurring themes or topics.

# The Dataset

The data comes from Berkshire Hathaway’s shareholder letters available on their website. The letters come in both HTML and PDF format. Using Python, I managed to scrape the text from annual letters dating back from 1977. See the code in the notebook linked here for more information. Note that Berkshire eventually caught on that I was a bot, and therefore denied me access to their reports. Fortunately, this was after I gathered all of my data, so while the code used to work, there is no way for it to work anymore without access. The letters are also saved as part of this Milestone Report.

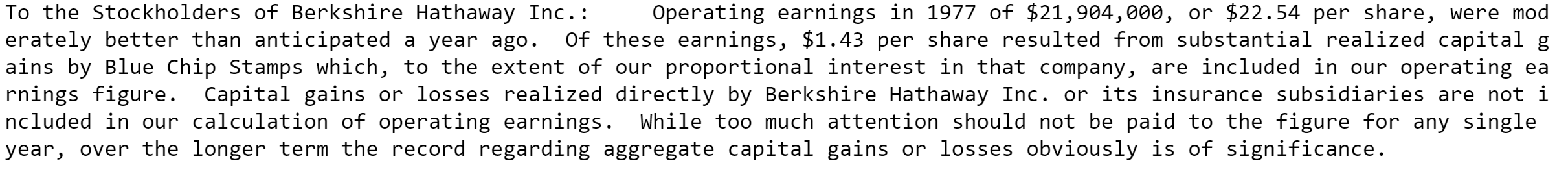
# Data Wrangling

Before doing any analysis on the letters, I needed to do some data wrangling to clean the output from the pages and PDFs the letters came from into useable text. Before any data wrangling, the files looked like the following sample.



Most of the cleaning for these files included removing HTML, anything before and after the actual contents of the letter, and non-ascii characters. A series of regular expressions removed unnecessary characters in tables and newline characters to make the text more readable. I skipped removing stop words and numbers because I want the ending summaries to read as close to actual sentences as possible.

The end result was a series of paragraphs in documents looking like this:

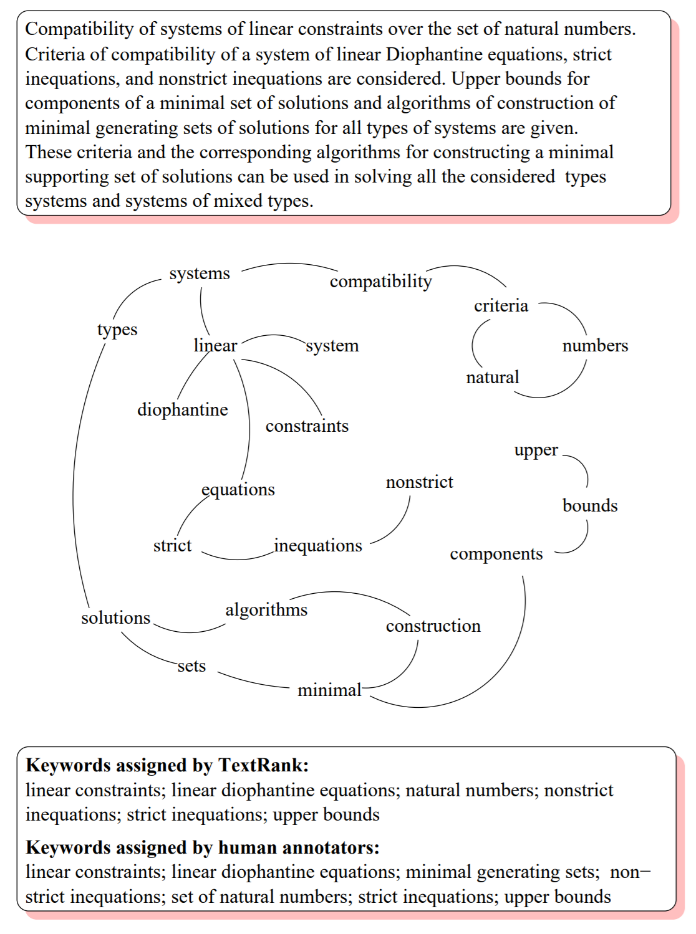


# Summarizing the Data

Word clouds provide a quick overview of text data. They take a group of words and create an image. The word cloud below represents the most common names of people and organizations featured in the annual letters. Berkshire appears multiple times, along with Charlie (Munger), and several of Berkshire’s subsidiaries in Geico, Nebraska Furniture Mart (NFM), and NetJets.



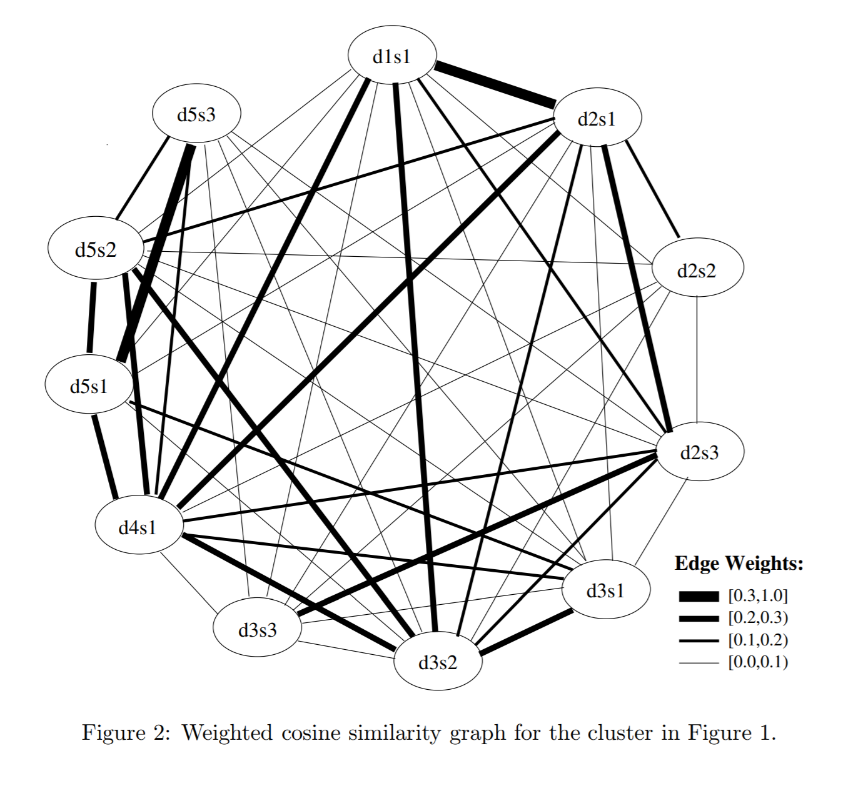
Word clouds provide a starting point for the actual “deliverable” – summaries of the letters. There are two ways to summarize data: *extractive* summarization and *abstractive* summarization. *Extractive* summarization selects the most important phrases from the text and extracts it to get the most important phrases. *Abstractive* summarization creates sentences from the text using deep learning and returns these new sentences as the summary. Computers are really good for doing extractive summarization, but abstractive summarization is much harder because it involves creating new text to explain the document so that humans can understand it. There have been attempts with deep learning to create abstractive summaries, but that is beyond the scope of this project.

I will try three different extractive summarization methods: Textrank, Lexrank, and LSA. [Textrank](https://web.eecs.umich.edu/~mihalcea/papers/mihalcea.emnlp04.pdf) is a graph-based ranking model similar to Google’s Pagerank algorithm. Each word links to other words based on the number of times the words appear near each other. The algorithm then assigns hidden weights to each of these links given how likely each of the linked words appears with each other. Suppose in the example to the right I start with “natural”. Natural is linked with “criteria” and “numbers”. Based on Textrank’s weighting, however, it will determine that “natural numbers” is a better keyword than “natural criteria”. The weighting itself can be a black box – see *Textrank: Bringing Order into Texts* by Rada Mihalcea and Paul Tarau for more details on the mathematical implementation.

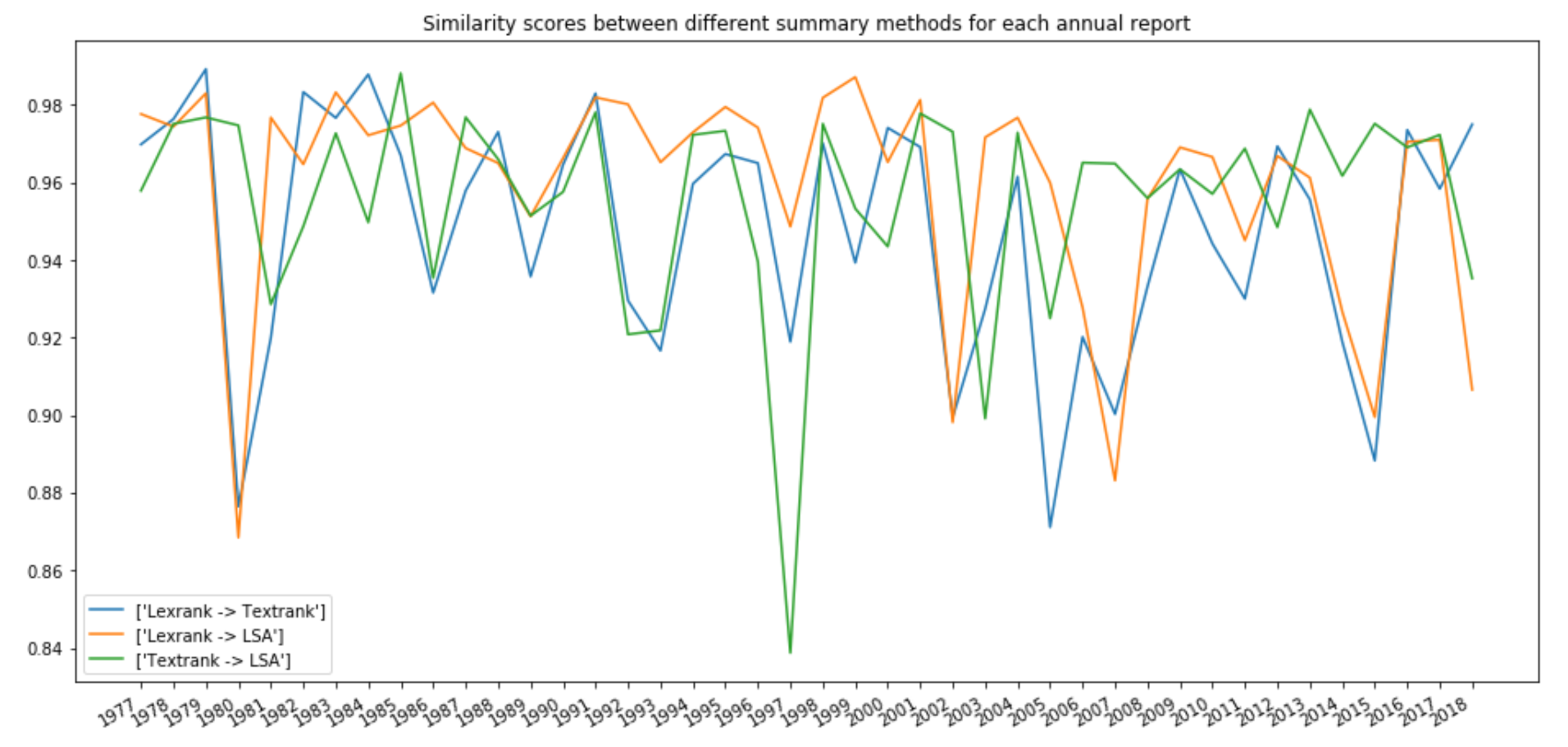
In the example shown above, connected words are good examples of keywords that can summarize documents. Examples of summarization of the text on the right include “linear Diophantine equations”, “upper bounds”, and “natural numbers”. Without any comprehension of the actual text, the graph shows that the text is at least about these three keywords. The algorithm will then look for sentences with these keywords to summarize the document.

[Lexrank](https://www.cs.cmu.edu/afs/cs/project/jair/pub/volume22/erkan04a-html/erkan04a.html) is very similar to Textrank. Both use the same graph-based weighting model, but differ in the method of generating weights. Lexrank uses a method of cosine similarity weighted by term frequency – inverse document frequency, otherwise known as TF-IDF, to weight terms that have more meaning instead of just looking at document frequencies. The end result of applying the Lexrank algorithm to a series of documents is a graph similar to the picture on the right here. Each circle is a document and the edges represent the cosine similarities between each of the documents. In this example, d1s1 (document 1, sentence 1) and d2s1 (document 2, sentence 1) have a high weighted cosine similarity and would be good candidates for summarizing the corpus of documents.

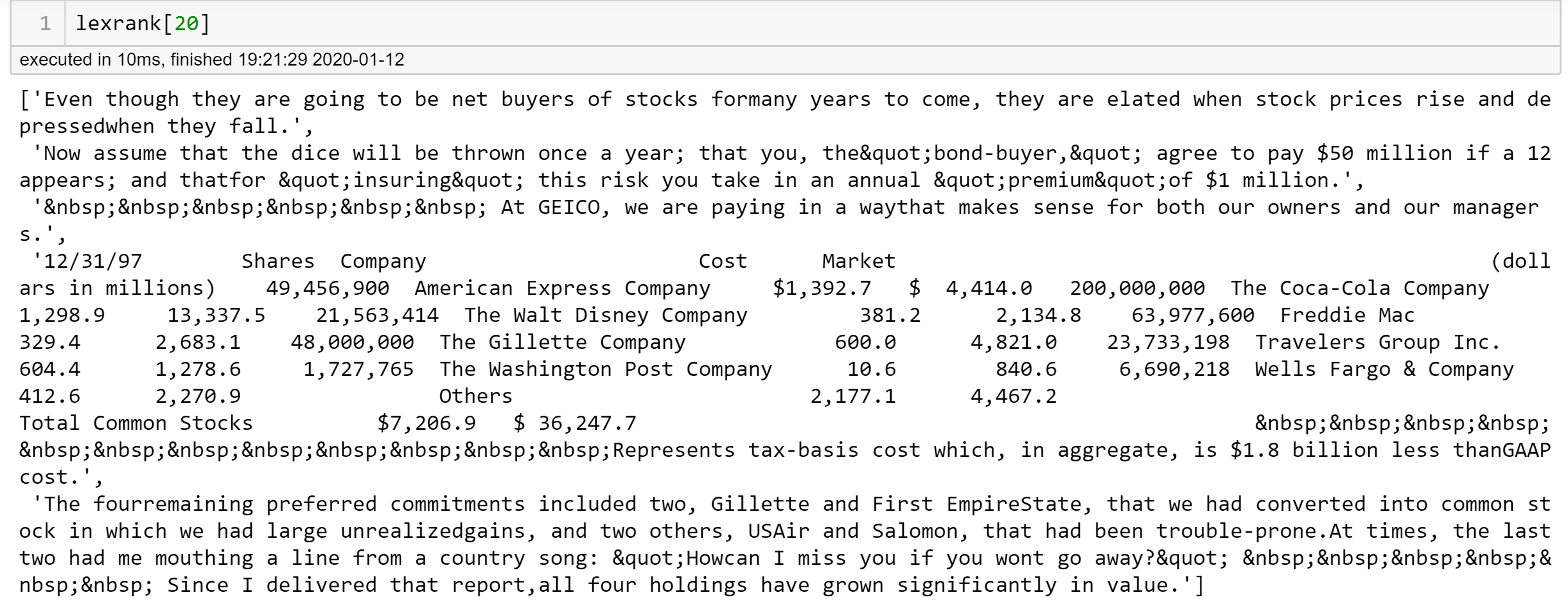
See LexRank: Graph-based Lexical Centrality as Salience in Text Summarization by Gunes Erkan and Dragomir R. Radev for graph.

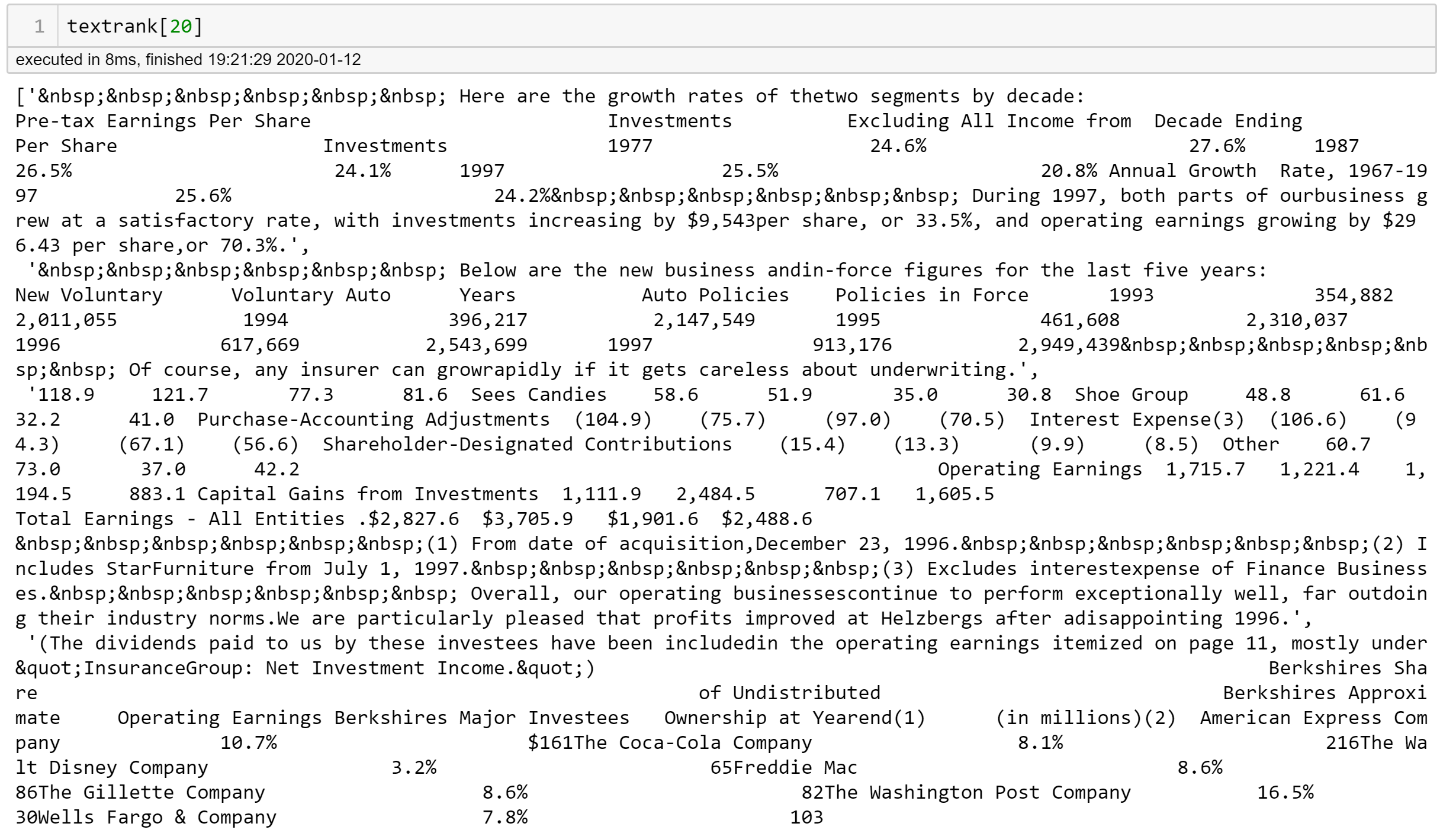
 [*LSA*, or *Latent Semantic Analysis*](http://www.kiv.zcu.cz/~jstein/publikace/isim2004.pdf), analyzes similarities between the meanings of a series of documents and produces a series of concepts related to the documents. Like Lexrank and Textrank, LSA creates a term-document matrix consisting of word frequencies for each term in each document. Each word is a row while each document is a column. After creating this matrix, a linear algebra technique called singular value decomposition reduces the number of words, while keeping the same meaning in each of the documents. By reducing the number of words in each document, it is easier to find similarities between different documents. Sentences that keep the most words from the reduction are considered the “most important”. These reduced words and phrases also generate topic models that will be shown later.

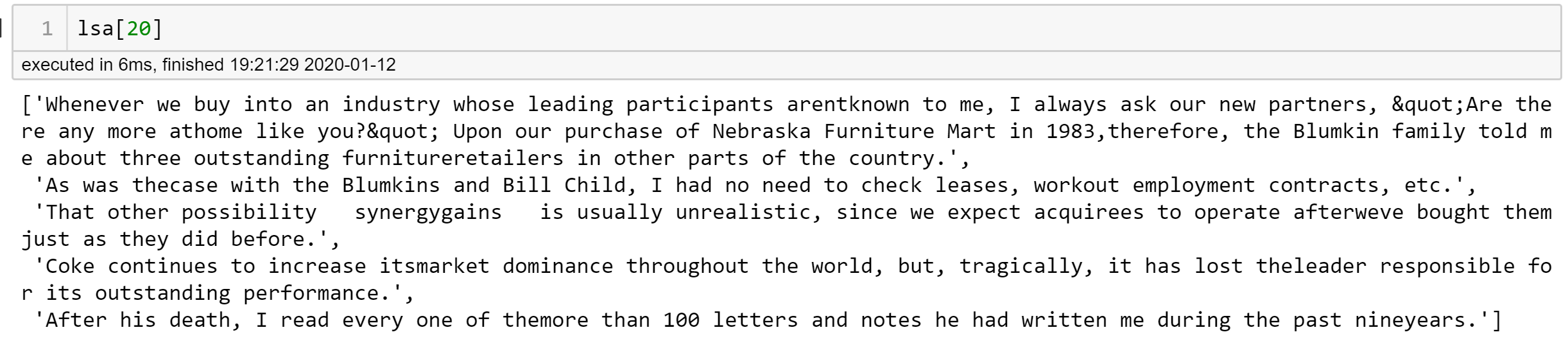
For each annual report, I used each of these three techniques to summarize each year in the corpus by extracting the five most meaningful sentences according to its method. Then, I used the Spacy library to create a summary score to see how similar the summaries are to each other. Spacy has a large vocabulary of words that can capture similar phrases even if the words do not match, so the techniques should produce similar summaries according to Spacy. The below graph plots these similarity scores between the three different methods. Each line is a comparison between two of the methods for each year in the annual report. The divergences, as shown in the year 1997, provide a good contrast to see the different sentences each technique deems as important.



Here are the summaries for each method in 1997:







I think this provides a pretty good contrast on the goals of the summarizers. Lexrank and Textrank pick sentences that contain many key terms throughout the document, such as the company names, but I would not pick those sentences if I wanted to summarize the entire annual report. The sentences in the LSA summary are much cleaner. They don’t have the dreaded non-breaking spaces or sentences that are actually part of a table. Content wise, they catch the big events: the acquisitions of Star Furniture and Dairy Queen, along with the death of Coca-Cola’s CEO. For this dataset, LSA catches the most signals and ignores the most noise.

# Topic Modeling

Milestone Report 2: Topic Modeling

Topic Modeling

* Rationale for topic modeling, how it differs from summarizing
* Differs in output, output in TM is clusters and latent embedding.
* TM works on entire corpus.
* Summarization is for each individual document.
* Different types of topic modeling, tools used, expectation for results
* Applying topic modeling to corpus
* Measuring coherence of topic models

Recommendations

// Treat this as a journal paper

\*It would be a shame to have all of this knowledge lost\*

\* provide quick details, add reference

\* provide rationale for each decision

Make an outline

Make word document to fill outline

PowerPoint is simplified version of outline

Thursday -> have PPT ready

Sunday -> have Word Document ready