Text as Data

Hossein Kazemi
Senior Advisor to the CAIA Association & the FDP Institute
Michael & Cheryl Philipp Professor of Finance, Isenberg School of Management

February 2020 Version 4

The spam folder of my email account at the University of Massachusetts contains this recent email:

> Subject: [SPAM: 93%] Check your Eligibility for a Complimentary Cellphone
>
> Dear Sir/Madam
> After a comprehensive review, it has been determined that you may be eligible for a free cellphone. Go here to claim your free cellphone.

The university's email server has determined that there is a 93% probability that this email is spam and sent it to my spam folder, preventing me from getting my free cellphone. ☺

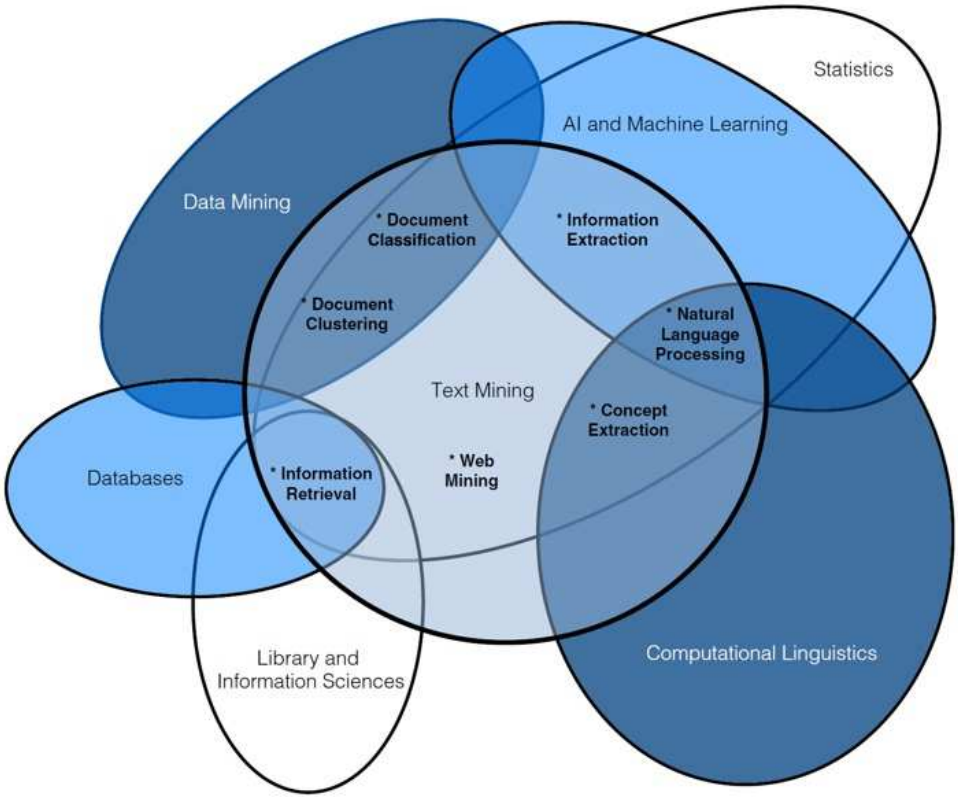
How did the email server assign such a probability to this email? Surely, UMass does not have thousands of employees reading faculty emails, deciding the probability that an email will be spam or not. The server uses a text-mining algorithm to make that determination. In this note, I will discuss the very basic steps of text mining and will explain how the server came up with a 93% probability.

Mining text is difficult. Text represents unstructured data. This reflects the fact that text does not appear in a structured form that we can present using an Excel table with columns and rows having specific meanings. Of course, text does have a structure, but it is not of the type that can be fed into an algorithm for analysis. The structure of a text is linguistic, which has evolved through time so that the written or spoken words can be understood by humans and not computers. Therefore, text must be pre-processed before it can be fed into a computer algorithm.

Among social sciences, finance and economics have been lucky in having access to lots of structured data. We have national accounting data, stock market data, accounting data, and so on. Having access to so much structured data, led economists and financial analysts to appreciate the usefulness of text and other unstructured data rather late in comparison to other areas of social science such as political science. However, the search for information advantages by investors and analysts has created an explosion of activities and discoveries in

the applications of natural language processing (NLP), and, in particular, textual analysis in the financial industry.

NLP and text mining are highly complex and specialized areas that require researchers to bring a variety of expertise into a text mining project. The following Venn diagram gives a visual description of these sets of skills.



However, it is not difficult to present a very brief and high level of description of NLP and text mining. The purpose of this note is to present such a description. The goal is to familiarize the reader with some of the terminology and the basic steps employed in a text mining project.

NLP vs. Text Mining

While NLP and text mining are different, they have much in common. Some sources consider text mining to be a subset of NLP, while others consider them to be highly related but not a subset of each other. In this note, I will focus on text mining. For the purpose of this note, we define text mining as an artificial intelligence technology that uses NLP to transform the unstructured text in documents and databases into normalized, structured data suitable for analysis using machine learning algorithms.

According to Linguamatics, a provider of an NLP based AI platform, text mining identifies facts, relationships, and assertions that would otherwise remain buried in the mass of textual big

data. Once extracted, this information is converted into a structured form that can be further analyzed or presented directly using tables, mind maps, charts, etc. Text mining employs a variety of methodologies, including NLP, to process the text. The structured data created by text mining can be integrated into databases, data warehouses, or business intelligence dashboards and used for descriptive, prescriptive, or predictive analytics. NLP, therefore, is a much broader area (e.g., includes voice recognition), which aims to enable machines to understand and analyze natural human language.

Text Mining

One of the earliest text mining papers in the area of finance is Niederhoffer (1971). In this study, Niederhoffer manually "mined" the headlines appearing in the New York Times. He ranked the headlines in terms of their sentiments and importance. He found out that large changes in stock prices followed days with important world events. Also, the sentiment of the headline seemed to have some predictive power, but the effects were small and did not last very long.

Text mining has applications in many disciplines. One of the most famous and influential studies was done by Mosteller and Wallace (1964), where textual analysis was employed to determine the authorship of some of the Federalist Papers when historians could not agree whether Hamilton or Madison had written them (It turned out that it was Madison).

Businesses have come to appreciate the power of text mining to increase revenue, reduce cost, and mitigate risk. For instance, recently, Admiral, a U.K. based insurance company, offered discounts to its car insurance clients if they agreed to give the company access to their Facebook profile. Admiral had come to realize that through text mining of Facebook pages of its customers, it could improve its underwriting practices. Admiral had developed an algorithm that analyzed its customers' posts, and by analyzing the style of writing of each user, the algorithm could uncover positive and negative traits. The company believed that these personal traits could be used as predictors of a customer's driving behavior. Facebook, however, refused to give the company access.

These three examples highlight different applications of text mining. Niederhoffer used text mining to determine if the sentiments can predict the stock market. Mosteller and Wallace used text mining to explore, understand, and classify the text. Admiral attempted to use text mining to learn about the personal traits of the authors of Facebook posts. However, regardless of the goal of the study, the researcher must develop a numerical representation of the text since computers and analytical techniques can only be applied to numerical values.

The starting point of text mining is a document containing text. For example, when working with a database consisting of thousands of emails, we could refer to each email as a document. If the goal is to determine if a document (email) should be classified as spam or not, then we

need to develop a numerical representation of the text, which is then used to classify the document as spam or not.

A Simple Spam Detector

Suppose an email server has received millions of messages over the last decade and wants to develop its spam filter. The IT department asks all users to mark emails that they consider to be spam. After several months, the IT department will have a large sample of emails labeled as spam or not spam. The data can then be used to train an algorithm, which, when fed a new email, will be able to assign a spam probability to the email. But how do you feed an email to an algorithm?

The first step is to breakdown each email (i.e., document) into a series of tokens. For example, the Python package "Typing" will read a text and break it down into its parts:

tokenize("for a complimentary") -> { "for", "a", "complimentary"}

Next, we create a table where the columns refer to the above 3 tokens, and the rows refer to each email. The cells will tell us how many times the words "for", "a" or "complimentary" appeared in each email. Here, we have the tabulated information for emails #1459 and #1460.

Table 1

Documents	for	a	complimentary
1459	6	4	1
1460	10	3	0

Next, we count the number of times that each word appears in the emails that are in our training set while noting if the email was marked spam or not. Notice that we are ignoring the sequence of these words as they appear in an email. That is, we do not distinguish between "for a complimentary" and "complementary for a." Also, we may decide to ignore upper and lower cases and consider them to be the same.

A summary of our counting results may look like this.

Table 2

Type	for	a	complimentary
Spam	99%	97%	60%
Not Spam	98%	95%	5%

We can see that words "for" and "a" appear in almost all emails. So, they are not useful for determining if an email is spam or not. However, we see that 60% of spam emails contain the

word “complimentary”, while only 5% of nonspam emails contain this word. Therefore, if a new email arrives with the word “complimentary” in it, the server will assign the following spam probability to it:

*Probability(Spam **Given that it contains** “complimentary”) = 60%/(60% + 5%) =92%*

The probabilistic algorithm used above is called Naïve Bayesian. In this very naïve example, we used a “bag-of-words” approach based on N-gram (n=1) tokenization. The bag-of-words approach assumes that documents are just random collections of words drawn from the set of all available words. Therefore, it ignores the structure and placement of the word.

N-gram tokenization indicates the degree of granularity of the tokenization process. Consider the phrase, “After a comprehensive review, it has been determined.” The tokenization can be performed with different levels of granularity:

n=1 (unigram) returns: {“After”, “a”, “comprehensive”, “review”, “it”, “has”, “been”, “determined”}

n=2 (bigram) returns: {“After a”, “a comprehensive”, “comprehensive review”, “review it”, “ it has”, “has been”, “been determined”}

n=3 (trigram) returns: {“After a comprehensive”, “a comprehensive review,” ….)

It seems obvious that using higher values of *n* will force our algorithm to account for the placement of words and the structure of sentences. However, this will come at the expense of having a much smaller training set. Very few spam emails may include the phrase “After a comprehensive review it has been determined” and we may never see another email with exactly the same words in the future.

The previous discussion was an oversimplified approach to document analysis. In a more sophisticated text mining application, the key point is to understand how words and punctuations appear together to express certain meanings. The incorporation of grammar rules in the text analysis is certainly important, but to create a workable representation of data, we need to simplify these issues.

Representing text data

To further illustrate how a document can be numerically represented, consider the following two documents, each consisting of one sentence:

1. Risk management is the key to asset management.
2. Alternative assets can help reduce downside risk.

The following table is a numerical representation of these two documents

Table 3

	risk	management	is	the	key	to	asset	alternative	can	help	reduce	downside
D1	1	2	1	1	1	1	1	0	0	0	0	0
D2	1	0	0		0	0	1	1	1	1	1	1

Using this simple mechanism, we have moved from unstructured data to structured data, which can now be analyzed through the applications of machine learning and data mining algorithms. This matrix is often referred to as the Document Term Matrix.

Punctuation could play an important role in a certain context (e.g., in twitter or email messages, we may include ":)"). To create a more parsimonious numerical representation of the corpus (i.e., collection documents), we can choose to reduce the number of features. For instance, in the above documents, "assets" and "asset" were considered to be the same. If every single word were to be included in our table, the dimension of the problem would quickly exceed the computational capability of our computers.

Even the simple representation presented above allows us to make some quick calculations. For instance, we can apply algorithms that measure the similarity/dissimilarity of tabulated data. This will allow us to measure how similar these two documents are. Earlier, we mentioned the application of text mining to the Federalist papers. The approach adopted in that research relied on the same basic principles. Interestingly, the hidden structures of those papers were uncovered by paying attention to punctuation and ordinary words that appear in most texts. Based on a training set that consisted of labeled papers, it was discovered that Madison and Hamilton had different styles in using punctuation and ordinary words regardless of the subject.

Tokenization, PoS Tagging, Stop Words, Stemming, and Lemmatization

We have already discussed tokenization in the previous section. It is a process that consists of breaking up documents (as words, N-grams, or phrases) into elements called tokens, which are used as input to text mining procedures.

After tokenization and depending on our interest, we may continue with a Parts of Speech (PoS) tagging. PoS refers to the process of tagging words within sentences into their respective parts of speech. By applying PoS tagging, we will be able to extract those parts of speech that are of interest to us. For instance, consider the sentence "Risk Management is the key," which has five tokens. Using a PoS tagger, we tag parts of speech to each of the tokens (e.g., {(Risk, Noun}, {is, Verb}, …).

After the tokenization process, we may remove punctuation and common words as well as combine similar words (e.g., "asset and "assets", or "investment" and "security"). The words

that are important to the construction of proper sentences (e.g., “the” or “and”) but carry no specific meanings are called “stop-words.”

After removing numbers, punctuation, and stop words, we can further simplify the numerical representation of our text using a stemming or a lemmatization process. For example, in many languages, words get transformed into various forms when being used in a sentence. For example, the word "go" might get transformed into "goes”, “going” “went” or “gone”. It is necessary to convert these words into their base forms, as they carry the same meaning. Stemming is a process that helps us in doing so. The most common stemming algorithm for the English language appears in Porter (1980).

Another common pre-processing step is lemmatization. The stemming process may lead to inappropriate results. For example, the word “producing” may be transformed into “produc,” which has no meaning. To avoid this problem with stemming, one could use lemmatization. In this process, an additional check is being made by looking through the dictionary to extract the base form of a word. However, this additional check slows down the process.

The basic goal of the processes mentioned above, and many others, is to produce an accurate numerical representation of a document while keeping the dimension of the numerical representation manageable.

Term Frequency and Inverse Document Frequency

Suppose we want to mine newspaper articles that focus on China. We are not looking for a specific term or phrase. Rather, we are looking for articles that are unusual and cover a somewhat unique story. A simple approach known as *Term Frequency and Inverse Document Frequency* (TFIDF) can provide a quick and yet powerful measure of the unusualness of thousands of articles. This is how it works.

The first step is to get a broad picture of each document (i.e., a newspaper article). For instance, we can count the number of times each token (i.e., a word or a phrase) appears in each article. This will not tell us much as there are two problems with using simple counts: (a) longer documents will contain more words, and (b) common words such as “the” are likely to have the highest count. To overcome the first problem, we can divide the count of each token by the total number of tokens that appear in each document. This is called term frequency. Still, common words such as “the” and “and” are likely to have the highest frequencies. To overcome the second problem, we need to identify the unusual tokens that appear in our corpus.

While the phrase “profit margin” is not considered unusual for articles from the business section, it will be unusual if it appears in stories from the sports section. To measure how unusual a token is, we can calculate the percentage of the documents that contain each token. For example, “profit margin” may appear in 60% of business section articles while it may appear

in only 3% of sports section articles. The measure used for this purpose is referred to as inverse document frequency (IDF) because it turns out that it is more helpful to use the logarithm of the inverse of the above percentage (e.g., log(1/0.60) for "profit margin"). Therefore, a token with a large IDF is considered unusual because it does not appear that frequently in our corpus.

The final step is to examine each document for the term frequency of the token that was deemed to be unusual. For instance, if "coronavirus" turns out to be highly unusual (has high IDF) but there are only a few documents for which "coronavirus" has high TF, then those articles will require further examination.

Using the results displayed in Table 3, we can see the TF of the token "management" in document 1 is 2/8 as the token "management" appears twice and there are 8 token in the document. IDF of the token "management" is log(2/1) since there are 2 documents, and the term "management" appears in 1 of those documents. Clearly, a common word such as "is" will have an IDF that is close to zero because it is likely to appear in all documents. For example, if there are 100 documents and the token "is" appearing in 99 of them, its IDF will be log(100/99)=0.01. On the other hand, a rather rare word such as "downside" is likely to have a large IDF. For example, if "downside" appears in only 2 of the 100 documents, its IDF will be log(100/2)=3.9.

Finally, if we multiply TF by IDF of terms appearing in a document, we can gain significant insights into the importance and uniqueness of that document. The measure Term Frequency - Inverse Document Frequency (TFIDF) is given below

*TFIDF = Term Frequency * Inverse Document Frequency*

We can see that it will have a high value in the case of a high term frequency and a low frequency in the collection. In other words, for a term that frequently appears in one document but rarely in other documents, it will have a high value for TFIDF. Thus, the document that contains that rare word must be special and unique.

Word List and Sentiment Analysis

Sentiment analysis is an important application of textual analysis. Suppose we want to measure the sentiments conveyed by the New York Times headlines over a period of several years. The goal is to find out if headlines with positive (negative) tones precede an increase (a decrease) in the stock market. A common approach in this area is to compile word lists that share common sentiments (e.g., positive, negative, neutral, uncertain, etc.). Using such lists, an analyst can count words associated with each sentiment that appear in each day's headline and create a measure that informs the analyst about its net sentiment.

In measuring the tone or sentiment of a document, analysts typically count the number of words associated with a given sentiment word list normalized by the total number

of words in the document. Therefore, higher proportions of positive words in a document indicate a more optimistic tone. To create such word lists, researchers rely on specialized dictionaries. An important step in the process is to decide which source (i.e., dictionary) should be used to identify and calculate the proportion of the targeted sentiments or attributes. For example, the Harvard General Inquirer offers a group of lists that are suitable for sentiment research in sociology and psychology. On the other hand, the dictionary produced by Loughran and McDonald (2011) has been used for research in accounting and finance.

One of the earliest dictionary-based sentiment research is Tetlock (2007), which analyzed the media sentiment and the stock market. The paper uses word counts in the Wall Street Journal's widely read "Abreast of the Market" column. Counts from each article measuring seven different sentiments based on the Harvard IV-4 psychosocial dictionary were then tabulated. Armed with the time series of the daily sentiment scores, Tetlock creates a single measure of overall sentiment that he calls the pessimism factor. Next, the author measures the predictive power of this pessimism factor and reports that high pessimism significantly negatively forecasts one-day-ahead returns on the Dow Jones Industrial Average. Since the publication of Tetlock (2007) finance and accounting paper has been using Loughran and McDonald (2011) dictionary as it is more suitable for research in these areas.

Conclusion

In this short note, I provided a very brief outline of text mining. The goal was to demonstrate how text (unstructured data) can be given a numerical representation (structured data). We also discussed the most common pre-processing steps that are taken to make the numerical representation accurate but manageable. Of course, this short note barely scratches the surface. Also, we completely ignored any discussion of dozens of algorithms that can be applied to the numerical representations. These algorithms along with insights from the Natural Language Processing field allow us to go much further in mining text data. For example, they could help us measure the sentiment conveyed by a set of documents (e.g., tweets related to a corporate event), which can then be incorporated in the investment decision (e.g., whether act on a merger announcement).

For those who are interested in learning about textual analysis, the following sources are recommended:

- Provost, F. and T. Fawcett. (2013). *Data Science for Business*. Sebastopol, CA: O'Reilly Media Inc, Chapter 10. This is the main textbook for Financial Data Professional program
- Einstein J. (2018). *Natural Language Processing.* https://github.com/jacobeisenstein/gt-nlp-class/blob/master/notes/eisenstein-nlp-notes.pdf

- Jurafsky, D. and J.H. Martin (2013). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* *https://web.stanford.edu/~jurafsky/slp3/edbook_oct162019.pdf*
- The Stanford Natural Language Processing Group, https://nlp.stanford.edu/

References

- Niederhoffer, V. (1971). "The Analysis of World Events and Stock Prices." *The Journal of Business*, 1971, Vol. 44, Issue 2, 193-219
- Mosteller, F. and D.L. Wallace (1964). *Inference & Disputed Authorship: The Federalist*. MIT Press
- Porter, M.F. (1980). "An Algorithm for Suffix Stripping," *Program*, Vol. 14 3, pp.130-137.
- Tetlock, P. C. (2007). "Giving Content to Investor Sentiment: The Role of Media in the Stock Market." *Journal of Finance* 62 (3): 1139–68
- Loughran, T. and B. McDonald. (2011). "When Is a Liability Not a Liability? Textual Analysis, Dictionaries, and 10-Ks." *Journal of Finance* 66 (1): 35–65.