Build Your Own Search Engine

An Introduction to Machine Learning Methods in Text Mining

Evan Hernandez (ehernandez4@wisc.edu, 907-092-9055)
David Liang (dliang23@wisc.edu, 907-034-1251)
Alec Yu (amyu@wisc.edu, 907-075-3653)

December 2016

Executive Summary

Search pervades everyday life in the twenty-first century. Students, researchers, and ordinary people alike enjoy the privilege of record-speed information retrieval from the Internet. Despite its fundamental importance, search is complicated. It leans heavily on the subject of text mining—the extraction of pertinent information from natural language documents—a subject which faces many challenges, both theoretical and practical. How does one model the content of a textual document? The same words appear in many documents, but each document may use those words differently. What determines a document's semantics? What about semantic similarity? Most importantly, how can one measure the semantic features of a document in real time? It turns out that many of these questions can be addressed at least in part by machine learning. In particular, the issue of semantic categorization can be reduced to a multiple-classification problem, one that is accurately solved by support vector machines.

We address these questions in the present paper, namely the questions of textual document representation and categorization. We will discuss methods for representing textual data, isolating the semantics of a textual document, and predicting attributes of new documents with learned classifiers. Specifically, we will discuss: term-frequency/inverse-document-frequency representations of text and measures of similarity between different texts; semantic noise reduction of many documents via latent semantic analysis; and supervised categorization of textual documents using k-nearest-neighbors and support vector machines. At the end, the reader will be tasked with building a simple search engine, whose implementation will depend on the methods of text mining discussed in the background.

Contents

1 Background

1.1 Problem Description and History

Consider a law firm that wants to build a strong defense for its client: thousands of legal documents are available to the firm, but only a small fraction of the information contained in these documents is useful for the given case. How would lawyers go about gathering pertinent information or evidence? On one hand, the lawyers could read every word in every document available and then hope to remember it all by the end. On the other hand, such brute force is inhumanly difficult and exorbitantly expensive. More to the point, not all available information is useful. In fact, chances are that the vast majority of legal documents are irrelevant to the case at hand. For example, if the lawyers are handling a case on tax evasion, they probably do not want to read about domestic laws. The firm ultimately needs an automated and highly efficient method for organizing textual information and for retrieving the pertinent information quickly.

This problem of organizing textual information is not new. Humans began writing somewhere around 5000 B.C. and since then have created a monolithic amount of written information. For hundreds of years, the only way to research pertinent information was to visit a library and page through book after book. The process was slow and oftentimes unfruitful. Then, with the advent of the internet, the world rejoiced at the new availability of information and yet faced an organizational crisis. What should be done with this monolith of textual information? In what ways might all this information be useful? Most importantly, how could this information be made accessible to the general public? These questions emphasized the need for search engines, programs that could troll through large masses of web content and pick out only the relevant bits of information in real time. During the 90s, companies like Yahoo!, Ask Jeeves, and eventually Google sprang up, each offering unique methods for indexing webpages and providing speedy information retrieval. Search engines have since become a staple and keystone for day-to-day life in the twenty-first century.

Underpinning much of search is the concept of **text mining**. Text mining refers to the organization and extraction of information from natural language documents. Note that search and text mining are not one and the same. Search engines typically undertake a number of steps to produce results, including indexing and query enrichment. By contrast, text mining is a more general collection of methods for extracting data from textual documents. Some examples of text mining include part-of-speech tagging, concept extraction, and sentiment analysis. In this paper, we will focus on the representation of textual documents as vectors and the categorization of textual documents into predefined categories. We will see that, with the right precautions, document categorization works out to be a fairly straightforward classification problem.

Document categorization has many immediate applications. Lawyers may wish to automatically sort large quantities of text into semantic categories (domestic law, tax law, etc.); companies may wish to categorize incoming email and eliminate spam; search engines may want to categorize new queries to restrict their search space; and so forth. Despite its usefulness, document categorization is not trivial. It is not immediately obvious what determines whether a document belongs to a category. Language is by nature ambiguous. Some studies from cognitive science suggest that this ambiguity is evolutionarily advantageous because it eases the language learning process for young children, but that ease does not surface for computers. In particular, polysemy—the presence of multiple meanings for the same word—makes it difficult for computers to disambiguate between pairs of sentences like "I climbed the steep bank," and "I deposited a check in the bank." On the other hand, language is often redundant and dilutes information with unnecessary clarifiers like pronouns and tense. Worse yet, the categorizer may have limited access to information about the documents. In the case of web search engines, the webpages will not come with category labels; it is up to the search engine to organize the data into semantic categories.

To address these problems, we need an effective way to encode the semantics of a document without falling victim to ambiguity or getting lost in the noisy redundancy of natural language. We present methods for doing so in the proceeding section.

1.2 Representation of Textual Data

In order to extract information from documents, we must first define what we mean by document. For our purposes, a **document** is a sequence of natural language tokens. In this sense, the sentence "I ate an apple" is as much a document as the entire play *Hamlet*. It is worth noting that although our definition of document is limited to text, other definitions are typically more general so as to include any collection of information in written or electronic form. In fact, many of the methods for text document representation and categorization that we discuss here will apply to other types of documents as well.

As one might expect, raw strings are not conducive to mathematical manipulation and therefore are not conducive to large-scale search. Performing string comparisons is both slow and inaccurate; the user is looking for a document semantically similar to his query, not for an exact textual match to his query. The clear alternative is to represent documents as vectors by defining a global list of features that a document may or may not have, and then encoding each document in terms of these features. Thus, for a corpus with m features, each document will be encoded as an m-dimensional vector. The list of features is typically determined by characteristics of the entire collection of documents.

Here, we will employ the term-document model. In the term-document model, each feature corresponds to a term from the collection of all terms taken over all documents. Its value is given by the number of times that term appears in the document. For example, let $d = [d_1\ d_2\ ...\ d_m]^T$ be the document vector given by some new article, and $\mathcal{V}$ the collection of all terms in the corpus. If term $v \in \mathcal{V}$ appears in the actual document k times, we have $d_v = k$, noting that we let each term correspond to some index of the

vector. Unfortunately, this encoding is still meaningless: a document might contain the word "the" in 1256 different places, giving it a large feature value, but that does not mean the word "the" is semantically salient. To remedy this, we need to weight each term inversely to how often it is used throughout the whole corpus, under the assumption that rarer words better identify the semantics of a document. This encoding is called the **term frequency-inverse document frequency (tf-idf)** because each term frequency feature is multiplied by the inverse of its corpus (document) frequency.

The motivation behind the use of this statistic is that we hope to find documents with the feature we are looking for, and we?ll prioritize documents when those terms are relatively exclusive. In short, this statistic combines term frequency inverse document frequency. The term frequency is simply the raw number of occurrences of a word in a given document, while inverse document frequency is the inverse of the proportion of documents in which the term is present. Tf-idf, then is the product of these two statistics. A high tf-idf is achieved by a low document frequency (high inverse-document frequency) meaning that few documents contain this feature, and a high term frequency which comes from a document having a high number of the feature.

The next challenge we are faced with is the size of the feature space embodied that our data model defines. We can use mathematical techniques to reduce the dimensionality of this feature space while preserving relative structure. Additionally, we will employ a technique called ?stemming? to combine categories of similar words like ?connect,? ?connecting,? and ?connection? into a single category so that multiple categories don?t exist for very similar words. We will also exclude all words with a frequency of three or less throughout the entire corpus because infrequently mentioned words are likely unimportant. Furthermore, we?ll exclude all words counted as ?stopwords? which are essentially commonly used words like ?the? and ?and? because most documents will include these and these terms are unlikely to be informative. Finally, to mitigate the problem of variable document lengths, we normalize each document vector.

1.3 Extracting Semantics from Textual Data

LSA and stuff.

1.4 Categorization of Textual Documents

We have finally arrived at the point where we can discuss the methods of classification. Through classification, we hope to determine which classes a new data entry belongs to. To do so, we require a classifier, a model learned from training data. Various classifiers exist, each with their own advantages and disadvantages. In this section, we will be discussing and comparing the least squares, k-nearest neighbors, and support vector machine classifiers. One thing to note is that we are dealing with binary (two-class) data: there are as many classes as dimensions. In practice, two types of multi-class classification are used for least squares and support vector machines: one versus one and one versus all. In one versus all, one classifier is learned that determines whether the data is in that

category or not. In one versus one, a classifier is built to compare each pair of classes, resulting in K(K-1)/2 classifiers which is far more computationally expensive. However, the advantage to one versus one classification is that it accounts better for overlapping categories than one versus all thus providing better classification accuracy. Because data can have multiple classes in the text classification, a new document can take the categories of the top k categories, or all categories above a given threshold.

The first classifier that we will inspect is the least squares classifier, a supervised learning method that takes an overdetermined system (lots of data, noise present) and minimizes over the sum of squared differences between the given categories and the corresponding modeled values, computed by multiplying the data feature matrix by a weight vector.

A method of solving the least squares problem is provided through the pseudoinverse, which relies on the singular value decomposition of a matrix.

Least squares has great performance when it comes to linearly separable data, but this is often not the case in the real world. In fact, often times there is noise in addition to outliers, and this is where the least squares classifier suffers because it tries to minimize classification error, so it takes into account extreme outliers and tries to minimize that error too.

The second classifier we shall discuss is the k-nearest neighbors classifier. This classifiers chooses the k points from the training data that are closest (in euclidean distance) to the new entry and classifies it based off of the majority vote. In reference to the figure above, if k = 3, then the new point would be classified as red since 2 of its 3 closest neighbors are red; if k = 5, it would be blue. This method works quite well for text classification. In fact, it is the most commonly used technique for a number of reasons. The first is that it is quite simple to implement. Another benefit is that it requires no training or precomputation: all of the computation occurs during classification. However, this is also quite a disadvantage, because every new data entry must be compared to every training data member in the classifier, which can be slow with large amounts of data in large feature spaces, as is the case with text classification. Another drawback is that results vary as you change k. In practice, k is determined to be ?elbow? of the graph comparing k to classification error. But to create this graph, every member of the training data needs to be classified for each value of k, which is quite costly.

Finally, we have arrived at the support vector machine (SVM). The support vector margin hopes to define the hyperplane in the feature space that produces the greatest margin between the two classes. However, there is often noise, producing overlap in the feature space, so support vector machines instead try to minimize classification error, which can be done iteratively. After the classifier is trained, it computes the classification of a new data entry by taking the sign of the inner product of the learned weight vector and the data vector.

We have ranked the SVM as the best classifier for our text classification problem and we have various reasons for that. When dealing with text classifiers, each word in the text is an entry in the vector of text features. With large text files, we are dealing with feature vectors with thousands of features. When this occurs, many problems can arise that cannot be handled properly by least squares and KNN classification such as overfitting

and vector sparsity. By applying the SVM to this problem, however, we can mostly void these issues. First of all, the SVM maximizes the margin space between each data set. This process is independent of the dimensionality of the feature space. Therefore, it helps us avoid the issue of overfitting. Secondly, sparse data does not adversely affect the SVM. One way to see this is to do a random rotation of the coordinate axes, which would leave the problem itself unchanged and give the same solution, but would make the data non-sparse.

2 Warmup

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text gives you information about the selected font, how the letters are written and an impression of the look. This text should contain and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

like this

all letters of the alphabet

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text gives you information about the selected font, how the letters are written and an impression of the look. This text should contain and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

like this

all letters of the alphabet

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text gives you information about the selected font, how the letters are written and an impression of the look. This text should contain and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

like this

all letters of the alphabet

3 Final Thoughts

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text gives you information about the selected font, how the letters are written and an impression of the look. This text should contain and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

like this

all letters of the alphabet

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text gives you information about the selected font, how the letters are written and an impression of the look. This text should contain and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

like this

all letters of the alphabet

A Appendix

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text gives you information about the selected font, how the letters are written and an impression of the look. This text should contain and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

like this

all letters of the alphabet

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text gives you information about the selected font, how the letters are written and an impression of the look. This text should contain and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

like this

all letters of the alphabet

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text gives you information about the selected font, how the letters are written and an impression of the look. This text should contain and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

like this

all letters of the alphabet

A.1 Warmup Solutions

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text gives you information about the selected font, how the letters are written and an impression of the look. This text should contain and it should be written in of the original language. There is no need for special content, but the length of words should match the language. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text gives

like this

all letters of the alphabet

like this

you information about the selected font, how the letters are written and an impression of the look. This text should contain and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

all letters of the alphabet

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text gives you information about the selected font, how the letters are written and an impression of the look. This text should contain and it should be written in of the original language. There is no need for special content, but the length of words should match the language. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text gives you information about the selected font, how the letters are written and an impression of the look. This text should contain and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

like this

all letters of the alphabet

like this

all letters of the alphabet

A.2 Lab Solutions

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text gives you information about the selected font, how the letters are written and an impression of the look. This text should contain and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

like this

all letters of the alphabet

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text gives you information about the selected font, how the letters are written and an impression of the look. This text should contain and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

like this

all letters of the alphabet