An Introduction to Topic Modeling as an Unsupervised Machine Learning Way to Organize Text Information

Robin M Snyder
RobinSnyder.com
robin@robinsnyder.com
http://www.robinsnyder.com

Abstract

The field of topic modeling has become increasingly important over the past few years. Topic modeling is an unsupervised machine learning way to organize text (or image or DNA, etc.) information such that related pieces of text can be identified. This paper/session will present/discuss the current state of topic modeling, why it is important, and how one might use topic modeling for practical use. As an example, the text of some ASCUE proceedings of past years will be used to find and group topics and see how those topics have changed over time. As another example, if documents represent customers, the vocabulary is the products offered to customers, and and the words of a document (i.e., customer) represent products bought by customers, than topic modeling can be used, in part, to answer the question, "customers like you bought products like this" (i.e., a part of recommendation engines).

Introduction

The intelligent processing of text in general and topic modeling in particular has become increasingly important over the past few decades. Here is an extract from the author's class notes from 2006.

> *"Text mining is a computer technique to extract useful information from unstructured text. And it's a difficult task. But now, using a relatively new method named topic modeling, computer scientists from University of California, Irvine (UCI), have analyzed 330,000 stories published by the New York Times between 2000 and 2002 in just a few hours. They were able to automatically isolate topics such as the Tour de France, prices of apartments in Brooklyn or dinosaur bones. This technique could soon be used not only by homeland security experts or librarians, but also by physicians, lawyers, real estate people, and even by yourself. Read more for additional details and a graph showing how the researchers discovered links between topics and people." Text Mining the New York Times* http://slashdot.org/article.pl?sid=06/07/29/0634232&from=rss [as of Sat, Jul 29, 2006]

In 2006, the author had little idea that he would do significant work in that area just a few years later. Processing text in intricately related to natural language but the applications of topic modeling, as we shall see, are not limited to just text.

The common theme in this paper is that of sophisticated text analysis (matching, grouping, etc.) for decision making using probabilistic models and distributed processing. This includes probabilistic topic modeling and various aspects of natural language processing and computational linguistics. In June, 2014, the author attended the Association of Computational Linguistics conference, held in Baltimore, MD. Practically unheard of a few years ago, that conference was the biggest ever, with over 1,400 at-

tendees, from all over the world, and all the big companies - Google, Baidu, Amazon, Facebook, Microsoft, Yahoo, Facebook, IBM, etc., looking for students and others to hire for sophisticated text analysis - topic modeling, sentiment analysis, etc.

The author has written software code to do many of the things described in this paper, and has used many Open Source software systems that make it easier to do the things described in this paper, but such details are beyond the scope of this paper.

Some actual topic modeling results and information visualization of such results were not available yet at the time of the paper submission, but will be presented at the session at the conference.

Expert systems

Topic modeling has its roots in expert systems. It has long been conventional wisdom that "expert systems" do not replace the experts. Rather, "expert systems" do several things.

1. Expert systems amplify the intelligence of non-experts - allowing non-experts to behave much more like experts. This has eroded job security in fields where the "experts" were not really experts - such as loan approval, risk assessment, athletic talent assessment, etc., and statistics-based decisions are better than the so-called "experts". It is often the case in such fields that the computer, with relevant data, machine learning techniques, etc., can make better predictions than the so-called "experts". Such jobs then become less common and/or become lower paying jobs.

2. Expert systems help real experts, such as doctors, lawyers, etc., make decisions in that by providing a list of alternatives and probabilities, the expert is less likely to overlook and/or be unaware of viable options in a given situation. For example, if an expert system provides a doctor with possible diagnoses for given symptoms, the overworked and time-stressed doctor should be less likely to overlook a possible condition that might otherwise later lead to a lawsuit in that the "expert" doctor "should have known".

Expert systems have two primary means implementing expertise. One is a rules based model whereby rules are used. The other is a statistical pattern based model where patterns, and lots of data are used. In practice, both methods are used. In machine learning, a lot of effort is put into what is called feature extraction (e.g., data scrubbing, picking out parts of an image, etc.), and then the statistical models are used on the extracted data.

Computational linguistics

Computational linguistics is the study of using computation to perform natural language processing. Natural language processing is inherently difficult, the common example being the following.

- Time flies like an arrow.
- Fruit flies like a banana.

Another being the following.

John saw Mary with a telescope.

Who has the telescope? If a human cannot tell, how could a computer? An upper bound for recognizing parts of speech in a language is about 96% since human experts in the field tend to disagree on the assignment of parts of speech tags.

But just like weather prediction, though chaotic and not exactly predictable, is useful if the forecast is pretty good out to 48 hours, natural language processing and computational linguistics can provide useful results.

Natural language processing has traditionally been rules-based, with tokenizers, parts of speech taggers (for nouns, verbs, etc.), grammars, both fixed and probabilistic, etc. A few years, Google came in and easily beat every rules-based system by using a probabilistic pattern-based machine learning technique that had millions of documents of translated works (e.g., United Nations transcripts, etc.) on which to train.

Google makes their translation system, based on machine learning techniques, available to everyone for browser web use for free, and for other commercial type uses for a minimal fee. Their free translation system is great for anyone learning a language. They use crowd sourcing to help improve the system, whereby anyone who sees something incorrect can help provide a better translation. But be aware that some phases do not translate perfectly. Idioms may not translate well. And the case, upper, lower, etc., may not match as the translations are based on machine learning matching.

Machine learning

Machine learning is a collection of techniques based on statistical, with some rules, usually via feature extraction, that are used to do pattern recognition. The field of artificial intelligence, a better name for which would have been machine intelligence, which got a bad reputation for promising too much that was not delivered, was somewhat vindicated by the appearance and success of machine learning, which one can take as a new term for what was previously called artificial intelligence.

But machine learning is not the answer for everything. One does not just get a lot of data and apply computing power to the data. In practice, a potentially very complicated and somewhat intuitive/intelligent process called "feature extraction" is used to extract features and then the machine learning works on the "features". So, to recognize a car, one might develop techniques to extract and thereby recognize features such as wheels, windows, etc. The field of syntax and parsing in natural language attempts to extract features on which other techniques can be applied.

Deep learning

Andrew Ng is leader in the field of machine learning. He has many good YouTube videos describing his word and, of course, the work of his many students. A good YouTube video by Andrew Ng is at:

https://www.youtube.com/watch?v=W15K9PegQt0

Here is a summary of some aspects of his video as interpreted by the author. Ng, a leader in the field of machine learning and deep learning, did work for Google, a short walk from his Stanford office, but now appears to now be doing work for Baidu, the most well known Chinese search engine. Baidu had a big presence at the Computational Linguistics conference in Baltimore.

"Machine learning" has made more and more progress in the past decade, not necessarily by better algorithms but, by feeding the algorithms more data. In the era of applying "Machine Learning" to "Big Data", there was initial good progress, and progress continues to be made, but that progress has leveled off. Better methods are needed. Ng has led the way in what he calls "Deep Learning". He (and his students) have solved problems including having machines learn to grasp objects, 3-D perception without multiple stereoscopic images, etc. His primary working assumption is that there is a general learning algorithm in the brain and he is looking for it. The learning algorithm of the brain appears to make use of edge detection methods (e.g., "Sparse Coding") to chunk/segment what is seen rather than just looking at the low level parts (e.g., pixels). A problem with traditional "machine learning" is that it does not look at the individual small parts of the data. Rather, much work goes into manually tweaking code to do "feature extraction" so that the machine learning algorithms can then make use the features. As an example, parsing rules for a sentence attempt to extract features (e.g., nouns, verbs, phrases, etc.) from the text and then make use of these features, either by more rules or statistical machine learning (e.g., language translation).

Ng led the Google Brain project. Instead of the usual 10 million neuron connections, they used 1,000 million (1 billion) connections and are expanding to 10,000 million (10 billion) connections. With access to millions of YouTube videos, which included over 10,000 cat videos, the Google Brain learned to recognize the concept of a "cat", without, of course, knowing much else but that there are "cats" on the Internet. They used 16,000 computers running for 3+ days in the experiment. The media hyped this, but the Google Brain also learned to recognize faces, etc.

Ng sees one of the many near-future uses of Deep Learning as that of natural language understanding. He has worked with Chris Manning (from Stanford, a leader of the field of natural language processing).

The concept of "edge detection" appears to be important in eliminating what is less important from what is more important, but after that some patterns still need to be recognized. Looking for search terms, etc., appears to be a form of "feature extraction" that can be used to identify true positives in terms of false negatives (recall) and false positives (precision) in the search process.

So the "Deep Learning" technique is a (deeply) layered neural network approach that recognizes and categorizes patterns. A neural network, loosely based on an analogy the neurons in the brain, consists of layers of "neurons" that have forward and backward feedback mechanisms that help in recognizing patterns. Such pattern recognition approaches are categorized under the general umbrella term of machine learning.

The neural network technique is similar to linear and nonlinear regression techniques whereby one hypothesizes a linear or non-linear equation that fits the data. The differences are that instead of fitting data to pre-defined equations, neural networks separate patterns into groups and do not require a predefined equation - the "patterns" are "learned" by the neural network.

The described "Deep Learning" appears to be an operational methodology of what Jeff Hawkins, inventor/developer of the Palm Pilot handwriting system, describes qualitatively in his book "**On Intelligence**" [1] (a very interesting read).

Every new advance in technology makes possible new economies that are not at this time possible or feasible. One goal, then, is to identify those economies that can use these new technologies. A classic example is that, years ago, when Toshiba announced a tiny hard drive, Steve Jobs (and Apple engineers) decided that consumers would want their own enormous collection of songs on their own small personal device, even though no consumer had actually asked or even knew they would want such a device.

Data and problem solving

Data tends to come in two varieties. Structured data is the traditional data of name, address, phone number, etc. (though address is sometimes unstructured). Unstructured text data can be processed using various techniques - topic modeling, probabilistic modeling, etc.

Some methods for text/data analysis are the following.

- Topic modeling - clustering of topics distributions in text, trends over time, etc.
- Sentiment analysis - determine positive, negative, neutral comment according to some question
- Recommendation engine - recommend additional products based on previous purchases

Problem solving: For any problem, it helps to start with the questions one wants to answer. That is, identify the problem(s). Such problems (and associated solutions) tend to come in two varieties. Those that will cut costs, and those that will increase revenue. To cut costs, one should have an idea of where the costs are arising (e.g., too many phone calls, mailings, returns, etc.). To increase revenues, one should have an idea of where and how people would buy more or get others to buy, etc.

Some examples of problem solving are the following.

1. What additional products might this customer buy? Recommendation engines help answer this question.

2. How can the satisfaction with a product be summarized for the seller and for other customers? Summarization engines and sentiment analysis can help.

3. What are the traits in common when a customer abandons a shopping cart? Cluster analysis, user experience/interface analysis, etc., can help here.

4. Many sites will have automatons/robots answering chat questions and then switch to humans when appropriate. The data/text collected can be used to improve service, question answering, etc.

Text processing

Processing large amounts of text can require lots of computation. A common way in information retrieval to decrease the search time for text is to use an inverted dictionary, where each word has a list of documents in which it appears.

The Redis in-memory key-value database for distributed networked access can work well for inverted dictionary support of large amounts of text. Redis is being used by many big tech companies to speed up distributed web access or local text access within a more localized network.

An inverted dictionary stores every word and the documents in which that word appears. This provides very fast access to those document sets and Redis supports set operations such as intersection and union. In topic modeling, it may help to stem each word, do quick searches for just the stems of words, and then do more complete searches or topic models for those results for more context-sensitive matching.

To support topic modeling over time, the author designed and implemented a way, using Redis, for an inverted dictionary of stemmed words with date support.

At some point, a tagger such as the Stanford Part-Of-Speech tagger could be used to add ports of speech for searching and topic modeling. Such search tasks can be distributed to many computers at once, when needed.

Anyone attempting to match keywords with any "document" (e.g., email subject and body) would extract all the words from the document and then compare the words in the document to the the keyword list and decide whether it is a match.

However, the hard part is how to "decide" what is a "good" match. This is not obvious and the key part of any such method. Most simple systems simply return the search results and let the user decide what is a good match.

Traditionally, as in the example of language translation, rule-based systems have not done as well as machine learning approaches - which match patterns rather than having an understanding of the content. But an effective matching process appears to need the system to have some understanding of the material being searched.

The author calls the obvious matching technique a "syntactic" match which is based on symbol matching whereas a higher level and deeper matching a "semantic" match - which is a match by meaning and may or may not have an obvious "syntactic" match. A "semantic" match requires a deeper understanding of the material and a way to infer meaning from parts that logically connect together but whose connections are not obvious in a textual matching.

And this harder problem of meaning is why, once the large majority of documents are rejected as not being relevant, human inspection is required to do the final determination. Those "semantic" matches that are not correctly classified fall into the category of false-negatives.

Bayes Rule

Topic modeling as a direct application of the ideas of graphical models and Bayesian analysis.

The essential inter-related ideas of big data, computer science, and Bayesian and frequentist statistics are well described by a recognized expert in the field, Michael Jordan (machine learning statistician from Berkley, not the basketball player), in a recent YouTube video, at http://www.youtube.com/watch?v=LFiwO5wSbi8. A good book on the history of Bayes Rule, and also of frequentist statistics, is "The Theory That Would Not Die: How Bayes' Rule Cracked the Enigma Code, Hunted Down Russian Submarines, and Emerged Triumphant from Two Centuries of Controversy" by Sharon Bertsch McGrayne.

What kept Bayes Rule an academic oddity for almost 200 years was the lack of a computational way to handle some intractable integrals arising from the equations in hierarchical models. Perhaps this is one reason why Laplace leaned towards frequentist methods - the manual computations were just too involved to solve even simple problems that had practical significance. The use of Markov Chain Monte Carlo (MCMC) techniques along with a simplified variation, Gibbs Sampling, that works in many cases, together with increases in the storage and power of computers, has made Bayes Rule much more tractable for practical use. Some of these techniques took a while to become known because they were used during World War II in both cryptography and the atomic bomb project and, as such, both the techniques and the people who would have developed them were classified information not to be released to the public.

According to Jordan, a frequentist approach will average over the possible data to see if the sample result is within a certain limit (i.e., confidence interval) while a Bayesian approach will look at just the available data, and what is know about the past data, in making a decision. Jordan makes the analogy to physics of the early twentieth century, when physicist looking at the some data would classify it either as wave phenomena or particle phenomena and gradually came to an understanding that it was just different ways of looking at the same phenomena. The field of computer science has no such dichotomy. Jordan points out that, just like the wave-particle duality, statistics has two main ways of looking at the same phenomena - called Bayesian statistics and frequentist statistics. And is in the same way that specific phenomena might be better analyzed with either wave theory or particle theory, specific statistics problems may be better analyzed with either Bayesian statistics or frequentist statistics.

Jordan sees computer science and statistics merging in the next 50 years. Many algorithms of interest are now probabilistic algorithms. And once data becomes too large to look at all the data, and one needs results based on many factors, query results will (and sometimes now have) error bars associated with them. In computer science, a linear algorithm is needed to at least look at all of the data once. At some point, as databases become bigger and bigger, the only way to get sub-linear algorithms is to not look at all of the data, which requires probabilistic models.

Topic modeling

In terms of topic modeling, both LSI (Latent Semantic Indexing), a VBM (Vector Based Model) and LDA (Latent Dirichlet Allocation), a hierarchical (and generative) Bayesian inferencing method, use a BOW (Bag of Words) approach.

- A set has items that appear only once in a set.
- A bag has items that can appear more than once.

Since topic modeling almost always uses a BOW approach, each "word" and a count of the number of times that a word occurs in a document is used. Better results are usually obtained by modifying the count using tf-idf (term frequency - inverse document frequency) to not inflate the importance of repeated words in a document or collection of documents. There are variations but usually a logarithm function is used to not overly weight words that appear more and more times in a document. However, this is done in the preprocessing step so that this modification can be easily be omitted or changed.

The general approach used to pre-process a document for topic modeling includes the following.

1. Remove "stop" characters (such as period, comma, etc.), separate the text into words.
2. Remove common words that are in an "omit" list or patterns (e.g., common numbers), depending on the application.
3. Use stemming to consider some words with the same stem as the same word. For example, "car" and "cars".
4. Bi-grams (adjacent words that often appear together), tri-grams, etc. can be considered one "word".
5. Transform the resulting list of words into a bag of words (words with count) for that document.
6. Modify/weight the counts if desired (e.g., using tf-idf, etc.).

This pre-processing, or feature extraction, is the messiest part of the process and cleaning up this pre-processing, often with long lists available from different sources, etc., helps a lot in getting valid results. A lot of NLP (Natural Language Processing) techniques are used in this pre-processing.

The order of the words, unless used as bi-grams, tri-grams, etc., is not considered important to the analysis - probably because no one who has tried it has found it useful - probably because there is too much noise to discern any useful signal. And, to date, topic modeling has worked will without that complication.

Note that LSI, LDA, etc., is designed to be general and to smooth out differences in the document so that it may find a document relevant/similar even if the keyword of interest is not found in that document.

Since the introduction of LSI, LDA, etc., derivative works have appeared (usually in academia) that cover almost any conceivable alteration to the original model. This includes temporal orderings (e.g. date and time), word orders (e.g., bi-grams, tri-grams, etc.), hierarchical levels, etc.

In topic modeling, the entire corpus of documents is pre-processed in the above manner and then LSI, LDA, etc., is used. Document similarity allows these documents to be grouped/clustered into similar documents (using a variety of methods).

Methods to compare a new "document" to existing documents include the following.

- A search query is converted into a document in the same manner as described above.

- A dictionary of words with importance indicated by repetition is just a search query with those important words repeated and then processed in the above manner.

- An advertiser could supply search terms, a dictionary of words with frequency count, are examples of what they are interested in in terms of paragraphs, documents, etc. In each case, that information supplied is converted into a list of words with frequency count (i.e., bag of words) that represents a "document" and then document similarity is used to determine and do something with similar documents.

Topic modeling helps identify "similar" documents without knowing anything about the actual documents, one must specify which group or groups of documents that are of interest. This is, in contrast, to having humans manually identify those groups by annotating a small subset of documents. Other ways include, as mentioned above, search queries, a dictionary with frequencies, example text, etc.

Topic modeling is a very general idea that has found applications in, for example, DNA processing/research, image processing, etc.

For example, if customers are considered "documents" and the number of each item they have bought are considered "words" with associated "counts", then, without knowing the details of any customer or product, topic modeling can help answer questions like "customers like you bought these products" (i.e., document similarity and then most associated products) and "here are products similar to the one you are considering" (word/topic similarity), etc. This is the basis of recommendation engines and was a key part of the winning solution to the NetFlix competition a few years ago. Note: The "cold start" problem happens, for example, when a new product is introduced that has no history and a way is then provided to jump start this product, which is why, in the NetFlix competition solution, topic modeling is only part of the overall solution.

One is always free to integrate specific queries into the process but, for the problem being solved, this may help or hinder the process and make the results better or worse, depending on the application, the model, the implementation, etc.

It is still true that to solve any of problem, one must carefully identify and define the problem that one is solving and then, if off-the-shelf methods do not work well, one needs to create or adapt some model to solve that particular problem.

LDA

The LDA (Latent Dirichlet Allocation) model originated with Jordan, Ng, and David Blei. All have done many interesting YouTube videos that explain this and other related work.

The following is a commonly used general method for setting up documents for analysis.

1. Convert each document to text.
2. Remove all extraneous characters from the text using a list of characters to remove.

3. Pick out all of the words from the text of each document.
4. Convert words to their base word. For example, the word cars would be stemmed to car and considered the same word.
5. Remove stop words - words that are to be ignored - from a list of stop words.
6. If desired, consider select bi-grams (adjacent words), tri-grams, etc., to be the same word for computation purposes.

Thus each real document is transformed into a cleaned-up document for processing.

The above process converts each document to a "bag of words". A "set" of words would only contain a list of words where each word can appear only once. A "bag of words" includes each word and a count of the number of times it appears in the document. This "bag of words" document model can be modified by weighting the word count using a suitable measure, such as tf-idf (Term Frequency - Inverse Document Frequency), so that repeated words do not adversely influence the results.

From this starting point, two methods to compare the documents are the following.

1. LSI, Latent Semantic Indexing, a VBM, Vector-Based Model of document/word comparison.
2. LDA, Latent Dirichlet Allocation, a HBM, Hierarchical Bayesian Model, of document/word/topic comparison.

These methods provide a higher-dimensional way of comparing documents than is obtained just by looking at the intersection of words between documents.

In addition, the following method for similarity processing (again, having studied, coded, and customized it).

3. LCS, Longest Common Subsequence, a dual (i.e., the same) problem to the MED, Minimum Edit Distance between documents.n

Much text analysis and comparison can be done with the LCS, Longest Common Subsequence, problem, a dual problem to the MED, Minimum Edit Distance problem. The LCS/MED algorithm is used, for example, to compare two programs to see what has changed.

The methods such as LSI, LDA, etc., help automatically determine document similarity such that documents can be grouped together according to some criteria. Which similar documents are most relevant needs an appeal to an authority "outside the box", in the same manner as "Sentiment Analysis" requires some pre-determined authority on whether text fragments are positive, neutral, or negative.

By analogy, a computer program could be created to group monetary currency bills into similar types, resulting in piles of bills for the denominations $1, $5, $10, $20, etc. But such a program would have no idea which were more valuable unless some "outside information" were programmed into the system - so that, for example, $20 is worth more than $1.

Software

The author has been using Python and Python libraries to prototype and semi-automate most of the work. By semi-automate, it is meant that there is no convenient GUI (Graphical User Interface) with which a user can use a finished system. Instead, a collection of ad hoc scripts that can be easily changed (as necessary for research purposes) are used to automate what is being done. Python, using the scientific libraries for NumPy, SciPy, etc., and gensim, etc., is the language system and libraries used by many researchers and developers in the field - including many of the original authors of the relevant research. Such methods (e.g., the gensim libraries) already have built-in support for huge amounts of data, incremental addition of new information, and the effective use of multi-core processors and collections of machines that can all work on the task - LSI, LDA, etc.

Another Open Source machine learning / topic modeling software is Mahout - which runs on Hadoop using MapReduce and which has been greatly improved in the past few years. The symbol for Hapdoop is an elephant, an idea of the young daughter of the inventor and founder of Hadoop. Mahout is an Indian word (from India, not the United States) for an elephant rider. Mahout thus claims to assist in riding the Hapoop elephant.

Mallet, Java-based topic modeling software recommended by Dr. Jordan Boyd-Graber, computational linguist professor with many very useful and well done educational videos on YouTube.

The Standford Parts-Of-Speech tagger is a Java-based software for tagging parts of speech. This is useful, for example, if one wants to only look at nouns, verbs, etc., in a large corpus of documents. The Stanford Natural Language group has published a large number of educational and well-done YouTube videos.

Summary

This paper/session has discussed and/or demonstrated some ideas on the state of topic modeling as an unsupervised learning technique for relating documents in order to extract useful comparison information in order to make decisions.