# The Power of LDA Algorithms and How They Help Text Mine Your Documents

LDA can work its wonders on any set of documents: client satisfaction comments, documents obtained in discovery or due diligence, annual reports and more.

Rees Morrison, Juris Datoris, Law Technology News

June 8, 2017

With text-mining software, which finds patterns and insights from collections of documents, one powerful capability identifies related words. The software **models** the words in a **corpus** (all the documents) into topics. **Latent Dirichlet Allocation** (LDA) is one of the **algorithms** which carries out such topic modeling. In the legal industry, LDA can work its wonders on any set of documents: client satisfaction comments, documents obtained in discovery or due diligence, annual reports, hot line replies, survey answers, and other sources.

## An Example of LDA

Let's start with an actual set of documents and see how LDA performs. The author gathered self-descriptions by thirty U.S. law firms, as in what they might use in recruitment brochures. Each self-description runs at least 150 words. After removing trivial words, we used LDA from a package of the open-source **R language** and told it to model five topics. The table below lays out the 10 words the software most closely associated with each topic, in declining order within each topic.

[LDA_Table-Article-201706080943.jpg]

Topic 1 appears to address client service and value ("provide," "providing," "value"); Topic 2 suggests depth of experience ("services," "years," "leading"); Topic 3 is the bragging topic ("recognize", "top," "best," "ranked"); Topic 4 focuses on substantive practices ("litigation," "real," "estate," "regulation"); and Topic 5 on engagement of lawyers ("pro" "bono"). Obviously, readers might pick alternative themes for the topics, but at least it the software assembles large amounts of text and isolating the key words. The software does not suggest a concept that pertains to the topics it creates.

LDA relies on several decisions and offers quite a few parameters that can be modified. For example, the results would change if we decided to added more self-descriptions; removed some words that appear often, such as “firm,” “legal,” or “lawyers”; or treated two-word combinations like “real estate” or “pro bono” as single terms. We could also combine synonyms, such as “lawyers” and “attorneys”, into one of them and we could treat single and plural forms the same (“firms” = “firm”). We could choose to show the top 15 words in each topic, or any other number, which might clarify (or complicate) our interpretation of what the topic is all about.

As for parameters, it is easy to modify the number of topics you ask LDA to extract. That number depends partly on the amount of text you have in the documents, as well as the number of salient topics you foresee represent the range of key concepts in the corpus. You might pick a number (referred to cryptically as “**k**”) that generates topics to your desired level of interpretability or the one yielding the highest statistical certainty (i.e. **log likelihood**).

## How LDA Works

At its core LDA is guided by two principles. *Every document is a mixture of topics* containing words from several topics in particular proportions. For example, were this a two-topic model we could say "Law firm Morrison’s self-description is 90 percent topic 1 and 10 percent topic 2, while the Rees firm’s self-description is 30 percent topic 1 and 70 percent topic 2."

*Every topic is a mixture of words.* For example, a three-topic model of self-descriptions might have one topic that has to do with "quality," one “size,” and one "clients." Some of the words in the quality topic might be "experienced," "sophisticated," and "challenging," while the size topic might be made up of words such as "global," "hundreds," and "offices." Importantly, words can be shared between topics; "firm" appeared in two of our topics.

LDA works because it assumes the documents were “generated” according to a set of rules, one of which is called a **Dirichlet distribution**. That distribution can be thought of as the statistical probabilities of a document having a particular spread of topics, weighted by likelihood, and those topics have a particular spread of words. In effect, the LDA algorithm unravels the documents and topics based on that assumed model of document creation.

LDA mathematically estimates both of these at the same time: the mixture of words that is associated with each topic and the mixture of topics that describes each document. The algorithm starts by converting the entire corpus into a matrix whose rows are documents, columns are words and each element in a column is a count of a given word in a given document. That matrix has lots of zeros, of course, since many words appear in only a few documents. LDA "factorizes" this matrix of size n (number of words) by d (number of documents) into two matrices, documents/topics (n x k topics) and topics/words (k x d).

The “latent” part of LDA is because we only directly observe the words, whereas the topics are **latent variables**. The Dirichelet part of LDA restates the underlying generative model, and allocation is, well, allocating topics and words.

The algorithm checks and updates topic assignments, looping through each word in every document many times. For each word, its topic assignment is updated based on two criteria: How prevalent is that word across topics? How prevalent are topics in the document?

The per-topic-per-word probabilities are called **beta**, from the model. The software has turned the documents into a one-topic-per-term-per-row format. For each combination, the model computes the probability of that term being generated from that topic. For example, the term "litigation" has a beta[1] probability of being generated from topic 1, but a beta[2] probability of being generated from topic 2.

Besides estimating each topic as a mixture of words, LDA also models each document as a mixture of topics. We can examine the per-document-per-topic probabilities, called **gamma**. One step of the LDA algorithm assigns each word in each document to a topic. The more words in a document are assigned to that topic, generally, the more weight (gamma) will go on that document-topic classification. Each of these values is an estimated proportion of words from that document that are generated from that topic. For example, the model estimates that only about gamma[1] of the words in document 1 were generated from topic 1.

Assuming this generative model for a collection of documents, LDA then tries to backtrack from the documents to find a set of topics (and the words that define those topics) that are likely to have generated the collection. These intricate calculations and recalculations of probabilities depend on **Bayesian statistics.** As a huge simplification, with that statistical methodology, new information (a recalculation) feeds into the next calculation.

One internal methodology of LDA is known as **collapsed Gibbs sampling**. Here’s how it and LDA works:

The algorithm goes through each document and randomly assigns each word in it to one of the however many topics you prescribed (remember k?). Next, for each document the software goes through each word and for each topic t, computes two things: 1) the proportion of words in document that are currently assigned to a topic, and 2) the proportion of assignments to that topic over all documents that come from this word. If appropriate, it reassigns a word to a different topic, where you choose that topic based on the previously-calculated probabilities. In other words, it assumes that all topic assignments except for the current word are correct, and uses Bayesian techniques to update the assignment of the current word using the underlying model of how documents are generated. To repeat, the LDA calculations involve a Bayesian method of continually revising probabilities as more information comes in.

After repeating the previous step a large number of times, eventually the software’s assignments are pretty good and don’t change much (one of the parameters of LDA we did not include above can specify how many iterations the software should carry out). So it uses these assignments to estimate the topic mixtures of each document (by counting the proportion of words assigned to each topic within that document) and the words associated to each topic (by counting the proportion of words assigned to each topic overall).

## Interpretation of the Topics

As sketched above, fitting the model depends on sophisticated statistics. Once that is done, the interpretation of the topic word-sets takes most of the time. As the example at the beginning of this article shows, the words matched to a topic sometimes clearly indicate the gist of the topic, but sometimes you scratch your head. For this reason, when analysts carry out topic modeling, they spend time trying out various decisions and tweaking various parameters. Still, the capabilities of text mining software present significant opportunities for leaders of law firms and law departments.

---

*Rees Morrison, Esq. is a partner at Altman Weil with countless interests in legal analytics.*

---