Topic Modeling with Spark and LDA to Facilitate Legal Investigations

Ryan Chamberlain, Arthur Mak, James Route, Sayantan Satpati

chamberlain.ryan@gmail.com, arthur.t.mak@gmail.com, james.route@gmail.com, sayantan.satpati.sfbay@gmail.com

School of Information, University of California, Berkeley, Berkeley, CA

# **Abstract**

The process of legal discovery, which includes the examination of materials prior to a trial to find useful evidence, has become increasingly burdensome because of the rapid growth in electronically stored data. The traditional, manual review process scales poorly when faced with millions of documents, and common automated methods require manual labeling and can still be time consuming. We explore a fully unsupervised approach using Latent Dirichlet Allocation (LDA) to perform topic modeling on a cloud-based architecture running Spark and Hadoop, and test its viability against a public dataset of 1.2 million emails from the Enron investigation. Our five-node Spark cluster efficiently generated several topics of high relevance to a notional investigation, identified emails strongly associated with each topic, and generated keywords from the corpus to drive a more directed search. Our approach can be completed at a small fraction of the time and cost of traditional methods. While not a replacement, topic modeling can complement traditional methods by improving efficiency, providing better direction from the outset, and functioning as a check to help ensure completeness of the review process.

**Keywords**

Topic modeling, Spark, LDA, discovery

# **Introduction**

The process of legal discovery, which includes the examination of materials prior to a trial to find useful evidence, has become increasingly challenged because of the rapid growth in electronically stored data. The traditional, manual review process scales poorly when faced with millions, billions, or even more documents. The challenges of manual review were well publicized almost 15 years ago during the federal investigation of Enron, which at the time was considered the largest computer forensics investigation ever conducted (Iwata, 2002). To process all of the information, the U.S. Department of Justice (DOJ) staffed a special task force that pored over the data for four years; the DOJ also contracted with three data recovery firms for technical assistance (Linder, 2014).

Predictably, the challenges of discovery have only worsened in the intervening time. In 2010, the Examiner for the U.S. Bankruptcy Court in the Southern District of New York faced the task of reviewing approximately 3 petabytes of data—or the equivalent of 350 billion pages—as part of the legal proceedings against Lehman Brothers[[1]](#footnote-1). The Examiner used Boolean searches to reduce the page count to 40 million, and then employed more than 70 attorneys to conduct a manual review, which was still a daunting task for even a large group of reviewers (Baron, 2011).

As the capacity for digital storage grows, the traditional review techniques will become even less feasible, with an increased risk of missing key evidence. Boolean searches offer limited assistance when faced with an extremely large and mostly unknown dataset, as the examiner may have little insight into which keywords will return results relevant to the legal proceedings. Even after repeated successful searches there will likely be useful data that remains untouched and potentially undiscovered.

**One Popular Solution: Predictive Coding**

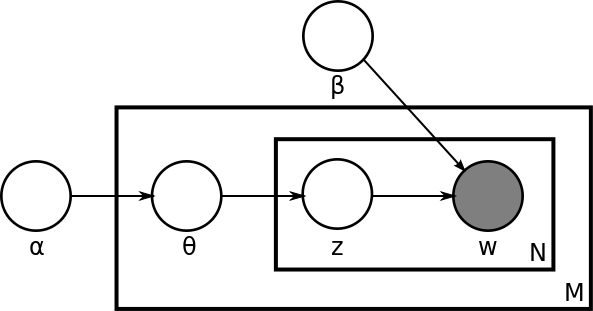
Predictive coding is one solution to the discovery problem that has risen to prominence since the early 2000s. This approach iterates between manual review and an automated, machine learning process to scan rapidly through document collections. Predictive coding first requires human experts to label a sample of documents as relevant or irrelevant. Then, the algorithm builds a model, such as a classification tree, from the labeled data and applies it to the rest of the dataset, predicting whether the remaining documents are relevant. After running the model, humans review a sample of the results to correct inaccurate labels, rebuild the model, and then rerun it. Following this iterative process, the algorithm provides the reviewers with a pared down set of potentially relevant documents, which are then reviewed manually (Baron, 2011; Barry, 2012; Remus, 2014).

Multiple experiments have produced results indicating that predictive coding yields accuracies equal to or better than manual review, and in an experiment with 60,000 documents, predictive coding was six times faster than Boolean searches coupled with manual review (Bordon, 2010). However, for very large datasets, predictive coding can still be very time intensive, and it is limited in its ability to provide useful information about the documents beyond classification.

**Our Solution: Topic Modeling Using LDA**

We elected to implement an alternative approach using topic modeling. Topic modeling can help examiners understand their dataset by grouping documents into a preset number of topics. Each topic is defined by a set of words that occur within the document text. For topics that are particularly relevant, the algorithm can identify documents that are most strongly associated with the topic. The topic’s related words can also seed Boolean searches to identify additional documents, which can be set aside for the manual review process.

We use the Latent Dirichlet Allocation (LDA) algorithm to perform topic modeling. We chose this algorithm based on its strong performance in previous studies related to discovery and document retrieval, as well as its integration into the Apache Spark computing engine (de Waal, Venter, & Barnard, 2008; George, Puri, Wang, Wilson, & Hamilton, 2014). LDA models a collection of documents as a set of probability distributions that explain a generative process of the collection (see Figure 1). It assumes that the collection is made up of a set of documents (M) with words (N) and a set of topics (𝛳). Each topic is a distribution over a set of words, and each document is a distribution of topics. These distributions are defined by latent variables (α and 𝛽) whose values are inferred by analyzing the entire document collection (Blei, Ng, & Jordan, 2003).



*Figure 1*

We evaluated the topic modeling-based approach using the Enron email dataset (hosted on [Amazon](http://aws.amazon.com/datasets/917205)), which we chose because of its public availability and its size—more than 1.2 million emails. The dataset was originally used in the Federal Energy Regulatory Commission’s (FERC) investigation of Enron, making it directly applicable to the task of discovery (Information Released, 2013; Enron Email Data, 2012).

**Tools and Libraries Used**

We implemented a cloud-based architecture for storage, preprocessing, and performing LDA, as the dataset is impractically large to run on a single machine. The architecture features Apache Hadoop and Spark, making it scalable to fit larger datasets. We chose the SoftLayer cloud so that we could implement our own object storage from scratch and more closely emulate a hybrid or private cloud architecture that a legal investigation might employ. A complete list of our tools is below:

*Table 1: Summary of Software Tools Used*

|  |  |
| --- | --- |
| **Tool** | **Rationale** |
| SoftLayer Cloud and Python Library | Primary cloud infrastructure enabling use of variable resources at different stages of the project. We used Python scripts to automate the setup and takedown of cloud resources. |
| Swift Object Storage | Efficient permanent storage location for the raw and intermediate data. SoftLayer’s Swift-based object storage was cost-efficient, reliable, and had programmatic API access. |
| Python Fabric | Provided an automated procedure for setting up the architecture and deploying the application. |
| Apache Hadoop | Distributed storage across our cluster of virtual machines. Also provided an efficient platform to preprocess data using MapReduce routines. |
| Natural Language Toolkit (NLTK) | Performed preprocessing on the text data, such as stemming. |
| Apache Spark/Scala/MLLib | Performed the LDA processing. |

**Setup and Execution**

Prior to running LDA, we ran several stages of preprocessing to clean the data and optimize it for use in Hadoop and Spark. Although we do not provide details on how to set up every component, our [GitHub repository](https://github.com/rmchamberlain/w251-project) offers instructions on how to replicate our build process. We also repeatedly tested our scripts to confirm that our infrastructure could be reliably rebuilt from scratch.

*1. Load Enron data from Amazon’s public distribution into Swift*

From Amazon’s public Enron dataset, we used the original data published by FERC, as its format required less preprocessing. Due to the size of the entire Amazon dataset (210 GiB), we set up a small virtual machine (VM) in the cloud to acquire the data and transfer it to our Swift object store.

*2. Create a temporary server and perform the initial cleaning stage*

The Enron data are organized into 151 folders (in ZIP format), each containing the emails sent by a particular Enron executive. Each email is an individual file, as are any attachments. We opted to exclude attachments from our analysis, as many were non-textual and unsuitable for use with LDA.

We deployed a single VM in Softlayer with sufficient memory (12 GB) to avoid expensive writes to disk. The VM retrieved each folder, extracted the contents, and cleaned individual emails by removing the footer and extraneous formatting characters. Each cleaned text file was saved back into Swift.

*3. Create a Spark cluster, perform final data cleaning, and load dataset into Hadoop Distributed File System (HDFS)*

We first set up a cluster of five SoftLayer VMs, each with four vCPUs, 16GB of memory, and 100GB of storage. We used a Python script and the Fabric library to automate this process and install and configure Spark, Hadoop, NLTK, and other dependent libraries on the cluster.

We then transferred the text files from Swift to HDFS, merging the files so that emails from the same original directory were concatenated into one file. This was a critical step, as HDFS is designed to work with large files, rather than many smaller ones. (Placing 1.2 million small files on HDFS and running operations on the data can cause the NameNode to run out of memory.) Furthermore, Hadoop spawns Map tasks based on the number of input files, which creates a great deal of overhead that slows execution when there are many inputs. In our preliminary tests, processing the un-merged files was dramatically slower and tended to cause the DataNodes to fail after running out of memory.

We performed final preprocessing as a MapReduce job (map only), using NLTK for tokenization, stopword removal, and stemming. This process prevented the LDA algorithm from taking common, insignificant words into account, increasing the chances of generating useful topics. Similarly, stemming reduced words to their root forms (e.g., ‘follow’ and ‘following’ are treated as the same word). These processes also decreased the vocabulary size, which improved the runtime of LDA. The clean data were written back to HDFS, with each file containing one email per line.

*4. Run LDA using Spark MLLib*

We chose Spark as our compute engine because MLLib contains an implementation of LDA, and because Spark’s in-memory processing and parallelization options offered significant speed advantages over Hadoop.

Spark’s MLLib made running LDA straightforward. The 151 input files were read from HDFS into memory, and a flatMap operation separated the emails within Spark’s RDD data structure. Spark calculated word counts across all of the documents, removed the most common words, and then ran LDA on the filtered document set. The results of LDA—the topics and their 20 most strongly associated words—were written back to HDFS. We ran LDA with two separate document-weighting schemes and specified a varying number of topics (as described later) to observe the effects on the resultant topic models.

**Results**

The topic-modeling pipeline run successfully under Spark processed all emails and provided a list of 20 topics in approximately two hours. (Preprocessing was slightly longer.) Each topic comprised a list of 20 words, along with the log-likelihood score of the word occurring in a document with the given topic. We judged seven of the topics as relevant to the investigation of Enron, whereas the remaining 13 varied from slightly useful to irrelevant. The LDA algorithm did not name the topics (they were simply lists of words) so we gave each one a name based on its contents, as is customary when discussing topic modeler output. Table 2 summarizes the 20 topics produced by the model, and the subsequent tables provide the word lists for five relevant and five irrelevant topics.

*Table 2: Summary of Generated Topics (High-Relevance Topics in* ***Bold****)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Structure** | Discussion | Legal Terms | Names | **Investing** |
| Names 2 | Communications | Recreation | Conversation | **Regulations** |
| Names 3 | **R&D** | IT | **Energy Transport** | Legal Terms 2 |
| Names 4 | **Business Terms** | IT 2 | **Business Partners** | Names 5 |

*Table 3: Stemmed Words from Five Topics of High Relevance*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Regulations** | **Investing** | **R&D** | **Energy Transport** | **Business Partners** |
| generat | million | futur | capac | citi |
| plant | stock | research | pipelin | permian[[2]](#footnote-2) |
| particip | news | trader | volum | tenn |
| polici | share | link | area | east |
| public | invest | weather | transport | elpo |
| govern | online | model | deliveri | nymex[[3]](#footnote-3) |
| ferc[[4]](#footnote-4) | fund | valu | hour | summer |
| industri | secur | natur | germani | tetco[[5]](#footnote-5) |
| suppli | technolog | data | total | west |
| regulatori | quarter | analysi | outag | socal |
| board | world | carrfut[[6]](#footnote-6) | firm | gate |
| competit | network | test | storag | sonat[[7]](#footnote-7) |
| member | earn | differ | data | appalac |
| action | billion | spread | nomin | questar4 |
| nyiso[[8]](#footnote-8) | analyst | high | flow | oklahoma |
| general | save | average | natur | winter |
| feder | investor | paper | meter | appalach |
| regul | money | forecast | paso | shpchan |
| committee | profit | chart | pool | hehub |
| nation | york | contain | peak | demarcat |

*Table 4: Stemmed Words from Five Topics of Low Relevance*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Names 2** | **Personal** | **Recreation** | **Conversation** | **Discussion** |
| kate | check | game | peopl | talk |
| larri | hotmail | play | thing | sure |
| syme | bass | sunday | realli | morn |
| lesli | home | season | littl | user |
| matt | yahoo | ticket | even | soon |
| elizabeth | night | event | friend | tomorrow |
| doug | weekend | pass | never | possibl |
| sager | travel | sign | much | move |
| stephani | great | player | tell | thought |
| campbel | matthew | fantasi | great | present |
| baughman | room | miss | life | done |
| stacey | lenhart | place | didn | suggest |
| jason | vacat | expect | school | peopl |
| karen | visit | agent | went | understand |
| clark | love | center | better | feel |
| russel | card | yard | recruit | feedback |
| murphi | hotel | holiday | true | appreci |
| gilbert | dinner | activ | love | problem |
| counterparti | happi | saturday | everi | status |
| roger | holiday | practic | around | password |

All of the topics, even those of low relevance, display sound coherence, indicating that the model has clustered the documents effectively. The high-relevance topics are filled with terms that are likely to be of use to an investigator, such as the names of regulatory agencies, business partners, and terms related to investments. Notably, five of the low-relevance topics consist almost solely of first or last names. At a glance, these do not appear to be very useful. However, these groupings may indicate people who communicate with each other frequently, and thus could be useful as a starting point for a network analysis.

We also note that only a minority of the generated topics would be relevant to an investigation. This is not a failure of the model, but rather is to be expected. In document review and discovery processes, it is often the case that a small number of documents will be relevant. Similarly, in the Enron corpus, one might expect a small fraction of the emails to contain useful data for an investigation; many emails might simply lack necessary detail, refer to aspects of work unrelated to criminal business practices, or discuss non-work matters. The topics generated by LDA reflect what is likely a lower proportion of relevant emails to review.

**Improving the Model**

We attempted to improve the output of the LDA algorithm by adjusting the number of topics and changing the term weighting for the vocabulary. We started by focusing on the number of topics. We initially chose 20, viewing this number as generating a manageable amount of data to review. We tried increasing this number to a 50 topics to see if the models generated additional useful topics or increased intra-topic cohesion. The results were in fact discouraging, as the additional topics tended to yield more lists of names or common terms used in conversation and business emails, rather than more relevant terms. In addition, LDA runtime increased roughly linearly with additional topics, up approximately five hours for 50 topics.

We then tried implementing term frequency-inverse document frequency (TF-IDF) weighting to test its effect on the topic outputs. LDA normally uses a bag-of-words model and whole-number term-frequency counts. However, the MLLib implementation of LDA accepts fractional term counts in its input, which allows us to use TF-IDF. The potential benefit of TF-IDF is that it might help LDA identify important words for the topic model and ignore less useful words. Our existing LDA code already calculated term frequencies, so we created an additional RDD in Spark to calculate IDF and used it to divide the term frequencies. The results from using TF-IDF were initially encouraging, but the algorithm only generated five useful topics, two fewer than the baseline method. The model offered an improvement by creating fewer topics that consisted solely of employee names, but it also created useless topics containing pleasantries and words commonly found in emails (e.g., please, thank, sincerely, etc.).

After running LDA under different parameters, we concluded that our original model with 20 topics offered the best combination of useful output and efficient calculation time. Additional topics and TF-IDF weighting offered little to no improvement and increased processing time.

**Implications and Conclusion**

Topic modeling on the Enron dataset ran in approximately two hours using a relatively small cloud-based cluster. This is a small fraction of the time and cost required to manually review documents or use a machine-assisted technique such as predictive coding. Topic modeling is not intended to replace a thorough review process; instead it offers an efficient method for exploring a document corpus, and can be synergistically combined with other technologies and review methods.

One use of the topic model is to generate terms to search the document set. In the discovery process, document searches may be used to reduce the set to a more manageable size, after which manual review or predictive coding can be conducted. An examiner must first determine the search terms, which can be challenging when facing a large, unknown document set. The topic model provides a starting point, as all of the words it returns are found in the document set. The topics themselves can also lead to additional search terms; a topic containing words related to energy regulation, for example, may suggest that other terms related to the topic will be fruitful. Even if an examiner has already conducted exhaustive searches, a topic model can help describe the contents of the document set and shed light on document clusters of interest that have yet to be discovered by searching.

The topic modeler and the underlying architecture we constructed can also be extended to create an iterative approach to reducing the document set while minimizing human intervention, saving both time and cost. In addition to generating topics, LDA can also return a set of documents most strongly associated with a given topic. This can be combined with a cloud-deployable search engine, such as ElasticSearch, to identify documents from relevant topics. The topic modeler can be run again on the subsets for each topic, generating smaller and more specific document sets for review. This approach will scale well to larger document sets, as more cloud resources can be marshaled proportionally to the corpus size.

In sum, topic modeling provided an efficient method to explore a document set that was too large to be checked by human reviewers in a reasonable amount of time. The LDA algorithm can work in concert with machine-assisted review, or can be used on its own to identify relevant documents. As the inevitable growth of digitally stored data makes human participation in the discovery process increasingly difficult, unsupervised methods running on an elastic architecture can help describe a corpus and identify relevant documents within relatively short timeframes, leaving more time and resources for experts to review a smaller subset of potentially relevant documents.

**Works Cited**

Bordon, B. (2010, October 1). The Demise of Linear Review. Retrieved from http://upc.utah.gov/materials/2015civil/The\_Demise\_of\_Linear\_Review.pdf

Baron, J. R. (2011). Law in the Age of Exabytes: Some Further Thoughts on 'Information Inflation' and Current Issues in E-Discovery Search. *Rich. JL & Tech.*, *17*, 9-16.

Barry, N. (2012). Man versus machine review: the showdown between hordes of discovery lawyers and a computer-utilizing predictive-coding technology. *Vand. J. Ent. & Tech. L.*, *15*, 343.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, *3*(Jan), 993-1022.

de Waal, A., Venter, J., & Barnard, E. (2008, January). Applying topic modeling to forensic data. In *IFIP International Conference on Digital Forensics* (pp. 115-126). Springer US.

Enron Email Data. (2012, February 14). Retrieved from https://aws.amazon.com/datasets/enron-email-data/

George, C. P., Puri, S., Wang, D. Z., Wilson, J. N., & Hamilton, W. F. (2014, May). SMART Electronic Legal Discovery Via Topic Modeling. In *FLAIRS Conference*.

Information Released in Enron Investigation. (2013, April 1). Retrieved from http://www.ferc.gov/industries/electric/indus-act/wec/enron/info-release.asp

Iwata, E. (2002, February 18). Enron case could be largest corporate investigation. *USA Today*. Retrieved from http://usatoday30.usatoday.com/tech/news/2002/02/19/detectives.htm

Linder, D. O. (2014). The Enron (Ken Lay and Jeff Skilling) Trial: An Account. Retrieved from http://law2.umkc.edu/faculty/projects/ftrials/enron/enronaccount.html

Remus, D. (2014). The Uncertain Promise of Predictive Coding. *Iowa Law Review*, *99*, 101.

1. Lehman Brothers was the fourth-largest investment bank in the United States when it declared bankruptcy in 2008 as a result of the subprime mortgage crisis. The bank’s collapse triggered investigations into its financial practices. [↑](#footnote-ref-1)
2. Bank [↑](#footnote-ref-2)
3. New York Mercantile Exchange [↑](#footnote-ref-3)
4. Federal Energy Regulatory Commission [↑](#footnote-ref-4)
5. Natural gas company [↑](#footnote-ref-5)
6. Futures company [↑](#footnote-ref-6)
7. Energy holding company [↑](#footnote-ref-7)
8. New York Independent System Operator [↑](#footnote-ref-8)