Machine Driven Patent Classification

Columbia University, Data Science Institute
Gabrielle Agrocostea
Francisco Javier Arceo
Abdus Samad Khan
Justin Law
Tony Paek

Abstract

United States Patent Trade Office (USPTO) categorizes patent applications according to the Cooperative Patent Classification(CPC) system. However, they are limited in their ability to identify underlying technologies to map relationships among patents. In our project we apply Latent Dirichlet Allocation, a generative probabilistic model which learns latent thematic structures in textual data, to infer underlying topics for the USPTO patents in 2014. The benefits from applying such models to patents are twofold, first it provides the ability to map a single patent to multiple latent classifications (i.e.,different topics), second it allows us to find relationships between patents by comparing their assigned topics. Additionally, we explore various performance measures and our results suggest that we can, in addition to learning an underlying subset of CPC categories, infer classifications that complement the existing CPC system through mixed membership models. Lastly, we leverage tools such as Python, Kibana, Elastic Search, and Shiny to visualize and validate our exploratory data analysis and model results.

Introduction

Patents are a set of exclusive rights granted by a sovereign state in exchange for a detailed disclosure of an invention. It protects the rights of inventors, often in the form of businesses, which require significant time and resources. Patents are invaluable assets of modern businesses, and it often constitutes a majority of a company's value. Companies such as IBM and Samsung possess more than tens of thousands of patents to gain comparative advantage and create barriers to entry for followers in the market.

As the number of patents filed every year is in the size of millions, organizing patents in a meaningful way becomes an increasing challenge. Entities that utilize a set of technologies should be able to search through historical patent data to see if there is any breach or an opportunity to file for a new patent, and the publicly disclosed nature of patents makes this data a great asset for investors hoping to gain incremental insights about businesses/industry of interest.

The USPTO provides CPC system (current) and USPC (United States Patent Classification) system (past) to categorize and group similar patents together. While the adoption of CPC was an improvement that enabled the patent office to adopt to the ever-changing trend of patents, it still has fundamental shortcomings. First of all, a patent gets mapped to one specific class, which means that any relationships among classes or the multifaceted nature of many of modern patents is disregarded. Secondly, the classification of patents is manually done by examiners, which is prone to subjective judgement and human errors.

This project explored estimating a topic model by utilizing Latent Dirichlet Allocation to automatically learn the underlying topics of patents and infer relationships to other patents to gain additional insights about the publishers of the patents and the patents themselves.

Data

Data Description

The original dataset consists of a set of zip files, the aggregate size of which amounts 250GBs. The unzipped files within each zip file consist of a concatenated list of xml-formatted files, each xml of which represents one patent. There are three main formats: SGML for year 2001, XML for 20022004, and ICEXML for 20052015. Within them, the attributes of each patent of interest to our project were identified to be the following: abstract, title, description, and claims. The abstract is a summary of the patent, the title identifies a name given to a particular patent, description contains the details of patents, and the claims define the extent, the scope and the protection conferred by a patent.

Preprocessing

As the scope of our project was confined to classify the utility patents, the patents were filtered so that only the patents with the utility application are retained (definition of all classes is given in the appendix). Thereafter, for the abstracts and the description of each patent, the contents were further processed so that only the textual elements of the attributes are retained. This step ignored the paragraph structures of descriptions and abstracts, since it does not affect the subsequent computation as topic modelling algorithms such as LDA utilizes a term-document matrix as the data structure for the input corpus.

Preprocessing Details

The original processing of each of the individual XMLs was done in a two-stage fashion where (1) each XML was parsed into smaller XMLs representing each individual utility patent and (2) each utility patent was parsed to retrieve only the textual elements and stored into a csv file for building a document-term matrix. The preprocessing of dataset was computationally intensive and as it required iterating through every single line of an XML file. The average processing time for each year was approximately 12 hours. While the computation time was long, this did not present a significant obstacle as the preprocessing was required only once. Lastly, while processing these XMLS into a CSV was a requirement for building the document-term matrix (input to our models), it was not useful for Exploratory Data Analysis (EDA). For EDA we employed a separate pipeline to parse and explore the data using Elasticsearch.

Exploratory Data Analysis

Elasticsearch

Elasticsearch was explored as a means to explore the massive dataset at hand. To repeatedly perform basic data exploration tasks on documents stored in flat files such as in XML or CSV formats is computationally-intensive because of the lack of indexing and the inability to extract only the specific fields. Scanning through the entire dataset is constrained not only by the memory available for the project (maximum 16GB RAM) but also the simple I/O limitation of having to read all files from disk every time.

Elasticsearch is an open-source distributed database that specializes in text documents while also functioning as a full-text search engine. It stores data only in JSON format which is similar to NoSQL databases where documents have a flat structure without tabular relations. When indexing documents, Elasticsearch computes the inverted index (or document-term matrix) of terms to enable fast retrieval of documents based on a string query. Elasticsearch also supports standard text preprocessing methods such as tokenization, stemming, stopwords, ngrams, and synonyms.

The Elasticsearch server was hosted on a threenode Ubuntu server on a Microsoft Azure free trial. Due to the limitations of the trial, the three virtual machines, each of which have 2 core processors with 3.5GB memory, were below the standard hardware specifications recommended for Elasticsearch. We initially considered three years of patent data to index (907k documents, using about 150GB of data within the server), but later decided to deal only with one year worth of data due to computational constraints faced in building a topic model. The amount of data we considered has been adequate for our needs and sufficient for basic data exploration and visualization through Kibana, which is an open source data visualization plugin for Elasticsearch that provides a website for users to interface with the data and produce plots and dashboards.

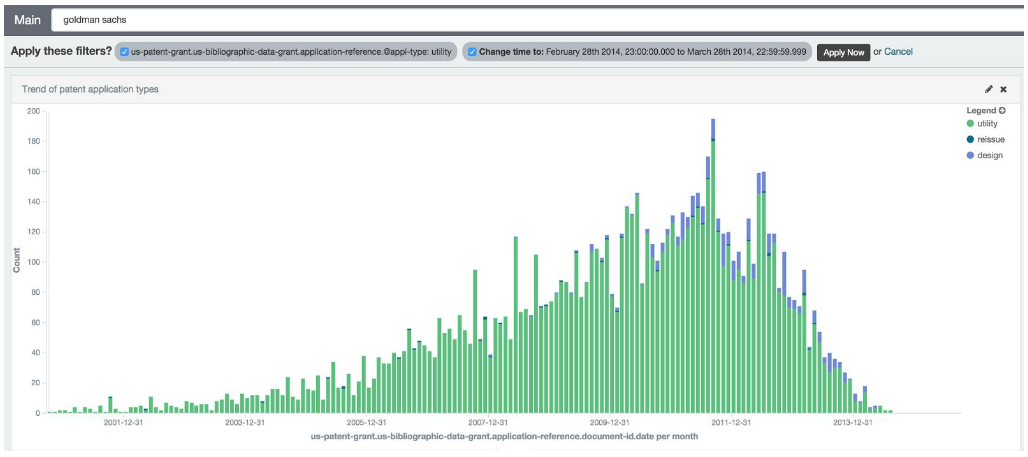
Main
goldman sachs
Apply these filters?
us-patent-grant.us-bibliographic-data-grant.application-reference.@appl-type: utility
Change time to: February 28th 2014, 23:00:00.000 to March 28th 2014, 22:59:59.999
Apply Now
or Cancel
Trend of patent application types
Legend
utility
reissue
design
Count
us-patent-grant.us-bibliographic-data-grant.application-reference.document-id.date per month

Figure 1: Number of Patents filed by Goldman Sachs across time on Kibana

EDA Observations

It is important to note that the last three years of data (2014-2012) was explored in Elasticsearch, patents issued only in 2014 were considered. All exploratory analysis and modeling presented in the subsequent sections is performed on this subset of data. There was a total of 327,012 patents in 2014. A vast majority of these patents were Utility patents (92%; Count: 301,641) and approximately 7.5% are design patents (7.2%; Count 23666) with the remaining 0.8% categorized as plant, reissue or sir (figure 2 below).

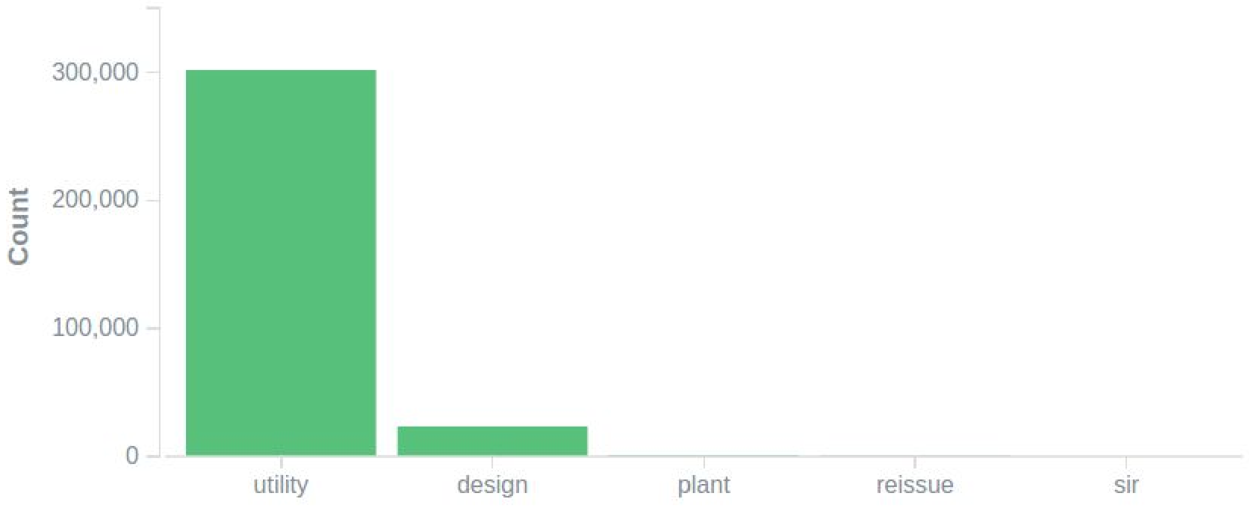

Figure 2: USPTO Classification Composition

Fields of Interest

Noted previously, the problem statement restricts us to use only the following attributes of the patents in the subsequent analysis: title, description, abstract and claims. These attributes formed the primary source of information in our analysis. It is crucial, therefore, to investigate each of these fields in detail.

Each of theses four, is a text field stored in UTF-8 encoding in the dataset. We computed average lengths, mode and percentiles respectively to get a sense of the center and the spread of the distributions across our dataset. The summary of the statistics is noted below.

Fields	Mean	Mode	25%	50%	75%	Range
Title	8	7	5	7	10	(1,74)
Abstract	110	140	81	113	141	(3,491)
Claims	73	30	26	41	82	(3,21338)
Description	8222	4000-5000	3711	6025	9672	(10,1261948)

Observations:
Description is the largest field with around 8000 words on average. We noticed an abnormal peak in the abstract field at 140 words. The following guideline provided by the USPTO explained the sharp decline in frequency after 150 words in the abstract field: “*The abstract shall be as concise as the disclosure permits (preferably 50 to 150 words)”*. We also noticed outliers in the claims and description fields. Relevant plots are given below.

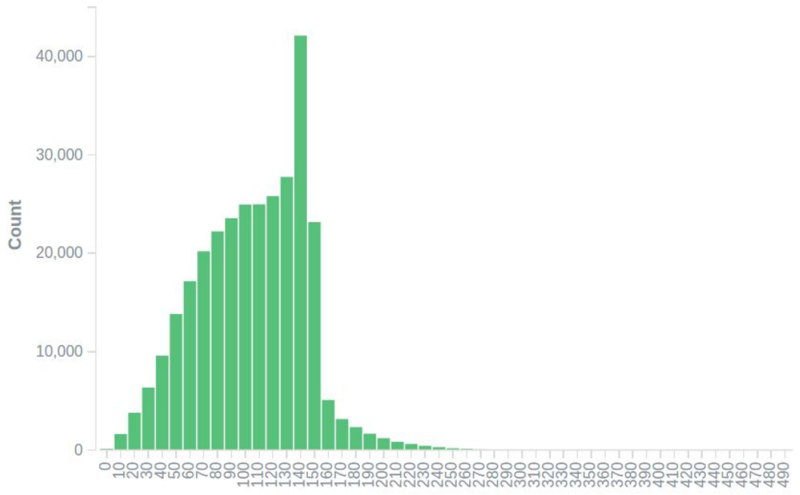

Figure 3: Length of Abstract

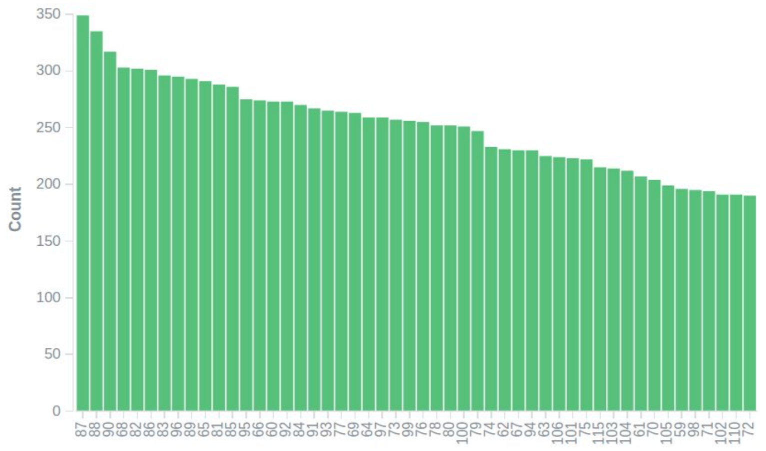

Figure 4: Length of Description

We also considered the distribution of unique terms in each field. We noticed generic terms relating to a particular field occurring with high frequency in that field, for example {method, include, claims, from, us, ...}. Such words have weak predictive power since they don’t vary which topics but are constant across words. A lot of these stopwords were removed manually. While the manual filtering of such words does not guarantee the removal of all stop words which do not add significant value to the contents of patents, we had reasonable confidence that topic modeling algorithms such as LDA were able to capture the insignificance of such words. A plot for abstract field is given below.

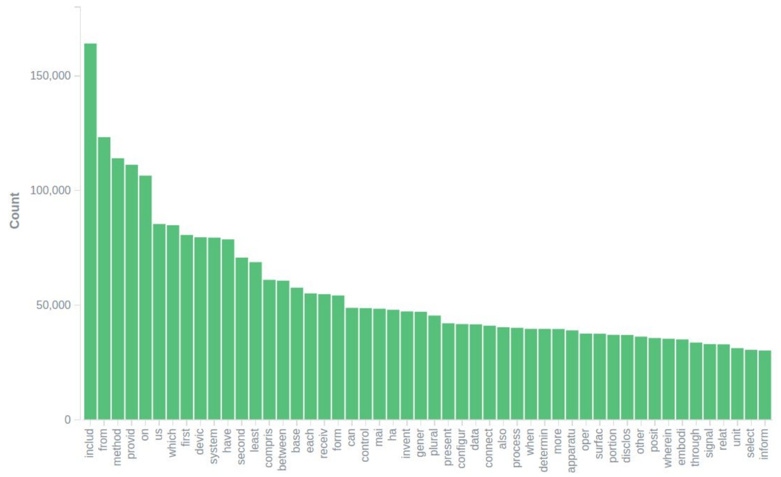

Figure 5: Frequently Occurring words in Abstract

Modeling

Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) is a well-known topic modeling technique, developed by David Blei, Andrew Ng and Michael Jordan. It is a generative probabilistic model which considers a document to be a bag-of-words, disregarding the sequence of words. In the model, each document is considered a bag of words with an associated distribution over topics, and each word is assumed to be a set of realizations from of its associated distribution. The associated distribution over topics is a mixture distribution since it is a distribution over topics and each topic is itself a distribution (over the vocabulary). An example of a topic and a document is given below:

Each topic is a multinomial distribution over words

Sample Topic (0.141,0.0179,0.0177,0.0154,0.0121,0.0117,...)
(color, colors, image, white, green, values,...)

Each document has an associated multinomial distribution over topics

Sample Document (0.01, 0.01, 0.36, 0.02, 0.01, 0.002,...)
(0, 1, 2, 3, 4, 5,...)

The model makes a generative assumption of the underlying structure of the corpus. That is, it assumes that each document in the corpus is generated in the following way:

- For each document we have an associated distribution over topics.
- We first randomly sample a topic from this distribution.
- Next we sample a word from the topic distribution.
- Repeating this many times generates each word and hence one document.

Training the LDA model essentially involves learning these two sets of probability distributions through latent dirichlet distributions of topics producing each document.

LDA Hyperparameters

LDA takes as input the following parameters:

$\boldsymbol{K}$ the number of topics

$\boldsymbol{\eta}$ the dirichlet prior of the per topic distributions over vocabulary

$\boldsymbol{\alpha}$ the dirichlet prior of the per document distributions over topics

$\boldsymbol{\alpha} = (\alpha_1,\alpha_2,...,\alpha_K)$

$\boldsymbol{\eta} = (\eta_1,\eta_2,...,\eta_V)$

Since we are not assuming any prior knowledge about the topics or the vocabulary it makes sense to set $\boldsymbol{\alpha}$ and $\boldsymbol{\eta}$ as symmetric. That is

$\alpha_i = \alpha \ \forall \ i$

$\eta_i = \eta \ \forall \ i$

Alpha ($\boldsymbol{\alpha}$)

For each document in the corpus, LDA returns a distribution over the *K* topics

$\theta_i \sim$ Dirichlet($\boldsymbol{\alpha}$) for $1 \leq i \leq D$

Each θ_i has dimension equal to the number of topics. Tuning parameter α will control the softness of clustering done by LDA. A large value of alpha implies that each document is likely to be a mixture of most of the topics, and not any single topic. While, choosing a small alpha pushes LDA towards hard clustering.

Eta ($\boldsymbol{\eta}$)

For each topic, LDA returns a distribution over vocabulary *V*.

$\beta_k \sim$ Dirichlet($\boldsymbol{\eta}$) for $1 \leq k \leq K$

Each β_k has dimension equal to size of the vocabulary and is the distribution over this vocabulary. Each β_k defines a topic. The tuning parameter η controls the degree of distinction between topics. A large value of η implies a peaky distribution with one mode at the center of β_k simplex. This implies that each β_i sampled will be close to this point ($1/V$, $1/V$,… $1/V$). The implication of this is that each topic, β_i is a mixture of many words. On the other hand, choosing a small η tends to create distinct topics.

We choose small values of α and η particularly we set them both to $1/K$ since it has been empirically observed that values of α and η set to $1/K$ work well in practice.

Number of Topics (K)

Setting a correct number of topics is a crucial aspect of building a right topic model. Too small of a value for K would fail to generate topics for patents in specificity, whereas too large of a value for K would introduce topics that may be too similar to one another, and inevitably classify documents of a high level of similarity into different topics. While setting the right number of topics for K is considered an 'art', a number of performance metrics such as cosine similarity and KL divergence were explored to select the right number of topics that has the best tradeoff between the specificity and the orthogonality of of topics.

LDA Algorithm: Batch LDA vs Online LDA

There exists two variations of LDA algorithm: Batch LDA and Online LDA. The difference between the two is that in the case of Online LDA, the model is trained on chunks of data in an iterative fashion, where each chunk is treated as a subcorpus. The model is usually updated after each chunk is processed. On the other hand, the original LDA algorithm or Batch LDA, processes the entire corpus as one. This step is performed multiple times (multiple passes through the data) to ensure that the model converges. An advantage of Online LDA vs a batch version is that Online LDA requires multiple passes through the data. While One pass through the dataset is not nearly enough for a batch LDA, this was not the case for Online LDA. One downsize of Online LDA is that it does not work well when there is a lot of topic drift as subsequent updates the model in a less significant manner.

Models were trained utilizing both of implementations. Online LDA was run on a non-distributed setting using our local machines and a Microsoft Azure instance while the Batch LDA was run on Apache Spark.

Gensim Implementation

Gensim is an open-source library for topic modelling on Python. It includes a well defined API to handle large corpus and implementation of models such as LDA and LSA. The library also provides distributed implementation and multicore versions of these.

Gensim provides an Online LDA implementation of the algorithm. It is important to highlight that we would suspect a topic drift in patent data, since technology changes over time. As mentioned previously, that in such cases Online LDA might perform poorly. However, in analysis we only consider patents of a single year therefore we it is safe to assume that topic drift would be minimal.

The LDA algorithm in Gensim takes in a Gensim type corpus object. Gensim provides various corpus implementations to select which vary in memory requirements/performance. For our project we used serialized corpus. Serialize corpus involves creating an iterator which streams documents from a lines one at a time, rather than storing the whole corpus in memory, Where corpus is basically a term-document matrix, which is built in the following way: for each patent, each patent description is tokenized and indexed; i.e., each word is uniquely mapped to to a numeric index value, then the document index value and term frequency are recorded.

Gensim LDA Implementation Parameters

Gensim's LDA implementation requires the following parameters as inputs. A brief description of these inputs is given below.

chunksize	the number of documents to load in memory
update_every	the number of chunks after which we want to update
number_of_passes	the number of times to train on the whole dataset

For our project, we tested several set chunksizes, update_every, and number_of_passes. Ultimately, we chose the model with chunksize set to 20,000, update_every to 40,000 and number_of_passes to 10. A large number of updates is important otherwise we get a poor topics/classifications. Further, a chunksize value of 20,000 implies that at most 20,000 documents are loaded into memory which ensures a controlled memory footprint.

Microsoft Azure

Microsoft Azure is a cloud computing platform created by Microsoft. We utilized Microsoft Azure to run non-distributed version of LDA(Gensim) on our dataset. We mainly used an A4 configuration which is a single machine with 8 core processor, 14 GB ram. There were much higher configurations available (16 core) and we did some experimentation with them in the initial phases, however using them for our project was not feasible. During running the model we ran into two main problems.

Bug in Multicore version of LDA
Firstly, when we scale the model by incorporating more data LDA multicore version crashed. This bug is well known and has been experienced by previous researchers as well (https://github.com/piskvorky/gensim/issues/376). From our research, we found that there are no existing patches to fix this. To go around this we decreased our dataset from the initial 5 years of data to 1 year of data, and resorted to only using a single core version of the algorithm.

Slow computation on Azure
Secondly, we were surprised to find out that Azure server was slower than our local computers. Azure instance would take sometimes more than twice as much time to compute a similar model as it would take our local machine. Upon investigation we found that this was mainly due to disk IO bottlenecks. When we create a VM on Azure, the OS disk created with is not optimized for computation. Since we were using a steaming corpus, our computation would require continuous IO operations, which led to a slow runtime of topic modeling on Azure.

LDA Models on Spark

Spark is an Apache open source cluster computing framework that is rapidly gaining traction to perform data analysis and processing work on large amounts of data that a single machine would be difficult to deal with. In contrast to Hadoop's MapReduce model that goes through reading and writing data from disk, Spark uses a multi-stage in-memory framework that allows data to be processed at far greater speeds. This increase in speeds allow machine learning algorithms that requires the same data to be iterated through multiple times to be implemented at feasible execution times.

Although still considered to be in the experimental stage, LDA model is one of the available machine learning algorithms in Spark and was introduced back in March 2015 in Spark 1.3 and as of the current 1.5.2 release, has both EM (Expectation-Maximization) and VI (Variational Inference) optimizers. Spark also contains various data preprocessing functions to allow for easy generation of document term matrices from raw text. However, the python library in the current implementation still lacks the crucial CountVectorizer function that exists in Java/Scala for one-to-one mapping of each unique term into term indices, and instead, relies on a HashingTF function that doesn't guarantee unique indices and prevents exact retrieval of distribution of terms on topics.

The 8 cpu cores limitation meant that the best cluster configuration was 4 nodes (n1-highmem-2), with each node having 2 cpu cores and 13GB RAM each. The working memory available for computation was 30GB and as the LDA models in Spark are batch LDA models which processes the entire corpus at each iteration, this meant even using only 2014 patent descriptions (~300k documents) was constantly pushing the limits of the cluster. It turns out that the cluster was unable to finish processing under a parameter setting of k = 150 or more, at 50 iterations and with a vocabulary size of 100k. The vocabulary size setting here meant that only the top 100k terms in the corpus were kept while everything else were discarded.

Still, the time spent from going from raw text to completing the model at lower settings were impressive. For example, a model using k = 100 with 20 iterations and a vocabulary size of 100k can be completed within two hours from raw text. This could be attributed to the fact that each of a node has a solid state drive with a replication of the data which meant reading data into memory were three to four times faster compared to a single machine with a single drive. The iterations themselves took around 5 minutes each and this is also significantly faster considering the fact that it's processing the whole corpus at once.

As a reference for the broader capabilities of LDA models on Spark, Databricks ran topic modelling on 4.5 million Wikipedia articles with 100 topics and a vocabulary size of 10k, and were able to achieve iteration times of only 25 seconds per iteration using 16-node clusters with 8 cpu cores and 61GB memory per node.

Results

Visualization

Visualization of the model results was done through LDAViz, which produces a two-dimensional visualization of topics and the top-30 most relevant terms for each topic. This tool was helpful in validating the quality of topics and whether they made intuitive sense. Figure 6 provides an example of our model visualization.

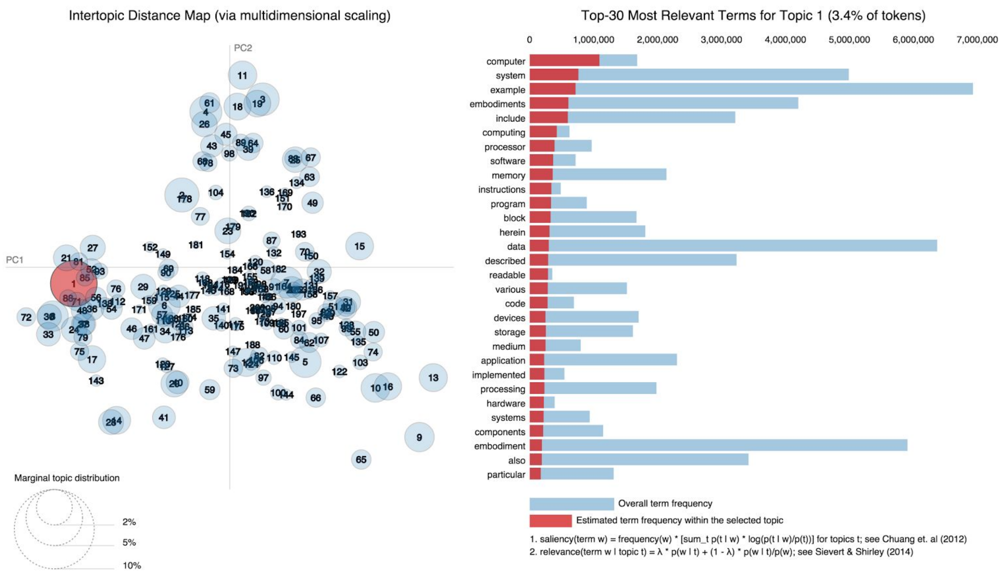

Figure 6: LDAViz for Patents

Performance Metrics

Average Length of words per topic

A crude measure of evaluating a topic model involves calculating an average length of words per topic. An intuition behind an average length of words per topic is that a topic with a higher average-length would indicate a more specific topic and a model with a higher average length of words per topic would indicate a model that has a more granular topic distribution. An equation for calculating an average length of words per topic is below.

$$\frac{\sum_{k=1}^{K}\sum_{w=1}^{W} WordLength_{k,w}}{KW}$$

Normalized Matrix Correlation/Cosine Similarity

The normalized matrix correlation/cosine similarity is captured by taking the 50 most probable words (and their weighting) for each and collecting the unique set of *L* words. We then vectorize the final collection of words and compute the cosine similarity and correlation for each topic across every other topic. Here we have calculated an (*L* x *L)* similarity matrix. Next we take the sum of the lower diagonal of this matrix and normalize it by (n^2 -n)/2, since this sum is bound between -(n^2-n)/2 and (n^2-n)/2, we can simply normalize this and we now have a metric bound between -1 and 1 that tells us the similarity of the matrix. Higher in absolute value tells us that the topics are more correlated.

KL Divergence

While the normalized matrix cosine similarity measures capture the orthogonality of topics by looking at the topic-word matrix, it does not incorporate the document-topic matrix. The Arun et al measure overcomes this deficiency by incorporating the document-topic matrix and calculates KL divergence of a document-topic matrix to a topic-term matrix. The abstract of a paper summarizes the measure succinctly below.

In proposing this measure, we view LDA as a matrix factorization mechanism, wherein a given corpus C is split into two matrix factors M1 and M2 as given by Cd w = M1d t x Qt w. The quality of the split depends on "t", the right number of topics chosen. The measure is computed in terms of symmetric KL-Divergence of salient distributions that are derived from these matrix factors. We observe that the divergence values are higher for non-optimal number of topics – this is shown by a 'dip' at the right value for 't'.

Although LDA is a probabilistic generative model, this measure assumes a document is generated by a non-negative factorization of topic-word matrix of M1 and a document-topic matrix of M2. Thereafter, singular values of M1 are computed by singular value decomposition in order to compute the distribution of variance in topics C1 and a distribution of of L*M2 is computed (C2) where L is the length of each document.

$$ProposedMeasure(M1, M2) = KL(C_{M1}||C_{M2}) + KL(C_{M2}||C_{M1})$$

CPC Similarity

The CPC similarity metric was a measure that took the 50 most significant terms from Elasticsearch for each CPC category and evaluated the overlap for each topic, and summed across all topics. This performance metric essentially indicates the degree of overlap between the learned topics of documents and the existing CPC classifications of patents.

Summary of Model Performance Metrics

	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6
Number of Passes	10	1	1	1	1	1
Number of Topics	200	200	200	150	100	50
Updated Every	40,000	1	1	1	1	1
Chunksize	20,000	5,000	500	5,000	5,000	5,000
Corpus Size	301,591	301,591	301,591	301,591	301,591	301,591
Number of CPUs	1	1	7	1	1	1
Topic Correlation	7.57%	0.34%	0.60%	2.29%	0.81%	0.50%
Topic Cosine Similarity	7.57%	0.34%	0.60%	2.29%	0.81%	0.50%
CPC Similarity	3.86%	1.89%	2.29%	2.80%	2.35%	2.00%
Abstract CPC Similarity	0.30%	0.17%	0.20%	0.24%	0.20%	0.18%
Average Word Length	6.78	7.20	7.01	6.84	7.05	7.18
KL - Divergence	2.342	2.925	2.813	2.882	2.325	1.423

New classification prediction using model results

There were also efforts to determine whether the model results were capable of assigning existing classifications to patents that were not assigned the same classification before. This is a different perspective of evaluating models but is actually more similar to evaluation methods currently used in research for unsupervised models. Researchers currently read through the terms within generated topics to identify the qualitative performance of the topics. The downside of this approach is that evaluation is subjective and difficult to determine which results are better. We used the following methodology:

1. Identify generated topics that possess keywords with a high overlap with the significant terms for each USPC class extracted using Elasticsearch (based on mean average precision, MAP).
2. For the topics that have been identified (> 0.35 MAP), find the patents that have been assigned a high weight to that topic (>0.8).
3. Check if these patents have already been assigned the USPC class that matches the topic. If not, then this may indicate that the USPC class should be actually assigned to this patent, thus generating a new classification that did not previously exist.

This method proved successful for some cases but unsuccessful for the majority that were reviewed. An example of a case in which this did work was a prediction of class 399 on patent 08668197. This patent describes an invention related to a sheet conveying apparatus which is why the original patent classification has only USPC class 271 which is sheet feeding or delivering. However, the patent also describes the use of this sheet conveying apparatus for an image forming apparatus such as copying machines and printers. So, one could make an argument that the patent should easily be classified as class 399 instead of class 271.

A case in which this method did not work was between USPC classes 257 and 438. The former is about active solid-state devices while the latter is about semiconductor device manufacturing processes. Most of the keywords between these two classes have a heavy overlap because they are both about semiconductors. Although class 438 does contain additional keywords that pertain to the manufacturing process that should be able to distinguish between these two classes, the weight (and thus appearance) of these terms were lower than the ones purely about semiconductors to allow the topic model to really separate these two apart.

This inspection suggests that keyword-based models may not be sufficient to distinguish classes that are of very similar in descriptive nature because it only depends on the words that appear in the document. Keywords that do not appear likely play a higher role in this case but the current state of research does not possess this meta-information to identify missing keywords.

Conclusion and Future Works

The exploration of various evaluation metrics, excluding the KL divergence, provided little guidance on which model to choose, yet this was no surprise since none of the metrics proposed could provide an explicit objective function to minimize or maximize. Rather, the evaluation metrics provided insight about the orthogonality of the topics, the similarity of topics to the most important words from the CPC classification system, and the information content (i.e., average word length) of topics. The KL divergence does, in fact, provide a loss function to minimize, yet it seems to favor simpler models with less topics that have little similarity to the CPC significant terms, which is inconsistent with our project objective and prior knowledge. Thus, our criteria for model selection is based on a subjective, holistic evaluation of all of these metrics. We ultimately decided on the model with 10 passes that had the highest CPC Similarity and Topic Correlation; additionally, this model was preferred most by the KL divergence for 200 topics.

While our methodology was able to successfully build a topic model that can complement the CPC classification system, the model performance does not go without its caveats. In many cases, the topic classification fails because the topics learned are sensitive to language conventions used in patent descriptions that span multiple fields. Given the current state of unsupervised learning, meta-information that could be helpful about patents cannot be taken into account to mitigate this problem and so we can only conclude success in patent classification through Latent Dirichlet to an imperfect degree. Caveats aside, we feel that our results do provide incremental value in complementing the CPC classification and can continue to provide insight on latent relationships to other patents based purely on the textual description of the patents.

References

1. Arun, R., V. Suresh, C. E. Veni Madhavan, and M. N. Narasimha Murthy. "On Finding the Natural Number of Topics with Latent Dirichlet Allocation: Some Observations." Advances in Knowledge Discovery and Data Mining Lecture Notes in Computer Science (2010): 391-402. Web.
2. Blei, David M., Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation." the Journal of machine Learning research 3 (2003): 993-1022.
3. Hoffman, Matthew, Francis R. Bach, and David M. Blei. "Online learning for latent dirichlet allocation." advances in neural information processing systems. 2010.
4. Řehůřek, Radim. "Gensim: Topic Modelling for Humans." Gensim: Topic Modelling for Humans. N.p., 7 Nov. 2015. Web. 22 Dec. 2015.

Appendix

USPTO Classification

The U.S. Patent and Trademark Office (USPTO) defines classification for patents based on its content. Each patent is classified as one of the following 6 categories:
Utility, Design, Plant, Reissue, DEF and SIR.Definition of each of these categories as on USPTO's website is given below:

Utility Patent
Issued for the invention of a new and useful process, machine, manufacture, or composition of matter, or a new and useful improvement thereof, it generally permits its owner to exclude others from making, using, or selling the invention for a period of up to twenty years from the date of patent application filing.

Design Patent
Issued for a new, original, and ornamental design embodied in or applied to an article of manufacture, it permits its owner to exclude others from making, using, or selling the design for a period of fourteen years from the date of patent grant.

Plant Patent
Issued for a new and distinct, invented or discovered asexually reproduced plant including sports, mutants, hybrids, and newly found seedlings, other than a tuber propagated plant or a found in an uncultivated state, it permits its owner to exclude others from making, using, or selling the plant for a period of up to twenty years from the date of patent application filing.

Reissue Patent
Issued to correct an error in an already issued utility, design, or plant patent, it does not affect the period of protection offered by the original patent. However, the scope of patent protection can change as a result of the reissue patent.

Defensive Publication (DEF)
Issued instead of a regular utility, design, or plant patent, it offers limited protection, defensive in nature, to prevent others from patenting an invention, design, or plant.

Statutory Invention Registration (SIR)
This document replaced the Defensive Publication in 198586 and offered similar protection.

According to a report by USPTO, approximately 90% of the patent documents issued by the USPTO in recent years have been utility patents.