Topic Model Analysis using Latent Dirichlet Allocation

Vinay Bysani vkb20 | Susmitha Batchu sb1543 | Nikita Balakrishnan nb580 | Srinivas Gaddam sg1240

Abstract

We intend to build a model to find topics from unlabeled document collection. In this growing age of digital data, labelling documents/data by human is not feasible. Automatic topic modelling is the key to organize huge amount of unstructured semantic data. Topic model analysis helps companies or organizations find structure in data and organize based on discovered topics and suggest customers based on his interests or usage patterns. Weve chosen NIPS dataset, which is a collection of all papers from NIPS Machine learning conference between 1987-2016. Once we discover topics among all the documents, we intend to analyze how topics evolved over time and predict future trends for topics. This gives us information on how subjects have been discussed and when new topics began taking shape among scientific community.

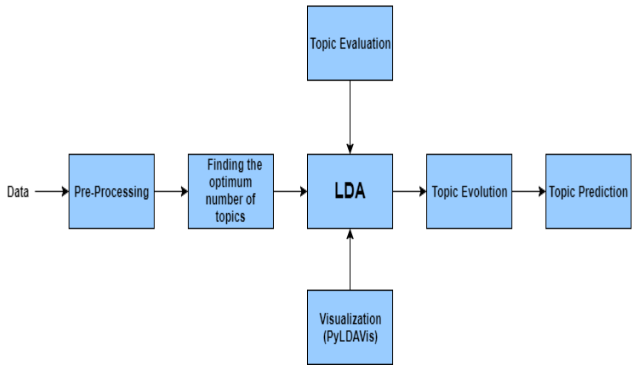
1. Prior Work

Topic model analysis is an efficient way to make sense of large volume of documents/text. It is the process of learning thematic structure from large document collections without human supervision. This has a major role in search engines. Search engine rankings are highly corelated with topic model analysis. While there are many types of topic modelling, the most commonly used and efficient model is LDA (Latent Dirichlet Allocation). Currently, Google is heavily investing on implementation of LDA using parallel computing.

LDA is a generative probabilistic model in which, each document is treated as a mixture of topics, where each topic is considered to be a distribution over a fixed vocabulary of terms. LDA is a mixture model, hence the topic clusters overlap (soft clustering) rather than being separated into distinct groups.

Most topics including LDA use bag-of-words(BOW) representation which assume that words are generated independently and hence, ignore phrases which may be crucial in some contexts. Topical N-grams is one such model which extracts topical phrases along with topics . However, including phrases will increase the complexity of the model and for simple Topic Modeling, BOW is good enough representation of data. Topics Over Time(TOT) is another variation of LDA which uses word co-occurrences along with the document time stamp to determine the topic of a document. In this model, each topic is associated with a continuous distribution of timestamps . We intend to use LDA with Bag of Words, as phrases will not have significant frequency compared to single word frequency(unigram), for topic modeling to first discover topics and then study the evolution of topics over time.

2. Technical details (theory/model):



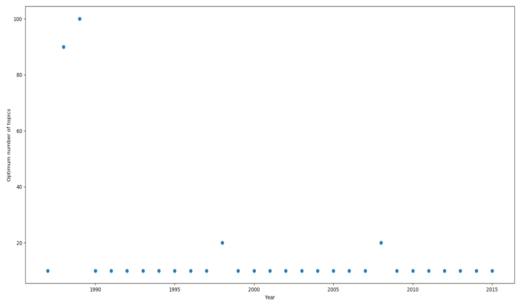
2.1. Number of topics:

In any clustering algorithm, knowing the right cluster size is vital. Optimum Cluster size majorly depends on dataset and how data is distributed. We have considered two metrics to determine the optimum number of topics
1. Silhouette Coefficient
2. Calinski Harabaz index

Silhouette Coefficient: This metric is used to study cohesion and separation of clusters. This metric takes into consideration, the inter-cluster distance and intra-cluster distance. The formula is given by (b-a)/max(a,b) where, a stands for intra-cluster distance and b is distance between data point and nearest cluster. Silhouette coefficients value ranges from [-1, 1]. Negative values near -1 mean that the data point is clustered wrongly. Zero value implies overlapping clusters. Optimum value is +1.

Calinski Harabaz index:CH index is another heuristic which is used to know how many clusters define the entire distribution. It is the ratio between within-cluster and between-cluster Dispersion. There is no acceptable cutoff for CH index, but it can be relatively compared and the higher the value, the better the solution. CH index is appropriate to use, where clusters are spherical in nature.

In this project, weve grouped documents based on year and calculated both the metrics (Silhouette Coefficient and Calinski-Harabaz index) for various cluster sizes. Weve plotted the optimum number of topics, based on the above metrics for each year as the below image. As can be seen from the graph, 10 is the optimum number of topics for this dataset.

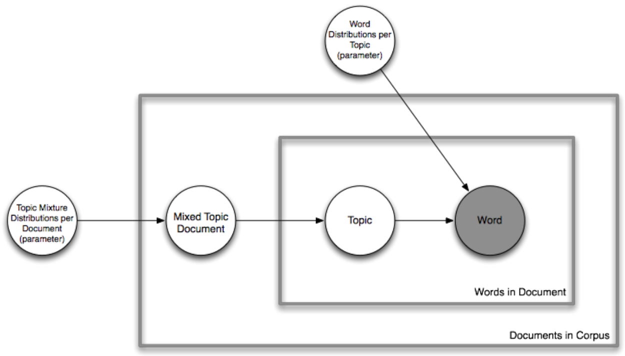


2.2. Preprocessing:

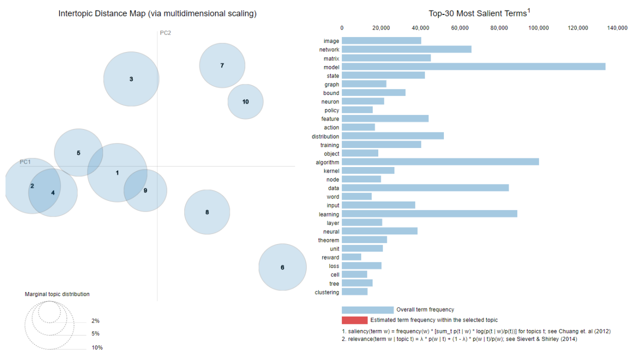
NIPS documents do need certain amount of preprocessing. The quality of output often relies on quality of input. The preprocessing techniques we used are stop word removal and Lemmatization. Stop words are those words that appear frequently across all documents, and tell us very little about the topics or context. This also reduces memory needed to store the data by huge factor. Lemmatization in linguistics, is the process of grouping words that have same origin. For ex: Talk, Talking, Talks all have same origin word Talk, which is called lemma. Since the nature of document is scientific, any sort of mathematical symbols need to be excluded. Using bigrams might yield better topic analysis as phrases are often critical in capturing the meaning of text, for example in Machine Learning, Machine and Learning are two words representing single topic. We appended bigrams to bag of words to include phrases but none of the top words contained phrases as they are less frequent compared to unigrams (single words).

2.3. Topic Discovery:

We have documents which do not have labels, finding appropriate topic for each document is very important and further steps rely on how well, we classify each document and label it. This problem can essentially be thought as a clustering algorithm clustering both words and documents. One more assumption is that a document can belong to multiple topics, ex: news article on Nasa space rover can fall under Science and Tech category and World News category alike. So, our aim is to build a mixture model, with soft clustering. One of the widely successful model in topic modelling is Latent Dirichlet Allocation (LDA). There is another variant of LDA called hierarchical LDA (hLDA), where topics can be viewed as a hierarchy or a tree. However, given the nature of NIPS dataset, which is very strict to Machine Learning, we chose to move further with LDA. LDA is a probabilistic model. In LDA, only the number of topics are given as input. It is a bag-of-words model. BOW is good enough for topic analysis. The only features the model sees are the words in documents, and there are lots of documents. The other features are latent or inferred. Each word will belong to a topic and hence, each document is a mixture of topics. LDA has three elements (Word, Topic and Context). LDA is majorly tuned using two hyper parameters: α - per document topic distribution, β - per topic word distribution. LDA model outputs probability distribution vector (over topics) for each document.


Word Distributions per Topic (parameter)
Topic Mixture Distributions per Document (parameter)
Mixed Topic Document
Topic
Word
Words in Document
Documents in Corpus

From the above metrics, weve used LDA to cluster all documents into 10 topics. We used pyLDAvis for interactive visualization of the topic clusters. Each topic is numbered 1-10, and since LDA is mixed model, we can see how topics overlap with each other and can infer how similar or dissimilar the topics are. This is a dynamic HTML page, and when we click on each topic, the top word distribution (top words of that topic) is shown in the below section.



There are various LDA frameworks available in the market. Some of which are:
1. lda-c - C implementation
2. GibbsLDA++ - C++ implementation
3. MALLET - Java implementation
4. plda - C++ parallel implementation

We use lda from genism for the following reasons.
1. Gensim LDA algorithm is streamed
2. It runs on constant memory. Irrespective of training corpus size, memory footprint is constant
3. It is distributed can make use of multiple machines to run parallely.

2.4. Topic Labelling:

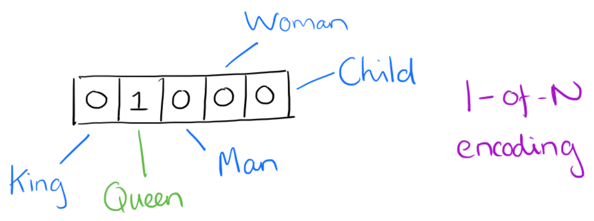
We start with the top 10 words for each of the 10 topics and start the topic labelling process with this.

Label generation

Word embedding - How does it work:

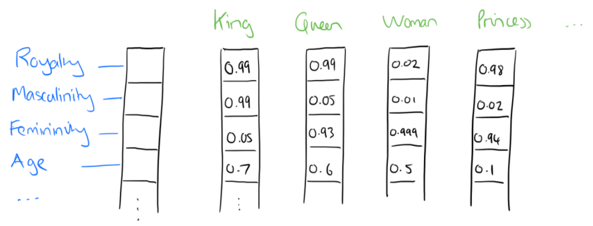
Word embedding is an important concept used for topic labelling. We have used Word2Vec in our project, to do word embedding. It is a method of extracting features out of text so that we can input those features into a machine learning model to work with text data. They try to preserve syntactic information.

So, lets take an example to understand the underlying process for word embedding.

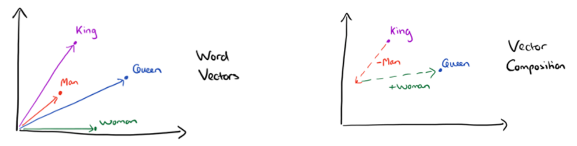


So, initially the word vectors for each of the above 5 words will be in this format considering there are only 5 unique words totally in the dataset. (1 Hot vector encoding) .The above vector is the vector representation for Queen before training.

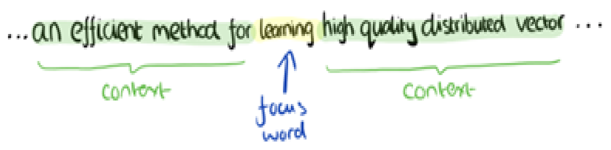
But after a few iterations of word embedding methods, the vectors will look like the ones represented below. The vector size will be equal to the number of unique words in the dataset. Each word is represented by a distribution of weights across all the elements in the vector. So instead of a one-to-one mapping between an element in the vector and a word, the representation of a word is spread across all of the elements in the vector, and each element in the vector contributes to the definition of many words. Such a vector comes to represent in some abstract way the meaning of a word.



Now consider a question like Man is to Woman as King is to ? This can be answered easily by King - Man + Woman = Queen (Vector operations) Check the figure given below for a pictorial representation of this case



So now, every word has a vector representation. Imagine a sliding window over the text, that includes the central word currently in focus (learning) , together with the four words that precede it, and the four words that follow it.


...an efficient method for learning high quality distributed vector ...
context
focus word
context

There are 2 types of word embedding CBOW (Continuous Bag Of Words) and Skip-gram.

There are 3 layers used - Input, hidden and Output.

CBOW : The training objective is to maximize the conditional probability of observing the actual output word (the focus word) given the input context words (input layer), with regard to the weights. In our example, given the input (an, efficient, method, for, high, quality, distributed, vector) we want to maximize the probability of getting learning as the output. We get a hidden layer by multiplying each input vector by a weight matrix (W1). Now this hidden layer is multiplied by another weight matrix (W2 - can be used to compute a score for each word in the vocabulary) to obtain the final output layer.

Skip-gram : Its just the reverse CBOW process. It is constructed with the focus word as the single input vector, and the target context words are now at the output layer. The training objective is to minimize the summed prediction error across all context words in the output layer.

So, after a lot of iterations through our corpus, we get trained vectors for each word that preserve the syntactic information.

The type of word embedding used by us in word2vec is skip-gram.

Doc2Vec and Word2Vec: In addition to the word embedding methods mentioned above, we train a doc2vec model on English Wikipedia articles and represent the embedding of a Wikipedia title by its document embedding. A doc2vec runs word2vec internally and so, word embedding is also done. The type of Doc2Vec used in our project is DBOW which is similar to the CBOW seen in word embedding.

The idea behind using both doc2vec and word2vec to generate title embeddings is that we observe that the two models favour different types of labels: doc2vec gives us headings of wikipedia pages which are more specific, while word2vec gives more abstract words . So, to have a balance of both, we use the two methods.

Given a topic, we measure the relevance of each title embedding (generated by either doc2vec or word2vec) based on the pairwise cosine similarity with each of the word embeddings for the top-10 topic terms, and aggregate by taking the arithmetic mean. Formally, the doc2vec relevance (rel_{d2v}) and word2vec relevance (rel_{w2v}) of a title a and a topic T is given as follows:

$$rel_{d2v}(a,T) = \frac{1}{|T|}\sum_{v\in T}\cos\left(E^{d}_{d2v}(a), E^{w}_{d2v}(v)\right) \quad (1)$$

$$rel_{w2v}(a,T) = \frac{1}{|T|}\sum_{v\in T}\cos\left(E^{w}_{w2v}(a), E^{w}_{w2v}(v)\right) \quad (2)$$

$E^{d}_{d2v}(x)$ - the document embedding of title x generated by doc2vec
$E^{w}_{d2v}(y)$ - the word embedding of word y generated by doc2vec
$E^{w}_{w2v}(z)$ - the word embedding of word z generated by word2vec
v εT - a topic term
T - the number of topic terms
cos() - cosine similarity function

To combine the strengths of doc2vec and word2vec, for each topic we generate a combined label ranking by summing the relevance scores using top-100 labels from doc2vec and word2vec. Now, we take the top 20 cosine similar words and project them as the probable labels for our topic.

$$rel_{d2v+w2v}(a,T) = rel_{d2v}(a,T) + rel_{w2v}(a,T) \quad (3)$$

Label Ranking

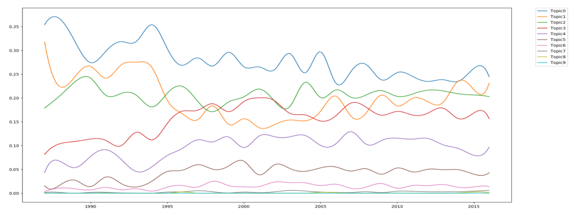
The next step after label generation is to re-rank them. The feature used for re-ranking is Letter Trigram. Letter trigrams simply capture character features of a word. Letter trigram vectors have reduced dimensionality, and are able to capture morphological variations of the same word to points that are close to each other in the letter trigram space. Our implementation is based on measuring the overlap of letter trigrams between a given topic label and the topic words. For each topic, we first convert each topic label and topic words into multinomial distributions over letter trigrams, based on simple MLE (Maximum Likelihood Estimation) . We then rank the labels based on the cosine similarity of all their trigrams with the topic words. This ranking by letter trigram forms our unsupervised baseline, as we found from other research papers that it has a much better performance as compared to other ranking methods. So the Letter Trigrams for the word say, image will be ima, mag, age.

2.5. Topic Evolution:

To study the evolution of topics over time, we first divided documents year wise and plotted how topic trend changed over time.
To do so, We have used two approaches to see how the topics have evolved over time[10]:

Average Topic Probability

The average probability of a topic in a particular year, for each topic - the topics probability associated with each paper was added and normalized by the number of papers published in that year. High average probability for a topic implies that, it either had maximum (or high) probability in many papers or considerable probabilities for most of them. Conversely, low average probability for a topic implies that very few papers are related to that topic. Weve used these average probabilities of each topic and plotted evolution over time as shown in the graph below. The data is discrete, and to make it smooth and continuous, weve used spline. Spline interpolation provides calculation simplicity, which is useful for further prediction.

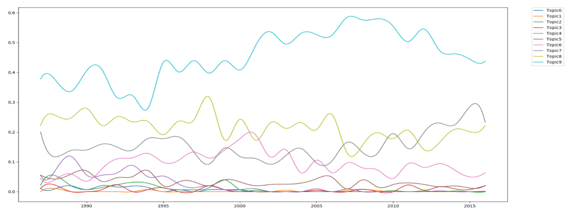


We got some interesting and useful insights from the graph. Topic 0 (Image processing) has the highest average probability over the years though it declined from 1995. Topic 1 (visual_system, neuron) which has very high probability in the 90s, dropped from 1995 and picked up from 2005 and attained higher probability than Topic 0 between 2013-2014. Topic 1, topic 2(optimization problem, matrix) and topic 3(Machine learning, Deep Learning) have similar probabilities from 1995 though they had different probabilities initially, which means that even though they had different popularity, ended up with similar interest in the scientific community. On observing topic 3s plot, we find that it started off with relatively less value, but had a sharp increase in the late 90s and has been steady ever since. This is in par with the increasing popularity of Machine Learning and Deep Learning from the last decade.

Maximum Topic Probability

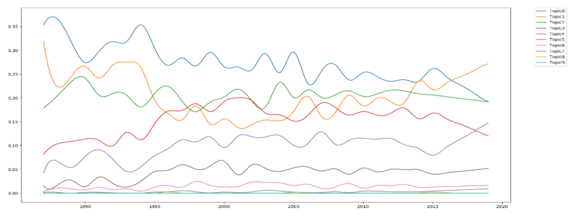
The downside with the first approach is that, a topic will have a good average probability only if it is present in significant number of papers. There maybe some topics which are the top topics in some papers but may have very less to zero probability in the rest. For such topics, we cannot observe the change in time using the average probability. To get Topic Prevalence, we only consider the Topic with Maximum probability. For every year, we get the count of papers in which, that particular topic has the highest probability and then divide it by the total number of papers published that year to get the percentage of the documents in which the topic is significant.

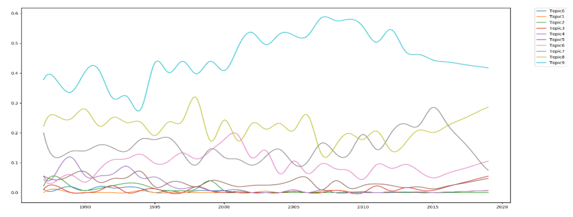
We can see from the graph that there is significant difference between the two graphs. For example, Topic 2(optimization_problem, matrix) has good average probability in the first graph, but very low probability in the second graph implying that it has high topic probability in few papers. From the labels of Topic 2 we can see that it is a more general topic, many papers may have the terms 'matrix' and 'optimization' and hence, it has good average probability but very less maximum probability.



2.6. Topic Prediction:

In order to predict topic trends for next few years, weve used data points from topic evolution. Curve fitting is basically building a model, based on available data, which can be used to predict over unknown data. Weve extrapolated the splines for the next 3 years. Since we have limited data (30 points/years), predicting for over 5 years, is not reliable.[?]





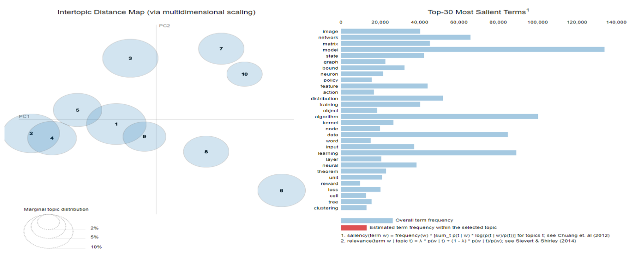
3. Evaluation

3.1. Datasets

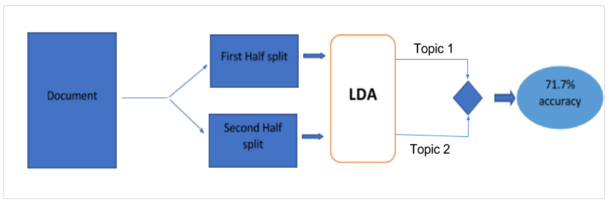
Neural Information Processing Systems(NIPS) is a Machine Learning and Neuroscience conference. NIPS dataset has all paper submissions from year 1987 - 2015. Dataset consists of 5811 conference papers and 11463 distinct words. The dataset is available in 2 formats (CSV and SQLite). Dataset is present in Bag of words representation and text document format.

3.2. Evaluation Strategy

Topic model evaluation methods other than human evaluation are found to be very poor indicators of the quality of the topics. We can consider human evaluation, but it could be tedious. To see how topics are clustered, how large a topic is and how well they are separated, weve used pyLDAvis python library to its potential for visualization of our topic model. Weve also used it to evaluate the generated labels. Weve looked at word distribution of each topic and checked with respect to labels generated. For example, the topic 6 has the word distribution as shown in the right section of the below image. All words related to Neuron, spike, signal are clustered together into single topic and these words doesnt seem to repeat in other topics. This shows that clustering is successful.

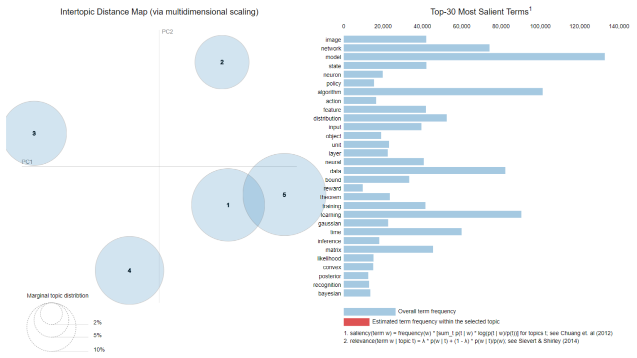


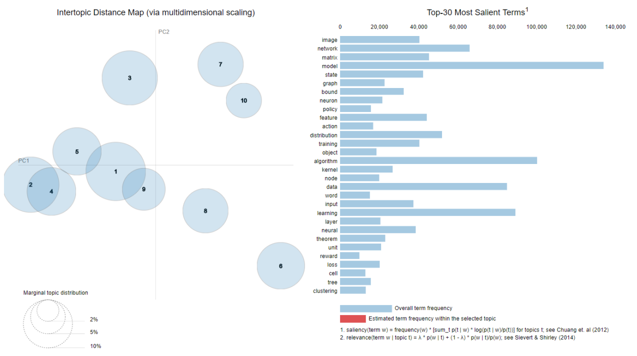
To evaluate the model, we split each document into two parts and each half is sent through LDA model. Each halfs topic distribution is compared to the other half and weve observed that both halves yielded same results for 71.7% of the documents in the corpus. This demonstrates consistency in the model.



3.3. Discussion of Results:

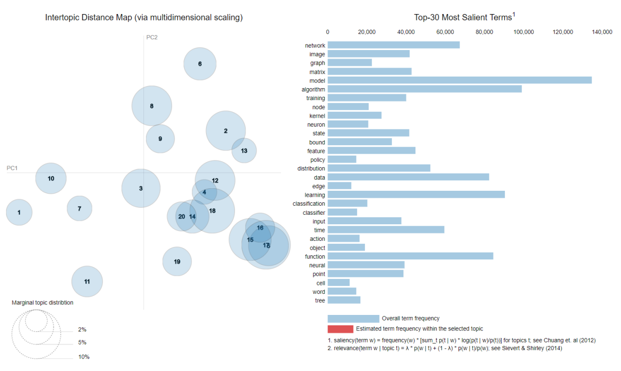
There are various results at every section of the project. Number of topics Even though, we came to conclusion that optimum number of topic size is 10, based on Silhouette coefficient and Calinski-Harabaz index, we tried the model with different cluster sizes (from 5 20). Weve shown sample cluster separation map for n = [5, 10, 20]. Weve observed that when cluster size is around 5, the clusters are very far apart and average cohesion within each cluster is large. The topics were too generalized in this case. Whereas when cluster size is around 15 20, the inter cluster separation is small and clusters overlap frequently. From the visualization tool (pyLDAvis), it can be seen that optimum number of topics = 10.





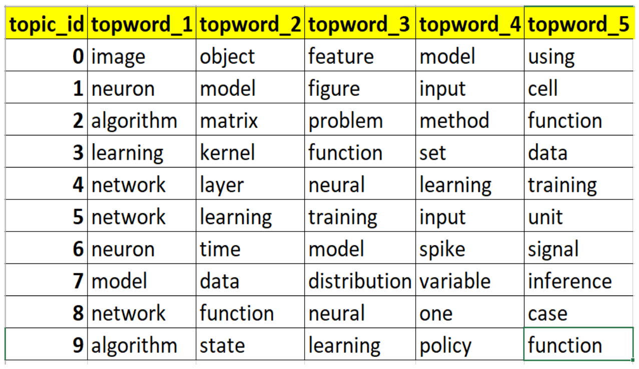
Once, we have topics clustered and labels generated, we have gone with human evaluation. Weve used the visualization tool, to verify if the top words in the cluster are matching with the label obtained.

Since dataset is strictly scientific and all scientific papers are highly structured, we plan to make use of this structure. Giving more weightage to words in title and abstract should give us more relevant analysis on topics. However, in NIPS dataset, abstract was missing for old

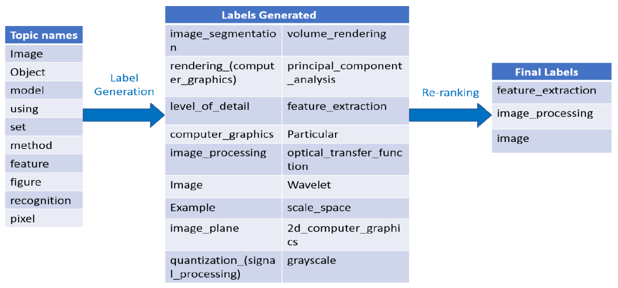


papers, and hence implementing the strategy did not give boost out results.

Sample result of top words for each topics:



topic_id	topword_1	topword_2	topword_3	topword_4	topword_5
0	image	object	feature	model	using
1	neuron	model	figure	input	cell
2	algorithm	matrix	problem	method	function
3	learning	kernel	function	set	data
4	network	layer	neural	learning	training
5	network	learning	training	input	unit
6	neuron	time	model	spike	signal
7	model	data	distribution	variable	inference
8	network	function	neural	one	case
9	algorithm	state	learning	policy	function



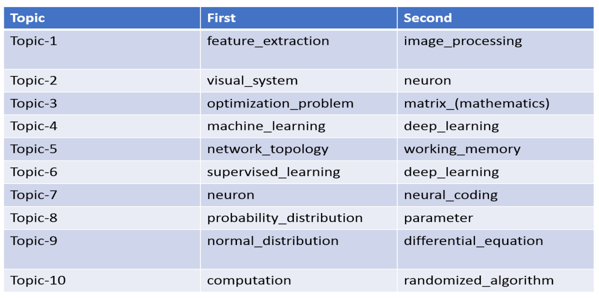
Topic names
Image
Object
model
using
set
method
feature
figure
recognition
pixel

Label Generation

Labels Generated	
image_segmentation	volume_rendering
rendering_(computer_graphics)	principal_component_analysis
level_of_detail	feature_extraction
computer_graphics	Particular
image_processing	optical_transfer_function
Image	Wavelet
Example	scale_space
image_plane	2d_computer_graphics
quantization_(signal_processing)	grayscale

Re-ranking

Final Labels
feature_extraction
image_processing
image



Topic	First	Second
Topic-1	feature_extraction	image_processing
Topic-2	visual_system	neuron
Topic-3	optimization_problem	matrix_(mathematics)
Topic-4	machine_learning	deep_learning
Topic-5	network_topology	working_memory
Topic-6	supervised_learning	deep_learning
Topic-7	neuron	neural_coding
Topic-8	probability_distribution	parameter
Topic-9	normal_distribution	differential_equation
Topic-10	computation	randomized_algorithm

The first table shows the 2 steps in topic labeling.It first calculates top 20 labels and then re-rank them.

The second table indicates the top-2 labels for every topic.

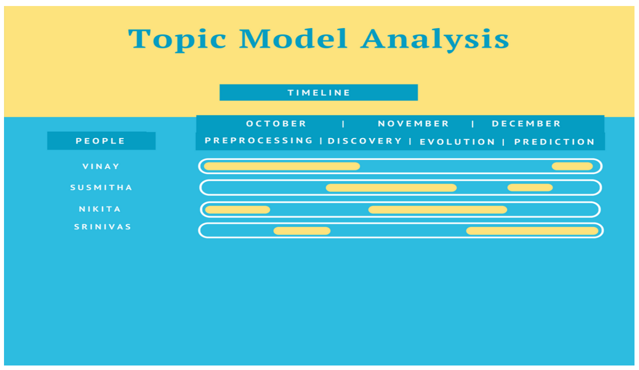
4. Conclusions:

Textual data is growing at an incredible rate each year, and most of it is unstructured and unlabeled. Categorizing and understanding them would help us gain insights. This would help large organizations build textual tools, search engines etc, to generate more relevant results. Weve demonstrated the capability of simple LDA model, to understand topics buried in a corpus of technical papers. Weve also understood how interest in those topics grew/diminished over period of time and used the data to predict trends for the next few years.

5. Proposed project timeline:

Weve divided entire workload into following four major sections:
1. Data Preprocessing
2. Topic Discovery and labelling
3. Topic Prediction
4. Topic Evolution.

We planned our work designation in such a way two of us are working on each section, and we can work collectively and effectively. We held meetings when there is progress and if we are stuck or needs a rethought. This way, all of us are involved in all aspects of design, implementation and debugging. Even though the below Gantt chart is concrete, all of us have touched upon all sections in different fashion. Weve completed preprocessing and discovery in October. However, labelling took considerable time to figure out best way to come up with labels that represent the topics. Then, weve observed how topics evolved over time, and extrapolated to future years. All of us are equally involved in evaluating and testing the model.



References

[1] Topical N-grams: Phrase and Topic Discovery, with an Application to Information Retrieval
`https://people.cs.umass.edu/œmccallum/papers/tngicdm07.pdf`

[2] Topics over Time: A Non-Markov Continuous-Time Model of Topical Trends
`https://pdfs.semanticscholar.org/b3bb/eecc76347f1f6921fe14a80b093f2cd9d3dd.pdf`

[3] pyLDavis Visualization
`https:https://github.com/bmabey/pyLDAvis`

[4]
`https://radimrehurek.com/gensim/models/ldamodel.html`

[5]
`https://arxiv.org/pdf/1612.05340.pdf`

[6] NIPS dataset from Kaggle
`https://www.kaggle.com/benhamner/nips-2015-papers/data`

[7] NIPS : Bag of Words Representation
`https://archive.ics.uci.edu/ml/datasets/NIPS+Conference+Papers+1987-2015`

[8] Automatic Labelling of Topics with Neural Embeddings
`https://arxiv.org/pdf/1612.05340.pdf`

[9] Word Embedding
`https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/`