Topic Modelling Analysis on VTechWork's Technical Reports

Ryland Hanson
Department of Computer Science
Virginia Tech
Blacksburg, Virginia, USA
rmhm1@vt.edu

Vinayak Kumar
Department of Computer Science
Virginia Tech
Blacksburg, Virginia, USA
vinayaksk@vt.edu

Abstract

Virginia Tech enables faculty, students and staff to upload various scholarly works such as journal articles, books, theses, and dissertations, through an online repository called VTechWorks. The works are published online in different collections based on the type of scholarly work and the department it belongs to. However, finding papers that belong to a specific subject area within the repository might become challenging since the platform filters out the paper directly on the keywords of the paper. This might be a problem if authors use different keywords that convey the same research area that the paper belongs to. We propose a topic modeling approach using Latent Dirichlet Allocation (LDA) that takes in a set of all technical papers belonging to the Computer Science department at Virginia Tech and used topic modeling to identify various topics that these papers could belong to. This approach was primarily run on a Hadoop environment, with libraries such as PySpark for data processing, various NLP libraries such as spark-nlp, NLTK and Gensim for tasks such as tokenizing and lemmatizing the incoming words, and used visualization libraries such as Matplotlib and pyLDAvis. We then use a combination of intrinsic evaluation metrics and human judgment to identify the best topics for the documents.

Keywords: Topic Modeling, NLP, PySpark, LDA

ACM Reference Format:
Ryland Hanson and Vinayak Kumar. 2022. Topic Modelling Analysis on VTechWork's Technical Reports. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Conference papers, dissertations, technical reports, and other equally important pieces of scholarly work are invaluable pieces of academic references for future research. Even within a specific subject, the number of different research topics that these papers may cover can be very large. As such, a repository of these scholarly works should allow anyone to look up papers that belong to a certain research area. However, even with an optimized search engine, it might be difficult to obtain a majority of the papers that belongs to the searched research topic. A lot of these papers may also inherently belong to a certain topic but due to the large volume of documents, the identification of these topics might get lost.

VTechWorks is an online repository that Virginia Tech handles that allows students, faculty, and staff to upload various pieces of research and scholarship including theses, dissertations, conference papers, and peer-reviews articles. The repository helps enable quick access and preservation of these pieces of scholarly work. The majority of the works submitted on this platform are made public. The platform allows for searching up specific fields when looking up specific papers, such as authors, titles, keywords, etc. These papers are grouped under different "communities" and "collections" that basically describe the college, department and type of work that the paper is categorized under.

However, if someone wants to quickly find out the distribution of possible research areas that the majority of the papers at Virginia Tech cover, relying on the search engine becomes a bit difficult for this sort of aggregation-based analysis. There is a filtering mechanism in the platform which allows users to filter the results based on predefined subject topics, but these subject topics are taken from the "keywords" section in a paper. While the authors of these papers may have included the most relevant research topics in the keywords that are common with other papers, there might be a vast array of keywords that have a similar meaning, but the keywords section might not be enough to grasp all of them. For example, one author may write the keyword "Data Analysis" while the author of another paper may write "Data Analytics" in their keywords. While both keywords indicate that the paper would fall under the same research topic, the filtering mechanism on the VTechWorks platform would not be able to identify the similarity between the two words, and hence group the two papers as two different subject topics.

To be able to consolidate and aggregate these papers into a set number of research topics, we would need to go through the text of the paper and identify groups of terms that could best represent the paper. By identifying these defined "topics" we can obtain meaningful insights about the various kinds of research that the majority of the different kinds of scholarly work at Virginia Tech tend to lie in, as well as easily label new incoming pieces of work into these identified research topics.

2 Background

Blei [2] proposed the usage of probabilistic topic models in the context of scanning through large quantities of documents to find a broad set of hidden themes. Blei primarily focused on describing the approach as a good alternative to the usage of tools such as search and links. While looking at documents and navigating to other linked documents through a search engine is helpful for searching up specific contexts within the data, the author argued that there was no efficient way to "zoom in" and "zoom out" of a large collection of documents to get a broader group of themes that the documents may cover.

The author then introduces the concept of topic models to tackle this issue, as topic models would help in discovering the main themes within a large and unstructured collection of documents which could lead to better organization and insights about the documents. The article primarily focuses on using the probabilistic topic model: LDA. LDA(Latent Dirichlet Allocation) is a form of generative probabilistic modeling, where data is treated as arising from generative methods that involve hidden variables. Data analysis is performed by using conditional (or posterior) distribution of these hidden variables over the observed variables. The author explains the concept of LDA in the context of topic modeling by analyzing articles from the *Yale Law Journal* and *Science* magazine and obtaining various document topics, where topics such as discrimination, contract law, genetics, and disease were found from these specific collections of documents. The author emphasized that there is no prior information about these subjects when the modeling is performed. The topics generated with the cluster of words are inherently found by the LDA model (and hence why this approach is considered an unsupervised learning method).

The system introduced by Asmussen et al [1] tackles a very similar problem to the one mentioned in this paper, where the topic modeling approach is used to identify themes behind relevant research papers in the context of exploratory literature review. Here the authors used the LDA algorithm, due to its consistent performance compared to other probabilistic topic model algorithms.

While the approach did initially seem to give favorable results for researchers to make an efficient literature survey, there were a few issues with the overall system mentioned in this paper. The preprocessing part of the system did not seem very well defined, primarily focusing on just stop words removal and stemming. Another possible problem that was identified is the scalability of the approach. While the system was primarily focused on scanning through large amounts of documents, the dataset chosen to validate the framework was around 650 papers in length, primarily focused on analytics and enterprise information systems research fields. This dataset might not have generated a large corpus for the topic modeling to act on, and there were certain topics with very few documents. While this may be due to the dataset size, this could also be due to the way the evaluation of the system was performed. They primarily focused on using perplexity [9] with cross-validation to identify the optimal number of topics. The cross-validation method would work well in this case, since it helps improve the reliability of the analysis as opposed to a static sampling of the dataset. The possible issue in this system is the use of perplexity.

There has been further research that does not encourage the use of perplexity in the context of evaluating the number of topics, due to it being inconsistent with semantic reasoning. Perplexity, which is a score based on the inverse of geometric mean per-word likelihood, is a decent statistical evaluation of topic models. However, from the studies of Chang et al[4], perplexity was found to not do a good job of depicting whether topics were coherent or not. In certain cases, these studies showed that perplexity could incur a negative correlation with human judgment in certain situations. The issue is that the qualitative understanding of the semantic nature of the learned topic is not looked at by using a metric like perplexity.

One of the more efficient methods of evaluating topic modeling includes using a metric called coherence score. There have been slight variations of the definitions of this metric but most of them primarily focuses on the similarity between the words in a given topic [7]. Based on the various similarity scores between the words, the top n scores are chosen for each topic. There are also other statistical methods of evaluating topic models, such as the OpTop metric mentioned by Lewis et al[6], which showed to outperform perplexity in their simulations.

It is important to note that these metrics did not guarantee better results as opposed to human judgment when finding the optimal number of topics. However, when larger amounts of documents and topics are required, it might not be possible to get human expertise to identify the topic themes, especially if there is a diverse set of topics spanning different areas of knowledge, produced from the topic modeling analysis.

Jacobie et al[5] introduced a topic modeling analysis on journalism research, where the LDA algorithm is used on

51,528 news stories from the *New York Times*, between the years 1945 and 2013, and filtered the articles based on key terms such as "nuclear" and "atomic". The analysis involved a series of preprocessing steps that included tokenization, lemmatization and part of speech tagging. The LDA model was fit on the processed text, and to find the optimal number of topics, the model was tuned according to the perplexity, alpha parameter, and human interpretability of the topics. The alpha parameter represents the document-topic density, where it is used to control the maximum number of topics that a document can be composed of. A higher alpha value indicates an acceptance of a higher number of topics per document in the LDA model.

The analysis focused on several variations of topic models, where they first identified the topics for the entire dataset from 1945 to 2013, topics that showed some pattern and variation in their use over time, and topics that had a more continuous presence throughout all the years. An in-depth comparison was performed between different LDA models with different 'k' topics which showed the best results in terms of statistical and human-interpretable evaluation. This analysis showed a very unique perspective on topic modeling analysis, particularly in the context of topic modeling variation over different time periods.

3 Design

To identify topics within the large collection of scholarly documents at VTechWorks, we propose to use the LDA algorithm to perform a topic-modeling analysis which would help lead to insights about the research areas that these documents cover, as well as possibly improve the existing filtering mechanism on the platform by using these general topics instead of the paper's keywords.

3.1 Data

For this analysis, we primarily focused on using the technical papers found in the Department of Computer Science at Virginia Tech, however, this approach can extend to include other pieces of work such as dissertations, presentations, and theses. To ensure that our framework would be as updated as possible, we proposed to web scrape the VTechWorks platform for the papers. We primarily extract the paper title, abstract, and date of the paper. The paper title would ideally be used as the primary key to uniquely identify each document.

The data would then need to be effectively stored and retrieved when necessary. Saving the data in a fault-tolerant distributed file system would help ensure data integrity and faster retrieval times during the modeling part of the framework. We proposed to use the Hadoop-Distributed File system (HDFS) as a good storage solution for this analysis.

Certain text preprocessing is required before we feed our data to the model. Most of the methods we use are similar to what is mentioned in the preprocessing steps of [5], but we would need to run a series of text-processing steps to see which methods would actually be necessary for our dataset. In our framework, we use a "pipeline" of sequential Natural Language Processing(NLP) steps to process the data and to create the features for our model. More information about these steps and the data can be referred to in the next section and in Figure 2.

3.2 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) is a generative model that enables sets of different observations to be described by unobserved groups which are able to explain why some parts of the data are similar. In the context of NLP, LDA is primarily used with documents. Each document is regarded as a distribution of corpus-wide topics. A "topic" is a distribution of words over a fixed vocabulary. These topics are generated from the collection of documents. See Figure 1 and 2 for an example of this. Refer to Appendix A for the formal notation of LDA.

LDA was introduced as a machine-learning algorithm by Blei et al [3] where the authors mention that the model makes the following assumptions: First, the k parameter (number of topics) is already known. Second, the word probabilities are estimated from the corpus in a matrix. In simpler terms, each document is considered as a "bag of words", where the order of the words and the grammatic role of the words are not considered. Third, the initial Poisson assumption of the model, where the length of the document should be Poisson distributed, is not critical to anything that follows in the model, and more realistic document length distributions can be used as needed.

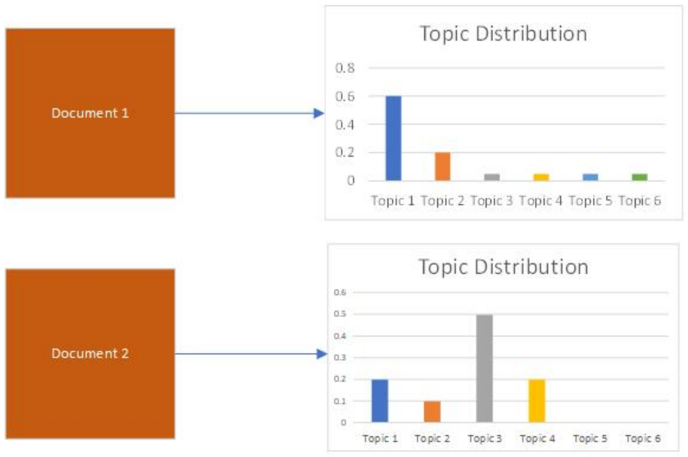


Figure 1. Topic Distribution for a set of documents

The output of the model would include the list of topics and the word distribution within each topic, as well as a list of documents and their respective topic distribution.

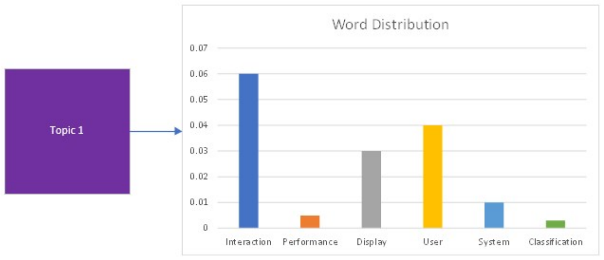


Figure 2. Word Distribution for a single topic

3.3 Optimization

There are three hyper-parameters that can be used to optimize the LDA model: The alpha(α) and beta (β) parameters, as well as the k number of topics. The LDA model can be run with various values of k which can give a different number of topics and different word distributions for each topic. We can then evaluate the model (using a metric or human judgment) to determine the best value for k.

As mentioned earlier, the α parameter is defined as the document-topic density which controls the number of topics that can represent a single document. Similarly, the β parameter is defined as the topic-word density that controls the number of words that can represent a single topic. Adjusting these values may lead to different results.

We can optimize our model by tuning these hyperparameters and getting a better result in terms of a metric and human judgment. We primarily focus on the coherence score [7] as our metric to evaluate the LDA model, since it is able to capture the semantic quality of the words in a topic. We first set k with a low seed value, and then increment it for different iterations of the LDA model, until we obtain the best coherence score for that value of k. We automatically set the values for the α and β parameters based on the incoming data and the current value of k.

The overall framework of the proposed system can be seen in Figure 3.

4 Implementation

4.1 Web Scraping

In order to create the corpus of documents on which to perform the topic modeling, technical papers had to be scraped from the VTWorks repository. Through the use of Python libraries such as Requests and Beautiful Soup, requests were sent to catalog pages of the repository to grab the list of document titles, links to the reports, and links to the next page. It is worth noting, that there is a timeout of a few seconds between the scraping for each set of 20 reports in order to ensure that requests are not sent too frequently to the website. For each given technical report, the Title, Date, and Abstract sections were scraped and added to a Pandas DataFrame for storage. Papers from the overall recent submissions, the department of Computer Science's technical reports, and thesis/dissertations were all scraped. Those from the department of Computer Science's technical reports were the main focus of the analysis performed, of which there were approximately 1,300 papers received.

4.2 Text Preprocessing

The text preprocessing stage performed prior to the modeling is an incredibly crucial aspect of making a quality LDA model. Each step helps in ensuring the words are in the correct format, extraneous words are discarded, compound words are joined, etc. Without these stages, the outputs of the generated topics will at best be of lesser quality, and at worst simply not make sense. In spark, the various processing stages must each be defined and then combined into a single pipeline that will allow the dataframe to be processed in order and then output as the final product to be fed into the later stages of model building.

The first step of processing documents in spark is to use the DocumentAssembler function. This function is relatively straightforward, and as the name suggests it is what is used to assemble the documents for further processing. This function takes as input the raw text column of the input DataFrame, and converts it into a Spark NLP [8] annotation format to be compatible with later functions. The sample call for this function is as follows:

```
from sparknlp.base import DocumentAssembler

documentAssembler = DocumentAssembler() \
     .setInputCol(text_col) \
     .setOutputCol('document')
```

The input column is specified, and then the name of the output column is given.

The next stage is that of tokenizing the data. This stage is quite straightforward as the raw input text is broken up into smaller units, or "tokens", with which further computation will be easier. Generally, this means simply breaking up each paper's abstract into an array of Strings where each value is one word of the given text. What is important to note about how this function is called, is that the given input column of the tokenizer is the output column of the DocumentAssembler. This allows the chaining of multiple preprocessing steps to happen in a given order. To show what is meant by this, the sample function call for this step is given:

```
from sparknlp.annotator import Tokenizer

tokenizer = Tokenizer() \
     .setInputCols(['document']) \
     .setOutputCol('tokenized')
```

The next stage in the pipeline is a normalizing the text. The tokens are taken as input and then dirty characters such as

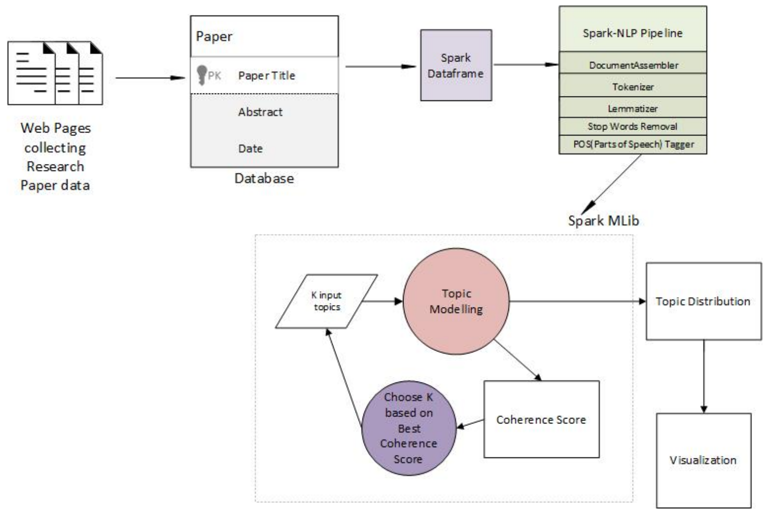


Figure 3. Framework Overview

punctuation, capitalization, etc. is removed and the words are all set to be lowercase.

Then, the cleaned and normalized tokens are fed into the next step of the preprocessing stage: lemmatization. This stage is a fairly complex one, where the words are used as input in a pretrained lemmatizer model which transforms all forms and inflections of a word to a single root. For example, an input word such as "better" is derived from a word "good", and thus should be transformed into this root word. This stage helps ensure that different variations of words are not recognized as seperate words, but instead belonging to the same root word and core idea.

The next step, and one of the most important, is stop word removal. Using the NLTK library, a list of extremely common, and generally non-meaningful, English words can be retrieved. Then all of the words of the input that matches those in this list will be removed to ensure these only those with importance will remain.

Finally, there is the NGramGenerator step. In this step, phrases made up of n number of words will be combined to be used as a single token. This step is extremely helpful in creating meaningful tokens as instead of the words "artificial" and "intelligence" being broken up and by themselves lacking much meaning, if they are joined as "artificial intelligence" then this becomes a very relevant word in many Computer Science reports.

Once all of the preprocessing steps are defined, the final pipeline can be created in which all of the stages are set and the order in which the document will be processed is given. The signature for creating this overall pipeline is given:

```
from pyspark.ml import Pipeline

pipeline = Pipeline() \
     .setStages([documentAssembler,
                 tokenizer,
                 normalizer,
                 lemmatizer,
                 stopwords_cleaner,
                 ngrammer,
                 finisher])
```

4.3 Modeling

After all of the text preprocessing has been performed upon the corupus of documents, next is the actual building of the LDA model. However, before the model can be created TF-IDF vectorization must be performed. The first step of TF-IDF is TF, or Term Frequency, in which a vector of word counts is created. This stage takes in the processed text, and outputs a new vector where each index corresponds to a specific word, and each entry gives the frequency in which that word appears in the corpus. This can be done in PySpark in the following way, where the processed text is in a column of a DataFrame called "final":

```
from pyspark.ml.feature import CountVectorizer

tfizer = CountVectorizer(inputCol='final',
    outputCol='tf_features')
tf_model = tfizer.fit(processed_text)
tf_result = tf_model.transform(processed_text)
```

Next is the IDF, or inverse document frequency, stage. After a vector of term frequencies has been created, in order to give less weight to words that appear extremely frequently in all of the documents scores are given to each entry in relation to how frequent each words are and the results are scaled. Thus, words that appear extraordinarily frequently, and are thus likely to be less impactful in general, are given lower scores and specific words that might stand out as a potential meaningful topic are given higher scores. This can be done in PySpark as follows:

```python
from pyspark.ml.feature import IDF

idfizer = IDF(inputCol='tf_features', outputCol='
    tf_idf_features')
idf_model = idfizer.fit(tf_result)
tfidf_result = idf_model.transform(tf_result)
```

Finally, it is time to pick the number of topics with which to fit the LDA model, the number of iterations of the algorithm to perform, and supply the model with the TF-IDF created above. This step is extremely easy, and can be shown:

```python
from pyspark.ml.clustering import LDA

num_topics = 12
max_iter = 100

lda = LDA(k=num_topics, maxIter=max_iter,
    featuresCol='tf_idf_features')
lda_model = lda.fit(tfidf_result)
```

5 Evaluation

Since Topic modeling is an unsupervised machine learning technique with results that are mainly intended to be human-interpreted, it is difficult to empirically evaluate model performance. However, it is possible to use the coherence and perplexity score to get an idea of how well a specific model may have done and thus get an idea of the optimal number of topics for a given corpus. Additionally, evaluating model outputs of topics by hand and visualizing the topics indeed helps with understanding the given results.

5.1 Selecting number of topics

As stated above, despite requiring much human judgment in evaluating if an LDA model is performing well there are indeed scores that can be computed across various numbers of topics in order to help in the selection. Thus, for the Computer Science technical report corpus, a model was fit and run with random initializations for 10 iterations at multiple different numbers of topics in order to help in the selection.

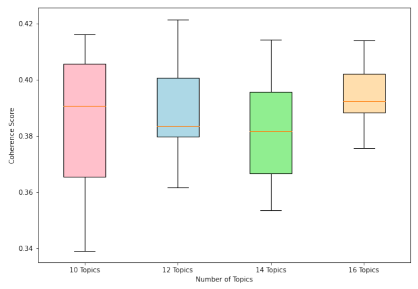


Figure 4. Coherence Scores

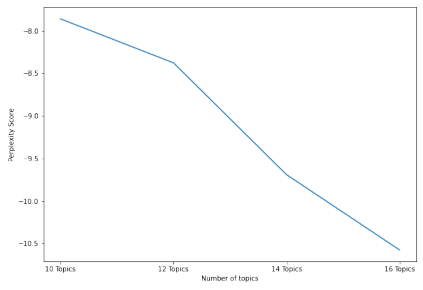
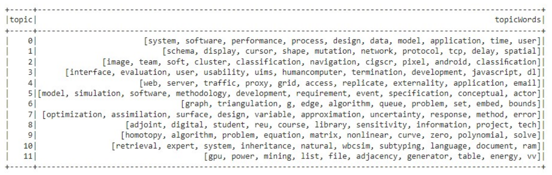


Figure 5. Perplexity Scores

While Figure 5 shows that the higher the number of topics the better the score, Figure 4 shows that perhaps 12 topics are the optimal number for the model. Since coherence score is generally a better measure for human-readable topics, those scores were given more weight when investigating the optimal choice. Upon further review of the actual topics output for the given corpus over the range of choices, it became evident that 12 topics generally made more human-readable output, and thus it was selected as the optimal number to use for further analysis.

5.2 Model Outputs

After running the LDA model, we were able to generate the topics with the associated words within each topic. In addition to spark, gensim models will be used in analysis due to the great functions provided for visualizations and computing topic distributions for a given document. Since two different implementations are used, we can give the results of both models to compare the quality of the topics created. The two results can be seen in Figures 6 and 8.



```
+-----+-------------------------------------------------------------------------------------------------------------+
|topic|                                                                                                   topicWords|
+-----+-------------------------------------------------------------------------------------------------------------+
|    0|                       [system, software, performance, process, design, data, model, application, time, user]|
|    1|                           [schema, display, cursor, shape, mutation, network, protocol, tcp, delay, spatial]|
|    2|             [image, team, soft, cluster, classification, navigation, cigscr, pixel, android, classification]|
|    3|      [interface, evaluation, user, usability, uims, humancomputer, termination, development, javascript, dl]|
|    4|                      [web, server, traffic, proxy, grid, access, replicate, externality, application, email]|
|    5|[model, simulation, software, methodology, development, requirement, event, specification, conceptual, actor]|
|    6|                               [graph, triangulation, g, edge, algorithm, queue, problem, set, embed, bounds]|
|    7| [optimization, assimilation, surface, design, variable, approximation, uncertainty, response, method, error]|
|    8|                   [adjoint, digital, student, reu, course, library, sensitivity, information, project, tech]|
|    9|                  [homotopy, algorithm, problem, equation, matrix, nonlinear, curve, zero, polynomial, solve]|
|   10|                [retrieval, expert, system, inheritance, natural, wbcsim, subtyping, language, document, ram]|
|   11|                                    [gpu, power, mining, list, file, adjacency, generator, table, energy, vv]|
+-----+-------------------------------------------------------------------------------------------------------------+
```

Figure 6. Spark Model Topics

5.3 Identifying Topic Labels

In addition to viewing the topics in a simple table format, there is a very useful library in Python called pyLDAVis that takes a Gensim LDA model as input and creates great visualizations. Thus, to get a better view of the topics generated we will use the Gensim model and explore the visualizations created by this library. While looking at the words that make up the shown topic, we can try and infer what these topics may actually represent.

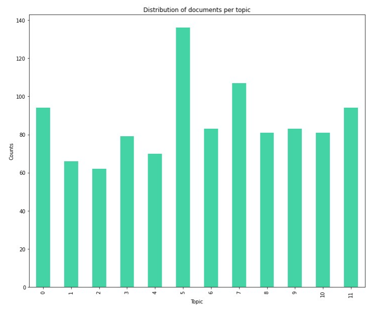
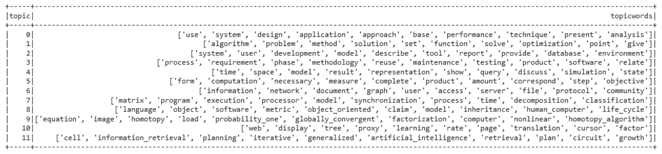


Figure 7. Document Frequency for all 12 Topics



```
+-----+--------------------------------------------------------------------------------------------------------------------------------------------+
|topic|                                                                                                                                  topicwords|
+-----+--------------------------------------------------------------------------------------------------------------------------------------------+
|    0|                                ['use', 'system', 'design', 'application', 'approach', 'base', 'performance', 'technique', 'present', 'analysis']|
|    1|                                     ['algorithm', 'problem', 'method', 'solution', 'set', 'function', 'solve', 'optimization', 'point', 'give']|
|    2|                                  ['system', 'user', 'development', 'model', 'describe', 'tool', 'report', 'provide', 'database', 'environment']|
|    3|                             ['process', 'requirement', 'phase', 'methodology', 'reuse', 'maintenance', 'testing', 'product', 'software', 'relate']|
|    4|                                        ['time', 'space', 'model', 'result', 'representation', 'show', 'query', 'discuss', 'simulation', 'state']|
|    5|                               ['form', 'computation', 'necessary', 'measure', 'complete', 'product', 'amount', 'correspond', 'step', 'objective']|
|    6|                                   ['information', 'network', 'document', 'graph', 'user', 'access', 'server', 'file', 'protocol', 'community']|
|    7|                 ['matrix', 'program', 'execution', 'processor', 'model', 'synchronization', 'process', 'time', 'decomposition', 'classification']|
|    8|                ['language', 'object', 'software', 'metric', 'object_oriented', 'claim', 'model', 'inheritance', 'human_computer', 'life_cycle']|
|    9|['equation', 'image', 'homotopy', 'load', 'probability_one', 'globally_convergent', 'factorization', 'computer', 'nonlinear', 'homotopy_algorithm']|
|   10|                                                ['web', 'display', 'tree', 'proxy', 'learning', 'rate', 'page', 'translation', 'cursor', 'factor']|
|   11|       ['cell', 'information_retrieval', 'planning', 'iterative', 'generalized', 'artificial_intelligence', 'retrieval', 'plan', 'circuit', 'growth']|
+-----+--------------------------------------------------------------------------------------------------------------------------------------------+
```

Figure 8. Gensim Model Topics

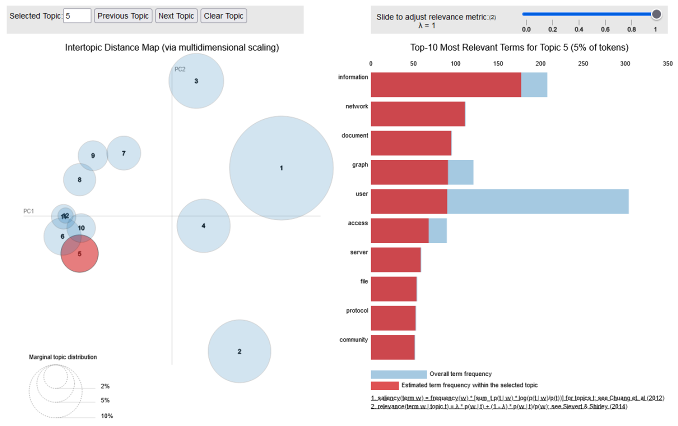

Selected Topic: 5
Previous Topic
Next Topic
Clear Topic
Slide to adjust relevance metric:(2)
λ = 1
0.0
0.2
0.4
0.6
0.8
1
Intertopic Distance Map (via multidimensional scaling)
PC2
PC1
1
2
3
4
5
6
7
8
9
10
12
Marginal topic distribution
2%
5%
10%
Top-10 Most Relevant Terms for Topic 5 (5% of tokens)
0
50
100
150
200
250
300
350
information
network
document
graph
user
access
server
file
protocol
community
Overall term frequency
Estimated term frequency within the selected topic
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Figure 9. Topic 5 Visualization

The left hand side of Figure 9 shows the topics scaled down to 2D via Multidemensional scaling (MDS) in order to visualize the distances, or similarities, between the various topics. Additionally, the size of the bubbles represent the dominance of the topic over the corpus of documents, and so the larger the bubble the more common the topic is in the documents. The right hand side shows the list of words and their relevancy for the chosen topic. For this topic #5 shown we can see words such as network, protocol, server, access, etc. which might represent a field such as networking.

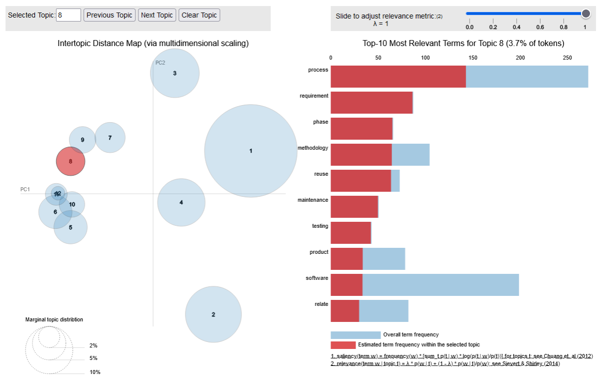

Selected Topic: 8
Previous Topic
Next Topic
Clear Topic
Slide to adjust relevance metric:(2)
λ = 1
0.0
0.2
0.4
0.6
0.8
1
Intertopic Distance Map (via multidimensional scaling)
PC2
PC1
1
2
3
4
5
6
7
8
9
10
12
Marginal topic distribution
2%
5%
10%
Top-10 Most Relevant Terms for Topic 8 (3.7% of tokens)
0
50
100
150
200
250
process
requirement
phase
methodology
reuse
maintenance
testing
product
software
relate
Overall term frequency
Estimated term frequency within the selected topic
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Figure 10. Topic 8 Visualization

In Figure 10 we can see that topic #8 is made up of words such as process, requirement, phase, software, maintenance, etc. All of these words seem very related to the software design life cycle (SDLC), and thus would likely represent a topic such as software design or software development.

5.4 Predicting a Document's Topic

In addition to high level results such as viewing the overarching topics for an entire corpus of documents, it is possible to look at a single document and identify the relevant topics within. We can take a look at a single document's abstract from the corpus given below:

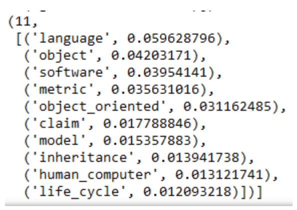
> *Most common object models of distributed object systems lack support for 'polymorphism' (an abstraction mechanism that represents a quality or state of being able to assume different forms). This lack of support restricts the development of new components and limits reuse of existing components that use these advanced features. In this paper, the Interoperable Common Object Model (ICOM), which focuses on a subset of object-oriented languages, specifically statically typed languages, is presented. The ICOM model is an attempt to elevate common object models closer to the object models of statically typed object-oriented languages by including support for polymorphism. Specific features of the ICOM object model include: remote inheritance, method overloading, and parameterized types.*

We can then look at this documents distribution of topics and their probability of relating to the content of this abstract, in addition to the words that make up the shown topics:

```
[(0, 0.06359949),
 (1, 0.036247957),
 (4, 0.22125278),
 (6, 0.03171684),
 (7, 0.03700903),
 (11, 0.5822022)]
```

Figure 11. Document's relevant topics

As we can see from Figure 11, the most relevant topics for this piece of text is #11 and #4. When inspecting those



```
(11,
 [('language', 0.059628796),
  ('object', 0.04203171),
  ('software', 0.03954141),
  ('metric', 0.035631016),
  ('object_oriented', 0.031162485),
  ('claim', 0.017788846),
  ('model', 0.015357883),
  ('inheritance', 0.013941738),
  ('human_computer', 0.013121741),
  ('life_cycle', 0.012093218)])]
```

Figure 12. Topic 11

topics, it is obvious that they represent ideas based on programming languages, and the properties of object oriented programming. This matches extremely well with what can be seen from the abstract which talks heavily on these concepts, and aims to present a new model for working with object oriented languages. Thus, we can confidently say that the LDA model has created a very good representation of what it is that this document aims to explore.

6 Conclusion

Using the topic modeling approach, we were able to identify a decent amount of possible themes and topics that could describe the technical papers in the Department of Computer Science at Virginia Tech. By understanding the broad themes of these pieces of scholarly work, we can better organize the documents at VTechWorks according to these identified themes.

However, our approach is not flawless and still needs to be improved. For example, not all topics were found to be meaningful. There was a topic that gave very generic words such as 'population', 'panel', 'past', 'personal', and 'spend', which does not particularly indicate any specific research area. Some of the topics also included a lot of words that did not relate to the labeled topic as well, or a topic sometimes had words that mixed different research areas. We might have restricted ourselves by focusing on just the technical papers. While we initially wanted to include other pieces of work such as theses and dissertations, our API call to VTechWorks was refused for certain documents, and hence we were unable to include more data for the topic modeling analysis.

The other limitation of this approach is the slightly disconnected implementation. While our proposed design initially focused on using the spark-nlp library, we faced a lot of issues in the analysis and visualization of the topics generated using this library, since there seems to be a large lack of visualization support for topic modeling in spark-nlp. Most of the visualization functions included in this library are restricted to supervised learning models like text classification. That was the reason why we had to emulate the processing steps in spark-nlp in the Gensim library to get the efficient bubble charts with pyLDAVis as seen in Figures 9 and 10.

Going forward, we ideally need to better optimize our learning model. While topic coherence gave us a small list of k possible topics, we still relied on human judgment to identify the exact optimal number of topics, which unfortunately is time-consuming and costly. This is especially true when the number of documents, the number of topics, and the corpus of words scale up to span a large number of subjects and research areas. Ideally, we would like to have a quantitative metric that is comparable to human judgment when it comes to identifying the best topics, but defining that metric is a very challenging problem that has yet to be solved.

Apart from extending our work to include other types of scholarly work, we would also like to look at obtaining documents from other departments at Virginia Tech. We could then ideally obtain the research topics for these documents for different departments. As mentioned earlier, we would most likely still need some human judgment for identifying and labeling the topics, and hence there might be additional costs incurred for having different human experts for each department.

Another possible extension of this work is to include a historical analysis of the topic distribution. While we did extract the date for each paper, due to the relatively small size of the dataset, we could not perform an appropriate topic modeling analysis when we aggregate the data based on time periods (such as on a yearly basis). If we increase our dataset by including other kinds of documents, we might be able to generate an efficient list of topics and obtain insights on the variations of topics over a specific period of time.

Our analysis can still provide a good baseline for the VTechWorks platform to improve its existing filtering mechanism. Once the topic model is refined, the platform can then include these topics as better filters for the documents. This topic model can also be used as a classification model to predict any new document's theme when a piece of work is submitted to the platform.

7 Appendices

A Latent Dirichlet Allocation

Figure 13 represents a probabilistic graphical model of the LDA. The parameters α and β are corpus level parameters, which are assumed to be sampled once in the process of creating the corpus.

$$\begin{gathered} P(W, Z, \theta, \phi, \alpha, \beta) = \prod_{j=1}^{M} P(\theta_j, \alpha) \\ \prod_{i=1}^{K} P(\phi_i, \beta) \prod_{t=1}^{N} P(Z_{j,t}|\theta_j) P(W_{j,t}|\phi Z_{j,t}) \end{gathered} \tag{1}$$

D: Number of Documents
N_d: Number of words in a given document
β: dirichlet prior on the per-document topic distribution
α: diriclet prior on the per-topic word distribution
θ_i: topic distribution for document i
ϕ_k: word distribution for topic k

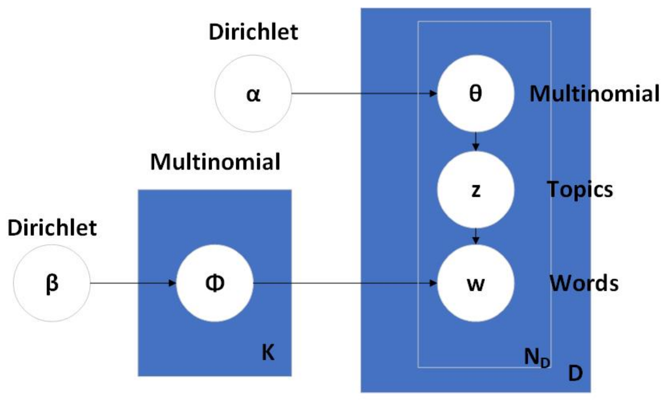


Figure 13. Graphical model representation of LDA

Z:Vector with topics of all words in all documents
W:Vector with all words in all documents
Z_{ij}:topic for the j-th word in document i
W_{ij}: specific word

Equation 1 refers to the probability of generating the original document from the LDA model. On the right side, the first two terms represent the Dirichlet distribution and the other two terms represent the multinomial distribution. The first and third term represent the distribution of topics while the second and fourth term represent the word distribution.

References

[1] Claus Boye Asmussen and Charles Møller. 2019. Smart Literature Review: A practical topic modelling approach to exploratory literature review. *Journal of Big Data* 6 (2019). https://doi.org/10.1186/s40537-019-0255-7

[2] David M. Blei. 2012. Probabilistic Topic Models. *Commun. ACM* 55, 4 (apr 2012), 77–84. https://doi.org/10.1145/2133806.2133826

[3] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *J. Mach. Learn. Res.* 3, null (mar 2003), 993–1022.

[4] Jonathan Chang, Jordan Boyd-Graber, Sean Gerrish, Chong Wang, and David Blei. 2009. Reading Tea Leaves: How Humans Interpret Topic Models. *Neural Information Processing Systems* 32, 288–296.

[5] Carina Jacobi, Wouter van Atteveldt, and Kasper Welbers. 2016. Quantitative analysis of large amounts of journalistic texts using topic modelling. *Digital Journalism* 4, 1 (2016), 89–106. https://doi.org/10.1080/21670811.2015.1093271 arXiv:https://doi.org/10.1080/21670811.2015.1093271

[6] Craig M. Lewis and Francesco Grossetti. 2022. A Statistical Approach for Optimal Topic Model Identification. *Journal of Machine Learning Research* 23, 58 (2022), 1–20. http://jmlr.org/papers/v23/19-297.html

[7] David Newman, Jey Lau, Karl Grieser, and Timothy Baldwin. 2010. Automatic Evaluation of Topic Coherence. *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the ACL*, 100–108.

[8] David Talby. 2017. Introducing the natural language processing library for apache spark - the databricks blog. *Databricks* (Oct 2017). https://www.databricks.com/blog/2017/10/19/introducing-natural-language-processing-library-apache-spark.html

[9] Hanna M. Wallach, Iain Murray, Ruslan Salakhutdinov, and David Mimno. 2009. Evaluation Methods for Topic Models. In *Proceedings of the 26th Annual International Conference on Machine Learning* (Montreal, Quebec, Canada) *(ICML '09)*. Association for Computing Machinery, New York, NY, USA, 1105–1112. https://doi.org/10.1145/1553374.1553515