FIT5212 - Data Analysis for Semi Structured Data

Assignment 1

Text Classification and Topic Modelling

Udit Arora
31167799
uaro0001@student.monash.edu

Table of Contents

Libraries Used:

- **Pandas:** used for data structure and data analysis
- **Numpy:** used for working with arrays and numerical transformations
- **String:** used to get punctuations for preprocessing
- **Torch:** used for optimizing neural network method
- **TorchText:** used for processing text for neural network input data
- **Sklearn:** used to build confusion matrix and model building
- **Nltk:** used for removing stop words for preprocessing
- **Genism:** used for preprocessing for Machine learning and topic modelling
- **PyLDAvis:** used for visualising the topic models

Part 1: Text Classification

The goal of this process is to classify the textual data into classes based on the models built with the training data and predict for the test data. For this assignment we will classify 'Abstract' into three classes namely 'InfoTheory', 'CompVis' and 'Math'.

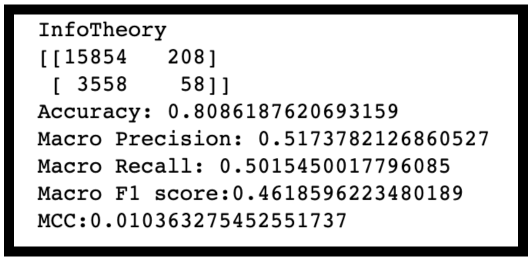
Preprocessing input data

- Start by removing stop words and punctuations obtained from the 'String' package.
- Next step is tokenizing the 'Abstract' column.
- Remove the words with low frequency in machine learning.
- Filtering of empty rows and replacing the text as 'NA' and numerical labels with empty values are replaced by '0'.

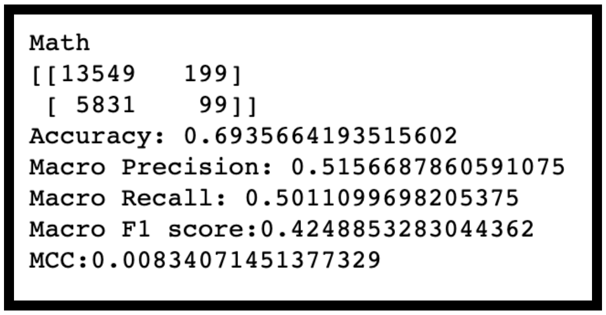
Neural network method

After preprocessing the data, we have defined a common RNN function which can be reused based on input data for each of the three classes. Also a binary accuracy function is defined in the common code.

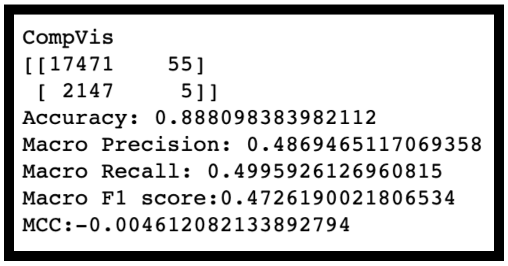
The input training data is loaded with 'Abstract' column as text data and class columns as labels. After that we build a vocabulary for the text data using the 'build_vocab' function. We create batch iterators for the class labels and define the input parameters for the RNN function to build the model. We use the optimizer to update various hyperparameters that will help reduce the losses in much less effort and it is used to train the model. The model is trained for the classes using the respective train data, then the same model is run using the test iterator for predicting the test data from the 'Abstract' column. We then generate the confusion matrix using the test value and predicted value of the model. Using the sklearn library we calculate the 'Recall', 'precision', 'f1 score', and 'accuracy'' for all the three classes.



```
InfoTheory
[[15854   208]
 [ 3558    58]]
Accuracy: 0.8086187620693159
Macro Precision: 0.5173782126860527
Macro Recall: 0.5015450017796085
Macro F1 score:0.4618596223480189
MCC:0.010363275452551737
```



```
Math
[[13549   199]
 [ 5831    99]]
Accuracy: 0.6935664193515602
Macro Precision: 0.5156687860591075
Macro Recall: 0.5011099698205375
Macro F1 score:0.4248853283044362
MCC:0.00834071451377329
```

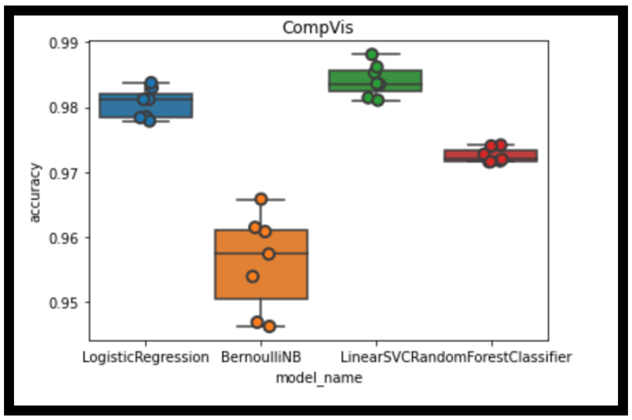


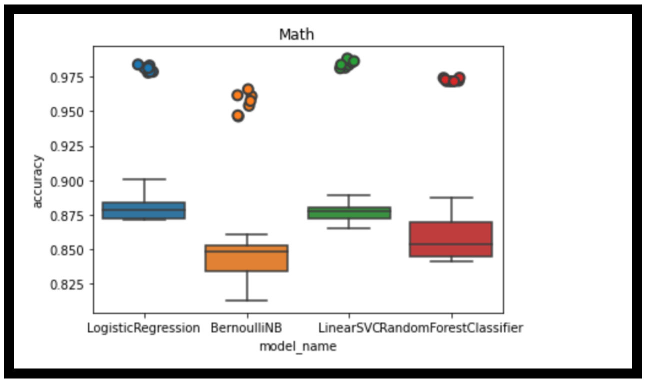
```
CompVis
[[17471    55]
 [ 2147     5]]
Accuracy: 0.888098383982112
Macro Precision: 0.4869465117069358
Macro Recall: 0.4995926126960815
Macro F1 score:0.4726190021806534
MCC:-0.004612082133892794
```

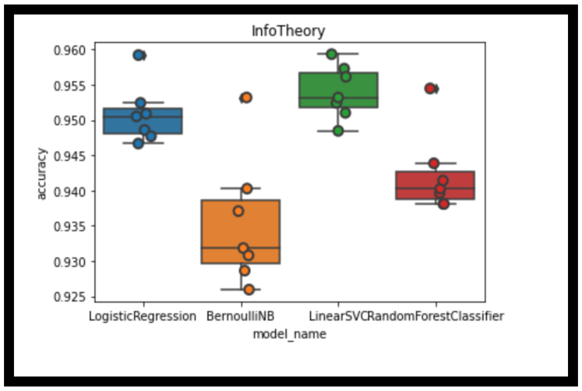
Machine learning method

Machine learning is a method of data analysis that automates model building. It is based on the idea that systems can learn from data and identify patterns and make decisions. In this assignment we use four different type of algorithms namely Logistic regression, Bernoulli NB, Linear SVC and Random Forest classifier on the training data to find the best model performing on it and then use the best model for classification of the classes.

The three classes and text column are saved separately from the training and testing data and are converted into a list for further processing. The text is vectorized by using TfidfVectorizer, Lemma tokenizer with the help of regex function. The training for ‘Abstract’ and the classes is defined using the ‘fit_transform’ function. We have processed all the four machine learning models for the training data with respect to the three input classes.

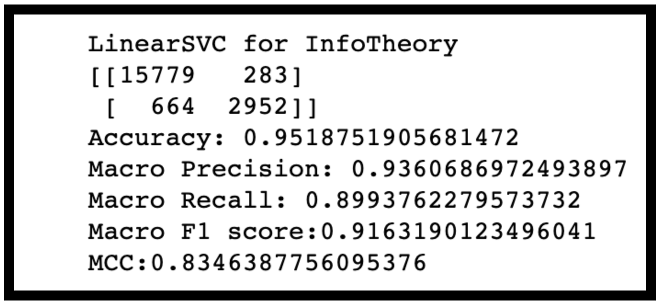




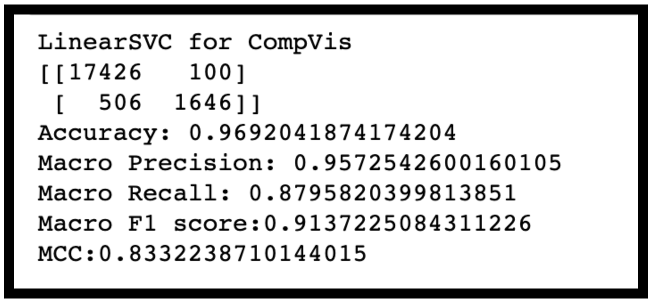


Analysing the performances of different models on the training data, it can be concluded that Linear SVC has the highest accuracy and is chosen to build on the test data.

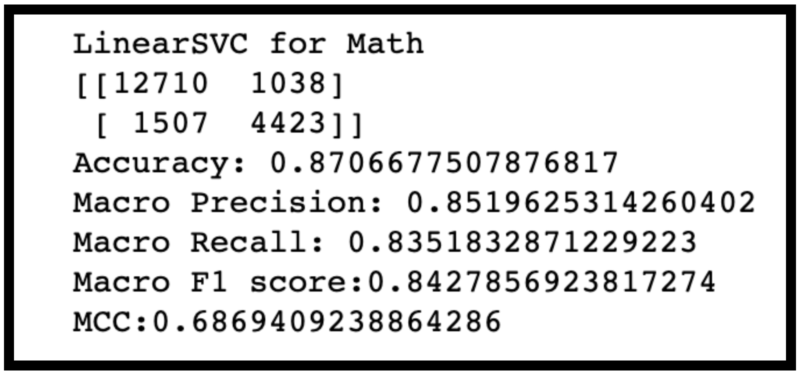
Linear SVC model is built using the training data for the three classes and it is used to predict the class values for the test data.



```
LinearSVC for InfoTheory
[[15779   283]
 [  664  2952]]
Accuracy: 0.9518751905681472
Macro Precision: 0.9360686972493897
Macro Recall: 0.8993762279573732
Macro F1 score:0.9163190123496041
MCC:0.8346387756095376
```



```
LinearSVC for CompVis
[[17426   100]
 [  506  1646]]
Accuracy: 0.9692041874174204
Macro Precision: 0.9572542600160105
Macro Recall: 0.8795820399813851
Macro F1 score:0.9137225084311226
MCC:0.8332238710144015
```



```
LinearSVC for Math
[[12710  1038]
 [ 1507  4423]]
Accuracy: 0.8706677507876817
Macro Precision: 0.8519625314260402
Macro Recall: 0.8351832871229223
Macro F1 score:0.8427856923817274
MCC:0.6869409238864286
```

Conclusion

As evident from the confusion matrices, the accuracy of machine learning method Linear SVC is higher as compared to the RNN method for all the three classes.

Part 2: Topic Modelling

Topic modelling is grouping of the similar topics together. We have preprocessed the data and performed two different variations for running LDA models. Both of these models are run on the first 1000 articles and 20000 articles in the training data sets.

Preprocessing input data

The input data is read and the 'Abstract' field of the data is converted into a list for processing, then the sentence is split into words, then lemmatized and finally tokenized. So as to get different data sets of different sizes to analyse 1000 and 20000 articles, a separate function is created which takes as an argument the number of articles, then returns the data set with that many records.

Model preparation

In this assessment, two different models are used on these two different data sets of 1000 and 20000 articles. The two models are different in the way that the first model(model 1) is created using bigrams and number of topics = 10 whereas in the second model(model 2) we don't use bigrams and number of topics = 40.

Model 1

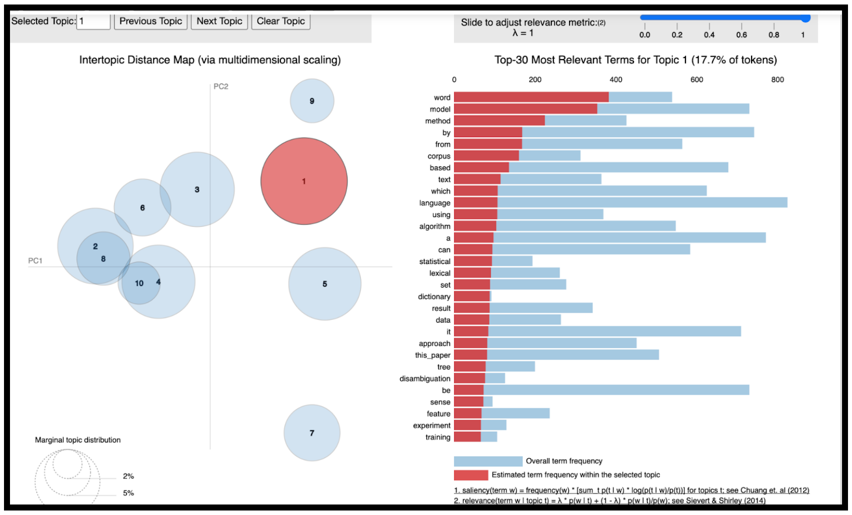
For model 1, we create bigrams using the function 'Phrases()', the bigrams are kept which have occurred for at least 20 times and are added to the tokens list. After that a dictionary representation is created for the tokens. From the dictionary we filter out the less frequent words and index the dictionary. After this we create different models using this dictionary using the LdaModel() function and train it with the training data of 1000 articles and 20000 articles. We then save these models and find out the top topics for these models and calculate the average topic coherence by dividing the sum of topic coherences of all the topics by the number of topics.

Model 2

For model 2, we don't use any bigrams. We create tokens by passing the number of articles to the predefined function i.e for 1000 articles and 20000 articles, and get the tokens generated by it. After that we create a dictionary representation of the tokens. We remove the less frequent words from the dictionary and generate a corpus. We now create a model using the LdaModel() function and train it with the training data of 1000 and 20000 articles. As done for the previous model we calculate the average topic coherence for both the models and save the model as well.

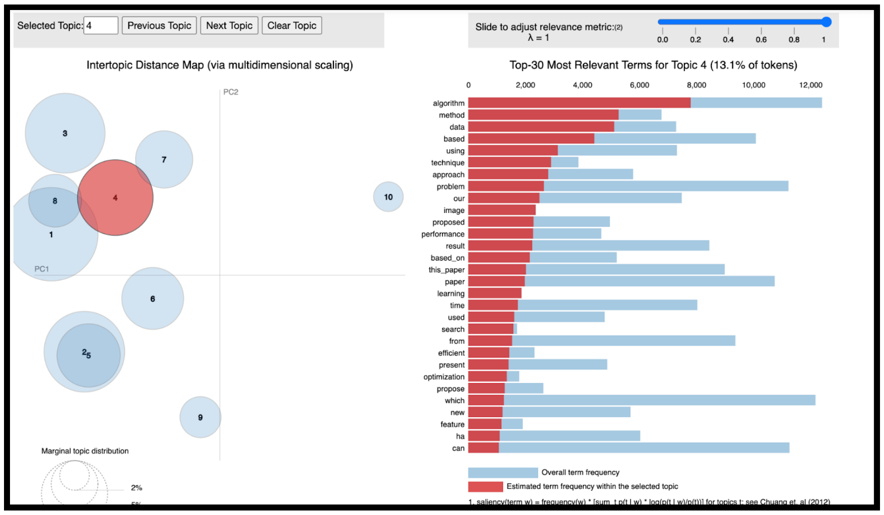
Topic Modelling Visualisations

For model 1 with 1000 articles:



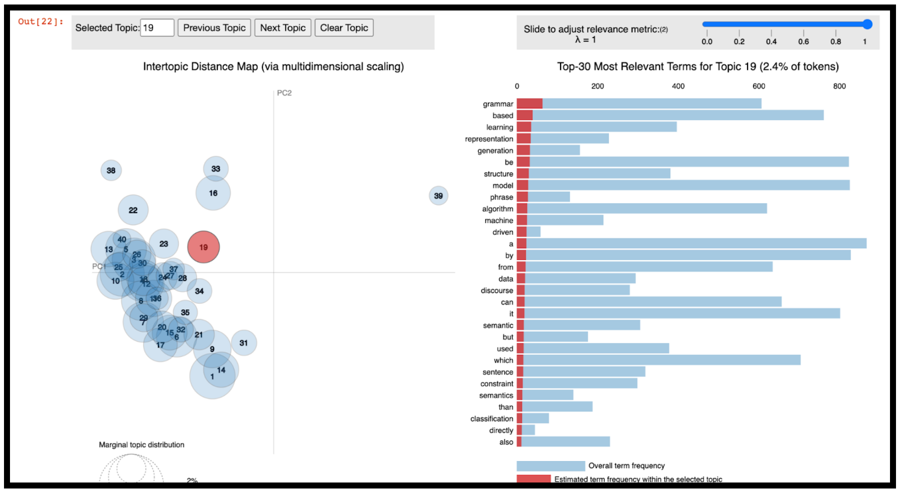
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

For model 1 with 20000 articles:

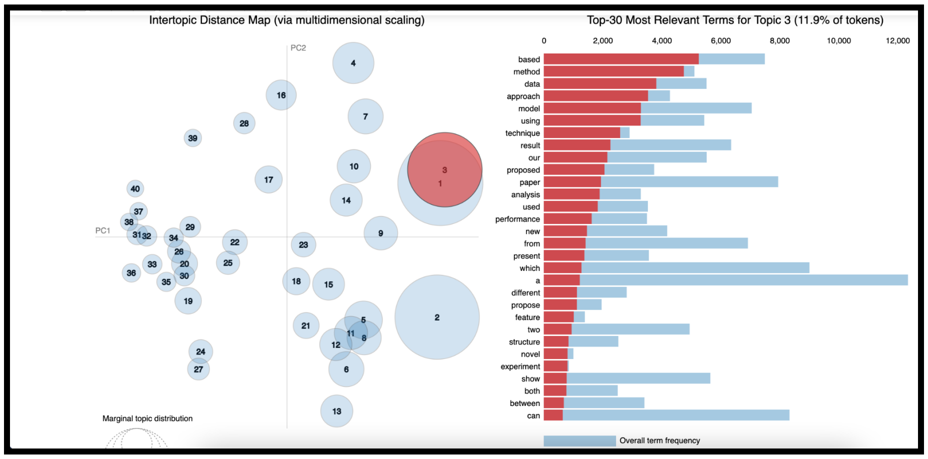


1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)

For model 2 with 1000 articles:



For model 2 with 20000 articles:

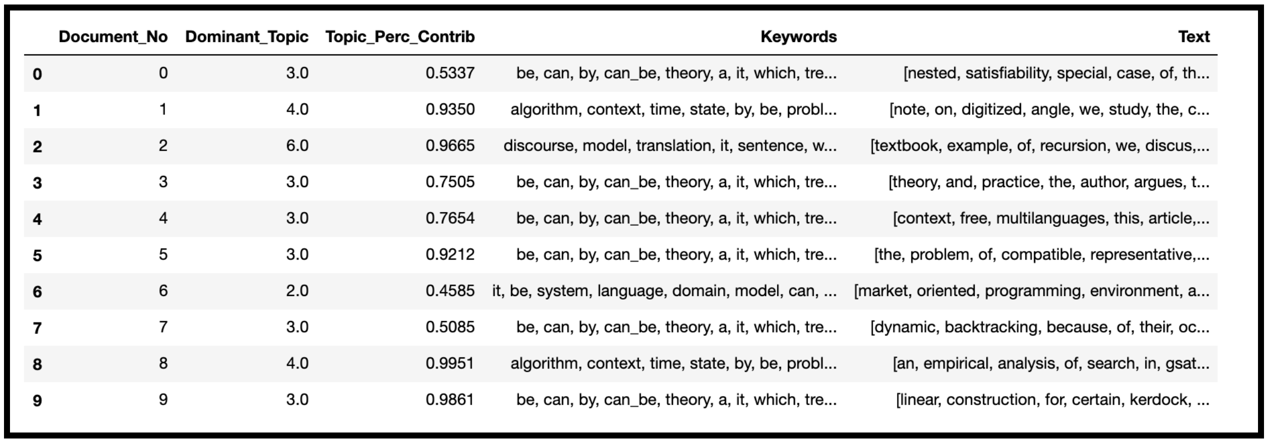


The above visualisations give a better way to understand the topics generated by different models. The visuals show the top topics generated by a model, and the top keywords of the selected topics in the sidebar. The sidebar also shows the frequency of the words, in all the topics and in the selected topic.

Dominant topics and their corresponding articles

In this we map the dominant topics with their corresponding articles generated with each of the models with different configurations.

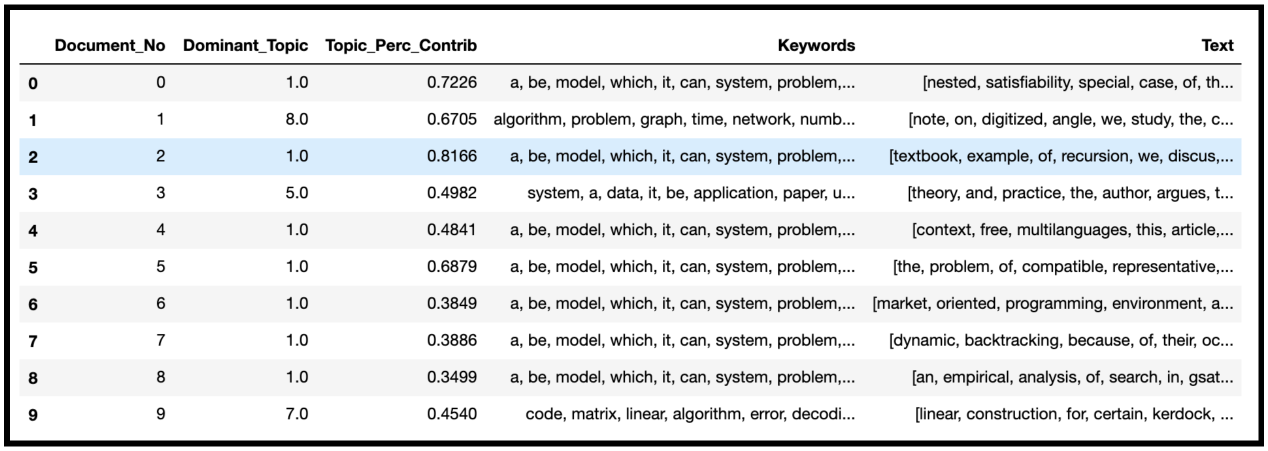
Model 1 with 1000 articles:



	Document_No	Dominant_Topic	Topic_Perc_Contrib	Keywords	Text
0	0	3.0	0.5337	be, can, by, can_be, theory, a, it, which, tre...	[nested, satisfiability, special, case, of, th...
1	1	4.0	0.9350	algorithm, context, time, state, by, be, probl...	[note, on, digitized, angle, we, study, the, c...
2	2	6.0	0.9665	discourse, model, translation, it, sentence, w...	[textbook, example, of, recursion, we, discus,...
3	3	3.0	0.7505	be, can, by, can_be, theory, a, it, which, tre...	[theory, and, practice, the, author, argues, t...
4	4	3.0	0.7654	be, can, by, can_be, theory, a, it, which, tre...	[context, free, multilanguages, this, article,...
5	5	3.0	0.9212	be, can, by, can_be, theory, a, it, which, tre...	[the, problem, of, compatible, representative,...
6	6	2.0	0.4585	it, be, system, language, domain, model, can, ...	[market, oriented, programming, environment, a...
7	7	3.0	0.5085	be, can, by, can_be, theory, a, it, which, tre...	[dynamic, backtracking, because, of, their, oc...
8	8	4.0	0.9951	algorithm, context, time, state, by, be, probl...	[an, empirical, analysis, of, search, in, gsat...
9	9	3.0	0.9861	be, can, by, can_be, theory, a, it, which, tre...	[linear, construction, for, certain, kerdock, ...

For model 1 trained on 1000 articles, we can see that not for all of the documents the keywords given by the model are not so comprehensible set of words and related much to the topic of the document specified. For example, in the image above for document number 6 we have some keywords system, language, domain and model. The title of the document number 6 is *'A Market-Oriented Programming Environment and its Application to Distributed Multicommodity Flow Problems',* we can see that the model has returned the keywords related to the document. But for document number 0 there are not any comprehensible keywords through which we can get the topic of the model. Overall the identification of topics of the documents from this model isn't very appropriate.

Model 1 with 20000 articles:

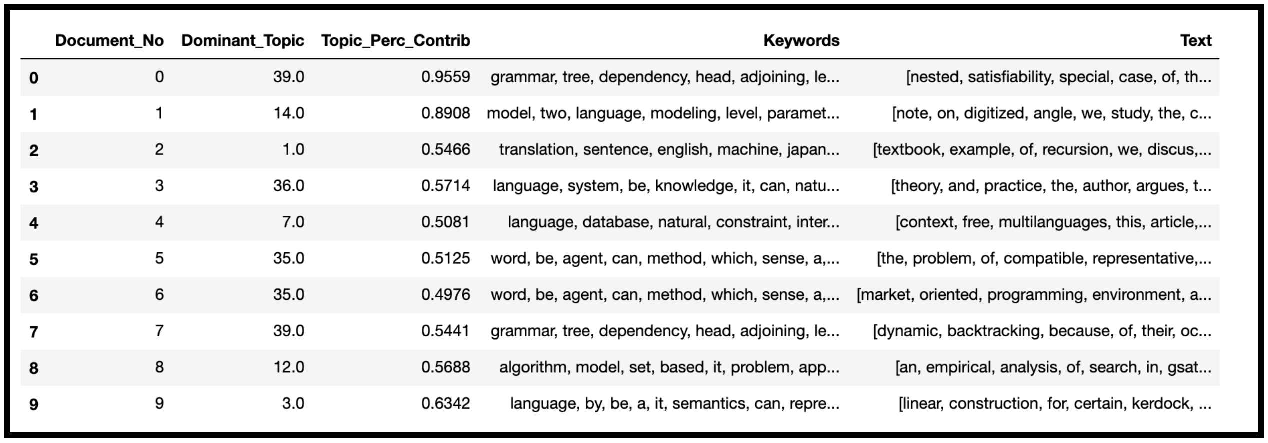


	Document_No	Dominant_Topic	Topic_Perc_Contrib	Keywords	Text
0	0	1.0	0.7226	a, be, model, which, it, can, system, problem,...	[nested, satisfiability, special, case, of, th...
1	1	8.0	0.6705	algorithm, problem, graph, time, network, numb...	[note, on, digitized, angle, we, study, the, c...
2	2	1.0	0.8166	a, be, model, which, it, can, system, problem,...	[textbook, example, of, recursion, we, discus,...
3	3	5.0	0.4982	system, a, data, it, be, application, paper, u...	[theory, and, practice, the, author, argues, t...
4	4	1.0	0.4841	a, be, model, which, it, can, system, problem,...	[context, free, multilanguages, this, article,...
5	5	1.0	0.6879	a, be, model, which, it, can, system, problem,...	[the, problem, of, compatible, representative,...
6	6	1.0	0.3849	a, be, model, which, it, can, system, problem,...	[market, oriented, programming, environment, a...
7	7	1.0	0.3886	a, be, model, which, it, can, system, problem,...	[dynamic, backtracking, because, of, their, oc...
8	8	1.0	0.3499	a, be, model, which, it, can, system, problem,...	[an, empirical, analysis, of, search, in, gsat...
9	9	7.0	0.4540	code, matrix, linear, algorithm, error, decodi...	[linear, construction, for, certain, kerdock, ...

For model 1 trained on the first 20000 articles from the training data we can see that not for much of the documents the training words are comprehensible. For example, for document number 9 as shown above the keywords give a good idea about the topic of the article which is *'A linear construction for certain Kerdock and Preparata codes'* . But for documents like document 0 '*Nested satisfiability*' and document 7 '*Dynamic Backtracking*' the keywords are not at all comprehensible so that doesn't make any sense for the topic of the article.

Comparing the above two models trained different sets of data sizes, it can be said that although both models are not very appropriate in identifying the topics of the documents but given the larger size of training data for the second model, the second does identify topics better than the first one.

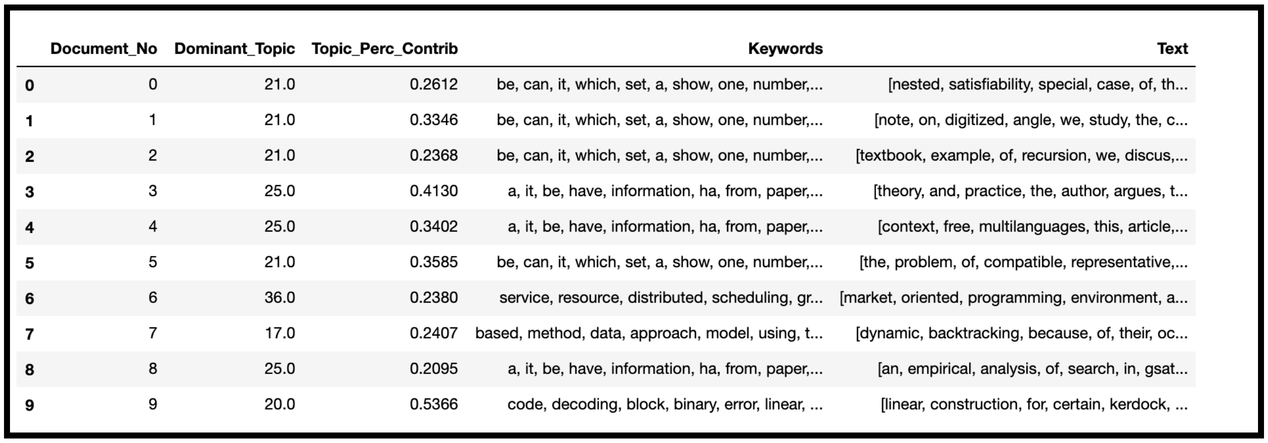
Model 2 with 1000 articles:



	Document_No	Dominant_Topic	Topic_Perc_Contrib	Keywords	Text
0	0	39.0	0.9559	grammar, tree, dependency, head, adjoining, le...	[nested, satisfiability, special, case, of, th...
1	1	14.0	0.8908	model, two, language, modeling, level, paramet...	[note, on, digitized, angle, we, study, the, c...
2	2	1.0	0.5466	translation, sentence, english, machine, japan...	[textbook, example, of, recursion, we, discus,...
3	3	36.0	0.5714	language, system, be, knowledge, it, can, natu...	[theory, and, practice, the, author, argues, t...
4	4	7.0	0.5081	language, database, natural, constraint, inter...	[context, free, multilanguages, this, article,...
5	5	35.0	0.5125	word, be, agent, can, method, which, sense, a,...	[the, problem, of, compatible, representative,...
6	6	35.0	0.4976	word, be, agent, can, method, which, sense, a,...	[market, oriented, programming, environment, a...
7	7	39.0	0.5441	grammar, tree, dependency, head, adjoining, le...	[dynamic, backtracking, because, of, their, oc...
8	8	12.0	0.5688	algorithm, model, set, based, it, problem, app...	[an, empirical, analysis, of, search, in, gsat...
9	9	3.0	0.6342	language, by, be, a, it, semantics, can, repre...	[linear, construction, for, certain, kerdock, ...

This model provides a comprehensible set of keywords through which we can know what an article is about for a good part of the data set. For example document 4 *'Context-free* *multilanguages'* the keywords like *'language'* gives us an idea about the topic of the article but for documents like document 0 '*Nested satisfiability'* the keywords don't identify the document correctly, in fact the keywords are nowhere related to the content of the article.

Model 2 with 20000 articles:



	Document_No	Dominant_Topic	Topic_Perc_Contrib	Keywords	Text
0	0	21.0	0.2612	be, can, it, which, set, a, show, one, number,...	[nested, satisfiability, special, case, of, th...
1	1	21.0	0.3346	be, can, it, which, set, a, show, one, number,...	[note, on, digitized, angle, we, study, the, c...
2	2	21.0	0.2368	be, can, it, which, set, a, show, one, number,...	[textbook, example, of, recursion, we, discus,...
3	3	25.0	0.4130	a, it, be, have, information, ha, from, paper,...	[theory, and, practice, the, author, argues, t...
4	4	25.0	0.3402	a, it, be, have, information, ha, from, paper,...	[context, free, multilanguages, this, article,...
5	5	21.0	0.3585	be, can, it, which, set, a, show, one, number,...	[the, problem, of, compatible, representative,...
6	6	36.0	0.2380	service, resource, distributed, scheduling, gr...	[market, oriented, programming, environment, a...
7	7	17.0	0.2407	based, method, data, approach, model, using, t...	[dynamic, backtracking, because, of, their, oc...
8	8	25.0	0.2095	a, it, be, have, information, ha, from, paper,...	[an, empirical, analysis, of, search, in, gsat...
9	9	20.0	0.5366	code, decoding, block, binary, error, linear, ...	[linear, construction, for, certain, kerdock, ...

For the model 2 with 20000 training articles very few articles are defined by using the topic model, for example the document 6 *'A Market-Oriented Programming Environment and its Application to Distributed Multicommodity Flow Problems'* the keywords give us an idea about the topic of the article but for many articles like document 0 *'Nested satisfiability'* the keywords are not comprehensible enough to justify the topic.

Comparison of models

The models 1 and 2 differ quite a lot in identifying the topics, for model 1 we use less number of topics and use bigrams to train the model whereas for model 2 we use more number of topics and no bigrams to train the model. The use of bigrams in determining the coherence between words seems to be an important part of model development as getting the context between words is important to identify what the abstract says about the topic. Also using less number of topics to train the model is a negative point as it links together some keywords which have very little coherence, thus causing lower performance of model 1. Not using bigrams in model 2 caused generation of a lot of unnecessary and redundant keywords which don't give any input into determining the topic of a document.

Each of these models were performed on two sizes of training data one with 1000 and other with 20000 articles. As shown in the visuals created for both of these models on the two data sets, the models on less training data set have topics less correlated to each other and on the training data with high articles 20000 in training data more number of topics are uncovered with a lot of them overlapping thus sharing significant keywords.