Identifying keywords and topics from questions on Stack Exchange Sites

Sayan Das
4/6/20

What is the goal?

- To identify keywords. topics and summaries from user questions on *Stack Exchange.*
- Use the results to optimize sorting of user queries, providing quicker and more relevant answers.

What is Stack Exchange?

- An online platform for sharing technical knowledge and collaboration on projects.
- Serves as an umbrella for actively used sites such as *Stack Overflow*, *Super Users*, and *Ask Ubuntu* containing a rich collection of text data.
- Provides a network for questions and answers across a wide range of technical topics.
- Contains basic as well as complex queries.

Approach

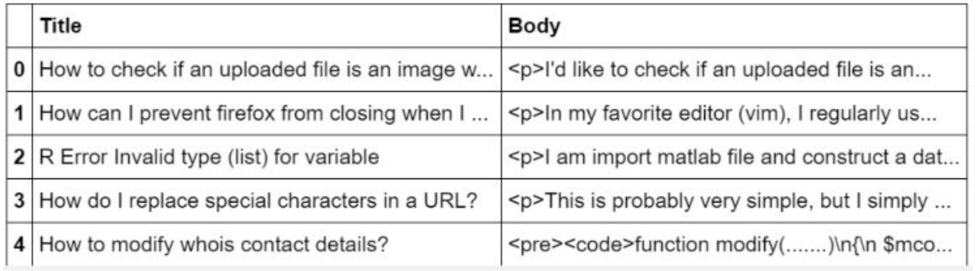
- Use topic modeling to identify keywords related to questions using *Latent Dirichlet Allocation (LDA).*
- Analyze the text data using *Natural Language Processing (NLP)* techniques to find insights.

Who is this useful to?

- Stack Exchange itself as well as similar platforms such as *Quora*, *Reddit*, *LinkedIn groups* etc., could use this model or adapt the model to their specific domain.
- The methodology can be extended to search engines such as *Google* and *Bing*.
- Furthermore, similar analysis can be applied to other text data such as comments, reviews, item descriptions on platforms such as *Yelp*, *Amazon* and *Twitter*.

Data Description

- Questions from users on Stack Exchange websites
- Contained **420,668 rows** and **two columns**:
  - *Title* of the question
  - *Body* of the question
- Snippet of the dataset is shown below



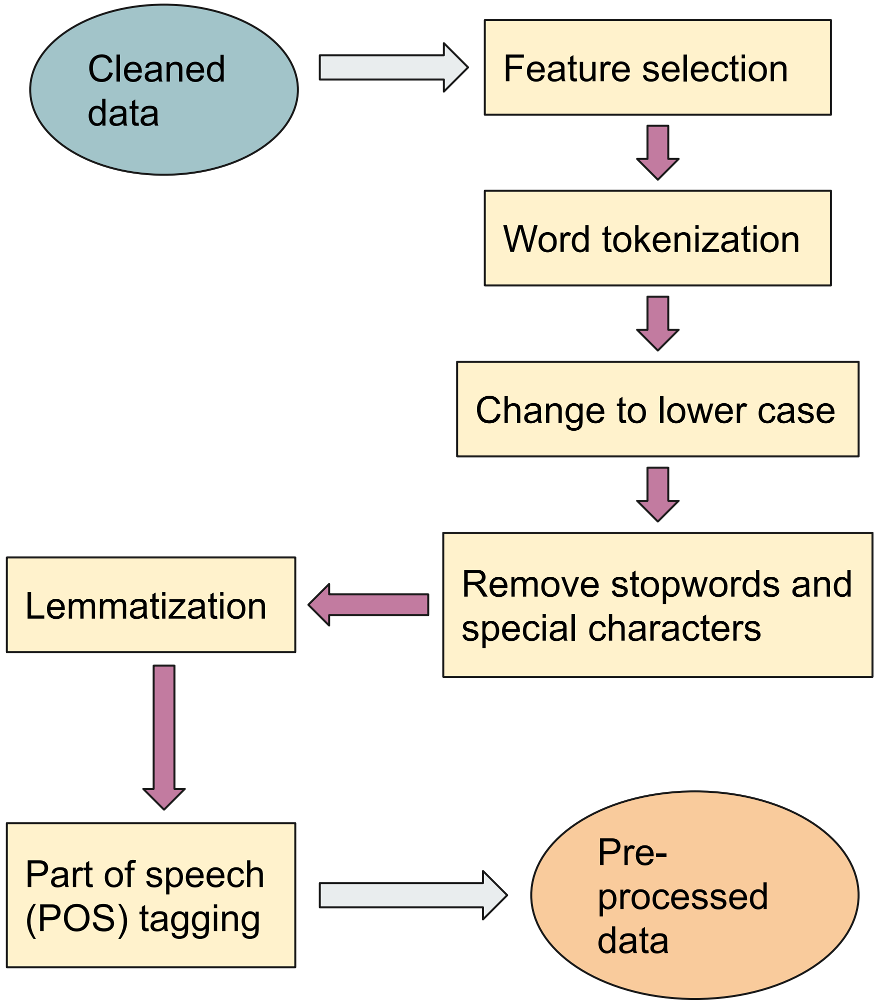
	Title	Body
0	How to check if an uploaded file is an image w...	<p>I'd like to check if an uploaded file is an...
1	How can I prevent firefox from closing when I ...	<p>In my favorite editor (vim), I regularly us...
2	R Error Invalid type (list) for variable	<p>I am import matlab file and construct a dat...
3	How do I replace special characters in a URL?	<p>This is probably very simple, but I simply ...
4	How to modify whois contact details?	<pre><code>function modify(.......)\n{\n $mco...

Data Cleaning Process

- Removed missing values:
  - 110 missing rows were excluded from the dataset.
- Removed HTML tags
  - Tags such as *<p>*, *</p>*, *<code>*, *</code>* don't provide useful information.
  - Most of the tags were contained in the *Body* variable.
- Removed newline characters

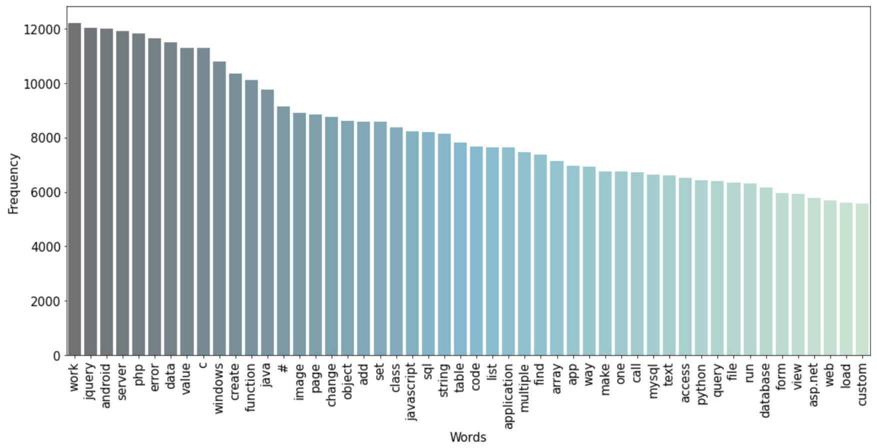
Preprocessing

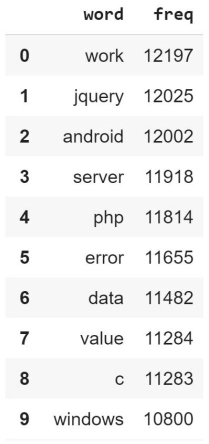
- Essential steps to prepare the text data for topic modeling
- The final model is only as good as the quality of the data it receives
- Used Python's *nltk* NLP library to feature engineer and format the data



Feature selection

- *Title* was picked as the main text feature for the following reasons
  - Tokenizing over 450,000 text documents for both *Title* and *Body* took up too much computational resources.
  - Initial analysis with just 500 observations revealed that *Title* texts contained more crucial keywords which would be important for the topic model.

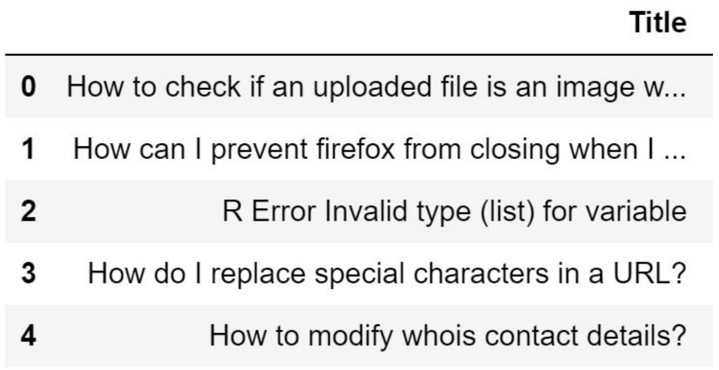




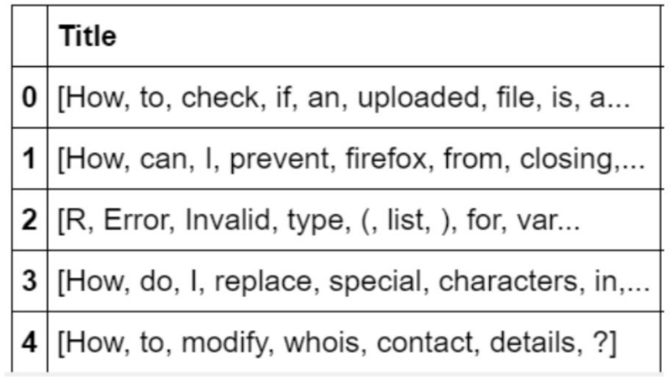
	word	freq
0	work	12197
1	jquery	12025
2	android	12002
3	server	11918
4	php	11814
5	error	11655
6	data	11482
7	value	11284
8	c	11283
9	windows	10800

Word tokenization

- Splits a sentence or text document into tokens which could be words, special characters, punctuations etc.
- Tokenization process shown below.



	Title
0	How to check if an uploaded file is an image w...
1	How can I prevent firefox from closing when I ...
2	R Error Invalid type (list) for variable
3	How do I replace special characters in a URL?
4	How to modify whois contact details?



	Title
0	[How, to, check, if, an, uploaded, file, is, a...
1	[How, can, I, prevent, firefox, from, closing,...
2	[R, Error, Invalid, type, (, list,), for, var...
3	[How, do, I, replace, special, characters, in,...
4	[How, to, modify, whois, contact, details, ?]

Convert tokens to lowercase

- Conversion done for the topic model to treat two words such as ‘Java’ and ‘java’ as the same for practical purposes.

Removing stopwords and special characters

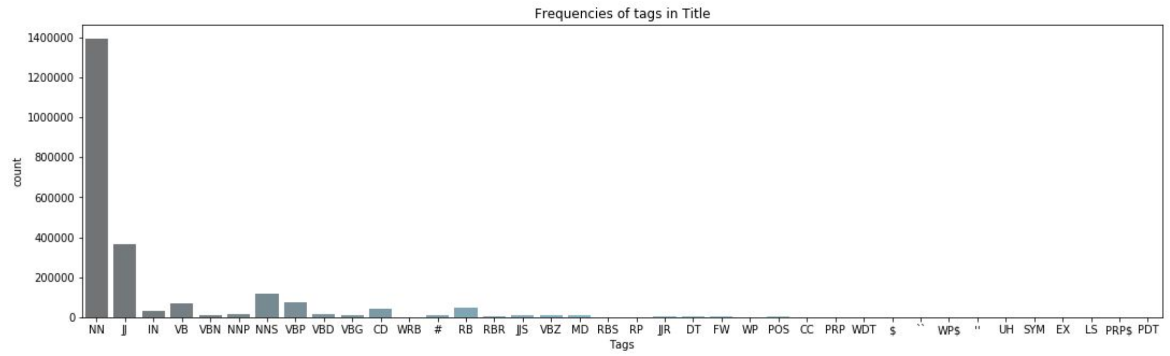
- Stopwords provide very little information in terms of context during modeling.
- A list of all common stopwords were removed using Python’s *NLTK* library.
- Additional tokens such as *‘I’*, *‘The’*, *‘(‘*, *‘$’*, *‘+’* were removed iteratively by monitoring the highest occurring tokens in the dataset. These tokens are manually stored in a list and removed.

Lemmatization

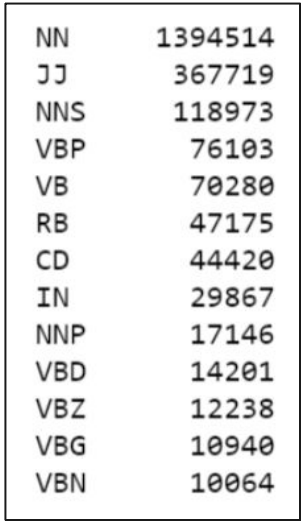
- NLP technique to reduce different variations of a word to a **single root word** provided a context such as *noun* or *verb* or *adjective*.
- In our case, the chosen context was a **verb.** Ex. *runs*, *ran* and *running* are all changed to the root word **run.**
- Lemmatization is done to reduce the number of unique words in the corpus so that the model treats different versions of a word in the same context.
- The verb context was chosen so that all possible verbs can be collected and filtered out, keeping only the nouns which provide more information for topic modeling.

Note: The *noun* filtering process is described in the next section (POS tagging).

POS tagging



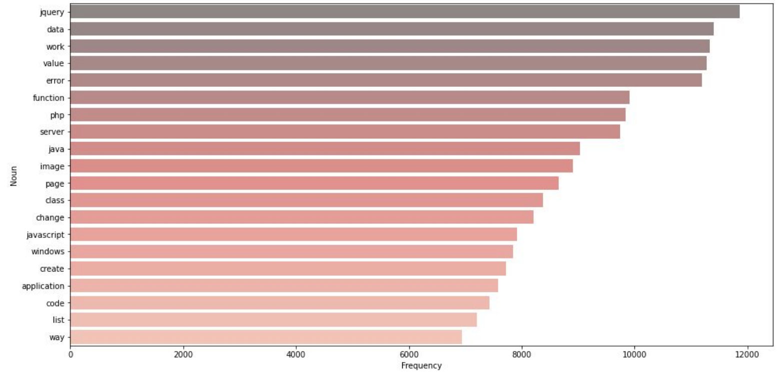
- Common POS tags
  - *Nouns*: NN (singular), NNS (plural), NNP (proper), NNPS (plural proper)
  - *Verbs*: VBP (singular), VBN (past participle), VBD (past tense)
  - *Adjectives*: JJ, JJR (comparative), JJS (superlative)
  - *Adverbs*: RB, RBR (comparative), RBS (superlative)
- Nouns were by far the highest occurring tags - *see graph (top-right) and table (right)*
  - Only nouns were kept in the data set since they provide the best context for topic modeling.
  - Useful for topic modeling since the goal is to capture technical keywords such as *Java*, *HTML*, or *iOS*.
- Adjectives were the second most frequent category
  - Contained some useful words such as **external** or **static** which described verbs or nouns.
  - But also contained a lot of lesser insightful words such as **possible** or **wrong**.

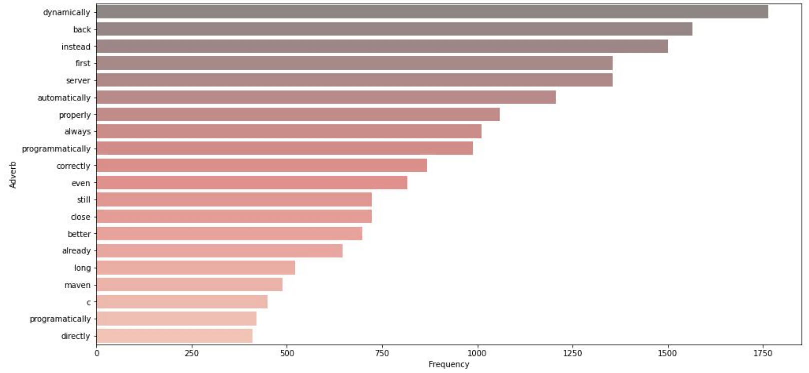


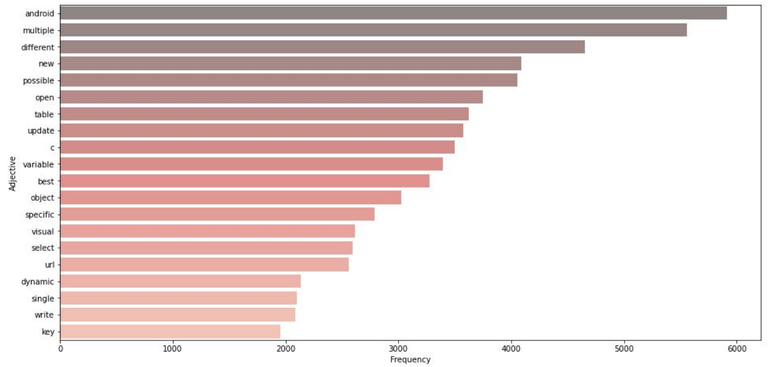
```
NN      1394514
JJ       367719
NNS      118973
VBP       76103
VB        70280
RB        47175
CD        44420
IN        29867
NNP       17146
VBD       14201
VBZ       12238
VBG       10940
VBN       10064
```

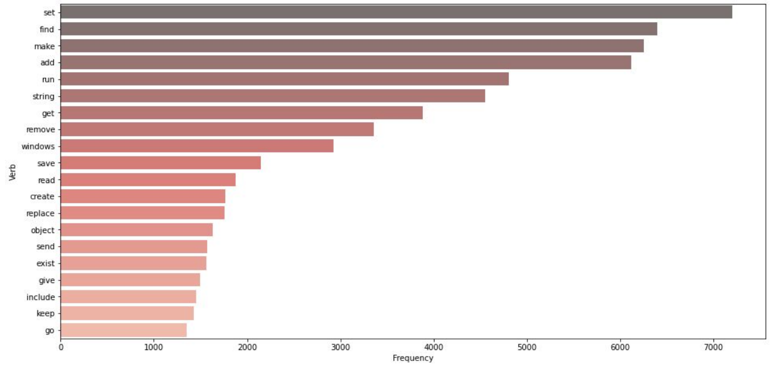
POS tagging (contd.)

- Nouns
  - The 20 highest occurring nouns in the dataset are shown in the top-left bar chart below.
  - Contains several crucial words such as *jquery*, *data*, *value*, *error*, *function*, *php*, *server*, *java*, *class*, *application*, etc.
- Adjectives/adverbs/verbs
  - Few adjectives provide meaningful keywords whereas most of the verbs and adverbs don't provide any insight at all.

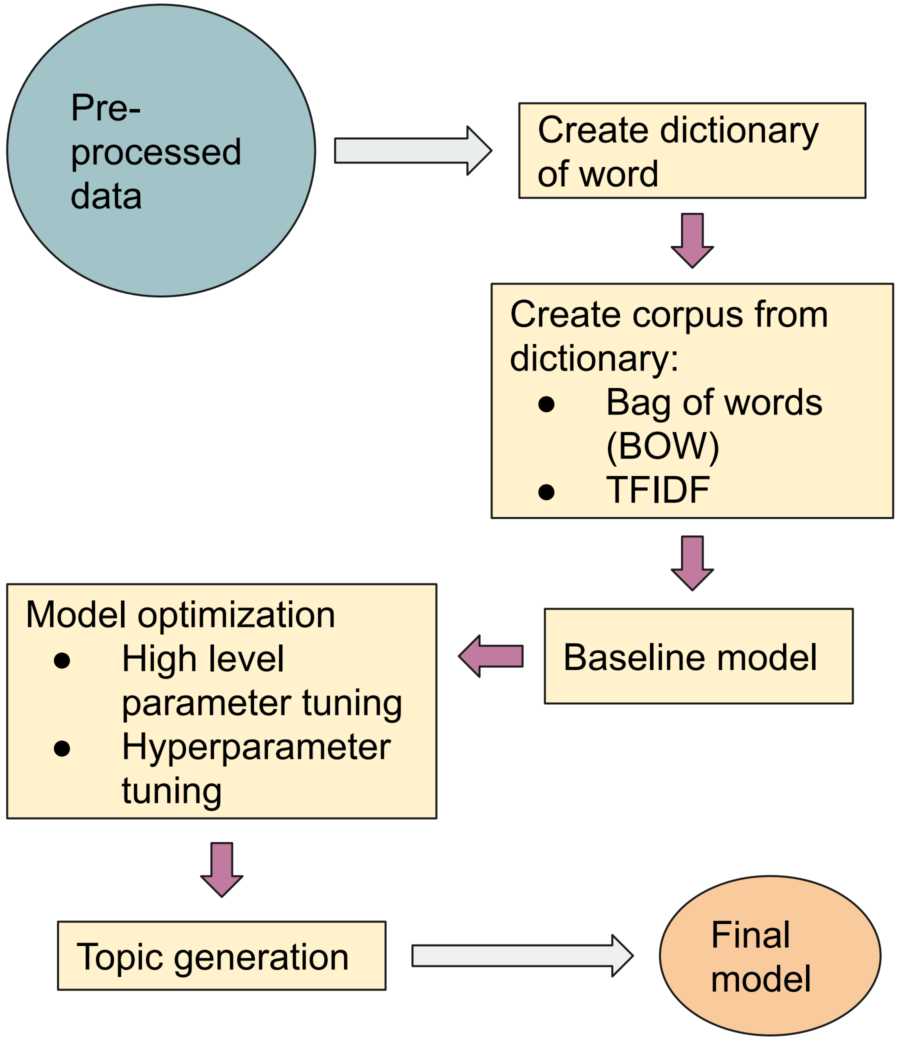








Topic modeling

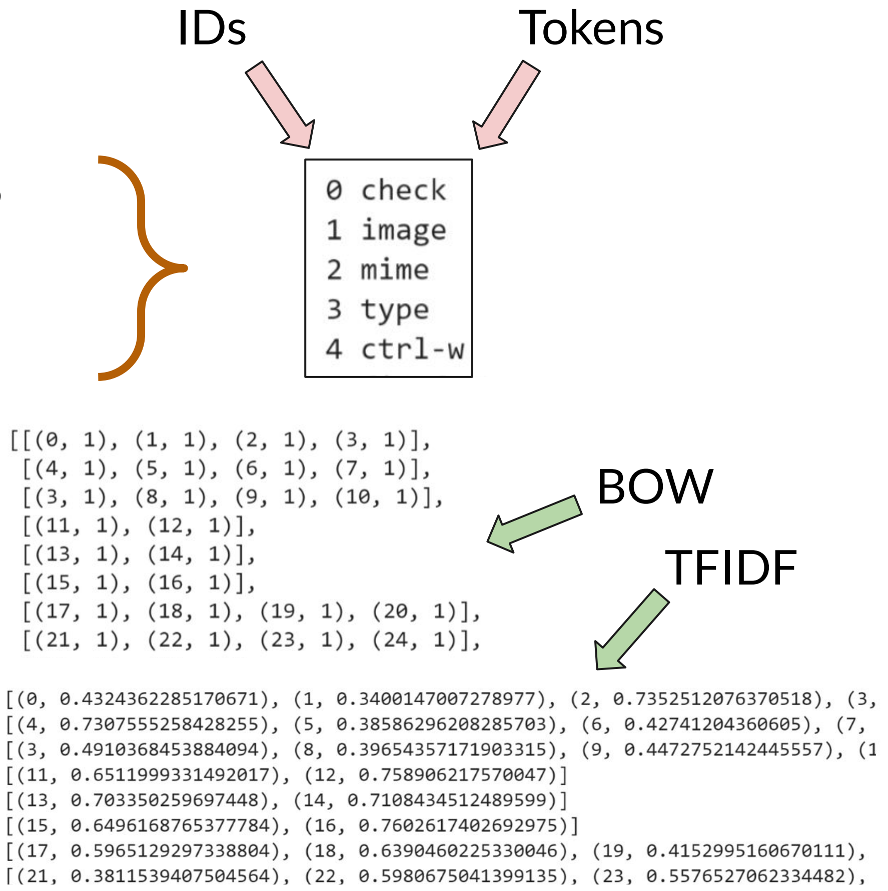


Creating dictionary of words

- Assigns a unique identifier to each token in the dataset
- Serves as a lookup table for all unique tokens (*see table on top-right*).

Creating a corpus

- BOW
  - Contains IDs representing each token and the frequency of that token in each document (*see table on middle-right*).
  - Each document consists of a list of tuples where the first number is the ID and the second the frequency.
- TFIDF
  - Same as BOW except the token frequencies are weighted decimal values (*bottom-right*).
- **TFIDF was selected after evaluating models using both methods**



```
0 check
1 image
2 mime
3 type
4 ctrl-w
```

```
[[(0, 1), (1, 1), (2, 1), (3, 1)],
 [(4, 1), (5, 1), (6, 1), (7, 1)],
 [(3, 1), (8, 1), (9, 1), (10, 1)],
 [(11, 1), (12, 1)],
 [(13, 1), (14, 1)],
 [(15, 1), (16, 1)],
 [(17, 1), (18, 1), (19, 1), (20, 1)],
 [(21, 1), (22, 1), (23, 1), (24, 1)],
```

```
[(0, 0.4324362285170671), (1, 0.3400147007278977), (2, 0.7352512076370518), (3,
[(4, 0.7307555258428255), (5, 0.38586296208285703), (6, 0.42741204360605), (7,
[(3, 0.4910368453884094), (8, 0.39654357171903315), (9, 0.4472752142445557), (1
[(11, 0.6511999331492017), (12, 0.758906217570047)]
[(13, 0.703350259697448), (14, 0.7108434512489599)]
[(15, 0.6496168765377784), (16, 0.7602617402692975)]
[(17, 0.5965129297338804), (18, 0.6390460225330046), (19, 0.4152995160670111),
[(21, 0.3811539407504564), (22, 0.5980675041399135), (23, 0.5576527062334482),
```

LDA modeling parameters

- Dictionary parameters
  - *No_below*: Ignores words that occur in less than the specified number.
  - *No_above*: Ignores words that occur in more than specified percentage of documents.
  - *Keep_n*: Only considers the specified number of most frequent words in the corpus for modeling.
- Gensim LDA model high level parameters
  - *Num_topics*: Number of topics to generate.
  - *Passes*: Number of iterations of modeling. Each iteration goes through all documents once and updates the word distributions for each topic.
- Gensim LDA model hyperparameters
  - **Alpha**
  - **Beta**

LDA evaluation metrics

- Topic interpretability
- Word distribution in each topic
- Word distribution in other topics
- Overall word distribution
- Coherence scores (*c_v* and *c_umass*)

Baseline model

- Parameter values
  - **No_below**: 10
  - **No_above**: 0.5
  - **Keep_n**: 1000
  - **Num_topics**: 20
  - **Passes**: 10
  - **Alpha**: 0.25
  - **Beta**: 0.01
- Results
  - **C_v** score of 0.4289
  - **C_umass** score of -11.9389
  - Fairly interpretable topics but too few to describe all documents accurately.
  - More uniform distribution of words in topics.

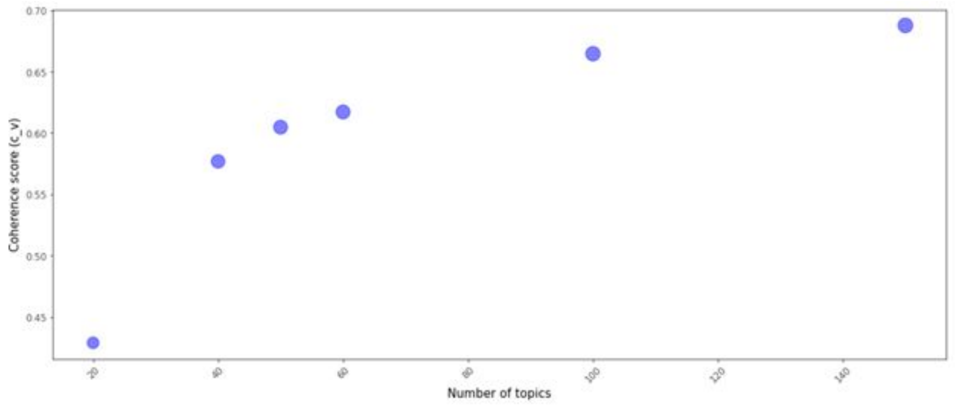
Final model

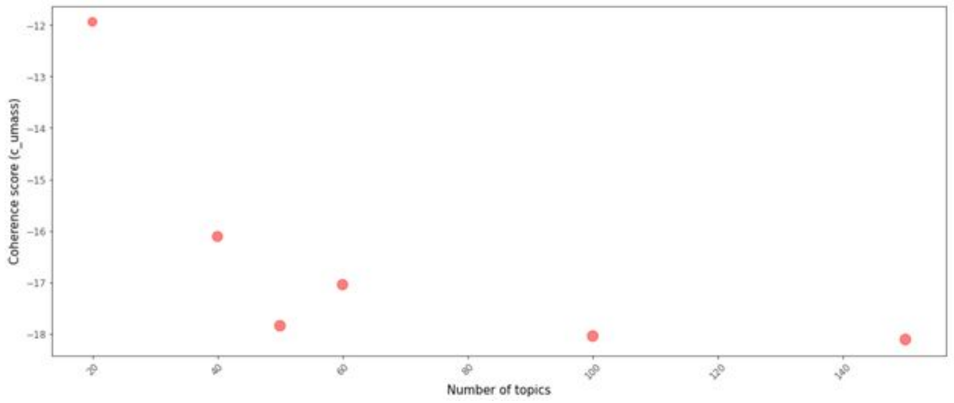
- Parameters values
  - **No_below**: 10
  - **No_above**: 0.5
  - **Keep_n**: 10000
  - **Num_topics**: 40
  - **Passes**: 50
  - **Alpha**: 1.25
  - **Beta**: 0.1
- Results
  - **C_v** score of 0.6107
  - **C_umass** score of -17.1428
  - Topics are less distinct but provide a more accurate description of documents.
  - Slightly right-skewed word distributions across topics.

Model optimization

High level model parameters

- *Number of topics*
  - The top-right plot shows an almost **linearly increasing c_v coherence score** with increasing topics.
  - There is a slight indication of a flattening of the curve as we reach 150 topics.
  - The **c_umass score decreases with increasing number of topics**.
- *Keep_n*
  - Increasing this value from 1000 to 10,000 provided a less skewed word distribution per topic.

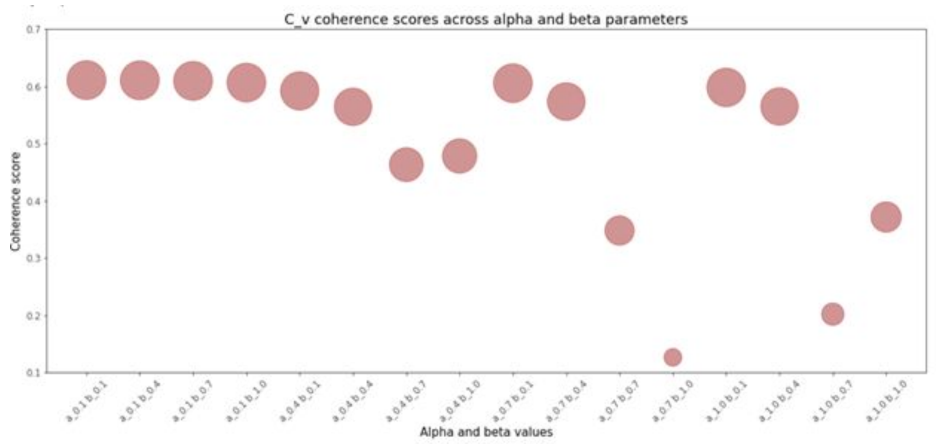


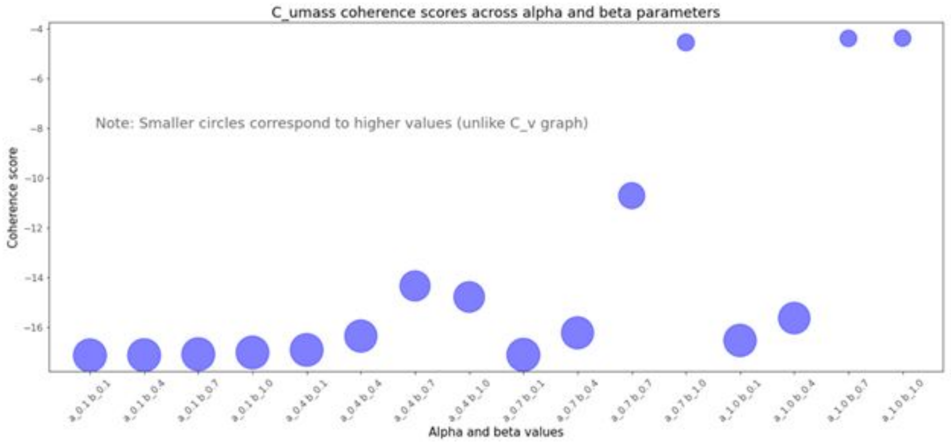


Model optimization

Hyperparameters

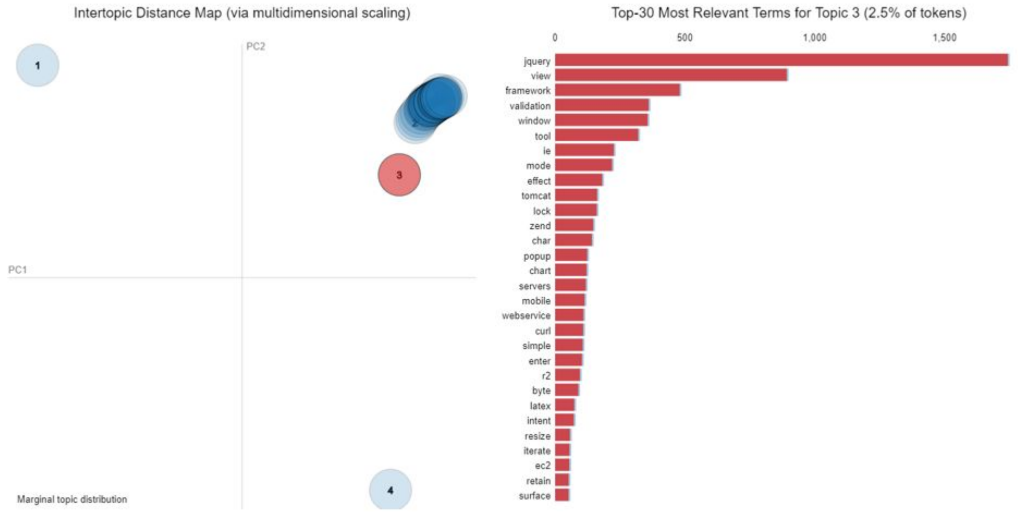
- *Alpha*
    - The best c_v score is **0.6107** (*top-right plot*) corresponding to an **alpha value of 0.1** and **beta value of also 0.1**.
    - General downward trend as beta is increased. The case is similar for alpha values but not as significant.
- *Beta*
    - General trend for the alpha parameter is slightly ascending (*bottom-right plot*)
    - Trend for beta is clearly upward trending with values such as 0.7 and 1.0 showing significantly higher scores.
    - A smaller u_mass score is more desirable.

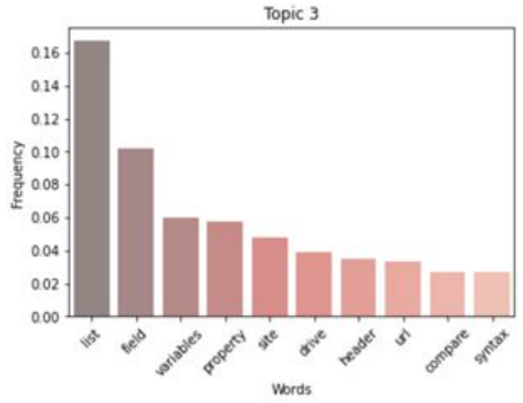


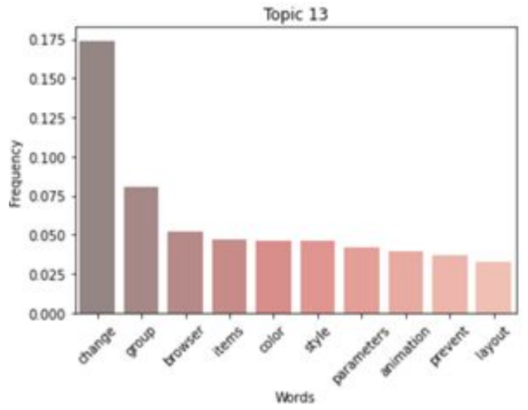


Topic generation

- Topics manually assigned based on word distributions for each topic using *PyLDAVis* graphs (*top-right plot*).
- Most topics contained more than one keyword to describe the prevalent themes contained within the topic.
- Word distributions for **topic 3 and topic 13** are shown as examples (*bottom-right graphs*). *Topic 3* was assigned keywords such as **data structures, websites** and **syntax** while *topic 6* was assigned the themes **web design, web layout** and **internet**.

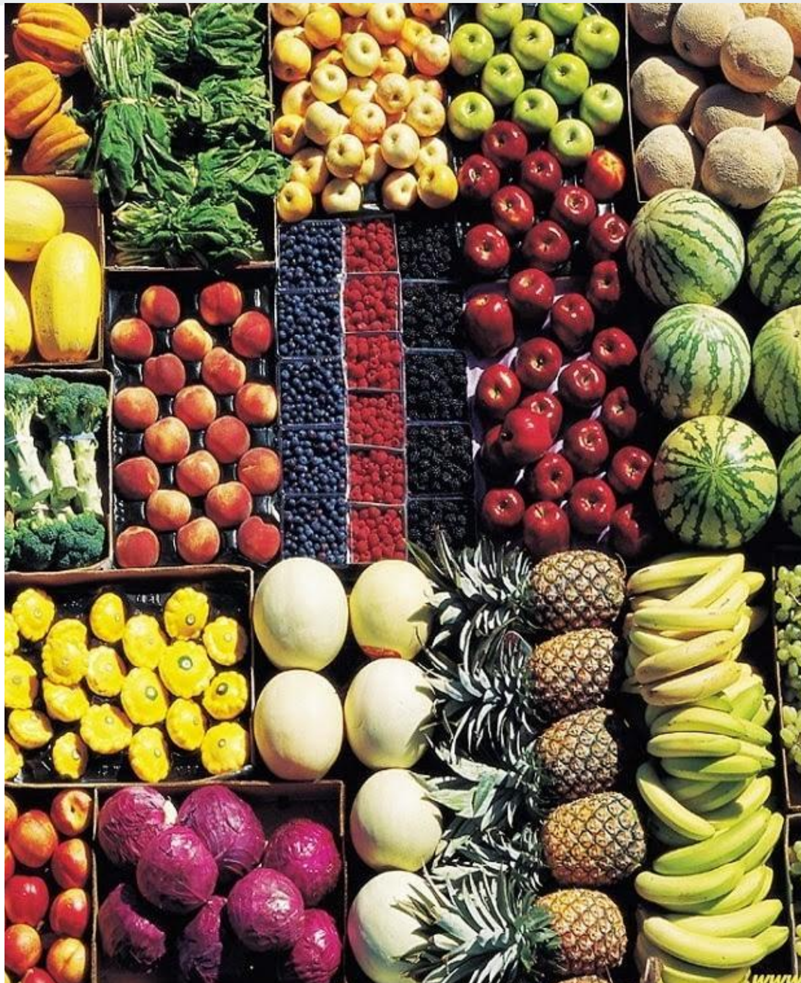






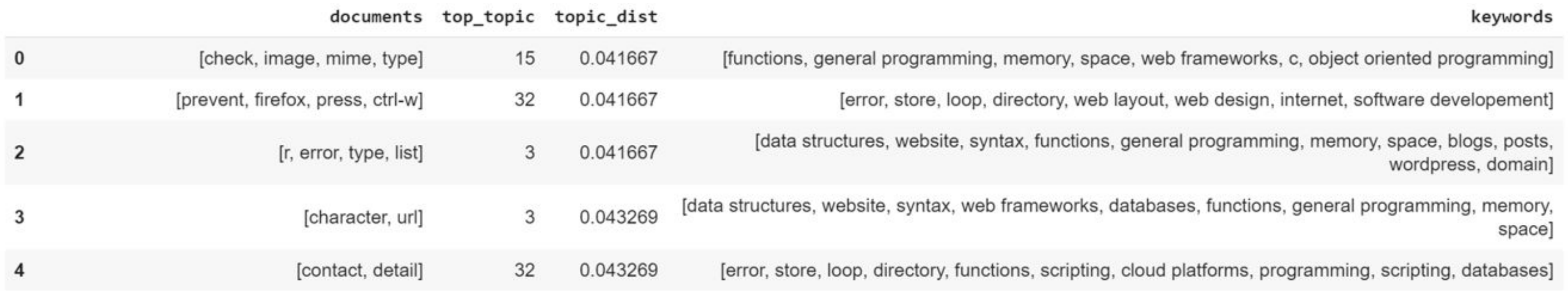
Model evaluation

- Prediction using training documents
- Prediction using test (unseen) documents
- Comparing train and test results



Predicting training documents

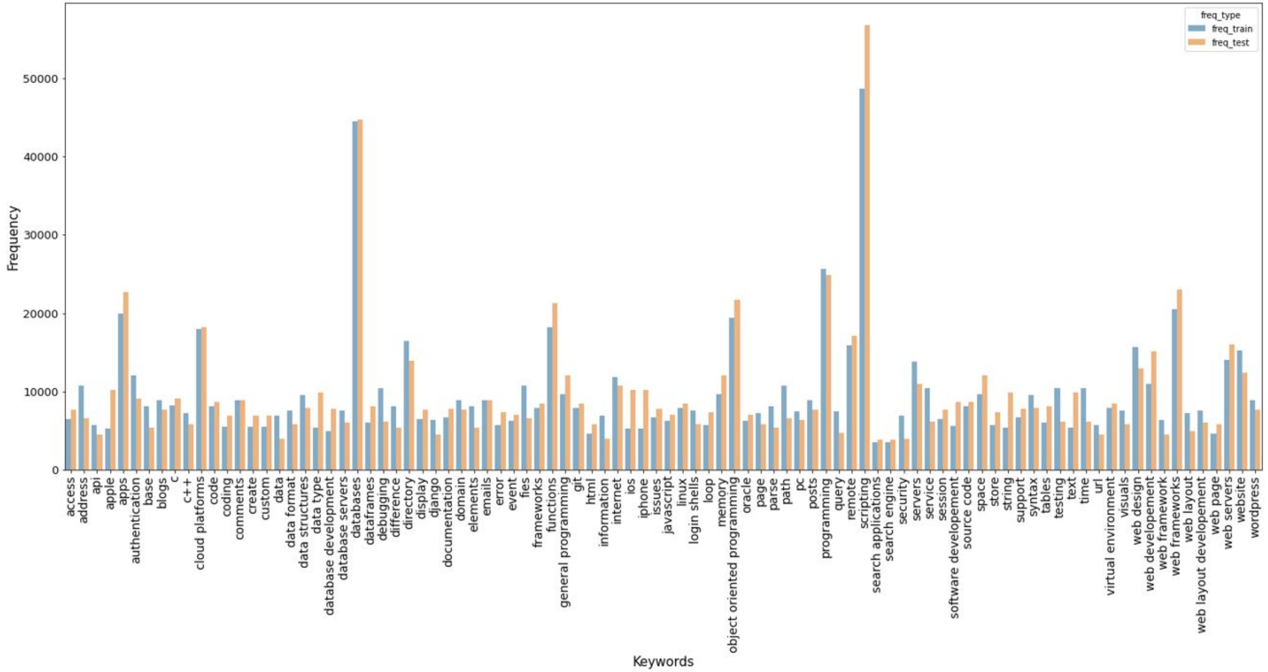
- The predictions were fairly specific due to the respective keywords covering a broad set of categories which could indicate some overfitting. Another reason is that the keywords are from the top three topics predicted for the documents instead of a single topic.
- In most cases, the keywords describe the respective documents fairly well.
- For the document **[‘r’,’error’,’type’,’list’]**, the keyword **data structures** describes **list** while **syntax** and **programming** describes **error, type** and **r**. However, the additional keywords such as space, blogs and post do not represent any of the document words which represents some of the error present in the predictions.



	documents	top_topic	topic_dist	keywords
0	[check, image, mime, type]	15	0.041667	[functions, general programming, memory, space, web frameworks, c, object oriented programming]
1	[prevent, firefox, press, ctrl-w]	32	0.041667	[error, store, loop, directory, web layout, web design, internet, software developement]
2	[r, error, type, list]	3	0.041667	[data structures, website, syntax, functions, general programming, memory, space, blogs, posts, wordpress, domain]
3	[character, url]	3	0.043269	[data structures, website, syntax, web frameworks, databases, functions, general programming, memory, space]
4	[contact, detail]	32	0.043269	[error, store, loop, directory, functions, scripting, cloud platforms, programming, scripting, databases]

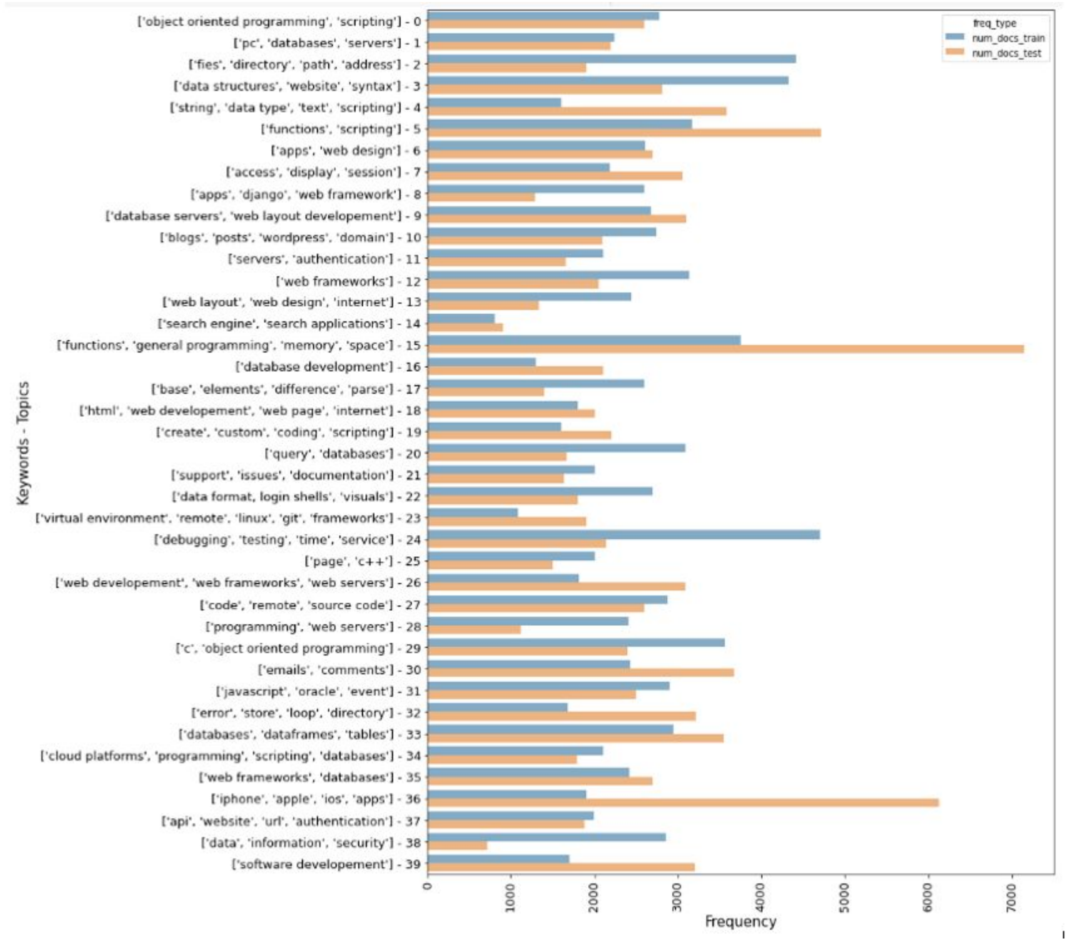
Predicting test (unseen) documents

- For most of the high frequency keywords, the **test numbers are higher** (*plot on right*).
- The sum of the frequencies of the top eight most frequent keywords for the test data (233,215) is approximately **30,000 higher than that of the train data (214,994)**.
- Considering that the total sum for all keywords are 885,299 and 884,673 for test and train data respectively, the difference between the top eight keywords seems significant, where the distance between the totals is about 600.



Predicting test (unseen) documents (contd.)

- Distribution showing the document frequencies for each topic (*right*).
- The most frequently occurring topics contain some of the same keywords which are the most frequent words in the predicted results.
- For example, the topic **['functions','general programming','memory','space']** contains words such as **functions** and **programming** which are within the top 8 most frequent keywords.



Conclusion

- **Modeling and prediction**
  - The final LDA model does a reasonable job in predicting unseen documents. However, it could sometimes over-predict a document.
  - Since a topic could be a combination of two or more themes, it would use extra words to explain a document whereby over-explaining certain documents.
  - In terms of predicting unseen data, the model provides similar results as the train data where both sets see the same eight most frequently occurring keywords such as **scripting**, **databases**, and **programming**. For lesser frequent words, the similarities between the sets reduce.
- **Insights**
  - Findings such as the most frequently occurring topics can be used by Stack Exchange or other forums or search engines to quickly filter search results. Less frequent topics, on the other hand, could be promoted in order to find answers for questions which might not be readily available.
  - The model can be applied to applications such as reviews, articles, social media comments etc., to provide users with relevant information. Additionally, the model can be used by search engines to display topics relevant to the typed word or words.