Topic Modeling and Latent Dirichlet Allocation (LDA)

Karnav Patel(17ce072)
Computer Engineering
Cspit, Charusat University
Changa

Nisarg patel(17ce080)
Computer Engineering
Cspit, Charusat University
Changa

Parth Patel(17ce081)
Computer Engineering
Cspit, Charusat University
Changa

Kishan Rupareliya(17ce094)
Computer Engineering
Cspit, Charusat University
Changa

Abstract

Topic Modeling provides a proper way to analyze big unclassified text. A topic contains a cluster of words that frequently used together. A topic modeling can connect words with same meanings and differentiate between uses of words with multiple meanings. This paper provides two categories that can be considered under topic modeling field. Initially discusses the area of methods of topic modeling, which contains four method and can be considered under this category. Latent semantic Analysis (LSA), probabilistic Latent Semantic Analysis (PLSA), Latent Dirichlet Allocation (LDA) and correlated Topic Model (CTM). The second category is called topic modeling evolution model, it considers an important factor time. Different models are discussed in this category, such as Dynamic Topic Models (DTM), Topic over time (TOT), Dynamic Topic Correlation Detection, Detecting Topic Evolution in scientific literatures etc.

Introduction

To have a best way of managing the explosion of electronic text document archives this days, it needs using new techniques or tools that manage automatically organizing, searching, indexing, summarizing and browsing big data. The basis of current research of ML and statistic, it has developed new

techniques for finding patterns of word in corpus using hierarchical probabilistic models. It is called Topic Models. Finding of patterns often reflect the underlying topics that are make together to form the documents such as hierarchical probabilistic model simply generalized to other type of data; Topic Models have been used to analyze other things like Images, Biological data and Survey Information of data rather than words[1].

The main advantage of topic modeling to discover patterns of word-use and how to connect documents that set similar patterns. The purpose of Topic Models is that term which can be working with documents and this documents are made up of topics where topic is probability distribution over words. In other words, topic model is a generativemodel for documents in specifies a simple probabilistic steps by which documents can be generated.

Make a new document by selecting distribution over topics than each term in that document could select a topic at any order depends on the distribution. After that a word from that topic.[2]

On the side of text analysis and text mining, Topic Models depends on the bunch of words assumption which is ignoring data from the ordering of words according to seungil and Stephen, 2010, each document in a given corpus is thus represented by a histogram containing the occurrence of words. The histogram is modeled by a distribution over a certain number of topics, each of which is a distribution over words in the vocabulary. By learning the distribution, a corresponding low – rank representation of the high – dimensional histogram can be obtain for each document [3] the various kind of topic models, Such as Latent Semantic Analysis(LSA), Probability latent semantic analysis(PLSA), Latent Dirichlet Allocation(LDA), Corelated Topic Model(CTM) have a successfully improved classification in the area of discovering topic modeling.[3]

As time passes, topics in a document corpus evolve Modeling topics without considering time will confirm topic discovery. Modeling topics by considering time is called topic evaluation modeling. Topic evaluation modeling can describe important hidden information in the corpus allowing identifying topics with appearance of time and marking their evaluation with time.

There are many of areas that may use topic evaluation models a markable example would be like a researcher want to choose research topic in a specific field, and would like to know how this topic has been evolved over time, and try

to identify those data that describe the topic in the second category, paper will review many important topic models.

These two categories have a better high level view of topic modeling there are many waste to best understanding the concept of topic modeling it will discuss inside each category. For ie. The four methods that topic modeling depends on are Latent Semantic Analysis(LSA), Probabilistic Latent Semantic Analysis(PLSA), Correlated Topic Model (CTM), Latent Dirichlet Allocation(LDA). Each of this methods will have general overview, the importance of these methods an example that can describe the general idea of using this method. On the other and, paper will mention the areas the topic modeling evolution provide such as Topic Over Time(TOT), multiscale topic tomography, Dynamic Topic Models(DTM), dynamic topic correlation detection, detecting topic evaluation in scientific literature and the web of topics. Furthermore, it will going to current overview of each category and provide example, if any, and some limitation and characteristic of each part.

Why is it require?

It is used to identify customer reviews by detecting recurring words and patterns. For example:

"*This mobile have superb camera and battery life.*"

From above review topic model will identify words like *'mobile', 'camera', 'battery'* and *'life'*. Then topic modelling can group this review with other review which speak about same similar things.

For example, if any company managing thousands of customer per day then topic modelling will make cluster of same thing happen in a day. Topic modelling will look for pattern and make cluster of comparable things.

In short, topic modelling algorithms churn out collections of expression and words that it thinks are related and determine what these relations mean, while topic classification will delivers topics with labels such as *'Features'* and lots of others.

The methods of topic modelling

In this section, some of the topic modeling methods will be discussed that deals with words, documents and topics. In addition, the general idea of each of these methods, and present some example for these methods.

- Latent Semantic Analysis
- Probabilistic Latent Semantic Analysis
- Latent Dirichlet Allocation(LDA)
- Correlated Topic Model

Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation model is reason for improving the way of mixture models that capture the exchange ability of words and documents both from the old way of PLSA and LDA[4].

In the data mining community there are several challenges to researches posed In the recent past as there are a lots of electronic document collection such as web, interesting blogs, news articles[5].

Latent Dirichlet Allocation (LDA) in an algorithm for text mining that is based on statistical topic models and it is very widely used. LDA is a generative model that tries to mimic what the writing process is. So it tries to generate document on given topic. There are lots of LDA models including: temporal text mining, author topic analysis, supervised topic models, Latent Dirichlet co-clustering and LDA based topic informatics[6].

In simple words, the idea being each document is a modelled mixture of topics and each topic is mixture of words with discrete probability of a particular word to appear In given topic. These probabilities provide a particular representation of a document. Here consider the document as a “bag of words” with no other words besides the topic.

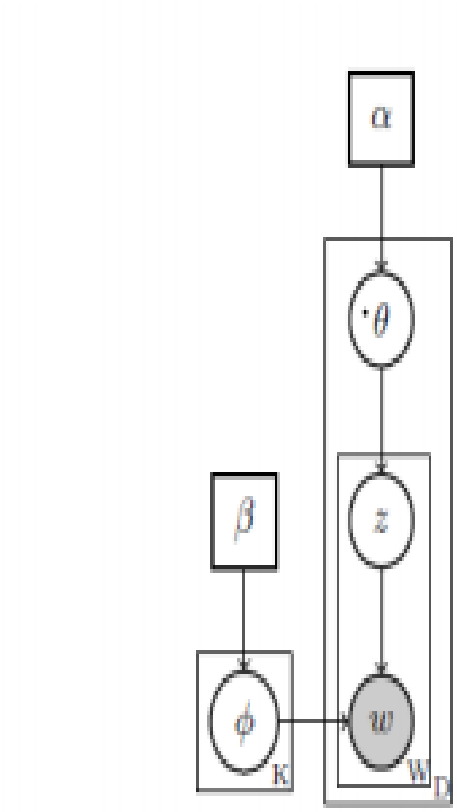
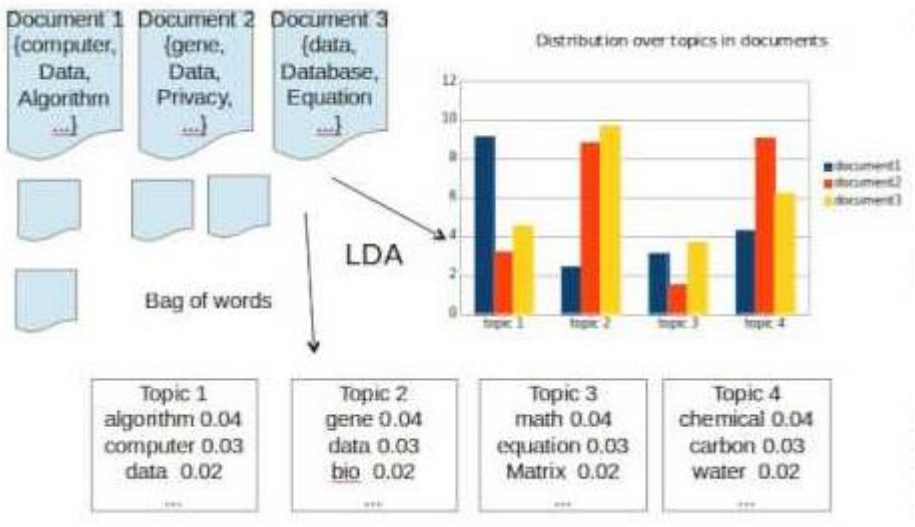


Figure shows the graphical representation of LDA models.

Latent Dirichlet Allocation (LDA) is relies on the assumptions: the distributional hypothesis and also the statistical mixture hypothesis that an organization may be determined. The aim of LDA is relate each document in our corpus to a group of topics which covers a decent amount of the words within the given set of text.



Example of LDA:

This is to provide example of use of LDA topic model on real data. By using the subset of TREC AP corpus containing 16000 documents. First remove the stop words in TREC AP corpus before running topic modeling. After that, use Em algorithm for find Dirichlet and conditional multinomial parameters for a 100 topic LDA model. The top words from some of the resulting multinomial distribution are illustrated below.

"ARTS"	"BUDGET"	"CHILDREN"	"EDUCATION"
New	Million	Children	School
Film	Program	Women	Students
Show	Tax	People	Schools
Music	Budget	Child	Education
Movie	Billion	Years	Teachers
Play	Federal	Families	High
Musical	Year	Work	Public
Best	Spending	Parent	Teacher
Actor	New	Says	Bennett
First	State	Family	Manigat
York	Plan	Welfare	Namphy
Opera	Money	Men	State
Theater	Programs	Percent	President
Actress	Government	Care	Elementary
Love	Congress	Life	Haiti

Data Pre-processing

LDA perform following steps:

1. **Tokenization:** Split the text into sentences and also the sentences into words. All words change to lowercase word and take away punctuation.
2. Words that have fewer than 3 characters are removed.
3. All **stop words** are removed.
4. **Words are lemmatized-** words in third person change into first person and words in future and past tenses are become tense.

5. **Words are stemmed-** words are reduced to their root form.

➢ Tokenization

The general meaning of tokenization is to spread text into minimal meaningful units. It's mandatory step before we start topic modelling. This step will basically split your whole document into sentences, symbols or other meaningful elements called tokens. The token is also words or number or punctuation. Tokenization does this task by locating word boundaries. Ending point of a word and beginning of the subsequent word is named word boundaries.

How tokenization works?

Tokenization have following functions for splitting sentences.

Name	Functions
Tokens	Tokenize a set of texts
Tokens_compound	Convert token sequences into compound tokens
Tokens_lookup	Apply a dictionary to a tokens object
Tokens_ngrams	Create ngrams and skipgrams from tokens
Tokens_remove	Select or remove tokens from a tokens object
Tokens_select	Select or remove tokens from a tokens object
Tokens_skipgrams	Create ngrams and skipgrams from tokens
Tokens_tolower	Convert the case of tokens
Tokens_toupper	Convert the case of tokens

Tokens_wordstem	Stem the terms in an object

➢ Stop words

Stop words are those words that are frequently utilized in sentences. Those words are removed after first two steps. They're not useful in find the context of a sentence. Those words make sentences lengthy and that they're not necessary in model. In python you've got predefine stop words list. For removing words we've library in python.

You can remove stop words by using Python library like

- **Stop Words with NLTK**
- **Stop Words with Gensim**
- **Stop Words with SpaCy**

Python's NLTK Library

The NLTK is one among the oldest and widely used Python library for removing stop words. You'll find list of stop words within the **corpus** module. To remove stop words from document, first of all divide document into sentences and then sentences to words and take away words if they're in list of corpus module.

Let's see example

```
from nltk.corpus import stopwords
nltk.download('stopwords')
from nltk.tokenize import word_tokenize

text = "Nick likes to play football, however he is not too fond of tennis."
text_tokens = word_tokenize(text)

tokens_without_sw = [word for word in text_tokens if not word in stopwords.words()]

print(tokens_without_sw)
```

In the above example, we import the stopwords collection with the help of nltk.corpus module. After that we import word_tokenize() method of nltk.tokenize class. Then we create text variable which contain a simple sentence. Then sentence in text variable is tokenized using word_tokenize() method. After that we will check through all words in the text_tokens list and check if words exists in stop words list or not. And if word is not exists in the stopword list, then it is returned and append to the token_without_sw collection. Then token_without_sw will be printed. The output will be

```
['Nick', 'likes', 'play', 'football', ',', 'however', 'fond', 'tennis', '.']
```

Adding or Removing Stop Words in NLTK's Stop Word collection

You can add or remove stop words as per your requirements to the predefined list or collection of stop words in NLTK. You can see list of stop words by following command.

print(stopwords.words(‘’english))

Output will be

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
```

- **Adding words to stop words**
  To add word in existing list of words use **append()**method.
  For example,
  all_stopwords.append(‘DWDM’)
  This will append DWDM word in existing stop words.

- **Removing words from stop words**
  To remove words from existing list of words, use **remove()** method.
  For example,
  all_stopwords.remove(‘DWDM’)
  This method will remove DWDM word from existing stop words list.

Python’s Gensim Library

The Gensim library is one among the widely used library for removing stop words from document. You have module genism.parsing.preprocessing for using genism library. From this library, remove_stopwords() method will remove all stop words from document.

Adding or Removing Stop Words in Gensim Stop Words Collection

You can add or remove stop words as per your requirements to the predefined list or collection of stop words in Gensim library. You can see list of stop words by following command.

Print(genism.parsing.preprocessing.STOPWORDS)

Output will be

```
frozenset({'her', 'during', 'among', 'thereafter', 'only', 'hers', 'in', 'none', 'with', 'un', 'put', 'hence',
'each', 'would', 'have', 'to', 'itself', 'that', 'seeming', 'hereupon', 'someone', 'eight', 'she', 'forty', 'muc
h', 'throughout', 'less', 'was', 'interest', 'elsewhere', 'already', 'whatever', 'or', 'seem', 'fire', 'howeve
r', 'keep', 'detail', 'both', 'yourselves', 'indeed', 'enough', 'too', 'us', 'wherein', 'himself', 'behind', 'ev
erything', 'part', 'made', 'thereupon', 'for', 'nor', 'before', 'front', 'sincere', 'really', 'than', 'alone',
'doing', 'amongst', 'across', 'him', 'another', 'some', 'whoever', 'four', 'other', 'latterly', 'off', 'sometim
e', 'above', 'often', 'herein', 'am', 'whereby', 'although', 'who', 'should', 'amount', 'anyway', 'else', 'upo
n', 'this', 'when', 'we', 'few', 'anywhere', 'will', 'though', 'being', 'fill', 'used', 'full', 'thru', 'call',
'whereafter', 'various', 'has', 'same', 'former', 'whereas', 'what', 'had', 'mostly', 'onto', 'go', 'could', 'yo
urself', 'meanwhile', 'beyond', 'beside', 'ours', 'side', 'our', 'five', 'nobody', 'herself', 'is', 'ever', 'the
y', 'here', 'eleven', 'fifty', 'therefore', 'nothing', 'not', 'mill', 'without', 'whence', 'get', 'whither', 'th
en', 'no', 'own', 'many', 'anything', 'etc', 'make', 'from', 'against', 'ltd', 'next', 'afterwards', 'unless',
'while', 'thin', 'beforehand', 'by', 'amoungst', 'you', 'third', 'as', 'those', 'done', 'becoming', 'say', 'eith
er', 'doesn', 'twenty', 'his', 'yet', 'latter', 'somehow', 'are', 'these', 'mine', 'under', 'take', 'whose', 'ot
hers', 'over', 'perhaps', 'thence', 'does', 'where', 'two', 'always', 'your', 'wherever', 'became', 'which', 'ab
out', 'but', 'towards', 'still', 'rather', 'quite', 'whether', 'somewhere', 'might', 'do', 'bottom', 'until', 'k
m', 'yours', 'serious', 'find', 'please', 'hasnt', 'otherwise', 'six', 'toward', 'sometimes', 'of', 'fifteen',
'eg', 'just', 'a', 'me', 'describe', 'why', 'an', 'and', 'may', 'within', 'kg', 'con', 're', 'nevertheless', 'th
rough', 'very', 'anyhow', 'down', 'nowhere', 'now', 'it', 'cant', 'de', 'move', 'hereby', 'how', 'found', 'who
m', 'were', 'together', 'again', 'moreover', 'first', 'never', 'below', 'between', 'computer', 'ten', 'into', 's
ee', 'everywhere', 'there', 'neither', 'every', 'couldnt', 'up', 'several', 'the', 'i', 'becomes', 'don', 'ie',
'been', 'whereupon', 'seemed', 'most', 'noone', 'whole', 'must', 'cannot', 'per', 'my', 'thereby', 'so', 'he',
'name', 'co', 'its', 'everyone', 'if', 'become', 'thick', 'thus', 'regarding', 'didn', 'give', 'all', 'show', 'a
ny', 'using', 'on', 'further', 'around', 'back', 'least', 'since', 'anyone', 'once', 'can', 'bill', 'hereafter',
'be', 'seems', 'their', 'myself', 'nine', 'also', 'system', 'at', 'more', 'out', 'twelve', 'therein', 'almost',
'except', 'last', 'did', 'something', 'besides', 'via', 'whenever', 'formerly', 'cry', 'one', 'hundred', 'sixt
y', 'after', 'well', 'them', 'namely', 'empty', 'three', 'even', 'along', 'because', 'ourselves', 'such', 'top',
'due', 'inc', 'themselves'})
```

- **Adding words to stop words**
  To add words in list apply **union()** method of genism.prarsing.preprocessong.
  For example,
  all_stopwords_gensim= STOPWORDS.union(set([‘DWDM’]))

- **Removing words from stop words**
  To remove words from existing list of words apply **difference()** method.
  For example,
  all_stopwords_gensim= STOPWORDS.difference(set([‘DWDM’]))

Python’s SpaCy Library

The Gensim library is one among of the widely used library for removing stop words from document. You have module en_core_web_sm for using genism library. For removing words you have Default.stop_words model in library.

Adding or Removing Stop Words in SpaCy Stop Words Collection

To add or remove stop words as per your requirements to the predefined list or collection of stop words in Gensim library. You can see list of stop words by following command.

print(all_stopwords)

Output will be

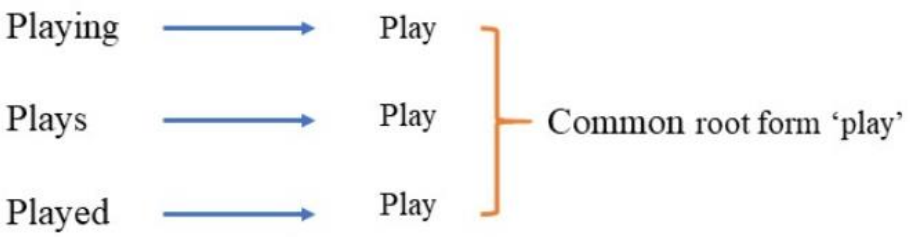
```
326
{'whence', 'here', 'show', 'were', 'why', 'n’t', 'the', 'whereupon', 'not', 'more', 'how', 'eight', 'indeed',
'i', 'only', 'via', 'nine', 're', 'themselves', 'almost', 'to', 'already', 'front', 'least', 'becomes', 'thereb
y', 'doing', 'her', 'together', 'be', 'often', 'then', 'quite', 'less', 'many', 'they', 'ourselves', 'take', 'it
s', 'yours', 'each', 'would', 'may', 'namely', 'do', 'whose', 'whether', 'side', 'both', 'what', 'between', 'tow
ard', 'our', 'whereby', "'m", 'formerly', 'myself', 'had', 'really', 'call', 'keep', "'re", 'hereupon', 'can',
'their', 'eleven', '’m', 'even', 'around', 'twenty', 'mostly', 'did', 'at', 'an', 'seems', 'serious', 'against',
"n't", 'except', 'has', 'five', 'he', 'last', '‘ve', 'because', 'we', 'himself', 'yet', 'something', 'somehow',
'‘m', 'towards', 'his', 'six', 'anywhere', 'us', '‘d', 'thru', 'thus', 'which', 'everything', 'become', 'herei
n', 'one', 'in', 'although', 'sometime', 'give', 'cannot', 'besides', 'across', 'noone', 'ever', 'that', 'over',
'among', 'during', 'however', 'when', 'sometimes', 'still', 'seemed', 'get', "'ve", 'him', 'with', 'part', 'beyo
nd', 'everyone', 'same', 'this', 'latterly', 'no', 'regarding', 'elsewhere', 'others', 'moreover', 'else', 'bac
k', 'alone', 'somewhere', 'are', 'will', 'beforehand', 'ten', 'very', 'most', 'three', 'former', '’re', 'otherwi
se', 'several', 'also', 'whatever', 'am', 'becoming', 'beside', '’s', 'nothing', 'some', 'since', 'thence', 'any
way', 'out', 'up', 'well', 'it', 'various', 'four', 'top', '‘s', 'than', 'under', 'might', 'could', 'by', 'too',
'and', 'whom', '‘ll', 'say', 'therefore', "'s", 'other', 'throughout', 'became', 'your', 'put', 'per', "'ll", 'f
ifteen', 'must', 'before', 'whenever', 'anyone', 'without', 'does', 'was', 'where', 'thereafter', "'d", 'anothe
r', 'yourselves', 'n‘t', 'see', 'go', 'wherever', 'just', 'seeming', 'hence', 'full', 'whereafter', 'bottom', 'w
hole', 'own', 'empty', 'due', 'behind', 'while', 'onto', 'wherein', 'off', 'again', 'a', 'two', 'above', 'therei
n', 'sixty', 'those', 'whereas', 'using', 'latter', 'used', 'my', 'herself', 'hers', 'or', 'neither', 'forty',
'thereupon', 'now', 'after', 'yourself', 'whither', 'rather', 'once', 'from', 'until', 'anything', 'few', 'int
o', 'such', 'being', 'make', 'mine', 'please', 'along', 'hundred', 'should', 'below', 'third', 'unless', 'upon',
'perhaps', 'ours', 'but', 'never', 'whoever', 'fifty', 'any', 'all', 'nobody', 'there', 'have', 'anyhow', 'of',
'seem', 'down', 'is', 'every', '’ll', 'much', 'none', 'further', 'me', 'who', 'nevertheless', 'about', 'everywhe
re', 'name', 'enough', '’d', 'next', 'meanwhile', 'though', 'through', 'on', 'first', 'been', 'hereby', 'if', 'm
ove', 'so', 'either', 'amongst', 'for', 'twelve', 'nor', 'she', 'always', 'these', 'as', '’ve', 'amount', '‘re',
'someone', 'afterwards', 'you', 'nowhere', 'itself', 'done', 'hereafter', 'within', 'made', 'ca', 'them'}
```

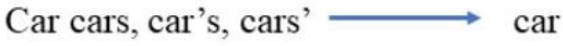
- **Adding words to stop words**
  To add words in list apply **add()** method of sp.Defaults.stop_words.
  For example,
  all_stopwords = sp.Defaults.stop_words
  all_stopwords.add(“DWDM”)

- **Removing words from stop words**
  To remove words from existing list of words apply **remove()** method.
  For example,
  all_stopwords = sp.Defaults.stop_words
  all_stopwords.remove(“DWDM”)

➢ Word lemmatized

In this step past and future tense will be change into present tense and third person change to first person.





Using above mapping a sentence could be normalized as follows:

➢ Word stemmed

in this step words are change to its root form.

original_word	stemmed_words
connect	connect
connected	connect
connection	connect
connections	connect
connects	connect

➢ APIs for Topic Modeling

- Google Prediction API
- AlchemyAPI
- DatumBox
- Microsoft Azure Machine Learning
- BIgML

➢ Industrial applications of Topic Modeling

- Living Lab
- Bio-informatics
- Text categorization
- Chatbot
- Sentiment
- IOT

- Summarization
  - Opinion summarization
  - Meeting summarization

Conclusion

In this paper we have well defined model like Latent Dirichlet Allocation topic model. The main conclusion is that LDA topic modeling applied to poetry does not extract clear and defined themes as it does when applied to scientific texts. Topic modeling extracts LDA-topics according to the contextual use of words. In this paper we also include deep information about Tokenizer.

References

[1] Blei, D.M., and Lafferty. J.D. “Dynamic Topic Models”, *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, PA, 2006.

[2] Steyvers. M., and Griffiths, T. (2007). “Probabilistic Topic Models” In T. Landauer, D McNamara, S. Dennis, and W. Kintsch (eds) *Latent Semantic Analysis*: A Road to Meaning. Laurence Erlaum

[3] Hofmann, T., “Unsupervised learning by probabilistic latent semantic analysis”, *Machine Learning*, 42 (1), 2001, 177-196.

[4] Blei, D.M., Ng, A.Y., and Jordan, M.I., “Latent Dirichlet Allocation”, *Journal of Machine Learning Research*, 3, 2003, 993-1022.

[5] Ahmed, A., Xing, E.P., and William W. “Joint Latent Topic Models for Text and Citations”. ACM New York, NY, USA, 2008.

[6] Zhi-Yong Shen, Z.Y., Sun, J., and Yi-Dong Shen, Y.D., “Collective Latent Dirichlet Allocation”, Eighth IEEE International Conference on Data Mining, pages 1019-1025, 2008.