**Auto extraction of descriptive metadata-**

**A generative statistical model with covariance scoring**

By

**Deepti Chevvuri**

A creative component submitted to the graduate faculty

in partial fulfillment of the requirements for degree of

MASTER OF SCIENCE

Major: Information Systems

Minor: Computer Science

Program of Study Committee:

Dr. Sree Nilakanta, Major Professor, Dept. of Supply Chain and Information Systems

Dr. Kevin Scheibe, Dept. of Supply Chain and Information Systems

Dr. Pavan Aduri, Dept. of Computer Science

Iowa State University

Ames, Iowa

2017

**Table of Contents**

[ACKNOWLEDGMENTS 4](#_Toc486430665)

[PROJECT BACKGROUD 5](#_Toc486430666)

[1.1 Introduction 5](#_Toc486430667)

[1.2 Research Objective 6](#_Toc486430668)

[1.3 Previous Study 7](#_Toc486430669)

[DATA AND RESEARCH BACKGROUND 9](#_Toc486430670)

[2.1 Scope of the Project 9](#_Toc486430671)

[2.2 Background 9](#_Toc486430672)

[2.3 Data Source 10](#_Toc486430673)

[DATA PREPARATION 11](#_Toc486430674)

[3.1 Data Format Prior to Cleansing and Processing 11](#_Toc486430675)

[3.2 Data Processing 12](#_Toc486430676)

[3.3 Data Format After Processing 12](#_Toc486430677)

[PROCESS FLOW 13](#_Toc486430678)

[4.1 Overview 13](#_Toc486430679)

[4.2 Process Steps 15](#_Toc486430680)

[4.2.1 Topic Word Creation of Representative Data Set 15](#_Toc486430681)

[4.2.1.1 Data Formatting 16](#_Toc486430682)

[4.2.1.2 LDA Model 16](#_Toc486430683)

[4.2.2 Topic Word Creation of Unclassified Text 17](#_Toc486430684)

[4.2.3 Scoring Algorithm and Clustering 17](#_Toc486430685)

[4.2.4 Summarizing 18](#_Toc486430686)

[4.2.5 Customizable Descriptive Metadata 20](#_Toc486430687)

[EXPERIMENT 22](#_Toc486430688)

[5.1 Experimental Dataset 22](#_Toc486430689)

[5.2 Stop Words 22](#_Toc486430690)

[5.3 Experiment 22](#_Toc486430691)

[5.4 Experimental Results 23](#_Toc486430692)

[5.4.1 Computing the Confusion Matrix 23](#_Toc486430693)

[5.4.2 Computing Precision and Recall 24](#_Toc486430694)

[5.4.3 Computing Accuracy 25](#_Toc486430695)

[CONCLUSIONS 28](#_Toc486430696)

[RECOMMENDATIONS 29](#_Toc486430697)

[REFERENCES 30](#_Toc486430698)

[APPENDIX 31](#_Toc486430699)

**Table of Figures**

Figure 1: Learning Process of LDA 9

Figure 2 Article format before processing 10

Figure 3: Cleansed and processed article. 11

Figure 4: Process Overview 13

Figure 5: Topic Words creation of Representative dataset. 14

Figure 6: Scoring Process 17

Figure 7: Summarization Technique Employed 18

Figure 8: Processing of Customizable Descriptive Metadata 19

Figure 9: Confusion Matrix 23

Figure 10: Precision and Recall for each category 24

Figure 11: Accuracy vs Category 25

Figure 12: Topic Words associated with each category 25

Figure 13: Summary of each article along with the frequency of topic words used to draw the summary 26

# **ACKNOWLEDGMENTS**

I would like to thank my committee chair, Dr Sree Nilakanta and my committee members, Dr. Kevin Scheibe and Dr. Pavan Aduri for their guidance and support throughout the course of this creative component.

In addition, I would also like to thank my family, friends, colleagues, the department faculty and staff for making my time at Iowa State University a wonderful experience.

An ode

To parents, Amma and Nanagaru, for their untiring effort

To a brother, Srinivas, for his lifelong support

To a mentor, Stalin, who is no less than an expert

To friends, Laha and Manas, for offering me their relentless comfort

To a person I always try to be, a person better than the person I am today.

**CHAPTER 1**

# **PROJECT BACKGROUD**

# **Introduction**

Descriptive metadata is information that describes data, identifies data with respect to its Title, author, category, keywords, summary etc. The importance of descriptive document metadata has long been identified in the fields of bibliographic control and data management, within the disciplines of library and information science (LIS) and computer science [Dempsey & Heery 1998, Burnett, et al. 1999]. However, the rapid generation of data today along with lack of standards and mechanisms to tag the generated data with appropriate metadata has rendered most of such data ineffective, thus making data retrieval (search) and data utilization extremely difficult. This problem has crippled many data scientists as well, who are authors of many research publications who concluded “that researchers often fail to develop clear, well-annotated datasets to accompany their research (i.e., metadata), and may lose access and understanding of the original dataset over time” [ Carol Tenopir, Suzie Allard, Kimberly Douglass, Arsev Umur Aydinoglu, Lei Wu, Eleanor Read, Maribeth Manoff, Mike Frame, et al. 2011].

According to The International Journal of Digital Curation, Metadata is used to both improve accessibility and discoverability; and to control authentication procedures, creating audit trails to ensure that material cannot be accessed or altered by those not authorized to do so [Sarah Higgin, et al. 2011]. Metadata is an asset to big data as it improves its potential usability. Having no usability renders all data worthless.

# **Research Objective**

After acknowledging the potential of metadata, this study has been designed to focus on understanding the current tools and methodologies used to generate descriptive metadata, thereby, developing a new approach towards generating descriptive metadata. In the study the problem is viewed from a linguistic stand point, where unclustered textual data is processed using Natural Language Processing techniques, such as Latent Dirichlet Allocation (LDA), to create topic word clusters which are used to categorize text (articles) into various categories based on an indigenous scoring algorithm. Subsequently, automatic text summarization is applied on the set of clustered texts to draw descriptive metadata for each or a group of texts (articles). The rationale behind the approach is that unclustered text can be grouped through topic recognition and segmentation with an expectation that articles pertaining to a category have a determined set of topic words around which their text revolves and where the co-occurrence of multiple topic words act as a significant factor for related material. In order to quantify this a unique scoring algorithm has been defined which enables us to hierarchically score and arrange topic words according to their relevance. When compared to other information retrieval techniques, apart from considering the co-occurrence of topic words this method also considers the relevance of such co-occurring topic words.

# **Previous Study**

The academia particularly those associated with R&D have made lot of efforts to automate the generation of metadata. For instance, SeerSuite- an open source search engine and digital library has unique services which generate metadata for research documents. The metadata extraction is automated for Header Extraction, Citation extraction, Table extraction and Figure Extraction [Wu, et.al IAAI 2014; Teregowda, IC2E 2013]. This research however does not include steps to find descriptive metadata.

Another significant approach towards metadata, which unlike the above tools included methods for generating descriptive metadata is Goods, a project at Google which extracted metadata to understand data’s provenance as well as its similarity to other data [Halevy A., Korn F., Noy N., Polyzotis.N, Roy S., Whang S. et al 2016]. The study initially creates a record of frequent tokens from which the potential tokens are found using s HyperLogLog algorithm. Once the potential tokens are identified they grouped similar texts through the concept of Locality Sensitive Hashing.

Various other studies concentrate on developing better metadata standards and principles, such that the new data generated is associated with appropriate metadata from the stage of its creation, which could be used for content search and comparison along the future. For instance, a study was developed to evaluate a Semantic Web-based approach that integrates the BRIDG model with ISO 21090 data types that generated domain-specific templates that supported clinical study metadata standards development [Jiang et al. 2016].

Greenberg, J. (2015) developed the MetaDataCAPT’L, which uses various methods to quantify the cost and value of metadata, and views metadata as an asset containing contextual knowledge about the main content in the data. Greenberg’s paper helps in realizing the conceptual understanding and predictive value provided by metadata.

**CHAPTER 2**

# **DATA AND RESEARCH BACKGROUND**

# **2.1 Scope of the Project**

The study concentrates and confines to determine a new methodology to automate the process of drawing descriptive metadata for textual information. Though descriptive metadata can be defined for non-textual information like images, videos, audio clips etc. such is beyond the scope of this study.

In this study a supervised learning algorithm is used to group unclustered text, which identifies repeated or overlapping topic words between the representative dataset and the test data set. In this study importance is given to the weighted covariance of multiple topic words where the weightage of each topic word is defined according to its predominance in determining the category.

# **2.2 Background**

Latent Dirichlet Allocation (LDA), a method to determine underlying topics in a document is used in this study. LDA is a “Bag-of-Words” model which backtracks from textual content to the set of topic words that are likely to have generated it. Being a generative model (probabilistic model), it promises a set of parameters that define the data and which are significantly smaller in number when compared to the complete textual content. The learning process of LDA is illustrated in figure 1,

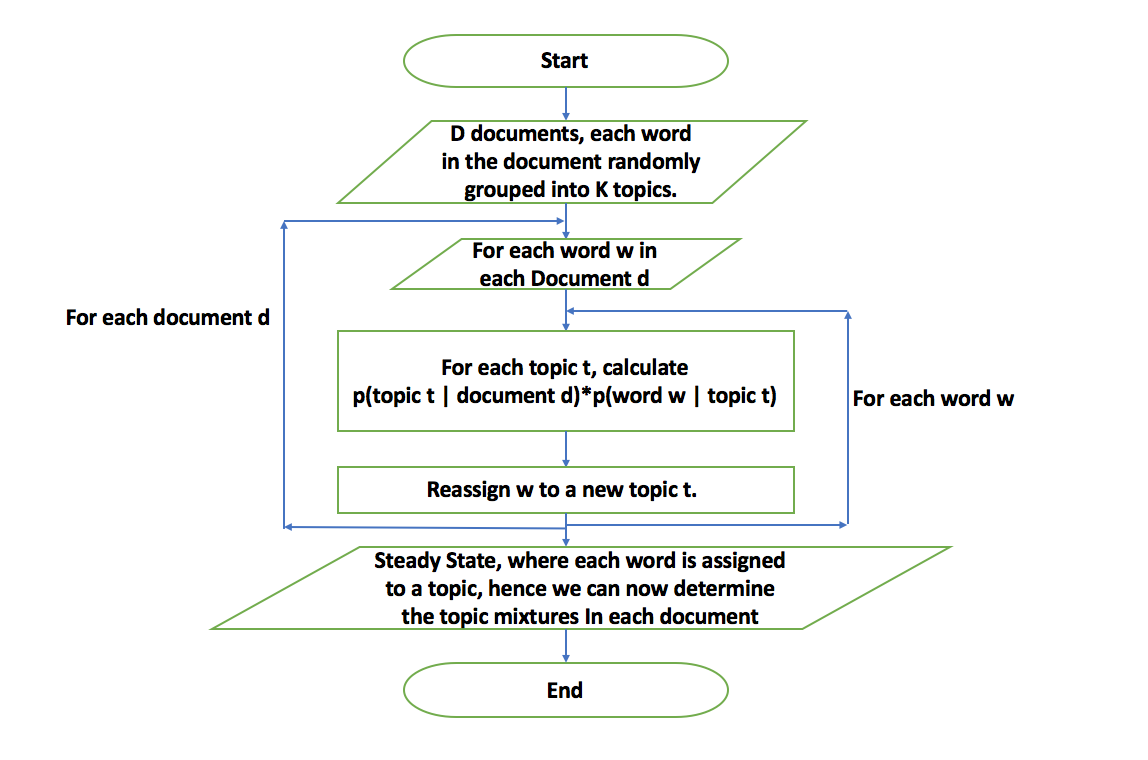


Figure 1: Learning Process of LDA

# **2.3 Data Source**

This tool builds metadata for unclassified textual datasets. A potential and easily available source for such data are news articles found in Lexis Nexis (provides research tools for academic and public libraries). Articles from The New York Times published between the dates of 01/01/2017 to 02/01/2017 provide the primary resource of the study. These articles broadly classify into four categories- Politics, Movies, Conservation, and Sports. Labeled training data of 473 documents was used in the learning process. Word count in each of these documents varies between 50 to 2000 words. Finally, the model was tested and evaluated on a test data set of 1840 unclassified documents from the same source.

**CHAPTER 3**

# **DATA PREPARATION**

# **3.1 Data Format Prior to Cleansing and Processing**

The document sets for each of the categories is downloaded. Each of the articles contain the body of the article along with other information such as byline, section (the section of the newspaper in which the article was printed), Length (word count of the body), document type, publication type, subject, scandals, persons (involved), country, Load-date etc., all of which are important with respect to the article’s provenance but are practically irrelevant for the current study. Also, the article’s body format seems to differ from article to article, a lack of consistency in the format hindered the processing of the files. The data format of one of the articles is shown in figure 2.

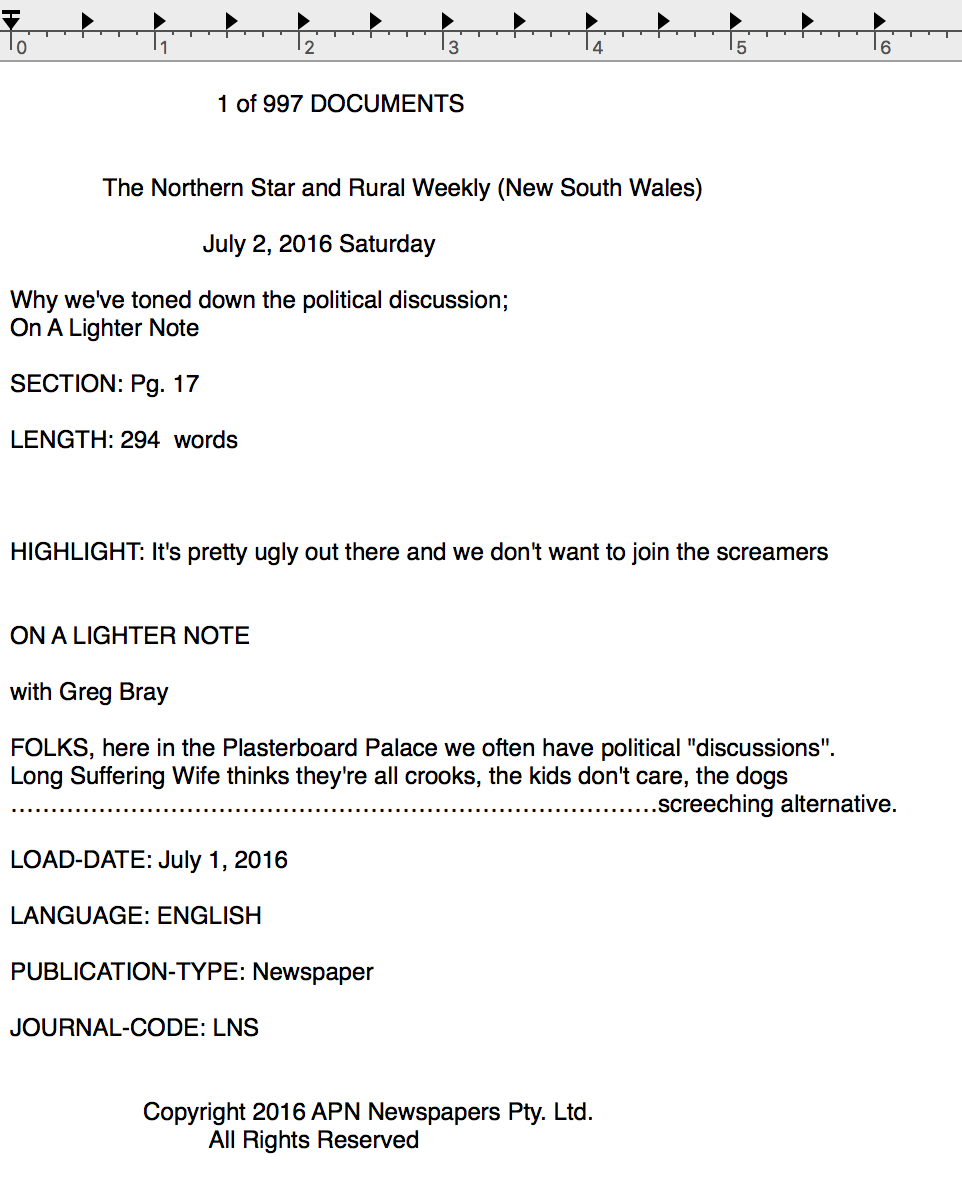


Figure 2 Article format before processing

# **3.2 Data Processing**

The first step in processing the files is to extract and remove irrelevant information from the articles. The removed data from each article is placed in a separate excel sheet for future use. The entire data set is processed and cleansed such that all the articles are in the same format and each article is converted into individual file its name containing a unique identification number. Also, the headline of the article is appended to the name of the article which helped in clear identification of the documents. A processed document is shown in figure 3.

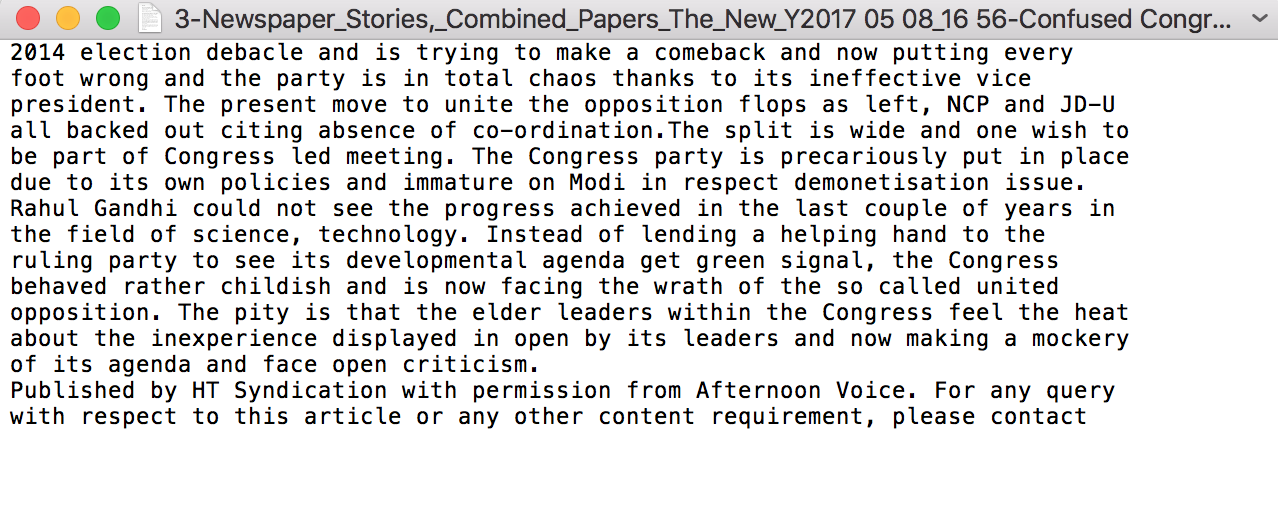
****

Figure 3: Cleansed and processed article.

# **3.3 Data Format After Processing**

The data set after processing contained folders of documents, each folder holding the documents classified and named as per their category, namely Conservation, Movies, Politics and Sports. Each file contained only the body of the article and name according to the format mentioned in section 3.2.

**CHAPTER 4**

# **PROCESS FLOW**

# **4.1 Overview**

The first step after processing the articles is to identify prominent topic words in each of the four categories, for which around 500 pre-classified articles are taken as training dataset. The processed training data set (representative data set) for each category is fed to the LDA which creates a list of 20 topic words associated with the category. The set of topic words of each category are essential for classifying unknown text(article) which has been explained in section 4.2.2 and 4.2.3. The limit for the number of topic words has been determined through a heuristic process, any number lesser or greater than which does not significantly increase the accuracy of the classification algorithm. The set of topic words found from the representative dataset act as the primary input while classifying an unknown textual content.

Each of the unclassified text(article) again goes through the “Topic Word Creation” process explained further in the paper, the topic words of the current document (unclassified article) are fed along with the topic words of each individual category (topic words of each category at a time) are fed to the scoring algorithm which scores the relevance of the new document to that category. The unclassified article is assigned to the category for which it gets the highest relevance score.

Once the new article is classified, the topic words of the representative data set are now used to find the summary of the article(s). The added advantage of using this approach is the importance given to the end users opinion while finding the summary, i.e. the user is given the ability to filter the summary text according to the topic words of his choice. This ensures that the relevance of the results returned depend on the topics of user’s interest.

This complete process is represented diagrammatically in figure 4, followed by a detailed explanation of each aspect of the process.

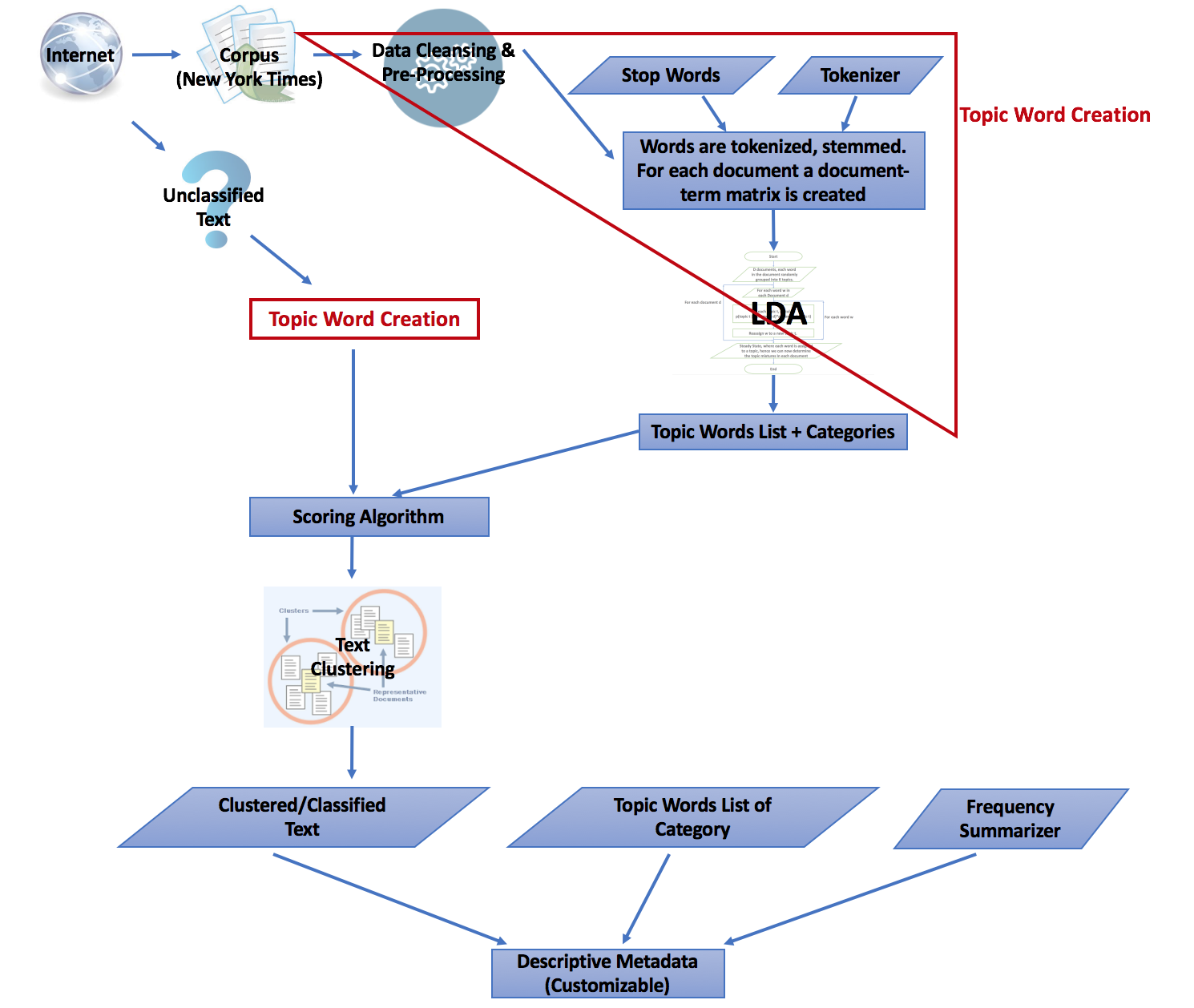
****

Figure 4: Process Overview

# **4.2 Process Steps**

The process steps for collecting, cleaning and processing the data have been covered in chapters before, the below steps are performed on the processed data sets.

## **4.2.1 Topic Word Creation of Representative Data Set**

By looking at Figure 4, we can understand that the input data set needs to be formatted before being fed to the LDA algorithm for Topic Word Creation. Therefor this step can be viewed and understood as two separate processes as shown in figure 5.

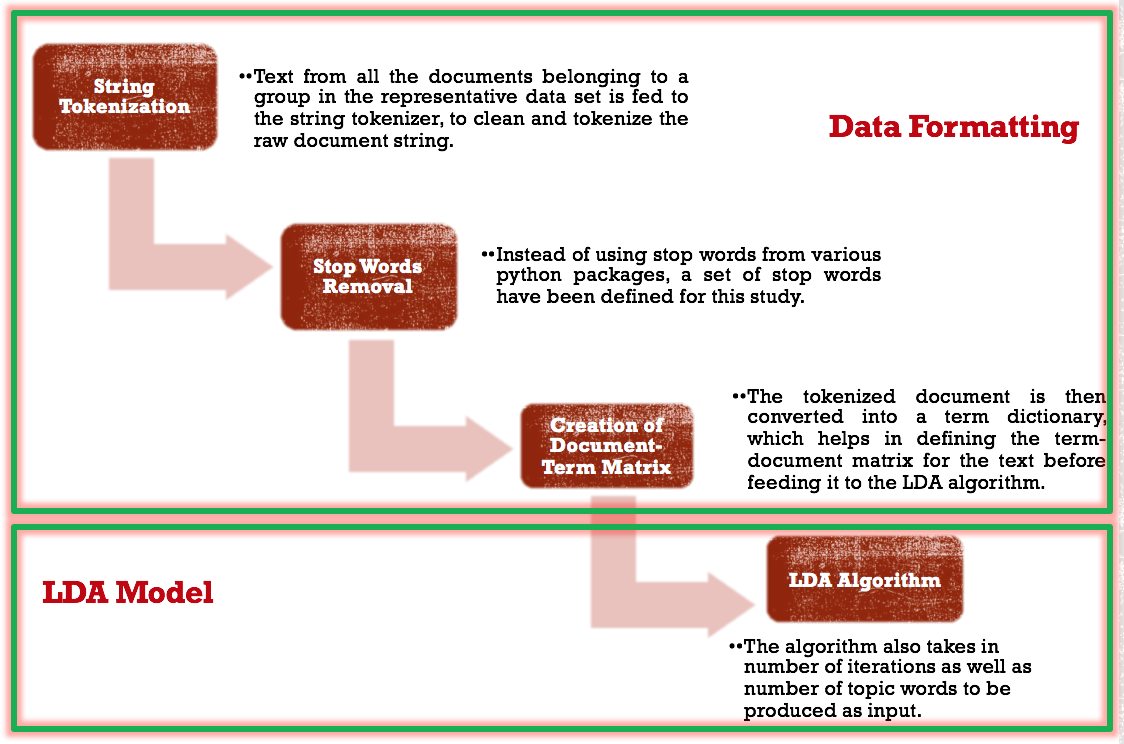


Figure 5: Topic Words creation of Representative dataset.

### **4.2.1.1 Data Formatting**

The inputs to this step are:

* processed representative dataset.
* The collection of stop words- instead of using the collection of stop words embedded in any of the python packages, a stop word collection has been separately created and used in this study. This is to enable the user to add or delete the stop words as per necessity. These stop words are listed in the appendix section of this document.
* A tokenizer, to clean and tokenize the raw document string. The tokenized document is the converted into a term dictionary, which helps in defining the term-document matrix for the text before feeding it to the LDA algorithm.

The representative dataset is a raw collection of words which are first tokenized and stemmed, from these stemmed words the stemmed stop words are removed. All the words which are more than two characters in length remaining in the tokenized word list are now considered to be a part of the term dictionary of the corpora. The term dictionary is now used to build the document-term matrix of the corpus which is fed to the LDA model.

### **4.2.1.2 LDA Model**

Gensim (“generate similar”) a software package for topic modelling is used to build the LDA model. The inputs to this step are:

* The corpus (document term matrix) created from the step above.
* The number of topic words to be output from the model. After a lot of heuristic this has been set to 20 for optimal performance of the model.
* The term dictionary- containing all stemmed words of more than two characters in length and not a stop word.
* The number of passes- the number of iterations the algorithm must make through the corpus before finalizing the set of topic words. Increase in this number results in increased computational cost but would also produce higher accuracy. An optimum value of 20 has been chosen to balance the importance of accuracy as well as the computational complexity. This is however a value that might change from dataset to dataset.

After the set number of iterations, LDA produces a set of 20 topic words for each category, further explanation of which is provided in the next step.

## **4.2.2 Topic Word Creation of Unclassified Text**

Each unclassified text article goes through the steps in 4.2.1, by the end of which each text is associated with 20 topic words each.

## **4.2.3 Scoring Algorithm and Clustering**

The inputs to this step are:

* All categories with their respective topic words list determined (form the representative data set) by the end of step 4.2.1.
* The topic words list determined for the unclassified text article by the end of step 4.2.2.

As mentioned before, the LDA algorithm results in 20 topic words for each category, say X1 to X20, which are arranged in the decreasing order of their relevance to their category, for example, if X1 to X20 are topic words of the category Politics, a document in which X1 is predominant has a higher probability to be grouped into the category of Politics than a document in which X20 is predominant.

The limit for number of topic words has been determined through a heuristic process, which changes form data set to data set. It depends on the number of categories as well as the number of documents of each category taken into picture. For the current study, 20 topic words per category have been determined, as it has been observed any number of topics words lesser or greater than this does not significantly increase the performance of the model.

For each topic word that is common between the topic words set of the representative dataset and the new unclassified text article the relevance score of the common topic word in both the sets if multiplied and cumulated. Figure 6 gives a better explanation of the same.

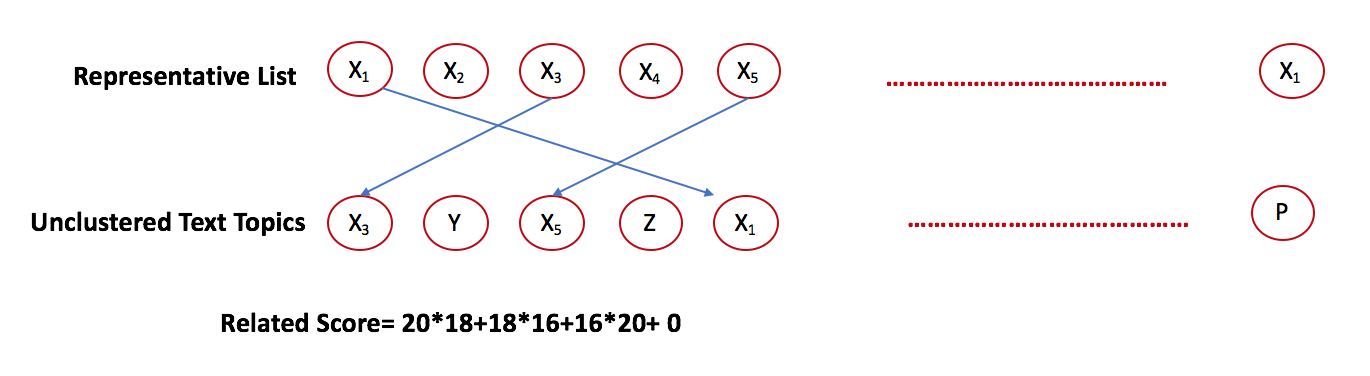


Figure 6: Scoring Process

From the scoring algorithm, we can identify that it gives importance to the co-occurrence of multiple “relevant” topic words, rather than just the co-occurrence of topic words. i.e. a document having X1 in moderate quantity is more relevant and has higher probability to be categorized correctly than a document with X5 in higher quantity. This is because X1 has is more relevant than X5.

## **Summarizing**

An extraction summarization technique has been implemented in this study to draw summary out of unclassified text or group of unclassified text. Extraction summarization works by selecting a subset of existing words, phrases or sentences in the original text to form the summary. There are various ways in which extractive summarization is employed, one of which is tweaked in this study.

In order to incorporate the relevance of the topic words associated with each category while drawing the summary, the topic words of each category are sent as an input to the summarization technique. The set of inputs for this method are:

* The clustered/ classified text which is classified in step 4.2.3.
* The set of topic words associated with the category into which the document is classified.
* Frequency summarizer

By the end of this step, the summarizer, with the help of the topic words determined by the scoring algorithm, selects a subset of words or phrases or sentences from the original text which serves as a summary, which is the Descriptive Metadata for the text. This process has been illustrated in figure 7.

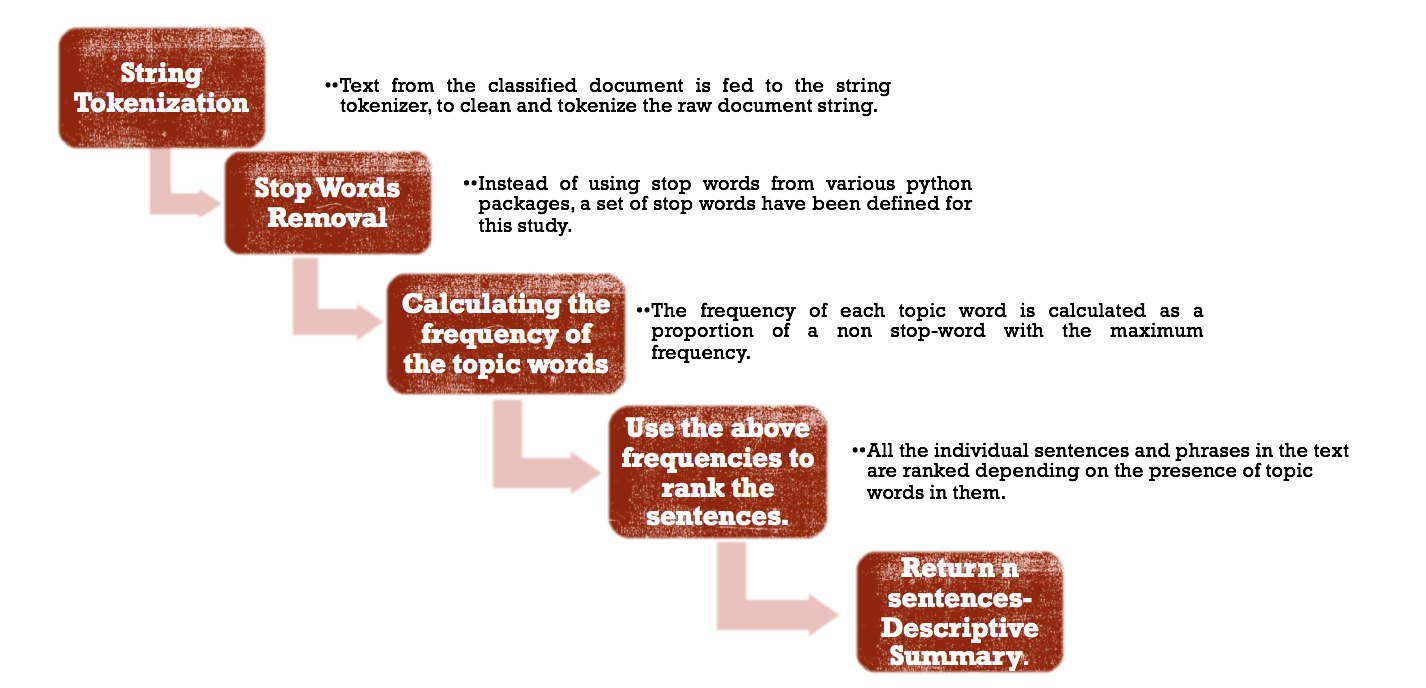


Figure 7: Summarization Technique Employed

## **4.2.5 Customizable Descriptive Metadata**

One of the added features of this study, is to tune the results according to the user’s interest, i.e. to provide a descriptive text summary which contains specific terms and eliminates nonessential terms. Such that the relevance of the results returned depends on the topic words of user’s choice, i.e. customizable descriptive metadata.

In order to incorporate user’s preference while processing the text for descriptive summary (which is descriptive metadata), step 4.2.4 is improvised with an additional input along with the other inputs:

* The clustered/ classified text which is classified in step 4.2.3.
* The set of topic words associated with the category into which the document is classified.
* Frequency summarizer
* The set of topic words of user’s preference

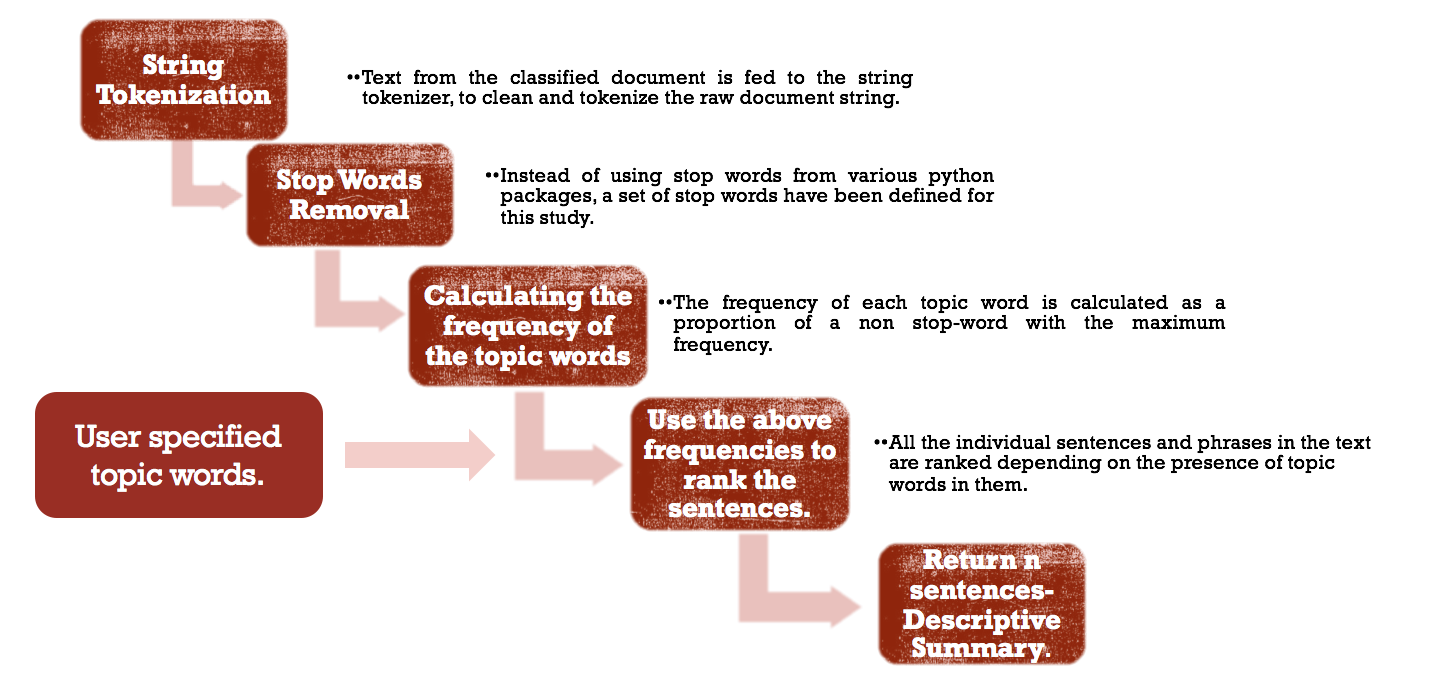


Figure 8: Processing of Customizable Descriptive Metadata

By the end of this step, the summarizer, with the help of the topic words determined by the scoring algorithm as well as the topic words suggested by the user, selects a subset of words or phrases or sentences from the original text which serves as a summary, which is the Descriptive Metadata for the text. This process has been illustrated in figure 8.

**CHAPTER 5**

# **EXPERIMENT**

# **5.1 Experimental Dataset**

As mentioned before this study builds a supervised learning model, for which the training data set consists of four different categories- Politics, Sports, Conservation and Movies. The training set consists of a total of 473 documents. The test data set however has varying number of documents for each of the categories. Conservation has around 476 documents, movies has 322 documents, Politics 496 and sports 546 documents. Both the training dataset and test dataset are New York Times’s articles drawn from Lexis Nexis. The word count in these documents vary between 50 to 2000 words.

# **5.2 Stop Words**

A set of total 429 words were included as stop words. The advantage of the study is that the stop words can be decided according to the user’s interest. This is to avoid jargons in industries to be considered as topic words, this is because the jargon might be repeated many a times in documents but would not be the required information a user is looking for. Thus, it helps in avoiding unnecessary bias towards specific topics.

# **5.3 Experiment**

As previously mentioned in the processes, the experiment was run for a total of 2313 documents 473 of which were the training dataset (Politics- 183, Conservation-81, Movies-55, Sports-154) and remaining were test dataset (Conservation- 476, Movies 322, Politics 496, Sports 546). It took a total duration of 105 mins to produce the classification result. And less that 2 minutes to produce the summary of a selected document or group of documents (which were classified correctly). While running the experiment multiple times multiple terms were tweaked to improve the overall performance of the model, they are:

* Number of passes through the text before outputting the topic words list.
* The number of topics words that could produce optimum performance.

# **5.4 Experimental Results**

A series of computations have been performed to analyze the performance of the study.

# **5.4.1 Computing the Confusion Matrix**

The output of the classification algorithm gives information like the original group, the classified group, the similarity score with which the document was classified and a Boolean value whether the document was classified correctly. With this information, a confusion matrix was built which is shown in figure 9.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Conservation (Actual) | Movies (Actual) | Politics (Actual) | Sports (Actual) |
| Conservation (Predicted) | 367 | 15 | 8 | 0 |
| Movies (Predicted) | 28 | 297 | 1 | 0 |
| Politics (Predicted) | 71 | 3 | 485 | 0 |
| Sports (Predicted) | 5 | 5 | 2 | 537 |
| Not Classified | 5 | 2 | 0 | 9 |

Figure 9: Confusion Matrix

The confusion matrix helps keep counters for the true positives, false positives and total number of instances for each class which are used in calculating precision and recall for each class.

# **5.4.2 Computing Precision and Recall**

Figure 9 gives all the values needed to compute precision and recall for each category. Note that the values in the diagonal would always be the true positives (TP). The false negative (FN) for each class would be the sum of all values in the corresponding row excluding (TP). The false positive (FP) for each class would be the sum of all values in the corresponding column excluding the main diagonal element (TP). TN for each class would be the sum of all the values of the confusion matrix excluding that class's row and column.

|  |  |  |
| --- | --- | --- |
| Class | Precision | Recall |
| Conservation | 0.94 | 0.77 |
| Movies | 0.91 | 0.92 |
| Politics | 0.87 | 0.98 |
| Sports | 0.98 | 0.98 |

Figure 10: Precision and Recall for each category

The precision for each category can be computed as

and, the recall for each category can be computed as

Figure 10 shows the precision and recall for each category.

# **5.4.3 Computing Accuracy**

Politics and Sports documents were classified with an accuracy greater than 90% but documents related to Movies and Conservation have an unusual high misclassification rate. On looking closely at the documents (human verification) it was identified that these topics were very sparse with topic words list which over lapped with the topic words list of Politics and Sports. A simple example to understand the scenario is that a movie can be about politics or can be about sports which makes the boundaries for the topic words of such category very wide and the term document matrix very sparse. This is the major contributing factor for a high misclassification rate of Movies and Conservation.

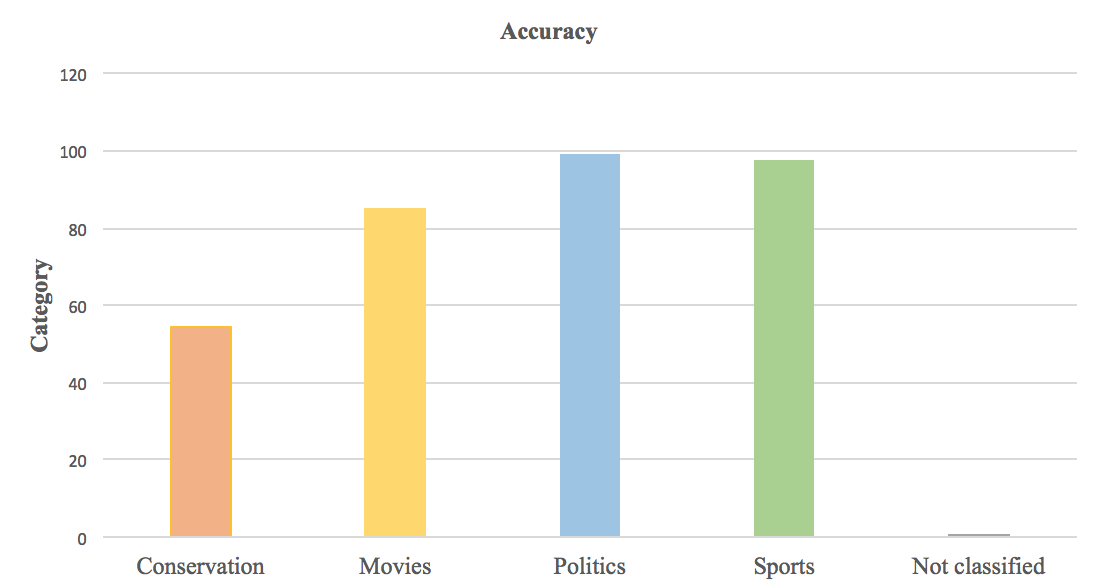


Figure 11: Accuracy vs Category

In figure 11, we also see a small percent of documents (less than 1 %) not classified, this is because the ten documents contributing to this have less than ten words in their document.

Figure 12, Figure 13 are screen shots of results. Figure 12 contains the list of topic words associated with each category. The topic words selected depend on the conglomerated text of all the articles belonging to a particular category.

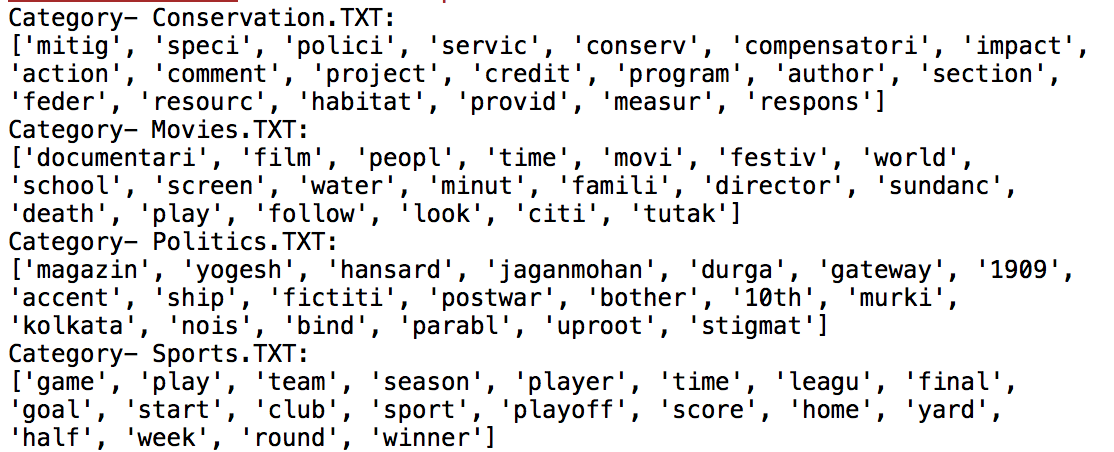


Figure 12: Topic Words associated with each category

Figure 13 contains the summary results for each document which show the summary of the text in the article along with the frequency of the topic words which have been used to pull the summary.

Here the results are limited to two sentences (starting with \*) out of all the sentences in the article, this is a customizable value. The words chosen to generate the summary can be customized by the user, currently only the topic words of the category into which the document is correctly classified are given relevance while drawing the summary. Any words of user’s interest can also be added to this list. An important point to be noted here is that the article or document must be categorized correctly for the summarization process, this is because the set of topic words used to summarize the article depends on the category into which the article is correctly classified.

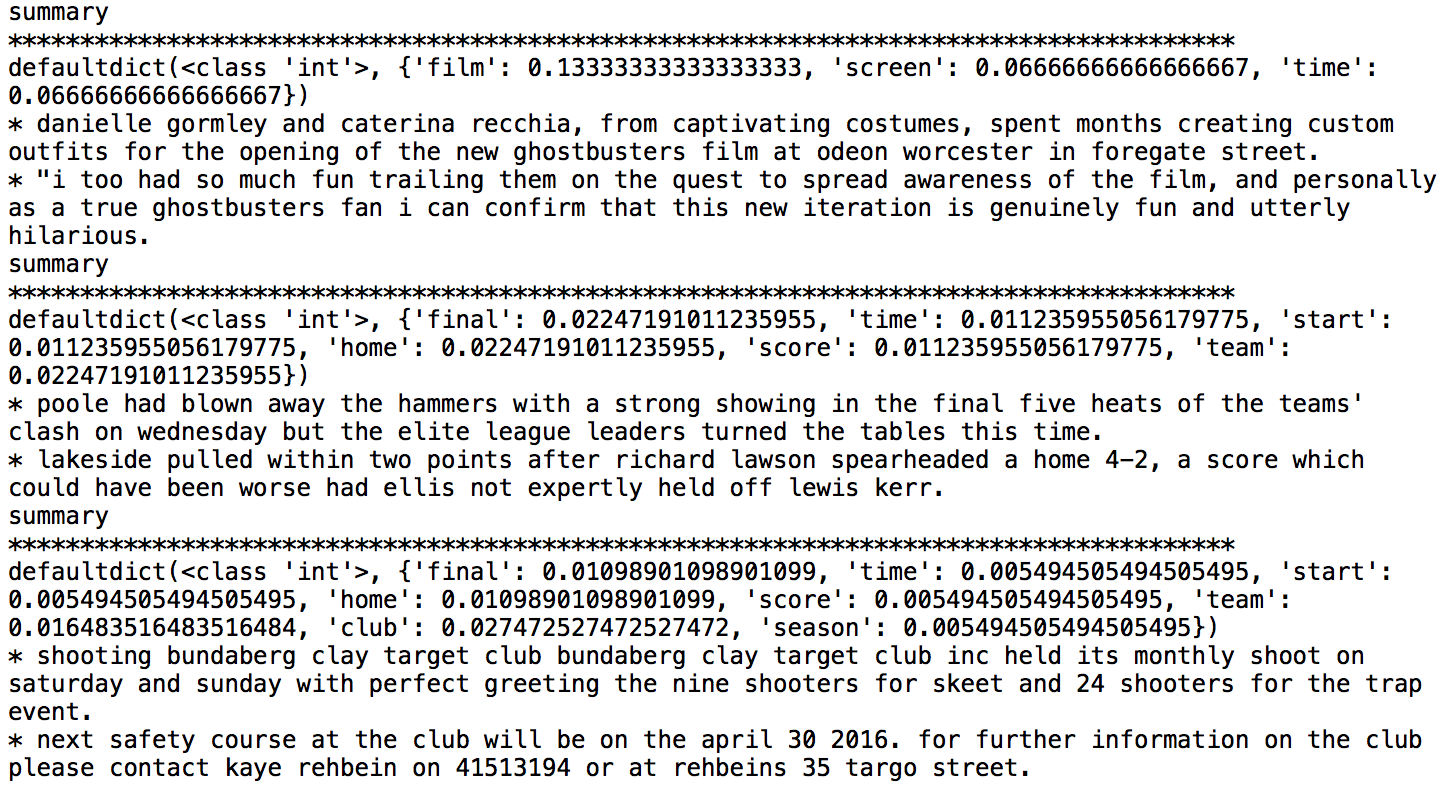


Figure 13: Summary of each article along with the frequency of topic words used to draw the summary

**CHAPTER 6**

# **CONCLUSIONS**

The primary idea behind this paper is that for documents belonging to a particular category or genre there are only a set of topic words around which the text revolves. If these topic words can be identified for text of known genre or category (representative text) any other unknown text can easily be identified and classified depending on the commonality between the topic words of the unknown text and the representative text. This can be true only if the supervised algorithm has learnt the topics of a category prior to being testes, i.e. for unknown text which doesn’t belong to any of the learnt categories cannot be correctly identified.

As an implementation of this idea, a NLP approach is designed where Latent Drichlet Allocation is used to identify the topic words of documents along with their relevancy index. The relevance index is used to calculate the similarity score between the representative dataset and test dataset. Once the category has been correctly identified, text summarization applied on the text aided by the topic words of the category helps build descriptive metadata for the text document.

The scoring algorithm uses an indigenous approach where the co-occurrence of multiple topic words is given more importance, then just the occurrence of topic words. Also, it has been identified that the probability of correct classification not just depends on the amount of similarity between the train and test data but it also depends on the category being classified. This is because categories like Movies are very sparse with overlapping content with other categories like Politics and Sports resulting in higher misclassification rates.

**CHAPTER 7**

# **RECOMMENDATIONS**

There are a few significant opportunities identified during the execution of this study which hold a lot of potential for future study.

* The current summarization technique is of extraction summarization type, which generally picks phrases or words from the text in the document depending on relevance, this approach can be changed to abstraction summarization type where the summary would be more close to a summary a human might interpret. This is an extremely active research area where this study can also be extended.
* A manual verification can be done to ascertain the classification of the unknown text, once classified correctly this can be fed back to the supervised algorithm which would further strengthen the algorithm by considering any additional topics which might be missing from the original representative dataset.

**CHAPTER 8**

# **REFERENCES**

1. Dempsey, L. and R. Heery (1998). *Metadata: A Current Review of Practice and Issues. Journal of Documentation*, 54 (2), 145-172.
2. Carol Tenopir , Suzie Allard, Kimberly Douglass, Arsev Umur Aydinoglu, Lei Wu, Eleanor Read, Maribeth Manoff, Mike Frame (June 29, 2011) *Data Sharing by Scientists: Practices and Perceptions*.
3. Sarah Higgins (2011) *Digital Curation: The Emergence of a New Discipline*.
4. C. Lee Giles (2014): *Scholarly Big Data: CiteSeerX Insights*
5. Wu, J.; Williams, K.; Chen, H.-H.; Khabsa, M.; Caragea, C.; Ororbia, A.; Jordan, D.; and Giles, C. L. 2014. Citeseerx: *Ai in a digital library search engine.* In IAAI 2014, co-located with AAAI 2014.
6. Caragea, C.; Wu, J.; Williams, K.; Gollapalli, S. D.; Khabsa, M.; Teregowda, P.; and Giles, C. L. 2014b. *Automatic identification of research articles from crawled documents.*
7. Alon Halevy, Flip Korn, Natalya F. Noy, Christopher Olston, Neoklis Polyzoti1, Sudip Roy, Steven Euijong Whang (2016*): Goods: Organizing Google’s Datasets*
8. Greenberg, J. (2015). *Metadata Capital: Conceptual Understanding, Predictive Value.*
9. Jiang G (2016), *Using Semantic Web technologies for the generation of domain-specific templates to support clinical study metadata standards.*

**CHAPTER 9**

# **APPENDIX**

**List of Stop Words:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a  about  above  across  after  again  against  all  almost  alone  along  already  also  although  always  among  an  and  another  any  everywhere  f  face  faces  fact  facts  far  felt  few  find  finds  first  for  four  from  full  fully  further  furthered  furthering  furthers  g  gave  mrs  much  must  my  myself  n  necessary  need  needed  needing  needs  never  new  new  newer  newest  next  no  nobody  non  noone  not  nothing  think  thinks  this  those  though  thought  thoughts  three  through  thus  to  today  together  too  took  toward  turn  turned  turning  turns  two  u  under | anybody  anyone  anything  anywhere  are  area  areas  around  as  ask  asked  asking  asks  at  away  b  back  backed  backing  backs  general  generally  get  gets  give  given  gives  go  going  good  goods  got  great  greater  greatest  group  grouped  grouping  groups  h  had  has  have  orders  other  others  our  out  over  p  part  parted  parting  parts  per  perhaps  place  places  point  pointed  pointing  points  possible  present  presented  presenting  until  up  upon  us  use  used  uses  v  very  w  want  wanted  wanting  wants  was  way  ways  we  well  wells  went  were  what | be  became  because  become  becomes  been  before  began  behind  being  beings  best  better  between  big  both  but  by  c  came  having  he  her  here  herself  high  high  high  higher  highest  him  himself  his  how  however  i  if  important  in  interest  interested  interesting  interests  presents  problem  problems  put  puts  q  quite  r  rather  really  right  right  room  rooms  s  said  same  saw  say  says  second  seconds  see  when  where  whether  which  while  who  whole  whose  why  will  with  within  without  work  worked  working  works  would  x  y  year  years  yet | can  cannot  case  cases  certain  certainly  clear  clearly  come  could  d  did  differ  different  differently  do  does  done  down  down  into  is  it  its  itself  j  just  k  keep  keeps  kind  knew  know  known  knows  l  large  largely  last  later  latest  least  less  seem  seemed  seeming  seems  sees  several  shall  she  should  show  showed  showing  shows  side  sides  since  small  smaller  smallest  so  some  somebody  someone  you  young  younger  youngest  your  yours  z | downed  downing  downs  during  e  each  early  either  end  ended  ending  ends  enough  even  evenly  ever  every  everybody  everyone  everything  let  lets  like  likely  long  longer  longest  m  made  make  making  man  many  may  me  member  members  men  might  more  most  mostly  mr  something  somewhere  state  states  still  still  such  sure  t  take  taken  than  that  the  their  them  then  there  therefore  these  they  thing  things |

**Pseudocode to Categorize as well as Summarize:**

#!/usr/bin/env python3

# -\*- coding: utf-8 -\*-

"""

Created on Tue May 9 13:15:47 2017

@author: deeptichevvuri

"""

# -\*- coding: utf-8 -\*-

"""

Spyder Editor

"""

from nltk.tokenize import RegexpTokenizer

from nltk.stem.porter import PorterStemmer

from gensim import corpora

import gensim

import re

import os

from summarizehelper import FrequencySummarizer

tokenizer = RegexpTokenizer(r'\w+')

# importing stop words list

with open('/Users/deeptichevvuri/Documents/CC/data/stop words.txt','r') as input\_buffer:

en\_stop=[]

for line in input\_buffer:

en\_stop.append(line.strip())

# Create p\_stemmer of class PorterStemmer

p\_stemmer = PorterStemmer()

# loop through document list

#for i in doc\_set:

a\_dir='/Users/deeptichevvuri/Documents/CC/data/Download3'

categoryTopics={}

groupTopics=[]

for name in os.listdir(a\_dir):

if os.path.isdir(os.path.join(a\_dir, name)):

groupTopics.append(name)

i=""

for file in os.listdir(os.path.join(a\_dir, name)):

org\_file = os.path.join(a\_dir, name)+'/'+file

if file=='.DS\_Store':

continue

with open(org\_file, 'r') as myFile:

for line in myFile:

i=i+line.strip()+' '

texts = []

# clean and tokenize document string

raw = i.lower()

tokens = tokenizer.tokenize(raw)

stemmed\_tokensfinal=[]

# remove stop words from tokens

stopped\_tokens = [i for i in tokens if not i in en\_stop]

# stem tokens

stemmed\_tokens = [p\_stemmer.stem(i) for i in stopped\_tokens]

for word in stemmed\_tokens:

if len(word)>3:

stemmed\_tokensfinal.append(word)

# add tokens to list

texts.append(stemmed\_tokensfinal)

# turn our tokenized documents into a id <-> term dictionarysssss

dictionary = corpora.Dictionary(texts)

# convert tokenized documents into a document-term matrix

corpus = [dictionary.doc2bow(text) for text in texts]

# generate LDA model

ldamodel = gensim.models.ldamodel.LdaModel(corpus, num\_topics=20, id2word=dictionary, passes=20)

myList=ldamodel.print\_topics(num\_topics=1, num\_words=20)

myList2=myList[0]

categoryTopics[name]=re.findall(r'"([^"]\*)"', myList2[1])

print("Category- "+name+":")

print(categoryTopics[name])

#test data

#for i in doc\_set:

a\_dir='/Users/deeptichevvuri/Documents/CC/data/testdata'

for foldername in os.listdir(a\_dir):

if os.path.isdir(os.path.join(a\_dir, foldername)):

i=""

with open('/Users/deeptichevvuri/Documents/CC/data/output.txt','a') as input\_buffer:

input\_buffer.write("Original Group\t"+"Classified Group\t"+" Similarity Score\t "+"Correct Classification\n")

for file in os.listdir(os.path.join(a\_dir, foldername)):

documentTopics={}

org\_file = os.path.join(a\_dir, foldername)+'/'+file

if file=='.DS\_Store':

continue

with open(org\_file, 'r') as myFile:

for line in myFile:

i=i+line.strip()+' '

texts = []

# clean and tokenize document string

raw = i.lower()

tokens = tokenizer.tokenize(raw)

stemmed\_tokensfinal=[]

# remove stop words from tokens

stopped\_tokens = [i for i in tokens if not i in en\_stop]

# stem tokens

stemmed\_tokens = [p\_stemmer.stem(i) for i in stopped\_tokens]

for word in stemmed\_tokens:

if len(word)>3:

stemmed\_tokensfinal.append(word)

# add tokens to list

texts.append(stemmed\_tokensfinal)

# turn our tokenized documents into a id <-> term dictionarysssss

dictionary = corpora.Dictionary(texts)

# convert tokenized documents into a document-term matrix

corpus = [dictionary.doc2bow(text) for text in texts]

# generate LDA model

ldamodel = gensim.models.ldamodel.LdaModel(corpus, num\_topics=20, id2word=dictionary, passes=20)

myList=ldamodel.print\_topics(num\_topics=1, num\_words=20)

myList2=myList[0]

documentTopics[foldername+'/'+file]=re.findall(r'"([^"]\*)"', myList2[1])

similarityScore=0.0

finalGroup=''

classification='True'

for groupType in groupTopics:

similarityIndex=0

for docTop in documentTopics[foldername+'/'+file]:

for catTop in categoryTopics[groupType]:

if docTop==catTop:

similarityIndex=similarityIndex+documentTopics[foldername+'/'+file].index(docTop)\*categoryTopics[groupType].index(catTop)

#print(similarityIndex/28.7)

currentSimmilarityScore=similarityIndex/28.7

if currentSimmilarityScore>similarityScore:

similarityScore=currentSimmilarityScore

finalGroup=groupType

if finalGroup!=foldername:

#print('wrong classification')

classification='False'

with open('/Users/deeptichevvuri/Documents/CC/data/output.txt','a') as input\_buffer:

input\_buffer.write(foldername+"\t"+finalGroup+"\t"+str(similarityScore)+"\t")

input\_buffer.write(classification+"\n")

if finalGroup==foldername:

summayTopics=categoryTopics[finalGroup]

fs = FrequencySummarizer()

print("summary")

print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

for s in fs.summarize(raw, 2,summayTopics):

print('\*',s)

**Pseudocode to summarizehelper used in above code:**

from nltk.tokenize import sent\_tokenize,word\_tokenize

from collections import defaultdict

from heapq import nlargest

class FrequencySummarizer:

def \_\_init\_\_(self, mincut=0.1, maxcut=0.9):

with open('/Users/deeptichevvuri/Documents/CC/data/stop words.txt','r') as input\_buffer:

en\_stop=[]

for line in input\_buffer:

en\_stop.append(line.strip())

self.\_stopwords = en\_stop

def summaryTopics(self, word\_sent, words):

freq = defaultdict(int)

for s in word\_sent:

for word in s:

if word not in self.\_stopwords:

freq[word] += 1

# filtering topic words only

m = float(max(freq.values()))

for w in list(freq):

if w in words:

freq[w] = freq[w]/m

else:

del freq[w]

return freq

def summarize(self, text, n, words):

"""

creates the summary using the topic words

"""

sents = sent\_tokenize(text)

assert n <= len(sents)

word\_sent = [word\_tokenize(s.lower()) for s in sents]

#print(word\_sent)

self.\_freq = self.summaryTopics(word\_sent, words)

print(self.\_freq)

ranking = defaultdict(int)

for i,sent in enumerate(word\_sent):

for w in sent:

if w in self.\_freq:

ranking[i] += self.\_freq[w]

sents\_idx = self.sentenceRanking(ranking, n)

return [sents[j] for j in sents\_idx]

def sentenceRanking(self, ranking, n):

""" return the first n sentences with highest ranking """

return nlargest(n, ranking, key=ranking.get)