

**MINOR PROJECT**

**TEXT SUMMARIZATION USING MACHINE LEARNING**

**STUDENT’S DECLARATION**

We hereby declare that the project work entitled **“TEXT SUMMARIZATION USING MACHINE LEARNING”** submitted to Maulana Azad National Institute of Technology, is a record of original work done by us under the guidance of Mrs. Saritha Khetawat, Department of Computer Science and Engineering, and this project work is submitted in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering, MANIT, Bhopal. The results embodied in this study have not been submitted to any other university for the award of any other degree or diploma.

Siva Likitha Valluru 141112009

**CERTIFICATE**

This is to certify that the above statements made by us are correct to the best of our knowledge.

Dr. Saritha Khetawat

**ACKNOWLEDGEMENT**

We would like to express our sincere gratitude to our supervisor, Dr. Saritha Khetawat, for providing her continuous and invaluable patience, guidance, suggestions, knowledge, and expertise throughout the course of this project.

**INDEX**

|  |  |
| --- | --- |
| **1. Abstract**  **2. Introduction**  **3. Methodology of Extractive Summarization**  **i. Preprocessing**  **a. Tokenization**  **b. Stop Words Elimination**  **c. Stemming**  **d. Term Weighting**  **e. Dimensionality Reduction**  **ii. Analyzing Features**  **iii. Method of Extractive Summarization**  **a. Feature Sets**  **b. Training and Testing Sets**  **c. Most Informative Features**  **d. Synsets**  **e. Extraction of Data**  **f. Weighted Sentences**  **4. Tools and Technology Used**  **5. Implementation of Extractive Summarization**  **i. Algorithm**  **6. Result Analysis of Extractive Summarization**  **7. Conclusion and Future Scope**  **8. References** |  |

***TEXT SUMMARIZATION USING MACHINE LEARNING***

**ABSTRACT**:

Text summarization is used for summarizing large datasets and corpus into brief sentences. It is not always possible for humans to manually read through every text. This can also be very time-consuming. Developing an algorithm to summarize text can save a lot of time and also prove to be effective. Naive Bayes Classifier is used to summarize text. For large texts, developers and users often cannot read every document and code. They may only rely and understand just the relevant parts corresponding to their task. Such quick skimming may lead to misinterpretation and confusion of context. Reading just the titles or headers doesn’t tell enough about the meaning. There is another possible case where users may read the entire document, disregarding the size of it. This takes too long. To resolve these two problems, we provide a solution which depicts a document into a concise and fluent summary. This paper presents a study that investigates how both extractive and abstractive summarizations can be implemented.

**INTRODUCTION:**

A summary is an informative way to cover important and relevant information of the text. Whether it is used for summarizing some parts or all parts of the text, a summary greatly reduces the reading time. A summary should be able to cover the different topics in a text without being completely redundant and wordy. Sentences shouldn’t be lengthy just to fill up the page. Summarization tools may search for headings and titles that reflect important topics in a document.

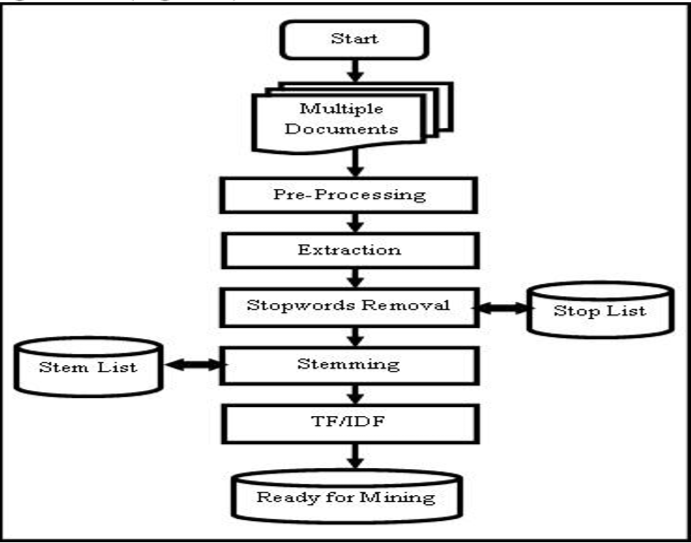
Text summarization methods are classified into extractive and abstractive summarization. Extractive summarization method consists of choosing important sentences and paragraphs from the original document. Sentences are ranked and weighted according to their importance by calculating their word features and frequency distribution. In this method, most important information is shown by displaying sentences with frequently occurring words. Abstractive summarization method, however, showcases information in clear natural language. Most important information is shown through short amount of text. Abstractive summaries use various methods to find new concepts and build a semantic representation that creates a summary which is closer to how a human might interpret.

**METHODOLOGY OF EXTRACTIVE SUMMARIZATION:**

Extractive summarization is divided into two categories: preprocessing and processing.

**I. Preprocessing**

Preprocessing is the process of turning original document into a structured and well-defined representation. It is the first and most important step done in any field of text mining.

******

Data is preprocessed using the following methods:

**A. Tokenization**

Desired document is imported into the program. Then the entire file is word tokenized. Word tokenization is the process of dividing a flow of text into meaningful elements also known as tokens. This list of words becomes the input for further processing.

**B.** **Stop Words Elimination**

Stop words are those words which contain little lexicon meaning. Most common stop words are articles and prepositions. They don't give any additional meaning to the text. Examples of stop words include words such as a, an, the, in, etc. Since stop words are not considered as key words in text summarizing, they're removed. There are various methods to remove stop words such as using Z-Methods and Mutual Information Method. However, this paper analyzes the classical method. In this method, stop words are already present in some predefined lists. From these lists, stop words are eliminated from the document.

**C. Stemming**

Stemming is the method of identifying the root or the *stem* of a word. For example, the words run, running, ran, and runner, all have a common stem: “run”. This method is used because it removes complexity from the program by removing prefixes and suffixes. An additional plus point is that stemming also saves space. Again, there are different ways to stem a certain word. There are different kinds of *stemmers* such as Lovins stemmer, Dawson stemmer, Porters stemmer, WordNetLemmatizer etc.In this paper, we will be using the WordNetLemmatizer. In this lemmatizer, rules are applied on each word. Whichever word passes all conditions, it is converted to its root.

Although stemming is very useful and pretty much reduces the size of the document by roughly 30 to 40 percent, errors may arise. Under-stemming and over-stemming, also known as false positive and false negative respectively, are the most common errors which are seen in text preprocessing. Under-stemming results when two words with the same stem are mapped to different ones. Over-stemming results when two words with different stems are mapped to the same root. Both affect the efficiency of preprocessing in the document.

**D. Term Weighting**

For every document, each term has its own importance level. For each term, a weight is mapped to it. We consider three major components that affect a term's importance:

1) Term frequency (TF) – expresses importance of a word by analyzing how it's distributed in the text

2) Inverse document frequency (IDF) – expresses importance of a word by analyzing how it's distributed in a *database* of texts

3) Document normalization – adjusts TF too see how the term is dispersed. For example, there is a difference between the same term being repeatedly used in a single paragraph and the same term frequently occurring in a large document.

**E. Dimensionality Reduction**

Dimensionality reduction is used to remove the infrequent occurrences of a word. It is similar to stop word removal, although this method removes frequent words which are irrelevant to the document. For example, infrequent occurrences could be those words which occur only once or twice. These one-time appearing words are also known as hapaxes. In a large database of documents, *document frequency* (DF) is used to determine the number of documents in which a word is present. Hapaxes are not needed, so these words are removed.

So, the constituents that make up preprocessing are tokenizing, removing stop words, stemming, calculating TF/IDF, and applying dimensionality reduction. Now, data is ready to be processed.

**II. Analyzing Features**

Summarizing a document means identifying meaningful contexts from the text, or in other words, identifying the *features* of the text. Following are the features that are considered while generating a summary:

1) Title word feature:

The title is one of the most prominent features in the text because it gives an idea about the document. There are great chances that sentences containing keywords from the title could be returned as part of the summary.

2) Sentence location feature:

Usually the first and last sentences also have a fair chance of being returned as prominent sentences. The first sentence usually sums up the rest of the document and is also known as thesis. The last sentence usually acts as a conclusion.

3) Sentence length feature:

Very large and very short sentences are usually not found in summaries.

4) Proper noun feature:

Proper noun is highly distinguished from a common noun, which usually refers to a group of entities. Proper nouns are names given to people, places, organizations, etc. Proper nouns are capitalized regardless of where they exist in a sentence.

5) Upper-case word feature:

Similar to proper noun feature described above, abbreviations, jargon, acronyms, etc. are also included in a summary.

6) Cue-Phrase feature:

Sentences containing cue phrases, such as, “in conclusion”, “therefore”, etc. also have a reasonable chance of being in a summary.

7) Font-based word feature:

If sentences contain bold, italic, or underlined words, then they're important.

8) Occurrence of non-essential information:

A document contains relevant and irrelevant information. The use of the word 'irrelevant' here applies to sentences which are redundant and extremely detailed. They're often discarded. Also, non-essential indicators exist throughout the document such as “in addition to”, “additionally”, “furthermore”, etc. They usually occur in the beginning of a sentence.

9) Discourse analysis:

Discourse analysis is probably one of the most important features in the list. For a steady and smooth flow of text, it's necessary to determine the course of the flow. Any sentences which deter from the course of this path are removed. This is again stating the above feature. Including non-essential information is not useful.

**III. Method of Extractive Summarization**

For this project, one of the most popular algorithms used is the Naive Bayes Classifier. Naive Bayes classifier is technique based on Bayes theorem. It gets the name 'naive' because it assumes that presence or absence of a particular feature is unrelated to the presence or absence of any other feature in a given class.

Drawbacks:

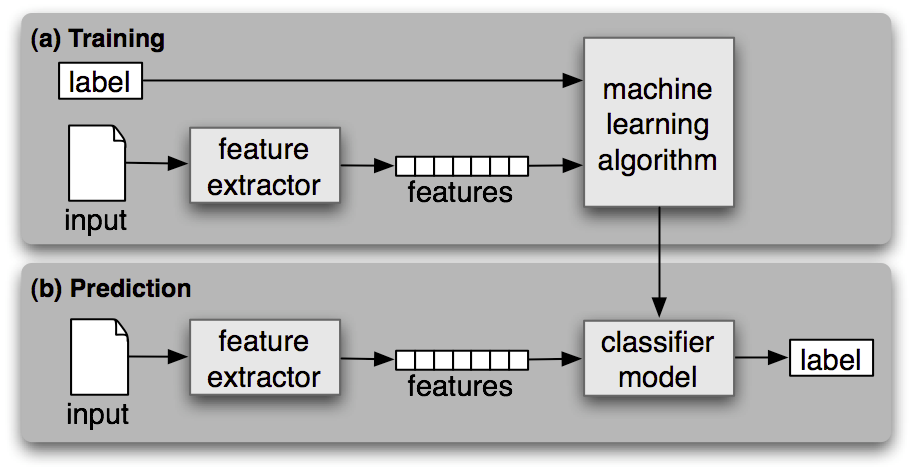
A drawback of the algorithm is that it makes assumptions that may or may not be true. Another drawback is that in case one feature depends on another, then results obtained Naive Bayes may not be true.

Advantages:

This algorithm is fairly easy to implement. It requires only a small training dataset to form relationships between given parameters. Accurate results are obtained in most cases.

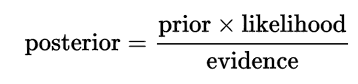
**A. Feature Sets**

When using any classifier, the first step is to decide what features of the text are relevant. Relevant features can be extracted through preprocessing. A feature set is a dictionary which maps the names of features to their values. Values are simple data types, such as booleans, strings, integers, etc. Next, we use a *feature extractor* by defining a function to process the data. By applying function on the data, we get *feature sets*. Feature sets are divided into a *training set,* a *validation test*,and a *test set*. A training set is a set of data that’s used to discover predictive relationships. A validation set is a set of data which is used to determine a stopping point. Validation set is also known as tuning set. However, we will not be considering a validation test since the training set is already large. A test set is a set of data that is used to weigh the performance of the relationships using some classifier X.



NaïvBayes classifier is *supervised* since it’s based on some labeled training corpora. Also, it is better suited for large sets of data. It is not recommended for small datasets. Data is reliable when it consists of variety and volume. Data is not generated at just one place. It must be extracted from different sources and databases. For example, consider sales marketing analysis in two different towns: A and B. A and B may require different essentials. Both need not wish for the same product. So, considering data from only A or only B is inaccurate data. For this reason, we accept diversified data.

After we get our data, we can construct feature sets. Feature sets are sets of features extracted from text with a corresponding label to each feature. Feature sets are used as input to models. During training, a feature extractor is used to convert each input value to a feature set. Feature sets capture the basic information about each input that should be used to classify it. Multiple feature sets along with their labels are fed into machine learning algorithms to generate a model. In the above figure, prediction means that the same feature extractor is used to convert test input to feature sets. Then these feature sets are fed into the model, which generates predicted labels using trained data set.





The posterior probability of a random or uncertain event is the conditional probability that is assigned after relevant evidence is taken into consideration. The prior probability, often called the prior, of an uncertain event is that one’s previous beliefs are taken into consideration before evidence is taken into account. Likelihood is the probability of an event given a set of parameters. In this case, it is the probability of *x* (predictor) given a class. In layman terms, what is the probability that event *x* will occur if *c* is true? Predictor prior probability is the prior probability of predictor. A predictor variable is used to predict another variable.

To assign a label, naïve Bayes calculates the prior probability for each label. This is determined by checking the frequency of each label in the training set. The contribution of each feature is then combined with prior probability. This will be a likelihood estimate for each label. The label with the highest likelihood estimate will become the input’s label.

**B. Training and Testing Sets**

After getting feature sets, we shuffle them to increase randomness. Feature sets are divided into training set and testing set. Usually, training set is about 70% of the feature sets. The remaining 30% is used as the test set.

**C. Most Informative Features**

After applying the algorithm on train and test sets, we extract the *most informative features.*

**D. Synsets**

WordNet is lexical database for the English language. It is used to group English words into synonyms known as *synsets*. Synsets provide short definitions of a word and also its usage in different sentences. Synsets are also known as synonym rings. WordNet is also referred as a dictionary and thesaurus.

By extracting synsets, we can find out the immediate synonyms of most frequently identified features.

**E. Extraction of Data**

Extract desired input text into the program. Then, text is preprocessed according to the given steps above.

**F. Weighted Sentences**

After preprocessing has been done, weights are assigned to each sentence. This means that sentences are ranked according to their importance. The sentences which are ranked the highest are returned. These sentences together constitute the summary.

**TOOLS AND TECHNOLOGY USED:**

**I. Hardware Requirements:**

Operating System: Windows XP, Windows Vista, Windows 7 (32bit or 64bit), Windows 8, Windows 10

RAM: Minimum of 1 GB recommended

Disk space: 850 MB free disk space is required.

**II. Software Requirements:**

Software used: Python 2.7, NLTK libraries (corpora, WordNet)

**IMPLEMENTATION OF EXTRACTIVE SUMMARIZATION:**

Program is implemented using Python 2.7, although it is compatible in Python 3.x except for a few minute pieces of syntax. For the implementation, since text processing is done, NLTK is required.

**Algorithm:**

1. Import libraries: math, NLTK, random
2. Import one of NLTK’s default libraries: movie\_reviews
3. Categorize each of the files in movie\_reviews according to their file id and category (whether review is positive or negative). Prepare a list of each of the documents according to the id and category.
4. Using FreqDist, traverse words in movie\_reviews for the most frequent word list:

for i in movie\_reviews.words():

words.append(i.lower())

words = nltk.FreqDist(words)

word\_features = list(words.keys())[:1000]

5. Create feature sets using a function:

def features (files):

sets=set(files)

feature={}

for i in word\_features:

feature[i]=(i in sets)

return feature

6. Out of the feature sets, create a training set and test set. Apply the Naïve Bayes classifier on both of the sets. Show the most informative features from the sets using classifier\_var\_name.most\_informative\_features(n). Ten features are used in this program.

7. Store the words in a separate list and derive synonyms using wordnet.

8. Now, import URLs of different movie reviews as sample texts. Preprocess the texts using the steps for preprocessing listed above.

9. Set weights for each sentence of text using TF-IDF (prominently) and return the sentences ranked highest.

10. The printed sentences constitute the summary.

**RESULT ANALYSIS OF EXTRACTIVE SUMMARIZATION:**

Extractive summarization has its advantages. It uses algorithms which don’t require large amounts of input. Naïve Bayes doesn’t require large datasets for training in order to immediately form relationships. However, using large datasets will give better accuracy.

As for the results, after algorithm is applied on the feature sets, we’ll get important and *prominent* features. These features are later used in determining ranks of sentences. Sentences with lowest ranks will be discarded. Lengthy and verbose sentences are also disregarded. TF-IDF – term frequency - inverse document frequency – is used to rank each sentence. Wherever previously extracted features are found, those sentences are assigned a higher priority. These higher priority sentences will be returned. The sentences together will form a summary.

Extractive summarization is very efficient in terms of time and space. It doesn’t require many algorithms to be applied and it’s fairly easy to implement, as compared to abstractive summarization. A disadvantage of this type of summary is that the context of the summary doesn’t always have to make sense. Meaning, if ranked sentences are out of context or if they’re not related to each other, then they won’t make sense together.

**CONCLUSION AND FUTURE SCOPE:**

Extractive and abstractive types of summarization both have their advantages and disadvantages. Extractive is better in some ways since it doesn’t deal with semantics and has a higher chance of success for this reason. It only selects important sentences based on statistical features. But a major drawback is that it suffers from inconsistency of contexts and proper balance in a summary.

Abstractive, on the other hand, reduces text size and generates a summary that satisfies semantic meaning. This means that the sentences are semantically related instead of just random sentences being together. A drawback is that it is very difficult to compute and often takes a long time for compilation process. Also, algorithms are very complicated and complex.

In conclusion, extractive text summarization is easier to build and compute. But abstractive text summarization is stronger, although difficult, because summary is semantically related.

**REFERENCES:**

Gupta, Vishal, and Gurpreet Singh Lehal. "A Survey of Text Summarization Extractive Techniques ." JOURNAL OF EMERGING TECHNOLOGIES IN WEB INTELLIGENCE 2.3 (August 2010): 258-68. Web.

Haiduc, Sonia, Jairo Aponte, Laura Moreno, and Andrian Marcus. "On the Use of Automated Text Summarization Techniques for Summarizing Source Code." 2010 17th Working Conference on Reverse Engineering (2010): 35-44. Web.

Patil, Aarti, Komal Pharande, Dipali Nale, and Roshani Agrawal. "Automatic Text Summarization." International Journal of Computer Applications 109.17 (2015): 18-19. Web.

Wong, Kam-Fai, Mingli Wu, and Wenjie Li. "Extractive summarization using supervised and semi-supervised learning." Proceedings of the 22nd International Conference on Computational Linguistics - COLING '08 (2008): 985-92. Web.

Siddique, Tanveer J., Dr. Multi - Document Summarization ( MLTA 2013). Powerpoint. N.p.: n.p., n.d. Print.

S, Vijayarani, Dr, J. Ilamathi, and Nithya. "Gaigole, Pritam C., L. H. Patil, and P. M. Chaudhari. "Preprocessing Techniques in Text Categorization." National Conference on Innovative Paradigms in Engineering & Technology (n.d.): 1-3. Web." International Journal of Computer Science & Communication Networks 5-17-16 (n.d.): 7-16. Web.

Gaigole, Pritam C., L. H. Patil, and P. M. Chaudhari. "Preprocessing Techniques in Text Categorization." National Conference on Innovative Paradigms in Engineering & Technology (n.d.): 1-3. Web.

A, Anil Kumar, and Chandrasekhar S. " Text Data Pre-processing and Dimensionality Reduction Techniques for Document Clustering ." International Journal of Engineering Research & Technology (IJERT) 1.5 (July 2012): 1-6. Web.

"Text mining." Wikipedia. Wikimedia Foundation, 06 Apr. 2017. Web. 08 Apr. 2017.

"Automatic summarization." Wikipedia. Wikimedia Foundation, 21 Mar. 2017. Web. 08 Apr. 2017.