A Framework To Automatically Categorize The
Unstructured Text Documents

Anshika Singh*
Bhaskaracharya College of Applied Sciences, Dwaraka, University of Delhi, New Delhi – 110075, Delhi, India; anshika1807@gmail.com

Abstract

Objectives: This paper presents an application of Text mining to automatically categorize the text documents from a predefined set of class or categories based on the content of the document. Method: In our framework, Association rule mining is used to generate frequently occurring word sets from training data set rather than the word sets in the whole document to derive a feature set to be used as training data. The new documents are then classified by applying classification method on the extracted feature sets. Findings: The results show that instead of using the whole word set extracted from the training document, frequently associated word sets extracted by applying association rules provide much accurate classification of the new document. Applications: The associated word sets not only relatively reduce the size of the training data but also relate to context of the document, which is further used to classify the new document. This approach can be used for efficient storage and retrieval of documents over the web or organizational databases. Hierarchical classification organizes the documents into tree like structure or meaningful hierarchy to form a single path structure which can add new categories or classes when needed. This approach can be combined with hierarchical structure for efficient browsing on search engines and provide a boulevard for future research.

Keywords: Association Rules, Classification, Text Categorization, Text Mining

1. Introduction

Data mining can be described as Knowledge Discovery from data1, i.e., significant extraction of valuable information or knowledge from an extensive assortment of data or simply database. Though, data mining itself is an integral part in knowledge discovery as shown in Figure 1.

Figure 1. Knowledge discovery from data.
Data mining can be categorized into the following, shown in Figure 2:

1.1 Data Collection
Collecting the data for processing usually stored in database or data warehouses in the form of spreadsheets, surveys, files etc.

1.2 Data Processing
comprises of two steps: Feature extraction to abstract out the significant features for a specific application and Data Cleaning and Integration to eradicate erroneous or missing data in order to ensure no inconsistencies presenting data and integrate the extracted data into a unified format which can be utilized for further processing.

1.3 Data Analysis or Analytical processing
efficient analysis of processed data and determining an ideal algorithm to figure out the best solution to the problem. A problem may further be divided into sub problems and represented as building blocks to generate the final outcome or conclusion.

Text mining can be addressed as Intelligent Text Analysis or Knowledge Discovery in Text (KDT) and defined as a knowledge intensive process which involves user interaction with a huge assortment of data to ascertain and evaluate the useful information by identifying and exploring interesting patterns of specific type in its core knowledge discovery process with the help of analytic tools.

Text mining is analogous to data mining, except the data mining presumes that data is already structured; data preprocessing focus on scrubbing and normalizing the data and creating table joins. Text mining extracts information from unstructured or semi-structured data in the documents in the data collection or databases. Here, the preprocessing aims to identify and extract the features for representing natural language text and transform these unstructured representative features into an intermediate format which is explicitly structured.

Text mining is an automated or partially automated processing of text. Text mining can be classified in three major steps: Data Collection, Data Preprocessing and Analysis as illustrated in Figure 3.

Figure 2. A model for Data mining.

Figure 3. A model for Text Mining.
linguistic phrases, key phrases etc. The other objective is to streamline the representative feature sets by identifying the features which are computationally efficient and useful for discovering patterns. The process is sometimes supported by normalization, validation, cross reference of features against constrained vocabularies or dictionaries, ontologies, lexicons or knowledge bases to generate a smaller subset of refined semantically rich features. The core functionality is the analysis of frequency sets, their associations, distributions, and co-occurrence, to discover the nature and relationships between the features as reflected in the Text corpus as a whole. Consequently, a document-term matrix is computed which is more structured representation from either text corpus or text database. Document-term matrix represents bag-of-words mechanism and consists of frequencies of each term for all documents in the corpus which is further used as input for analysis phase. In the analysis phase, core mining operations such as classification, clustering, relationship identification, syntax analysis, document summarization etc. are performed on the document term matrix and visualization tools are used to facilitate the navigation and exploration of concept patterns which use graphical approaches to express complex relationships between data.

Perhaps, Text mining process exploits various methodologies and techniques like data mining, information extraction, information retrieval, and corpus based computational linguistics. For instance, Text mining make use of Information Extraction technique for preprocessing a corpus of text documents into a more structured format which can further be used as input to data mining module as illustrated in Figure 4. Information Extraction (IE) is defined as the process which automatically obtains structured data from unstructured data, i.e., natural language document. IE perform the following:

(a) Term Analysis, i.e., identifying the terms in a document, where a term may consist of one or more words,
(b) Named-entity recognition, i.e., identifying the names in document, such as name of people, place, time, quantities and associated units, etc. and
(c) Fact extraction, i.e., identifying and extracting complex facts from documents such as relationships between entities or events. Information Retrieval techniques have extensively used the “bag-of-words” model for tasks such as document matching, ranking, and clustering.

1.4 FP-Growth Algorithm

FP-Growth algorithm is used for generating association rules based on divide and conquer method. This algorithm generates frequent item sets using the combination of data attributes, without any candidate set generation. The algorithm consists of two major steps:

1) Building a compressed data structure showing frequent item set called FP-tree.
2) Divide FP-tree into a set of conditional Database and mines each database separately, thus extract frequent itemsets from FP-tree directly.

Step 1: Constructing a FP-tree.

Input: A transaction database D.
Output: A FP-tree

Procedure:
1. Create the root of the FP-tree labeled as “NULL”.
2. Scan the transaction database D and finding minimum support (min_support) for each item.
3. Collect the frequent item sets (F_items) and create nodes for each F_item and map the path for each node.
4. Note: If two transactions share the common prefix items then the path for both will overlap and frequency count for that node will be incremented by 1. Otherwise, the paths for two transactions sharing common item but uncommon prefix items will be disjoint and link the nodes of shared item for both the transactions.

Figure 4. A text mining framework based on Information Extraction.
5. Sort frequent item sets, \( F_{\text{items}} \) in decreasing order of their min_support to form a sorted list of frequent item sets, \( L_{\text{freq}} \).

For each transaction \( T_{\text{id}} \) in \( D \), repeat steps 2 to 4 till all the transactions are mapped to FP-tree path.

For example, consider the transaction database \( D = \{ T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9, T_{10} \} \), where \( T_1: \{a,b\}, T_2: \{b,c,d\}, T_3: \{a,c,d,e\}, T_4: \{a,d,e\}, T_5: \{a,b,c\}, T_6: \{a,b,c,d\}, T_7: \{a\}, T_8: \{a,b,c\}, T_9: \{a,b,d\}, T_{10}: \{b,c,e\} \).

Firstly, create the root node of FP-tree as NULL. Then start scanning the transaction database \( D \).

Read \( T_1: \{a,b\} \) and create two nodes – \( a \) and \( b \) with the path \( \text{NULL} \rightarrow a \rightarrow b \) and set min_support of \( a \) and \( b \) to 1.

Read \( T_2: \{b,c,d\} \) and create three nodes – \( b \), \( c \), and \( d \) with the path \( \text{NULL} \rightarrow b \rightarrow c \rightarrow d \) and set min_support of \( b \), \( c \), and \( d \) to 1. Though item \( b \) is shared by both \( T_1 \) and \( T_2 \), the path for \( T_1 \) and \( T_2 \) are disjoint as they do not share common prefix. Add the links between \( b \) of both transactions.

Read \( T_3: \{a,c,d,e\} \): Since \( T_1 \) and \( T_3 \) share common prefix item \( a \), the path for both will overlap and min_support for node \( a \) will be incremented by 1. Create three nodes \( c \), \( d \), and \( e \) with the path \( a \rightarrow c \rightarrow d \rightarrow e \) and set min_support of \( c \), \( d \), and \( e \) to 1. Though items \( \{c,d\} \) is shared by both \( T_2 \) and \( T_3 \), the path for \( T_2 \) and \( T_3 \) are disjoint as they do not share common prefix. Add the links between \( c \) of \( T_2 \) and \( T_3 \), and \( d \) of \( T_2 \) and \( T_3 \) respectively.

Repeat the procedure till all the transactions in \( D \) are mapped to FP-tree path. Constructed FP-tree is given in Figure 5.

**Step 2: Generating frequent item sets:**

The FP-growth algorithm follows Bottom-up approach (i.e., from leaves towards the root) to generate the frequent item sets from FP-tree.

1. Firstly, extract prefix path subtrees ending in an item set to extract frequent item sets by decomposing the problem recursively and obtain the conditional FP-tree. While building Conditional FP-tree, we consider the transactions which contain only a particular item set and removing the rest from transaction database. For example. Prefix path subtree with path that contain node \( e \) is illustrated in Figure 6.

2. Then merge the results.

- **Algorithm for generating frequent itemset with FP-tree by pattern fragment growth:**

  In order to mine frequent itemsets from FP-tree, we call the method: \( \text{FP-growth}(\text{FP-tree, NULL}) \) which is implemented as follows:

  Input: FP-tree constructed in step 1 and minimum support threshold.

  Output: Complete set of frequent patterns

  \[
  \text{FP-growth}(\text{Tree}, P1) \\
  \text{if} \text{Tree contains a single prefix path} \ P \\
  \text{then for each combination set of nodes in} \ P \\
  \text{generate a pattern} \ P1 \cup P2 \text{ with min-support= minimum support of nodes in } P2;
  \]
else for each item \( a_i \) in multipath subtree \( Q \),
then generate a pattern \( P2 = a_i \cup P1 \) with
\[
\text{min\_support} = a_i.\text{min\_support}
\]
construct \( P2 \)'s conditional pattern base and con-
ditional FP-tree say \( \text{Tree}_{P2} \);

\[
\text{if} \quad \text{Tree}_{P2} = \emptyset
\]
then call \( \text{FP\_growth}(\text{Tree}_{P2}, P2) \);

Frequent item sets for transaction database \( D \):

| Suffix | Frequent itemsets |
|-------|-------------------|
| \( e \) | \{e\}, \{e, d\}, \{a, d, e\}, \{e, a\} |
| \( d \) | \{d\}, \{c, d\}, \{b, c, d\}, \{a, c, d\}, \{b, d\}, \{a, b, d\}, \{a, d\} |
| \( c \) | \{c\}, \{b, c\}, \{a, b, c\}, \{a, c\} |
| \( b \) | \{b\}, \{a, b\} |
| \( a \) | \{a\} |

1.1 Naïve Bayesian Classification for Text

The naïve-bayes classifier\(^5\) for calculating the
probability (PNB) that a document having attribute \( a_i \)
oblongs to class \( j(C_j) \) is given as:

\[
\text{PNB} = P(C_j) \Pi P(a_i | C_j)
\]

\( P(C_j) \) is estimated based on the fraction of each tar-
get class in the training data. \( P(a_i | C_j) \) is calculated after
m-estimate approach in order to avoid zero probability
value using the following equation:

\[
\frac{n_k + 1}{n + \text{vocabulary}}
\]

where \( n \): Total no of frequent word sets in all training
examples whose class value is \( C_j \),
\( n_k \): no. of occurrences of the frequent word set found
amongst \( n \),
\( \text{vocabulary} \): Total no. of distinct word sets found
within all the training data.

\( P_c \): Conditional Probability

Training_Naive_Bayes_Classifier\(^6\)(class, Document)
|
Vocabulary ← Extract_vocabulary(Document)
\( N \) ← Count_documents(Document)
For each class \( c \in \text{Category} \) do
\( N_c \) ← Count_documents_in_class (Document, Class)

\[
\text{Prior}[c] \leftarrow \frac{N_c}{N}
\]

Text ← Concatenate_text_of_all_documents_in_class
\( \text{Document, Class} \)
For each term \( \in \text{Vocabulary} \) do
\( T_t \) ← Count_tokens_of_term(text, term)
For each term \( \in \text{Vocabulary} \) do
\( P_c[\text{term}][\text{class}] \leftarrow \frac{T_{ct} + 1}{\sum_{t'}T_{ct'} + 1} \)
return Vocabulary, prior, \( P_c \)

Testing_Naive_Bayes_Classifier (Category, Vocabulary, \( P_c \), Document)
|
Weight ← Extract_Tokens_From_Document
(Vocabulary, Document)
for each class \( \in \text{Category} \) do
\( \text{score}[\text{class}] \leftarrow \log \text{prior}[\text{class}] \)
For each term \( \in \text{Weight} \) do
\( \text{score}[\text{class}] += \log P_c[\text{term}][\text{class}] \)
return \( \arg\max_{\text{class} \in \text{Category}} \text{score}[\text{class}] \)

2. Proposed Framework

Text categorization or Document categorization can be
defined as a method of assigning a target class or category
to the document from a predefined set of class or catego-
ries based on the content of the document. In our proposed
framework, we automatically classify the research papers
into a set of categories using Text mining. In our imple-
mentation, we used the tool Rapidmineral\(^2\) which provides
an integrated environment for text mining, as illustrated
in Figure 7. We initially start with a collection of research
papers of various disciplines to form a text corpus which
is simply an assortment of natural-language documents.
Each document is considered as a transaction in text cor-
pus and set of items represent frequently occurring words
in the transaction. The next step is the application of text
mining techniques in order to obtain the Document term
matrix for further analysis which starts with parsing and
preprocessing the text corpus. Once the Term-Document
matrix has been obtained, we discover the frequent item
sets from each document by applying FP-growth algo-

\text{Prior}[c] \leftarrow \frac{N_c}{N}

\]

Lastly, we apply Naïve Bayes classification on the
derived features sets to categorize the new documents.
2.1 Data Collection
Initially, we collected 34 research papers of three disciplines: Computer Science (CS), Electrical Engineering (EE) and Mechanical Engineering (ME) from the World Wide Web to form a text corpus.

2.2 Data Preprocessing
The preprocessing includes grammatical parsing, data cleaning, stemming, part-of-speech tagging of the text corpus in order to achieve the accurate calibration of volume and semantic level of features to depict the document meaning precisely. This process involves eliminating commas, punctuations, periods, stop words like “the”, “all” etc. and extracting the latent features such as words, terms, concepts, key phrases etc. Stemming refers to extraction of the <root> of the word based on dictionary or set of rules given by linguists. For example, the word “process” is extracted from the word “processing”. Part-of-speech is used to mark up the words equivalent to a particular part of speech. For example, identifying the words as nouns, verbs, adjectives, adverbs, etc. based on definition or context.

The next step is creating the Term document matrix, T by counting the number of occurrences of each term in each document d_j where d_j is a multidimensional vector of keywords. Each row t_i of T denotes occurrence of the term in each document D_j. The document d_j is represented as:

\[ d_j = \left[ \begin{array}{c}
    f_{1j} \\
    f_{2j} \\
    \vdots \\
    f_{nj}
\end{array} \right] \]

where f_{ij} represent no. of occurrences of i^{th} term in document j. This type of representing complete set of these n vectors for all documents is referred as Vector space model is shown in Figure 8. The Term-Document matrix is as follows:

\[ D = \left\{ \begin{array}{ccc}
    f_{11} & f_{12} & \cdots & f_{1n_d} \\
    d_{n_d} & \cdots & \cdots & \cdots \\
    d_{n_d} & \cdots & \cdots & \cdots \\
    \vdots & \vdots & \ddots & \vdots \\
    d_{n_w1} & d_{n_w2} & \cdots & d_{n_wn_d}
\end{array} \right\} \text{words} \]

There are several methods to calculate the weights of the words or terms. One of the popular methods is Term Frequency Inverse Document Frequency (TF-IDF) where TF represent the frequency of word or term in a document and IDF denote the measure of the rarity of a word or term within the complete set of documents. TF and IDF are represented as:

\[ IF_i = \frac{n_i}{\sum_{k=1}^{N} n_k} \quad \text{and} \quad IDF_i = \log \frac{N}{df_i} \]

where n_i = no. of occurrences of the considered terms, n_k = no. of occurrences of all terms in the document, N = no. of occurrences of the considered terms, df_i = no. of documents that contain term i.

TF-IDF method measures the statistical strength of the given term in reference to the query.
2.3 Analysis

Firstly, we generate an extensive feature set from the association rules obtained from FP-growth algorithm. Association Rules and Rules tree generated from application of FP-growth Algorithm is shown in Figure 9 and Figure 10 respectively. The feature set $L_i$ representing frequent word sets is generated from the term document matrix $T$ is generated using input as minimum support (min_support).

**Association Rules**

| Rule               | Confidence |
|--------------------|------------|
| [based] $\rightarrow$ [using] | 0.010      |
| [based] $\rightarrow$ [introduction] | 0.010      |
| [work] $\rightarrow$ [using] | 0.012      |
| [example] $\rightarrow$ [using] | 0.012      |
| [case] $\rightarrow$ [using] | 0.012      |
| [study] $\rightarrow$ [introduction] | 0.012      |
| [example] $\rightarrow$ [introduction] | 0.012      |
| [system] $\rightarrow$ [time] | 0.012      |
| [case] $\rightarrow$ [time] | 0.012      |
| [using, given] $\rightarrow$ [introduction] | 0.012      |
| [using, current] $\rightarrow$ [introduction] | 0.012      |
| [work] $\rightarrow$ [using, introduction] | 0.012      |
| [introduction, work] $\rightarrow$ [using] | 0.012      |
| [using, current] $\rightarrow$ [time] | 0.012      |
| [system] $\rightarrow$ [using, time] | 0.012      |
| [using] $\rightarrow$ [introduction] | 0.012      |

Figure 9. Association Rules Generation.

The we proceed with regular expression search for finding the matching feature set or word sets in $L_i$ with that of words in the new document and calculating the probability values of each feature set $L_i$ containing frequent word sets using Naïve Bayes method. Lastly, we apply the Naïve Bayes classification algorithm and calculate the probability of each class using conditional probability of the matched features sets or word sets and the probability values for each target class in order categorize the new document based on the target class having highest probability values.

3. Implementation and Results

To illustrate our framework, we collected 34 research papers of three disciplines: Computer Science (CS), Electrical Engineering and Mechanical Engineering from the World Wide Web to form a text corpus. Considering each document as a transaction and minimum support is two. We preprocess the text corpus to ensure that the data doesn’t contain any unnecessary word or filling the missing gaps in the data. Then the extracted data is transformed into a term-document matrix of training set which serves as an input for further application of FP-growth algorithm in the analysis phase. As a result, we get the frequent word sets and their number of occurrences in each predefined classes called Computer Science (CS), Electrical Engineering (EE) and Mechanical Engineering (ME). We find the probability of the frequent word sets using Naïve Bayesian classifier. After preprocessing the new document, we search for matching word sets in the feature sets obtained from training data with that of frequent word set of new document. We collected conditional probability of the matched word set for each target class and calculated the probability values for each target class using Naïve Bayesian Method. Unlike the word set of the whole document as training data, we used the frequent word sets created from FP-growth algorithm which
A Framework To Automatically Categorize The Unstructured Text Documents

employs association rules for much accurate classification of the new document.

We selected a random text file named “testme.txt” as input to our framework as shown in Figure 11 and the output shows the document belongs to the category: Electrical Engineering (EE). The results are illustrated in the Figure 12.

4. Conclusion

In our approach, we considered documents as transactions and the set of frequently occurring words as a set of items in the transaction. We used FP-growth to generate frequent occurring word sets from training data set rather than the word sets in the whole document to derive a feature set to be used as training data. The new documents are classified by applying Naïve Bayes method on these feature sets. The results show that our approach is an effective way to categorize the text document.

5. References

1. Han J, Kamber M, Pei J. Data Mining: Concepts and Techniques, Morgan Kaufmann Publishers Inc.: San Francisco, CA. 2012. P. 7–8. Available from Crossref
2. Miller Thomas W. Data and Text Mining: a Business Applications Approach. Pearson Edition. 2005. p. 104–124.
3. Berry Michael W. Survey of Text Mining: Clustering, Classification, and Retrieval. Springer; 2004. P. 24-43. Available from Crossref
4. Baeza-Yates R, Ribeiro-Neto B. Modern Information Retrieval. ACM Press: New York.; 1999. P. 287–88.
5. Kulkarni A R, Tokekar V, Kulkarni P. Identifying context of text documents using Naïve Bayes classification and Apriori association rule mining. 2012 CSI Sixth International Conference on Software Engineering (CONSEG): Indore. 2012. P. 1–4. Available from Crossref
6. Manning Christopher D, Raghavan P, Hinrich Schütze. Introduction to Information Retrieval. Cambridge University Press. 2009. P. 256–62.
7. Rapid Miner Studio 7.3. Available from https://my.rapidminer.com/nexus/account/index.html#downloads. Date accessed: 09/02/2017.
8. Feldman R, Fresko M, Hirsh H, Aumann Y, Liphstat O, Schler Y, Rajman M. Knowledge Management: A Text Mining Approach. Proceedings of the 2nd International Conference on Practical Aspects of Knowledge Management (PAKM98): Basel, Switzerland.; 1998. P. 91-9. PMid:9688329