Extracting Summary from Documents using K-mean Clustering Algorithm

Manjula.K.S[1] and D.Venkata Swetha Ramana [2]

R.Y.M.E.C, Bellary , INDIA

Summary
Extracting summary from the documents is a difficult task for human beings. There fore to generate summary automatically has to facilitate several challenges; as the system automates it can only extract the required information from the original document. This reduces the work to compress the original document and extract only essential information with one of the text mining technique known as “diversity”. This diversity helps to find the multiple means in the document. Document sentence use one of scoring technique MMR (Maximum Marginal Relevance) to get the quality text summary. MMR approach depends on the document sentences, and tries to apply restriction on the document sentence to get the relevance important sentence score by MMR, known as generic summarization approach. The generic summarize approach is employed with one of the clustering method known as K-Mean clustering to find the summary of the document. This method helps to process the data set through certain number of clusters and find the prior in the data sets. This helps to find the similarity of each document and generate the summary of the document.

Key words:
MMR (Maximum Marginal Relevance), K-Mean Clustering, Generating Summary

1. Introduction

The searching of important information from a large document is very difficult job for the users thus to automatically extract the important information or summary of the document. We have several techniques to process the summary of the document. The summary function document helps the users to get the relevant document based on there needs. These summary helps the users to reduce time instead of reading the whole document and it provide quick information from the large document. In today’s world to extract information from the World Wide Web is very easy. This extracted information is a huge text repository.
These approaches resolve some problems in employing information retrieval technique. The information retrieval technique will be based on keywords to search the desired information. This in turn requires more time to analyze the satisfied information. Thus we use one of the automatic text summarization processes, by scoring technique known as MMR, Where MMR is the information retrieval technique.

Manuscript received August 5, 2014
Manuscript revised August 20, 2014

This technique retrieves important sentence emphasize on high information richness in the sentence as well as high information retrieval. These multiple factors help to maximize coverage of each sentence by taking into account the sentence relatedness to all other document sentence.
These related maximum sentence generated scores are clustered to generate the summary of the document. Thus we use k-mean clustering to these maximum sentences of the document and find the relation to extract clusters with most relevant sets in the document, these helps to find the summary of the document.

Existing method:
To generate the summary of a system, concerns help to identify the connectors and components of the system. Our approach differs from other architectural work in that we rely on recovered software concerns to help identify components and connectors. A concern is a software system’s role, responsibility, concept, or purpose. Generally software documents are created alongside software development, testing or maintenance phase. Software goes through several changes. Hence retrieving the high level architecture of meaning of the software document becomes difficult task. All the present system emphasizes on either UML based technique or architecture diagram for extracting the architecture. However the diagrams are not searchable. Hence text level summarization of software architecture is needed. We posit that, by recovering concerns, we can improve the correctness of recovered components, increase the automation of connector recovery, and provide more comprehensible representations of architectures. Proposed Technique emphasizes on meeting this goal. The Drawback of The system depends upon clarity of the document and how well a document is written to present different aspects of software document. Hence dependency is based on the process of creating the document. Due to semantic measurement of the document, the process is slower for generating summary of larger document.

2. Proposed Method

Figure 1 shows the proposed approach of the system. Thus this method also finds the different features for single document. The input to this approach can be a file or document and source code. If the input is a document it extracts the words and sentences from the document, and for source code it extracts identifiers and comments. To automatically recover the architecture we use one of the machines learning based approach, In this approach we have different process to recover the architecture.

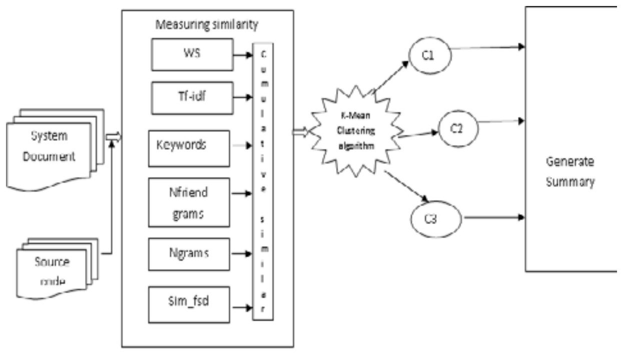

Figure 1: The approach of Generating Summary

Hence retrieval of high level meaning of the document becomes difficult task. All the present system either use UML based technique to extract the summary. However diagram is not searchable, hence text level summarization is needed. The drawback of existing system is the process is slower to generate the summary of large document. Thus we go to the proposed system to improve the performance of the system.
The input of the proposed system is purely a text document. Thus we apply all the technical blocks to the text document. These are the highest relevance or occurrences extracted from the document. These highest relevance are clustered with one of the mining technique method is k-mean clustering and extract the most relevant sets from those document. These essential clusters provide the summary of the document. The method for summary generation depends on extraction of high important sentence from the document, by MMR scoring technique. Thus MMR motivates us to choose some document features. These document sentence features are clustered using k-mean clustering algorithm to cluster the summary length. The proposed method use six different features it has shown in figure 1, these features identify the processed document like partitioning the document, stemming using porter's stemmer and removing stop words.
These features are as follows:

Word Sentence Score (WSS): The word scoring is calculated for summarization of the terms weight, term are nothing but words. The terms weight is calculated by number of words in each sentence divided by number of total sentence in the document. Where Wij is the term weight the term tij in the sentence si.

$$WSS(S_i) = 0.1 + \frac{\sum_{t_j \in S_i} W_{ij}}{HTFS} \quad | \, no.of \ sentences \ containing \ t_j >= \frac{1}{2} LS$$

Where WSS (word sentence score), TFS(total frequent sentence in the document),the sentence is represented by Si and Wij is the words in the sentence. The 0.1 is minimum score the sentence gets in the case its terms are not important.
Selecting N-Grams: Non-grammatical co-occurrence relations refer to the joint occurrence of words within a certain distance in the text. This broad definition captures several sub-types of co-occurrence relations such as n-grams.
N-grams: An n-gram is a sequence of n words that appear consecutively in the text. N-gram models are used extensively recognition the disambiguation tasks. In an n-gram model the probability of an occurrence of a word in a sentence is approximated by its probability of occurrence within a short sequence of n words. Typically sequences of two or three words (bi-grams or trigrams) are used, and their probabilities are estimated from a large corpus (corpus is a set of documents). These probabilities are combined to estimate the a priori probability of alternative acoustic interpretations of the utterance in order to select the most probable interpretation. The information captured by n-grams is, to a large extent, only an indirect reflection of lexical, syntactic and semantic relationships in the language. This is because the production of consecutive sequences of words is a result of more complex linguistic structures. However, n-grams have been shown to have practical advantages for several reasons: it is easy to formulate probabilistic models for them, they are very easy to extract from a corpus, and, above all, they have proved to provide useful probability estimations for alternative readings of the input.
The n-gram sequence module is computed by the following process. An n-gram "is a contiguous sequence of n items from a given sequence of text". In simplistic terms this means an n-gram is a collection of tokens where "n" equals the number of tokens contained within the collection. A token within this context can basically be any portion of data divided into smaller logical pieces by the n-gram creator. One of the simplest examples to consider is a text document. A token within a text document might represent each individual word within the document as delimited by spaces with all punctuation characters removed. However, many alternative tokenization strategies could be devised for the production of tokens from a given text document.
Term Sentence Frequency (TSF): The term frequency is very important feature. TF (term frequency) represents how many time the term appears in the document (usually a compression function such as square root or logarithm is applied) to calculate the term frequency. Thus term

sentence frequency is calculated by total number of term frequency divided by total number of sentence in the document.

The term Identifying sentence boundaries in a document is based on punctuation such as (. , (, “, [, {, etc) and split into sentences. These sentences are nothing but tokens.

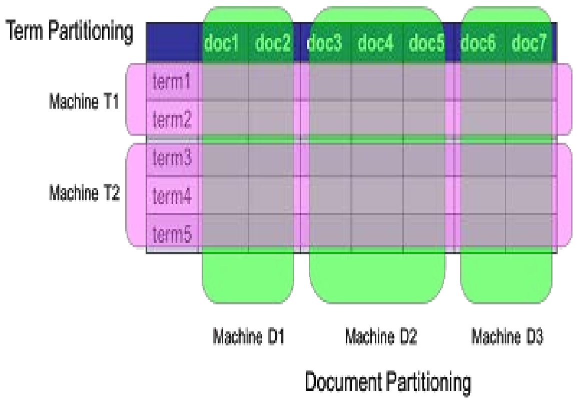

Figure 3.1: The Document Partitioning

Stemming: Once the documents have been tokenized, and apply a stemming algorithm to each token. Stemming is a means of reducing a word down to its base, or stem. For example 'search', 'searching' and 'searched' all get reduced to the stem 'search'.

Stemming greatly reduces the number of terms that the search engine has to index, which leads to a much more performance search experience. It also means that search queries do not have to contain exact word matches to the documents in the index. A search for the term 'fish' should match all documents that contained any word that can be stemmed to 'fish'. This improves recall; returning results of which more will be relevant to the user.

These tokenized sentences are splited into n-grams based on individual token. These splited n-grams should eliminate or filter the stop words from the selected grams.

Stop word filtering: In any document there will be many words that appear regularly but provide little or no extra meaning to the document. Words such as 'the', 'and', 'is' and 'on' are very frequent in the English language and most documents will contain many instances of them. These words are generally not very useful when searching; they are not normally what users are searching for when entering queries. Because of this it can be beneficial to remove these words from the index. This has the benefit of reducing the size of the index, as well as improving the performance of retrieving documents from the index.

Keyword Frequency: The keywords are the top high frequency words in term sentence frequency (TSF). The words score are chosen as keywords, based on this feature, any sentence in the document is scored by number of keywords it contains, where the sentence receives 0.1 score for each key word.

Nfriend Gram Frequency: The nfriend feature measures the relevance degree between each pair of sentences by the number of sentences both are similar to. The friends of any sentence are selected based on the similarity degree and similarity threshold. Thus the minimum threshold value is 0.3.

$$Nfriends(s_i, s_j) = \frac{s_i(friends) \cap s_j(friends)}{|s_i(friends) \cup s_j(friends)|} \,|\, i \neq j$$

Where i≠j in selecting the sentence of the document.

Ngram Frequency: This feature determines the relevance degree between each pair of sentences based on the number of n-grams they share. Thus n-grams are total number of unigrams or the count of unigrams in each sentence.

$$Ngrams(s_i, s_j) = \frac{s_i(ngrams) \cap s_j(ngrams)}{|s_i(ngrams) \cup s_j(ngrams)|} \,|\, i \neq j$$

Where i≠j in selecting the sentence of the document.

The Similarity to First Sentence (Sim_fsd): This feature is to score the sentence based on its similarity to the first sentence in the document. The first sentence in the document is very important sentence in the document. The sentence has board coverage of the sentence set (document) will get the high score.

Cumulative similarity : The sentence with different features and frequencies gets the highest scores , these scores are combined with bricks to calculate the cumulative similarity. We calculate the sentence cumulative by these scoring features between the sentences in the document. This cumulative similarity use TF-IDF (Term Frequency-Inverse Term Frequency) to find the similarity of different documents by cosine similarity algorithm.

TF-IDF is a technique used to retrieve relevant documents from a collection of documents. It normalizes the importance of a term across many documents of varying sizes. This makes use of TF-IDF to help score terms in documents.

TF-IDF is calculated by multiplying the term frequency by the inverse of the document frequency of that term. Term frequency is how often a particular term appears in this document, and document frequency is how many documents contain at least one instance of that term.

$$W_{ij} = tf_{ij} \times isf = tf(t_{ij}, s_i)\left[1 - \frac{\log(sf(t_{ij}) + 1)}{\log(n+1)}\right]$$

The term frequency represents how important a term is to a document. It makes sense that a document with many occurrences of a term is probably very relevant to a search for that term; hence the term frequency will be high. This

is normalized to take into account the length of the document, which prevents longer documents search results. The inverse document frequency acts to prevent very common terms from affecting the search results. A term that appears in all documents is not a good term to differentiate documents, so very common terms across all documents are penalized and contribute less to a particular document's score.
In this way "stop words, very common words such as 'and', 'the' or 'but', have little impact on a search. The excludes common stop words before this step so as to reduce the size of the index, but TF-IDF can help to lessen the impact of any corpus specific stop words that might not be included in the default stop word filter.
Cosine Similarity: Once terms have been scored for their relevancy to a document, this can calculate how relevant each document is to a query. To do this, we use a measure called cosine similarity.
Both documents and queries can be represented as vectors. These vectors have a dimension for every token in the entire index. The values of these dimensions will be the TF-IDF score for a token in the document/query being represented.
The angle, θ, between these vectors represents how similar they are to each other, which in turn tells us how similar a document is to a query, or another document. Identical vectors will have cos θ = 1, whilst completely opposite vectors will have cos θ = 0. This is much easier to visualize when dealing with just two dimensions, as shown in the diagram below. Although harder to imagine, vectors with a large number of dimensions behave in the same way.

3. K-Mean Clustering

Clusters are nothing but grouping the similar documents together. Thus we have many clustering technique ,in that one of the technique is k-mean clustering. The main region to use k-mean clustering is to group all the similar set of documents together by cumulative similarity ,and divide the document into k-clusters is to fine k centroids for each cluster. These centroids are placed in different location (not arranged properly) defines different result. Thus we go to next step to place them properly according to the given data and to group the nearest centroid. Thus we repeat this step until the complete grouping is done to the entire document. At this point we have to re-calculate k new centroids as center of previous step clusters. These k new centroids build the new data set points of nearest new centroid. As the loop is generated the k-centroids change their location step by step until no more changes are done.
Finally this algorithm aims at minimizing the proposed function

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2$$

where $\left\| x_i^{(j)} - c_j \right\|^2$ is a chosen distance measure between a data point $x_i^{(j)}$ and the cluster centre c_j, is an indicator of the distance of the n data points from their respective cluster centers.

4. Conclusion

The effective diversity based method combined with K-mean clustering algorithm to generating summary of the document. Two ways are used for finding the diversity: the first one is as preliminary way where the software document sentences are clustered based on the similarity - similarity threshold is 0.03 (The minimum value if non of the sentence are matched in the document). The second way is to apply the proposed method on each one sentence as summary sentence. The clustering algorithm is used as helping factor with the method for finding the most distinct ideas in the text. The results of the method supports that employing of multiple factors can help to find the diversity in the text because the isolation of all similar sentences in one group can solve a part of the redundancy problem among the document sentences and the other part of that problem is solved by the diversity based method which tries to select the most diverse sentence from each group of sentences as compared to the MMR approach used earlier. The advantages of the introduced method are: it does not use external resource except the original document given summary and deep natural language processing is not required. This method has shown good performance when comparing with the benchmark methods used in the study.
For future work is to plan to incorporate artificial intelligence technique with the proposed method to improve the performance of the system.

References

[1] S. Brin, and L. Page, "The anatomy of a large-scale hyper textual Web search engine",Computer Networks and ISDN System, 30(1–7): 107–117, 1998.
[2] J.Carbon ell and J. Goldstei, "The use of MMR, diversity-based reran king for reordering Documents and producing summaries", SIGIR '98: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 24-28 August. Melbourne, Australia, 335-336, 1998.
[3] G. Erkan, and D. R. Radev. "LexRank: Graph-based Lexical Centrality as Salience in Text Summarization", Journal of Artificial Intelligence Research (JAIR), 22, 457-479, AI Access Foundation. 2004.
[4] K Filippova, M. Mieskes, V. Nastase, S. P. Ponzetto and M. Strube. "Cascaded Filtering for Topic-Driven Multi-

Document Summarization”, Proceedings of the Document Understanding Conference, 26-27 April. Rochester, N.Y., 30-35, 2007.
[5] M. K. Ganapathiraju, “Relevance of Cluster size in MMR based Summarizer: A Report 11-742: Self-paced lab in Information Retrieval”, November 26, 2002.
[6] “The Document Understanding Conference (DUC)”, http://duc.nist.gov.
[7] A. Jain, M. N. Murty and P. J. Flynn, “Data Clustering: A Review”, ACM Computing Surveys.31 (3), 264-323, 1999.
[8] C. Jaruskulchai and C. Kruengkrai, “Generic Text Summarization Using Local and Global Properties”, Proceedings of the IEEE/WIC international Conference on Web Intelligence, 13-17 October. Halifax, Canada: IEEE Computer Society, 201-206, 2003.
[9] A. Kiani –B and M. R. Akbarzadeh –T, “Automatic Text Summarization Using: Hybrid Fuzzy GA-GP”, IEEE International Conference on Fuzzy Systems, 16-21 July, Vancouver, BC, Canada, 977 -983, 2006.
[10] W. Kraaij, M. Spitters and M. v. d. Heijden, “Combining a mixture language model and naïve bayes for multi-document summarization”, Proceedings of Document Understanding Conference, 13-14 September, New Orleans, LA, 109-116, 2001.