Automatic Recapitulation of Text Document

Aditi Konge[1], Manali Sarkar[2], Rashmi Hatwar[3], Vrushali Jain

[1,2,3,4] *Department of Computer Technology, Yeshwantrao Chavan College of Engineering, Nagpur, Maharashtra, India*

Abstract - *The information over the internet is increasing very rapidly making it difficult to take out concise information. Users have to read the whole document to determine that the given document is relevant or not. Thus it is necessary to build a system that could present human quality summaries. Automatic Recapitulation of Text is a tool to reduce the document from its original size by presenting the main points in a concise form. Recapitulation process involves interpretation, transformation and generation. To generate an appropriate summary, the input text is pre-processed which involves tokenization, stop-words removal and stemming. The appropriate features are extracted from the input data, tf-idf values for each word are computed and all the pre-processed input is transformed into a tf-idf matrix. In the proposed extractive-based approach sentences are given scores based on different feature and sentences with higher rating are selected for meaningful summary. It uses various Natural Language processing approaches for information retrieval. The proposed approach states the methodology for generating the relevant summary using WordNet.*

Index Terms - Automatic Recapitulation of text, Natural Language Processing, WordNet, extractive-based approach, tf-idf.

I. INTRODUCTION

Automatic Recapitulation is the procedure by which the precise and required information of a text is retrieved from the input text. Recapitulation is an act of summarizing and restating the main points of the input text. Due to digitalization the amount of electronic information is increasing day by day, it becomes a time and space consuming matter to handle such huge volume of data.

Automatic Text Recapitulation plays a major role by generating relevant and specific information from a large amount of data. A good document summary frees the system user from the need to read and analyze all of the text documents, and give the opportunity to focus his attention on aspects of the rapid and effective decision. Automatic Text Recapitulation reduces the manual efforts, produces a shorter version of large text documents by selecting most relevant information and hence it takes over the role of manual summarization. Moreover, Automatic Recapitulation can be very useful for neighbouring Natural Language Processing (NLP) tasks, such as Information Retrieval, Question Answering or Text Comprehension, because these tasks can take advantage of the summaries to save time and resources.

There are two types of summarization: abstraction-based summarization and extraction-based summarization. In abstract-based summarization, an abstract is created by interpreting the text contained in the original input document and generating summary that express the same in a more precise and concise way. In extractive-based approach some sentences that are important to the summary are taken out from the input text to form a meaningful summary.

Regarding to the current facts, applying semantic analysis approach in text summarization may achieve certain level of accuracy. In this paper, a practical approach is proposed for extracting the most relevant sentences from the original document to form a summary. The idea of our approach is to find out key sentences from the Keyword extraction using WordNet.

II. RELATED WORK

The problem of automatic recapitulation have been widely discussed in many papers and practical solutions. Most early work on single-document summarization is that of (Luhn, 1958), that describes research done at IBM in the 1950s. In his work, Luhn proposed that the frequency of a particular word in an article provides an useful measure of its significance. There are several key ideas put forward in this paper that have assumed importance in later work on summarization. Edmundson (1969) describes a system that produces document extracts. His primary contribution was the development of a typical structure for an extractive summarization experiment.

The simplest method for creating summaries is based on the assumption that the weight of the sentence depends on the weight of its words, calculated on the basis of their frequency in the text. There are numerous researches in text mining area, specialized in the automatic text summarization. Years ago, pair of researchers listed the trends of Automatic Text Summarization over the years such as statistical approach, natural language processing (NLP), semantic analysis approach, fuzzy logic.

Although there has been increased attention to different criteria such as well-firmness, cohesion or coherence when dealing with summarization most work in this NLP task is still concerned with detecting relevant elements of text and presenting them together to produce a final summary. Despite the fact that many approaches have been developed, some important aspects of summaries, such as legibility,

grammaticality, responsiveness are still evaluated manually by experts.

III. VARIOUS SUMMARIZATION METHODS

A. Machine Learning Method

In the 1990s, with the advent of machine learning techniques in NLP, a series of seminal publications appeared that employed statistical techniques to produce document extracts. While initially most systems assumed feature independence and relied on naive-Bayes methods, others have focused on the choice of appropriate features and on learning algorithms that make no independence assumptions. Other significant approaches involved hidden Markov models and log-linear models to improve extractive summarization. A very recent paper, in contrast, used neural networks and third party features (like common words in search engine queries) to improve purely extractive single document summarization.

B. Naïve-Bayes Method

Kupiec et al. (1995) describe a method derived from Edmundson (1969) that is able to learn from data. The classification function categorizes each sentence as worthy of extraction or not, using a naive-Bayes classifier. Let s be a particular sentence, S the set of sentences that make up the summary, and F1; : : : ; Fk the features.

Assuming independence of the features:

$$P(s \in \mathcal{S} \mid F_1, F_2, ..F_k) = \frac{\prod_{i=1}^{k} P(F_i \mid s \in \mathcal{S}) \cdot P(s \in \mathcal{S})}{\prod_{i=1}^{k} P(F_i)} \qquad (1)$$

The features were compliant to (Edmundson, 1969), but additionally included the sentence length and the presence of uppercase words. Each sentence was given a score according to (1), and only the n top sentences were extracted. Aone et al. (1999) also incorporated a naive-Bayes classifier, but with richer features. They describe a system called Dim Sum that made use of features like term frequency (tf) and inverse document frequency (idf) to derive signature words. The idf was computed from a large corpus of the same domain as the concerned documents.

C. Neural Networks and Third Party Features

In 2001-02, DUC issued a task of creating a 100-word summary of a single news article. However, the best performing systems in the evaluations could not outperform the baseline with statistical significance. This extremely strong baseline has been analyzed by Nenkova (2005) and corresponds to the selection of the first n sentences of a newswire article. Some of the used features based on position or n-grams frequencies have been observed in previous work. However, the novelty of the framework lay in the use of features that derived information from query logs from Microsoft's news search engine7 and Wikipedia8 entries.

D. Statistical / IR based Approach

This approach includes exploiting word-frequency Information of the text. Initially frequency of each word is calculated. According to this approach high frequency words are related to the topic of the document. Of course, this does not include stop words. Importance of sentence depends on number of occurrences of significant words and discriminating power of the words.

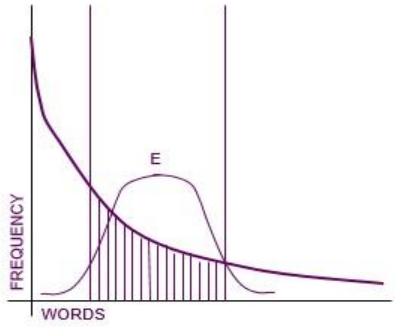

Figure 1. Resolving power of significant words

E. WordNet based Summarization

WordNet is a lexical word database with features of mapping similar words into synonym sets or synsets to denote semantic relationship between sets. Over the years, Princeton

University had managed to develop WordNet for English language containing large sets of words and classifications such as noun, verbs, adjective and adverb. WordNet contains words from the four syntactic word classes i.e. noun, verb, adjective and adverb. Except a connection between nouns and adjectives using attributes, there is generally no connection between the classes, so a comparison between different classes is not possible using WordNet. This means a similarity measure will only consist of noun-noun, verb-verb etc. comparisons. Semantic similarity has been computed using WordNet to perform graph searching. The results from the graph search is then fed into a method which computes the similarity. Relatedness between words can be determined by the use of hyponymy and meronymy. For example, a car and a steering wheel are related, it means the steering wheel is a meronym of car but they are not very similar. Semantic similarity only uses hyponymy to determine similarity. Since a car and a bicycle are both hyponyms of vehicle, they are considered similar.

IV. SYSTEM OVERVIEW OF PROPOSED APPROACH

The proposed work is the generation of extractive single document summarizer to improve the coherence of the summary text. Extractive approach works by selecting important sentences. Numerous methods are used for sentence selection. The most widely used method is sentence scoring method. This method assigns some numerical value to a sentence by summing all the feature values of the sentence.

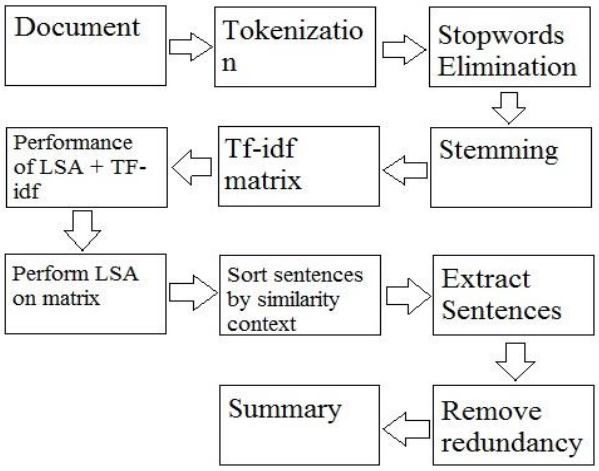

Figure 2. Representation of overall approach

Figure 2 shows the main phases of generating summary based on this proposed measurement such as Pre-Processing,

Feature Computation, Feature Ranking and Generating Summary. The document is pre-processed for eliminating the noises and wastes which exist in the original document so that the result from pre-processing phase can be processed for further phases. This phase consist of Sentence Extraction,

Tokenization, Stop Words Removal, Stemming. The further details will be described in Section 5.

V. PREPROCESSING FEATURES

A. Tokenization

Tokenization is the act of breaking up a sequence of strings into pieces such as words, keywords, phrases, symbols and other elements called tokens.Each sentence will be divided into number of tokens. Lexical analysis transforms the source document into a set of tokens. Non ASCII characters are removed since they do not carry implicit meaning. The final list of tokens is passed on to the next phase for further processing.

B. POS Tagging Module

The process of classifying words into their parts of speech and labelling them accordingly is known as part-of-speech tagging. POS tagging is done in the context of computational linguistics. For example, NN for singular nouns, NNS for plural common nouns. The POS tags and their description is given in figure 3.

Tag	Description
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
PDT	Predeterminer
POS	Possessive ending
PRP	Personal pronoun

Tag	Description
PRP$	Possessive pronoun
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	to
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Whdeterminer
WP	Whpronoun
WP$	Possessive whpronoun
WRB	Whadverb

Figure 3. POS tags with description

C. Removal of Stop words

These are the words which are strained from natural language sentences. Any group of words can be considered as stop words. Any word which does not contribute to the importance of a sentence can be removed and all those words are termed as stop words. Every word is compared to the group of stop words, which if matched is not considered as a new word in the vocabulary. So the vectors do not include stop words as a variable. 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.

D. Term Frequency-Inverse Document Frequency (tf/idf)

The weights of each word are calculated by using *tf/idf* values. Term frequency is the count of appearance of the word in the whole document. Inverse Document Frequency is a measure of the importance of the word based on the rarity of its occurrence. After finding both values of a word in a sentence, its value will be computed as a product, as shown in equation 2.

$$w_{i,j} = tf_{i,j} \times \log \frac{N}{df_i} \quad (2)$$

Where,

tfi,j : count of occurrences of i in j

dfi : count of documents containing i

N : total number of documents

Weights will be assigned to every word in a sentence.

After weight assignment, we have a n-dimension vector for every sentence. The coefficients of words are *tf/idf* values for words present in sentence.

E. Stemming

Stemming is a process of compressing affixed words using certain rules into its root form or so called stem. In this paper, stemming algorithm is used in word similarity measurement process for normalizing affixed words which have similar stem in order to calculate their base form similarity not based on the actual form. A human will easily observe that words such as run, runs, ran and running are all the different forms of the same word, i.e. they have the same base form, run. The WordNet database contains only the base form of each word, thus pre-processing is needed to transform every word to its base form.

F. Feature Extraction

Initially, the sentences are represented in the vectors of *n* dimension, where *n* is the count of words in the vocabulary of the document without the stop words. Each word is associated with a coefficient which represents the each word weight in the document. The collective weight of a word in a sentence defines the weight of the sentence. Sentence can be represented as a vector of n-dimension.

G. Cosine Similarity

After calculating the *tf/idf* values of the document, the similarity of sentences is to be identified with respect to every other sentence. To do this, we use the trigonometric function cosine. Every pair of vector in the *n*-dimensional space is related by angle θ (theta). Greater the angle lesser is the similarity of the sentence with each other. Cosine by definition varies from 1 to -1 as the angle varies from 0 to 180. Another easier way to find the cosine value of angles between the vectors is to find the dot product of the vectors and divide it by the product of their magnitudes.

H. Clustering

Once the cosine similarity between vectors is identified, group similar sentences so that sentences are picked from each group. To do this, one of the machine learning methods of clustering called k-means is used. In data mining the popular cluster analysis is K-means clustering. It is vector quantization method.

K-means clustering partitions n observations into k number of clusters. Based on the nearest mean value each observation fit in to the cluster. *K-means* is a unsupervised machine learning algorithm that takes number of clusters that has to be formed as input. Depending on the percentage of the summary required, the clusters numbers is calculated.

VI. WORD SIMILARITY ALGORITHM

The technique for computing a similarity value between two texts uses the similarity of the words in each text. Therefore, we need to be able to represent how closely the words relate. Some of the words which are presented to WordNet may not be present in the database, thus a similarity score for this word cannot be measured.

Result for Word Similarity Algorithm

Sr.	Tokens	Compare With	Score
2	relevance	approach	0.22
3	sentence	approach	0.12
4	text	approach	0
5	belong	approach	0.12
6	extract	approach	0.38
7	information	approach	0.18

Figure 4. Example Output of Word Similarity Algorithm

The figure 4 shows the individual similarity score of every token. After preprocessing of the text, every token is compared with the heading in the text and the similarity score is calculated.

To perceive the semantic value of documents or terms, word similarity measurement process should be adopted first. There are two main features for word similarity such as depth of words and Wu & Palmer measurement.

A. Depth of Words

To obtain semantic similarity of two words using synthetical WordNet, defining depth of those words is prioritized before further calculation. Depth of words are acquired by traversing the shortest distance paths from one word to another word using synonym sets or synsets graph. Essentially, the more paths from one word to another word are traversed, the more depth's value are given. The calculation is defined in formula (3):

$$depth(w_1, w_2) = min(paths(w_1, w_2)) \quad (3)$$

where wl and w2 are the words respectively; paths(wl , w2) is collection of paths value from w l and w2; min(paths (wl , w2)) is the minimum value of paths (wl , w2). For example,

the paths from word makan and minum are four (4), three (3) and six (6). Since the depth of words extracts the shortest distance path between two words, the value of depth (makan, minum) is three.

B. Wu & Palmer Measurement

Wu & Palmer calculates the similarity measurement of two concepts by enumerating the depths of those concepts in WordNet taxonomy. The measurement includes depth of Least Common Subsumer (LCS) and the depths of the respective words. The formula is defined as follows in formula (4):

$$sim_{wp}(w_1, w_2) = \frac{2 * depth(lcs(w_1, w_2)}{depth(w_1, w_2) + depth(w_2, w_1)} \quad (4)$$

Least Common Subsumer (LCS) is a shortest distance of two concept compared in lexical taxonomy. In this paper, we use our raw self implementation of WordNet which is considered as simple and incomplete. Thus, since the taxonomy relationship of our WordNet is still raw, we consider the depth of LCS is one as the depth of a root in taxonomy.

Result for Semantic Matrix

-	Promot	development	computerization
Promot	1	0.27	0.2
development	0.27	1	0.07
computerization	0.2	0.07	1

Figure 5. Example of Semantic vector derivation process

In the sematic vector derivation process each token in the input text is compared with every token. After completion of score computation, the average score is calculated by adding all the word similarity scores. The token which has score less than the average score are eliminated from the summary. Hence only the most relevant words are included in the final summary.

VI. CONCLUSION AND FUTURE SCOPE

An automatic extractive recapitulation tool should produce the output summary quickly with minimum redundancy or no redundancy. Two types of techniques can be used for the final summary evaluation. They are intrinsic evaluation and extrinsic evaluation. The major focus of the project was on relevance. The output summary produced by the recapitulation tool should be more relevant to the input text document provided which is achieved.

For the future work, the recapitulation can be performed on multi document inputs. The challenge in multi document recapitulation is that the information overlaps between different documents which makes the task difficult.

VII. REFERENCES

1. H. P. Edmundson; “New methods in automatic extracting,” in journal of the ACM, pp: 264-285, 2008.

2. Fang Chen, Kesong Han and Guilin Chen; “An Approach to sentence selection based text summarization," in proceedings of IEEE TENCON02, pp: 489-493, 2002.

3. Joel larocca Neto, Alex A. Freitas and Celso A.A.Kaestner; "Automatic Text Summarization using a Machine Learning Approach,” in proceedings of Advances in Artificial Intelligence: Lecture Notes in computer science book, vol 2507/2002, pp: 205-215, 2002.

4. Baxendale, P. B.; "Machine-Made Index for Technical Literature-An Experiment," in proceedings of IBM Journal of Research & Development, pp: 354-361, 1958.

5. Hovy, E.; “Text Summarization,” in proceedings of The Oxford Handbook of Computational Linguistics, Oxford University Press, pp: 583–598, 2005.

6. C. Jaruskulchai and C. Kruengkrai; “Text Summarization Using Local and Global Properties,” in proceedings of the IEEE/WIC International Conference on web Intelligence, Canada: IEEE Computer Society, pp: 201-206, 2003.

7. A. Kiani –B and M. R. Akbarzadeh –T; “Automatic Text Summarization Using: Hybrid Fuzzy GA-GP,” in proceedings of IEEE International Conference on Fuzzy Systems, Canada, pp: 977 -983, 2006.

8. Luhn, H.P; “The automatic creation of literature abstracts,” in proceedings of Advances in Automatic Text Summarization, MIT Press, pp: 15–22, 2005.

9. M. Hassel; "Evaluation of Automatic Text Summarization: A practical implementation," in proceedings of Licentiate Thesis, University of Stockholm, 2004.

10. S. Banerjee, T. Pedersen; "An adapted Lesk algorithm for word sense disambiguation using WordNet," in proceedings of the Third International Conference on Intelligent Text Processing and Computational Linguistics, pp:136-145, 2002.

11. Lin, C.Y., Hovy, E.; “Automatic evaluation of summaries using n-gram co-occurrence statistics,” in proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology, pp: 71–78, 2003.