Extractive Summarization of a Document Using Lexical Chains

Chapter *in* Advances in Intelligent Systems and Computing · August 2019

DOI: 10.1007/978-981-13-0514-6_78

5 authors, including:

Chirantana Mallick
Indian Institute of Engineering Science and Technology, Shibpur

Madhurima Dutta
Indian Institute of Engineering Science and Technology, Shibpur

Ajit Kumar Das
Indian Institute of Technology Kharagpur

Apurba Sarkar
Indian Institute of Engineering Science and Technology, Shibpur

Extractive Summarization of a Document using Lexical Chains

Chirantana Mallick *, Madhurima Dutta **, Ajit Kumar Das, Apurba Sarkar, and Asit K Das

Department of Computer Science and Technology,
Indian Institute of Engineering Science and Technology, Shibpur, India
chirantana9@gmail.com, madhurima.pg2016@cs.iiests.ac.in,
writetoajit@yahoo.com, as.besu@gmail.com, akdas@cs.iiests.ac.in

Abstract. Nowadays, efficient access of information from the text documents with high-degree of semantic information have become more difficult due to diversity of vocabulary and rapid growth of the Internet. Traditional text clustering algorithms are widely used to organise a large text document into smaller manageable groups of sentences but it does not consider the semantic relationship among the words present in the document. Lexical Chains try to identify cohesion links between words by identifying their semantic relationship. They try to link words in a document that are thought to be describing the same concept to gather information. This method of text summarization helps to process the linguistic features of the document which is otherwise ignored in statistical summarization approaches. In this paper, we have proposed a text summarization technique by constructing lexical chains and defining a coherence metric to select the summary sentences.

Keywords: summarization, extractive summary, lexical chains, hyponym, hypernym

1 Introduction

In the internet era, enormous amount of online information are available to readers in the form of e-newspapers, content from various news feeds, journal articles, transcription reports, social media streams, emails, etc. It is a tedious job for any individual to extract meaningful information from a large amount of text let alone multiple documents in a stipulated time. Hence, there is a need for automatic summarization system to digitally retrieve information from a vast number of online and offline sources in the form of a summary. A summary is a piece of text, containing a major segment of information compiled from one or more documents. Automatic text summarization has garnered a lot of interest in the recent years. It is the ability to encapsulate information automatically and present results to the end user in a compressed, yet complete form.

* Corresponding Author
** Corresponding Author

Summarization systems provide two canonical strategies namely, Extractive Summarization [24] and Abstractive Summarization [8] for constructing summaries from single source or multi source documents. The main challenge of "Extractive Summarization" is to select important sentences from the original document into the summary without any modification. It consists of concatenating source sentences into a summary. "Abstractive summarization" techniques tend to generate novel sentences based on information gathered from the document. These techniques are much harder to implement than extractive summarization techniques in general. Apart from these, several systems try to merge these concepts for providing summaries which are both extractive and abstractive in nature.

Modern extractive summarization systems work in three steps. The first step involves the construction of an intermediate representation of the original text. The second step is to analyze the intermediate representation and define a metric as a means of scoring the sentences. The score is commonly related to how well a sentence expresses some of the most important topics in the document or to what extent it combines information about different topics. Hence, in this step, the important sentences get classified. The third step involves selection of coherent sentences for construction of the summary. This step follows a greedy approach and generally involves selection of the top predefined number of weighted sentences.

A plethora of methods [5] [17] [26] have been employed to create an intermediate representation of the original document. There are "statistical approaches" which are employed to find word frequency, theme words, etc. Statistical approaches include identification of "topic words" or "topic signatures" [19] which are words directly related to the theme of a particular document. Metrics like term frequency and inverse document frequency, TF-IDF [28] also contribute to summarization systems. Centroid Summarization [25], used by open source multidocument multilingual text summarizer and popularly known as MEAD system works by computing the centroid of the document. Other than these, entropy, mutual information and other statistical models are used to find important terms or words in sentences. Emphasis is also given to dialogues, capitalized words, cue phrases and so on. In query focused summarization techniques [30], these metrics have been proved to produce meaningful results.

There are "semantic approaches" like lexical chaining [29], word sense disambiguation [12] which try to establish relationships between words or sentences and hence leads to partial understanding of the document. These are also known as "linguistic approaches". Several graph-based models like Pagerank [20], Lexrank [10], are also thought of as semantic approaches for text summarization. There are several similarity metrics like cosine similarity, Levenshtein distance [9], are used to compute similarity between pairs of sentences. Hence, in case of graph models, sentences are taken as vertices and the edge between two vertices bears the similarity measure as weight.

Other approaches include designing probabilistic models [6] to assign weightage to terms and sentences in general to determine the suitable sentences to go into the summary. Several systems also use Neural networks [16], Bayesian topic models [14], Markov chains [23] to generate summaries where documents are to be extracted from multiple sources. Over the years, research in this field has picked up pace and a lot of methods have been defined. In this paper, a summarization method has been defined using Lexical Chains to make a summary of a single document.

The rest of the paper is organised in the following manner. The concept of Wordnet and Lexical chains is thoroughly discussed in Section 2. The proposed method is discussed in Section 3. The results obtained are depicted in Section 4 and finally future work and conclusion are drawn in Section 5.

2 Background Study

2.1 Wordnet

WordNet [4] organizes lexical information in terms of word meanings, rather than word forms. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are linked by semantic relations [21] i.e. synonyms of each word, and also hyponym/hypernym (i.e. Is-A), and meronym/holonym (i.e. Part-Of) relationships and word forms are linked by lexical relations. The most frequently encoded relation among synsets is the super-subordinate relation (also called hypernymy, hyponymy or IS-A relation) [11].

The synsets provide a hierarchical tree-like structure for each term. WordNet in document clustering is used to improve the clustering performance for word sense disambiguation by Hotho et al. (2003) [13] and this work is extended by Sedding and Kazakov (2004) [27] using their part of speech tags. But the main bottleneck of both these approaches is the increase in dimensionality of the data. WordNet is a large lexical database of English organized by semantic relations. Suppose there is a semantic relation X between meaning $\{a_1, a_2, \ldots\}$ and meaning $\{b_1, b_2, \ldots\}$, then there is a relation X between $\{b_1, b_2, \ldots\}$ and $\{a_1, a_2, \ldots\}$. Lexical chain [15] is constructed by calculating the semantic distance between the words using WordNet. Lexical relationship exists between words while semantic relationship occurs between the concepts synsets represent. The relationship between canine and dog is an example of a semantic relationship. Wordnet precisely expresses this as hyponym from the synset containing the words canine, canid to the synset with the words dog, domestic dog, Canis Familiaris. Antonymy is an example of a lexical relationship. Dark is an antonym of light; however darkness and lightness, which belong to the same synsets as dark and light respectively, are not antonyms. Wordnet provides semantic and lexical relationship between noun synsets and lexical relationships:

1. Semantic Relationships
   (a) Hypernyms/Hyponyms: B is a hypernym of A if every A is a (kind of) B (eg. canine is a hypernym of dog) then A is a hyponym of B (dog is a hyponym of canine)

(b) Holonyms/Meronyms: B is a meronym of A if B is a part of A (window is a meronym of building) and A is a holonym of B (building is a holonym of window).
   i. Component-Object: root is a component of tree.
   ii. Member-Collection: bird is a member of a Flock.
   iii. Material-Object: Aluminium is the material the object airplane is made of.
2. Lexical Relationships
   (a) Synonyms: Synonym of a word is such that the other word is exactly the same in meaning. For example, movie is synonymous to cinema.
   (b) Antonyms: Synonym of a word is such that the other word is exactly the opposite in meaning. For example, shut is an antonym of open.

An example of wordnet relationship is depicted in Fig. 2.1.

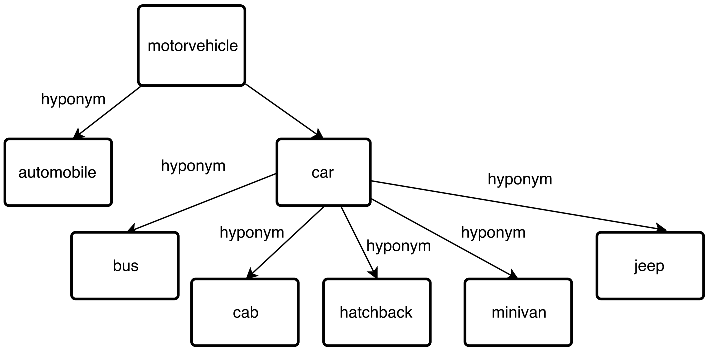

Fig. 1. Wordnet Hierarchy

.

2.2 Lexical Chaining

The various chaining algorithms used in text summarisation where their implementations are different. In general, they are characterised by certain rules based on how they perform chaining as well as discover chains. The idea of lexical chaining was first implemented by Morris and Hirst (1991) [22]. It mainly deals with the problem of word sense disambiguation (WSD). Lexical chains are created based on the same topic words of the document. In this approach, identities, synonyms, and hypernyms or hyponyms are the relations among words which cause them to be grouped into the same lexical chain and for this purpose WordNet is widely used in this method. Lexical chain is constructed by determining the semantic distance between the words using WordNet. There are four common steps are performed to construct desired summary using lexical chain.

1. **Segmentation of the original source text**: At first each sentence is identified by tokenisation from the input text document.
2. **Construction of lexical chains**: Lexical chain is constructed in three different steps. First, we select a set of candidate words and then for each candidate word find an appropriate chain relying on a relatedness criterion (like identities, synonyms, and hypernyms or hyponyms) among the other members of the chains. If any similar candidate word with certain relatedness criterion is found, insert it into the same chain and update the score of the chain accordingly.
3. **Identification of strong chains**: To identify the strong chain first the overall score of each chain is determined by adding the weights of each individual element in the chain. The chains with score more than the mean score for every chain in the document are selected and for each chain identify the representative word whose contribution to the chain is maximum.
4. **Extraction of significant sentences**: Finally pick the sentence that contains the first appearance of representative word in the text document.

Overall, lexical chains provide a better indication of discourse topic than does word frequency simply because different words may refer to the same topic.Even without sense disambiguation, this approach are able to derive concepts like war, campaign, warfare, effort, cause, operation, conflict, concern, carrier, worry, fear, scare or home, base, source, support, backing.

Consider the following sentences: "Sophia Loren says she will always be grateful to Bono. The actress Sophia revealed that the U2-singer, Bono helped her calm down when she became scared by a thunderstorm while travelling on a plane." Fig. 2.2 depicts the chain obtained from the sentences.

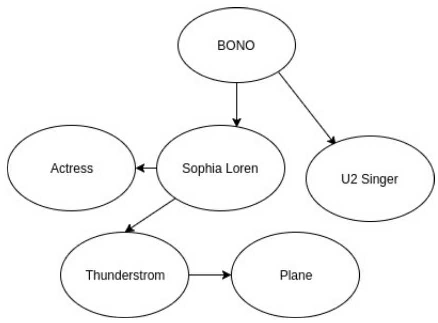

Fig. 2. Lexical Chain formed from the given sentences.

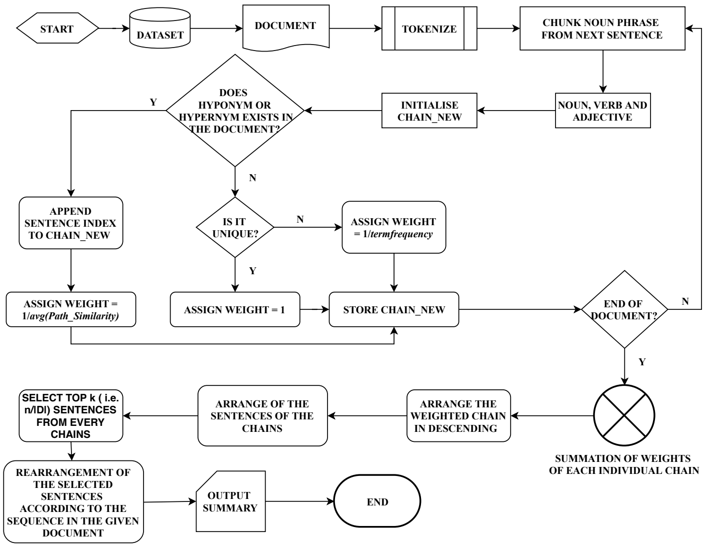

Fig. 3. Flowchart of the proposed summarization method

3 Proposed Methodology

3.1 Preprocessing

We have used news articles from BBC News Feed. The BeautifulSoup library [1] of Python [3] is used as the web scraping tool of our work. This library helps in extracting only the text portion of the HTML or XML document. The articles are successfully read into the program, in the form of a single piece of text or string. The texts of the document D are tokenized into individual sentences $\{S_1, S_2, \ldots S_n\}$ and each of them is Part of Speech tagged (i.e. CC, DT, JJ, NN, NNP, NNS, VB, VBN, VBG, RB; where CC is coordinating conjunction, DT is determiner, JJ is adjective, NN is singular noun, NNP is proper noun, NNS is noun plural, VB is verb, VBN is verb past participle, VBG verb past tense, RB is adverb) using the NLTK (Natural Language Processing Toolkit) library [7] of Python. After Part of Speech tagging, the stop words are removed from the sentences $S_i \in D$ $\forall i = 1$ to n. Thus, after removing stop words, the processed sentences will be of the form $i.e. S_i = \{w_i(CC) \cup w_i(DT) \cup w_i(JJ) \cup w_i(NNP) \cup w_i(NNS) \cup w_i(VB) \cup w_i(VBN) \cup w_i(VBG) \cup w_i(RB)\} \cap \neg STW$, where S_i is the ith sentence of the given document D and $w_i(X)$ is the word present in the sentence S_i with X tagging and STW is the set of all stop words of the document obtained using NLTK library. The NLTK library tags the words using the same tags as of Penn Treebank [2].

The flowchart of the proposed summarization technique is shown in Fig. 3.

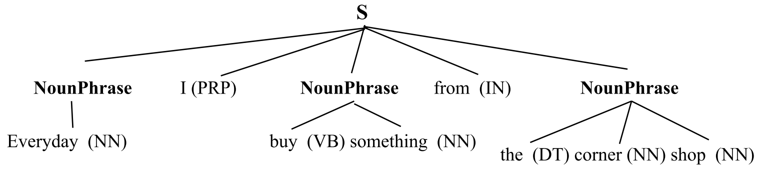

Fig. 4. Chunking a sentence into its noun phrases

3.2 Lexical Chain based summarization

In this proposed method, we have considered semantic relationship between the words within a sentence as well as among the sentences of the given document. After preprocessing of the sentences, with the help of regular expressions, we chunk out the noun phrases (w(NP)) from the document and subsequently the document is represented only by means of the noun phrases. We define noun phrases by using Expression (1), where the proper nouns are accumulated together.

$$\text{“} < DT > * < JJ.? > * < CC > * < NN.? > * < VB.? > * < NN.? > \text{”} \quad (1)$$

where DT refers to Determiners, JJ refers to adjectives, CC refers to Conjunctions, NN refers to Nouns, and VB refers to Verbs.

For the sentence, “Everyday I buy something from the corner shop.”, Fig. 4 shows the chunked grammars.

A hypernym-hyponym set (H) is constructed with the words having noun (NN) , verb (VB) and as well as adjective (JJ) tag of the sentence in the document as follows:

$$H = \{hr(w(NN)) \cup hp(w(NN)) \cup hr(w(VB)) \cup hp(w(VB)) \cup hr(w(JJ)) \cup hp(w(JJ)) : w(NN), w(VB), w(JJ) \in S_i \forall i\}$$

where $hr(w(NN)$ is the set of hypernyms of the words having noun tag and $hp(w(NN)$ is the set of hyponyms of the words having noun tag and so for the verbs. Next, we have assigned weights for the nouns, verbs and the adjectives of every sentences. For weight assignment we have considered the following conditions:

i If the word $w_i \in H$, then set the weight $(wt(w_i))$ as $1/avg(\sum$ path_similarity of between every possible pair of words $\in H)$, where avg is the notation for average. The weights of the words $\in H$ are less if the path_similarity between their hypernyms and hyponyms is high (i.e. the words with less uniqueness).

ii If the word w_i is unique (i.e term frequency $(tf(w_i))$ is 1), then set the weight $(wt(w_i))$ as 1.

iii If the term frequency $(tf(w_i)) > 1$, set the weight $(wt(w_i))$ as $1/tf(w_i))$.

Then a set of lexical chains is formed with nouns and verbs of the processed sentences along with their weights as shown below:

$$chain_i = \{w_i(VB), wt_(w_i)(VB)\} \cup \{w_i(NN), wt_(w_i)(NN)\} \cup \{w_i(JJ), wt_(w_i)(JJ)\}$$

$$\forall i = 1, 2, \ldots, n.$$

Algorithm 3.1: MULTISENTENCE-LEXICAL-CHAINING(D, n)

Input: D= the target text document and n= size of the summary .
Output: Summary of the document D.
1. Encode D into D' using UTF-8 format;
2. D' is tokenized into individual sentences (S_i) using the NLTK library;
3. The sentences are Part-of-speech tagged using the tags of Penn Treebank;
4. Chunk each sentences by means of grammar using Expression (1);
5. Extract Hypernyms, Hyponyms for each noun of each processed tokenized sentence and construct a set of Hypernyms, Hyponyms ;
6. Initialise *index_value* value at zero for the sentence index;
7. **for each** $S_i \in D'$ in order of appearance in encoded D **do**
   7.1. Remove the stop words from the sentence;
   7.2. Increase the *index_value* by 1;
   7.3. Assign the *index_value* to the sentence;
8. If similarity between the nouns of other sentences is found:
   Append those sentence indices to the same chain;
9. Else
   Construct a chain with that sentence index;
10. **for each** word w in D' **do**
   10.1. If the word is present in the Hypernyms, Hyponyms set:
      Assign weight of w as $1/avg(\sum$ path_similarity of between every possible pair of words $\in H$), where avg is the notation for average;
   10.2. If the word is unique in the given document:
      Assign weight of w as 1;
   10.3. Else
      10.3.1. Calculate TF for that word w in D';
      10.3.2. Assign weight of w as $1/TF(w)$;
11. Calculate overall weight for each chain;
12. Arrange the weighted chain in descending order in a list;
13. **for each** chain in the list **do**
   13.1. Arrange the sentences of the chain in descending order according to their weights;
   13.2. Store top k no of the sentences from the chain in summary, where k is the $n/|D|$ of the length of the chain ;
14. Rearrange the sentences in order of their occurrence in D in summary;
return (*summary*)

After this, the relationships between the words within the sentences are examined. This is done for every noun and verb from the noun phrases for their hypernyms and hyponyms if they are present in the passage. For every possible pair of sentences (S_i, S_j), we have examined the noun , verb and adjective from the noun phrases $\in (S_i, S_j)$ for matching of their hypernyms and hyponyms and if any matching is found, construct a chain by appending their corresponding sentence indices. For an example, if matching is found between sentences S_i and S_j of the document D then the chain contains i and j. In this way, all the noun phrases of all the sentences are examined and some of them are chained until and unless no pair of sentences is left for processing . In the next step, the weights of the individual chains are summed up and sorted in descending order. At the end, the top k highest weighted sentences from each chain are picked, where k is the $n/|D|$ (i.e. n is the summary size and $|D|$ is the no of sentences in the document

D) of the length of each chain until the length of the summary is reached. Here, we have considered every chain instead of only the highest weighted chain. There is a chance that similar type(lexically similar) of sentences can be included in the summary if we only consider the highest weighted chain whereas we can cover much more information from the given document by considering every chain in this proposed method. The final summary is obtained after arranging the selected sentences in order of their occurrence in the original document. The algorithm of the proposed method is depicted in Algorithm 3.1.

4 Results

The method has been thoroughly tested on the news articles generated by the BBC News feed. We have used the BeautifulSoup library [1] and the NLTK library [7] of Python 2.7 [3] for extracting and preprocessing of the document.

Evaluation using Rouge: Since early 2000s Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [18] is widely used for automatic summaries and also for the performance evaluation. For the first time it is used in DUC 2004. Some of the ROUGE methods i.e. ROUGE_N, ROUGE_L, ROUGE_W and ROUGES_U are commonly used to measure the performance. ROUGE_N is the N-gram ($N \geq 1$) recall between a system summary and a set of sample summaries. It is used to estimate the fluency of summaries. Here, the value of N is considered as 1 and 2. ROUGE_1 and ROUGE_2 denote the overlap of 1-gram and bi-grams between the system and sample summaries respectively.

Rouge L is the longest subsequence based statistics which is used to identify the longest co-occurring in sequence n-grams automatically. Suppose we assume that A is the set of sentences of the reference summary and B is the set of sentences of the candidate summary represented by the sequence of words and LCS based f-score indicates the similarity between A (of length m) and B (of length n).

The unbiased ground truth or the reference summaries of the news articles were obtained from our fellow peers and research scholars having different field expertise. The evaluation of our proposed summarization technique is done by comparing it with the ground truth summary, as listed in Table. 1. The size of the summary is set as 25% of the total sentences in the document and compared with the ground truth summary in terms of Recall, Precision and f-score. In general 25% size of the original document is preferable as an effective summary size. It is noticed that summary size contains 25% sentences of the document and ground truth summary sizes are almost same. This proves that the proposed technique provides the summary which is almost close to that obtained by the experts. In Table. 1, the result is compared with some state of the art methods that are widely available and with the summary obtained by the work presented in [28] which shows the superiority of the method.

Summarisaton Technique		Rouge-1			Rouge-2			Rouge-L		
		Recall	Precision	f-score	Recall	Precision	f-score	Recall	Precision	f-score
Existing Summarization Method	Method1	0.690	0.344	0.417	0.380	0.183	0.227	0.558	0.300	0.390
	Method2	0.610	**0.540**	0.170	0.330	0.300	0.310	0.380	0.340	0.360
	Method3	0.462	0.308	0.388	0.130	0.100	0.110	0.350	0.290	0.210
	Method4	0.503	0.210	0.360	0.120	0.060	0.280	0.350	0.200	0.250
	Method5	0.520	0.511	0.305	0.380	**0.520**	0.342	0.472	**0.437**	0.442
	Method6	0.501	0.338	0.420	0.170	0.120	0.140	0.340	0.250	0.280
	Method7	0.652	0.512	0.421	0.470	0.367	0.475	0.507	0.467	0.449
Proposed Work	lexical_chain	**0.751**	0.452	**0.579**	**0.490**	0.328	**0.422**	**0.516**	0.413	**0.478**

Table 1. Rouge values of Input summaries obtained from some already implemented summarization techniques and our proposed summarisation methods with respect to ground truth

5 Conclusion and Future Work

This paper is based on the semantic relationship among the sentences of the given document using WordNet and chaining multiple sentences lexically. We have also identified cohesion links between words by identifying their semantic relationship. In this paper, the word sense disambiguation of the given document is handled efficiently and provided with more suitable summary preserving the central idea of the original document. The results of applying the proposed approach on extractive summarization are quite promising.

But the proposed algorithm does not take care of anaphora resolution problem. We can include this for better improvement. The proposed method may select more than one similar type of sentences with high score is selected for the summary. To eliminate this kind of duplication we may consider any kind of clustering algorithm. As an extension of the paper, we will compare our method with existing state-of-art text summarization methods evaluating different performance measurement metrics.

References

1. Beautifulsoup documentation. `https://www.crummy.com/software/BeautifulSoup/bs4/doc/`. Accessed: 2017-11-29.
2. Penn treebank pos tags. `https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html`. Accessed: 2017-12-30.
3. Python 2.7.14 documentation. `https://docs.python.org/2/index.html`. Accessed: 2017-11-29.
4. Wordnet. `http://wordnet.princeton.edu/`. Accessed: 2017-12-30.
5. Nitin Agrawal, Shikhar Sharma, Prashant Sinha, and Shobha Bagai. A graph based ranking strategy for automated text summarization. *DU Journal of Undergraduate Research and Innovation*, "1"(1), 2015.

6. Regina Barzilay and Lilian Lee. Catching the drift: Probabilistic content models, with applications to generation and summarization. volume 34, pages 1–34, March 2008.
7. Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly, 2009.
8. V. Dalal and L. Malik. A survey of extractive and abstractive text summarization techniques. In *2013 6th International Conference on Emerging Trends in Engineering and Technology*, pages 109–110, Dec 2013.
9. Soumi Dutta, Sujata Ghatak, Moumita Roy, Saptarshi Ghosh, and Asit K Das. A graph based clustering technique for tweet summarization. In *2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions)*, pages 1–6. IEEE, 2015.
10. Gne Erkan and Dragomir R. Radev. Lexrank: graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, pages 457–479, 2004.
11. Christiane Fellbaum. *WordNet*. Wiley Online Library, 1998.
12. Abhishek Ghose. Supervised lexical chaining. Master's thesis, Indian Institute Of Technology, Madras., 2011.
13. Andreas Hotho, Steffen Staab, and Gerd Stumme. Ontologies improve text document clustering. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 541–544. IEEE, 2003.
14. Hal Daumé III. Bayesian query-focused summarization. *CoRR*, abs/0907.1814, 2009.
15. Aakanksha Jain and Atul Gaur. Summarizing long historical documents using significance and utility calculation using wordnet. *Imperial Journal of Interdisciplinary Research*, 3(3), 2017.
16. K. Kaikhah. Automatic text summarization with neural networks. In *Intelligent Systems, 2004. Proceedings. 2004 2nd International IEEE Conference*, volume 1, pages 40–44 Vol.1, June 2004.
17. Thomas K Landauer, Peter W. Foltz, and Darrell Laham. An introduction to latent semantic analysis. *Discourse Processes*, 25(2-3):259–284, 1998.
18. Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Proceedings of the ACL Workshop: Text Summarization Braches Out 2004*, page 10, 01 2004.
19. Chin-Yew Lin and Eduard Hovy. The automated acquisition of topic signatures for text summarization. In *COLING '00 Proceedings of the 18th conference on Computational linguistics*, pages 495–501. Association for Computational Linguistics Stroudsburg, PA, USA 2000, 2000.
20. Rada Mihalcea. Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 20. Association for Computational Linguistics, 2004.
21. George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.
22. Jane Morris and Graeme Hirst. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational linguistics*, 17(1):21–48, 1991.
23. Ani Nenkova, Sameer Maskey, and Yang Liu. Automatic summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts of ACL 2011*, HLT '11, pages 3:1–3:86, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

24. Ani Nenkova and Kathleen McKeown. *A Survey of Text Summarization Techniques.* Springer Science+Business Media, 2012.
25. Dragomir R. Radev, Hongyan Jing, Malgorzata Stys, and Daniel Tam. Centroid-based summarization of multiple documents. *Information Processing and Management*, 40:919–938, 2003.
26. Horacio Saggion and Guy Lapalme. Generating indicative-informative summaries with sumum. *Computational linguistics*, 28(4):497–526, 2002.
27. Julian Sedding and Dimitar Kazakov. Wordnet-based text document clustering. In *proceedings of the 3rd workshop on robust methods in analysis of natural language data*, pages 104–113. Association for Computational Linguistics, 2004.
28. Yohei Seki. Sentence extraction by tf/idf and position weighting from newspaper articles. 2002.
29. Tingting Wei, Yonghe Lu, Huiyou Chang, Qiang Zhou, and Xianyu Bao. A semantic approach for text clustering using wordnet and lexical chains. *Expert Syst. Appl.*, 42(4):2264–2275, March 2015.
30. Lin Zhao, Lide Wu, and Xuanjing Huang. Using query expansion in graph-based approach for query-focused multi-document summarization. *Information Processing and Management*, 45(1):35 – 41, 2009.