International Journal for Modern Trends in Science and Technology
ISSN: 2455-3778 |Volume No: 02 | Issue No: 02 | February 2016

Keyphrase Extraction using Neighborhood Knowledge

Vonguri Spandana[1] and Dr.Srinivasu Badugu[2]

[1]PG Scholar, Department of Computer Science & Engineering, Stanley College of Engineering and Technology for Women (Affiliated to Osmania University), Hyderabad, Telangana, India

[2] Associate Professor, Department of Computer Science & Engineering, Stanley College of Engineering and Technology for Women (Affiliated to Osmania University), Hyderabad, Telangana, India

Article History

Received on:	30-01-2016
Accepted on:	05-02-2016
Published on:	16-02-2016

Keyword
Neighborhood
TF-IDF
POS tagging
Stemming
JTEXPRO

ABSTRACT

This paper focus on keyphrase extraction for news articles because news article is one of the popular document genres on the web and most news articles have no author-assigned keyphrases. Existing methods for single document keyphrase extraction usually make use of only the information contained in the specified document. This paper proposes to use a small number of nearest neighbor documents to provide more knowledge to improve single document keyphrase extraction. Experimental results demonstrate the good effectiveness and robustness of our proposed approach. According to experiments conducted on several documents and keyphrases we consider top 10 keyphrases are most suitable keyphrases.

I. INTRODUCTION

Key phrases are words or group of words that capture the key ideas of document. Automatic keyphrase extraction is automatic selection of important keyphrases in the document. Appropriate key phrases can serve as a highly condensed summary for a document and they can be used as a label for the document to supplement or replace the title or summary, or they can be highlighted within the body of document to facilitate users fast browsing and reading. Key phrases extraction has huge arena. Document key phrase have been successfully used in the following IR and NLP tasks. Document indexing, document classification, document clustering and summarization. Key phrases are usually manually assigned by authors, especially for journal or conference articles. But since document corpus is increasing rapidly, automation of identification of key phrase process is necessary. Generated keyphrases do not necessarily appear in the body of the document. Though the keyphrases play a vital role in information extraction, only minority of documents are available with keyphrases. Some commercial software products use automatic keyphrase extraction. Example: word97 and Tetranet. Most previous works focus on keyphrase extraction for journal or conference articles, while this work focus on keyphrase extraction for news articles because news article is one of the popular document genres on the web and most news articles have no author-assigned keyphrases.

II. MOTIVATION

Existing methods conduct the keyphrase extraction task using only the information contained in the specified document, including the phrase's TFIDF, position and other syntactic information in the document. One common assumption of existing methods is that the documents are independent of each other. And the keyphrase extraction task is conducted separately without interactions for each document. However, some topic-related documents actually have mutual influences and contain useful clues which can help to extract keyphrases from each other. For example, two documents about the same topic "politics" would share a few common phrases, e.g. "government", "political party", and they can provide additional knowledge for each other to better evaluate and extract salient keyphrases from each other. Therefore, given a specified document, we can retrieve a few documents topically close to the document from a large corpus through search engines, and these neighbor documents are deemed beneficial to evaluate and extract keyphrases from the document because they can provide more knowledge and clues for keyphrase extraction from the specified document.

III. LITERATURE SURVEY

Several papers explore the task of producing a summary of a document by extracting key sentences from the document. This task is similar to the task of keyphrase extraction, but it is more difficult. The extracted sentences often lack cohesion because anaphoric references are not resolved. Anaphors are pronouns (e.g., "it", "they"), definite noun phrases (e.g., "the car"), and demonstratives (e.g., "this", "these") that refer to previously discussed concepts. When a sentence is extracted out of the context of its neighboring sentences, it may be impossible or very difficult for the reader of the summary to determine the referents of the anaphors. Keyphrase extraction avoids this problem because anaphors (by their nature) are not keyphrases. Also, a list of keyphrases has no structure; unlike a list of sentences, a list of keyphrases can be randomly permuted without significant consequences.
Most of these papers on summarization by sentence extraction describe algorithms that are based on manually derived heuristics. The heuristics tend to be effective for the intended domain, but they often do not generalize well to a new domain. Extending the heuristics to a new domain involves a significant amount of manual work. A few of the papers describe learning algorithms, which can be trained by supplying documents with associated target summaries. Learning algorithms can be extended to new domains with less work than algorithms that use manually derived heuristics. However, there is still some manual work involved, because the training summaries must be composed of sentences that appear in the document, which means that standard author-supplied abstracts are not suitable. An advantage of keyphrase extraction is that standard author-supplied keyphrases are suitable for training a learning algorithm, because the majority of such keyphrases appear in the

bodies of the corresponding documents.

Another related work addresses the task of information extraction. An information extraction system seeks specific information in a document, according to predefined template. The template is specific to a given topic area. For example, if the topic area is news reports of terrorist attacks, the template might specify that the information extraction system should identify (i) the terrorist organization involved in the attack, (ii) the victims of the attack, (iii) the type of attack (kidnapping, murder, etc.), and other information of this type where information extraction systems are evaluated with corpora in various topic areas, including terrorist attacks and corporate mergers.

Information extraction and keyphrase extraction are at opposite ends of a continuum that ranges from detailed, specific, and domain-dependent (information extraction) to condensed, general, and domain-independent (keyphrase extraction). The different ends of this continuum require substantially different algorithms. However, there are intermediate points on this continuum. An example is the task of identifying corporate names in business news

IV. PROPOSED SYSTEM

In order to extract keyphrases from neighborhood we need to identify neighborhood documents by using following

A. *Cosine similarity:*

The cosine similarity between two vectors (or two documents on the Vector Space) is a measure that calculates the cosine of the angle between them. This metric is a measurement of orientation and not magnitude; it can be seen as a comparison between documents on a normalized space. What we have to do to build the cosine similarity equation is to solve the equation of the dot product for $\cos(\theta)$: Cosine Similarity will generate a metric that says how related are two documents by looking at the angle instead of magnitude.

The cosine of two vectors can be derived by using Euclidean dot product formula: $a.b = \|a\|\|b\|\cos(\theta)$

(1)

Given two vectors of attributes, A and B the cosine similarity, cos(θ), is represented using a dot product and magnitude as

$$Cos(\theta) = \frac{A.B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(B_i)^2}} \qquad (2)$$

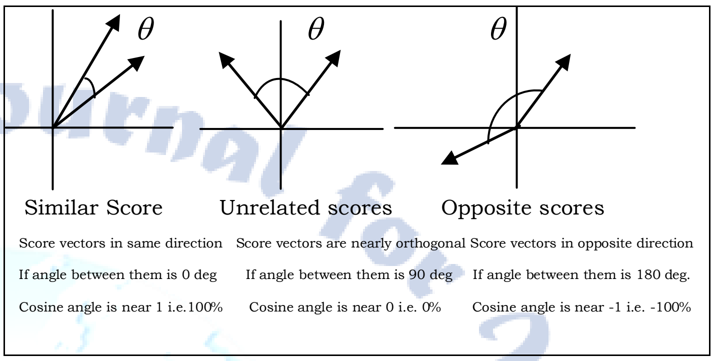
θ
θ
θ
Similar Score
Unrelated scores
Opposite scores
Score vectors in same direction
Score vectors are nearly orthogonal
Score vectors in opposite direction
If angle between them is 0 deg
If angle between them is 90 deg
If angle between them is 180 deg.
Cosine angle is near 1 i.e.100%
Cosine angle is near 0 i.e. 0%
Cosine angle is near -1 i.e. -100%

Fig 1: Cosine similarity values for different documents

Note that even if we had a vector pointing to a point far from another vector, they still could have an small angle and that is the central point on the use of Cosine Similarity, the measurement tends to ignore the higher term count on documents. Suppose we have a document with the word “earth” appearing 200 times and another document with the word “earth” appearing 50, the Euclidean distance between them will be higher but the angle will still be small because they are pointing to the same direction, which is what matters when we are comparing documents.

B. *TF-IDF:*

Tf-idf stands for term frequency-inverse document frequency, and the tf-idf weight is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. Variations of the tf-idf weighting scheme are often used Ɵby search engines as a central tool in scoring and ranking a document's relevance given a user query. One of the simplest ranking functions is computed by summing the tf-idf for each query term; many more sophisticated ranking functions are variants of this simple model.

Tf-idf can be successfully used for stop-words filtering in various subject fields including text summarization and classification.

Typically, the tf-idf weight is composed by two terms: the first computes the normalized Term

Frequency (TF), aka. the number of times a word appears in a document, divided by the total number of words in that document; the second term is the Inverse Document Frequency (IDF), computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears.
TF: Term Frequency, which measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length (aka. the total number of terms in the document) as a way of normalization:

$$TF(t) = \frac{\text{Number of times term t appears in a document}}{\text{Total number of terms in the document}}$$

IDF: Inverse Document Frequency, which measures how important a term is. While computing TF, all terms are considered equally important. However it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones,

$$IDF(t) = \log_e \frac{\textit{Total}\ \text{number of documents}}{\text{Number of documents with term t in it}}$$

The use of neighborhood information is worth more discussion. Because neighbor documents might not be sampled from the same generative model as the specified document, we probably do not want to trust them as much as the specified document. Thus a confidence value is associated with every document in the expanded document set, which reflects out belief that the document is sampled from the same underlying model as the specified document. When a document is close to the specified one, the confidence value is high, but when it is farther apart, the confidence value will be reduced. Heuristically, we use the cosine similarity between a document and the specified document as the confidence value. The confidence values of the neighbor documents will be incorporated in the keyphrase extraction algorithm.

V. SYSTEM ARCHITECTURE

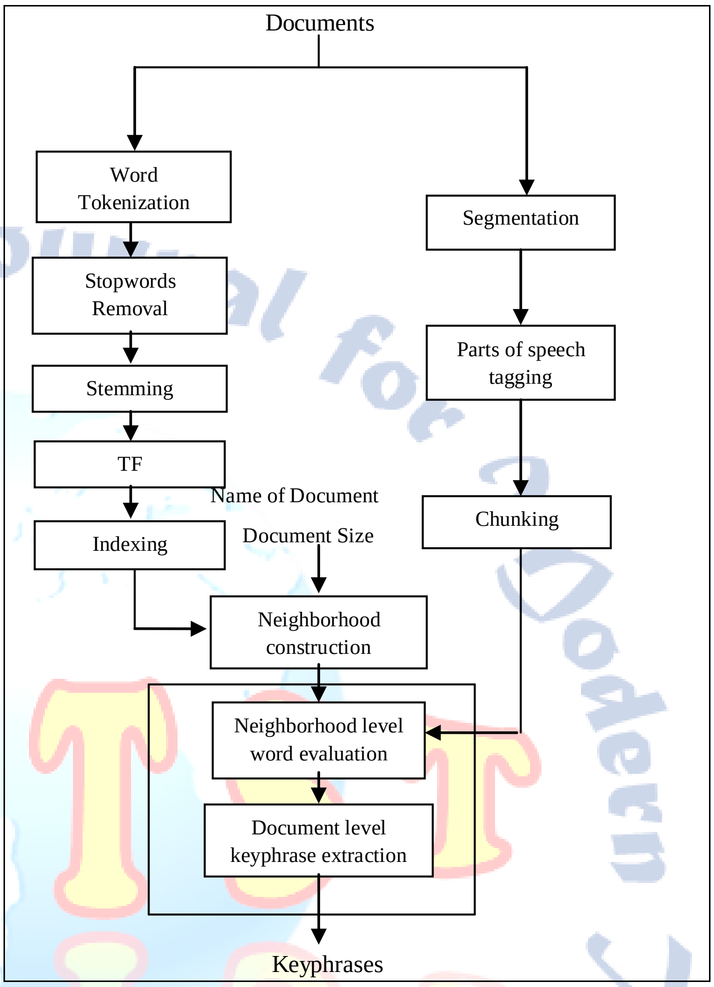

Fig 2: System Architecture for keyphrase extraction

VI. IMPLEMENTATION

In order to construct neighborhood documents we divide our work into following modules

1. INDEXING:

A document index improves the speed of data retrieval operations. Indexes are used to quickly locate data without having to search whole document.
Example: I am the king of cards and she is the queen of roses and we are the rulers of the world. You are the greatest king.
Output: King; card; great; queen; rose; ruler; world
Indexing helps in storing data to facilitate fast and accurate information retrieval. In order to form index system should undergo various phases.

TOKENIZATION:

Tokenization means splitting of text into meaning full words.

Example: I am the king of cards and she is the queen of roses and we are the rulers of the world. You are the greatest king.

No. of tokens in the above sentence are 26

Tokenization relies on punctuation and whitespace may or may not be included in the resulting list of tokens.

Tokens are separated by whitespace character, such as space or line break, or punctuation characters.

In java Text is divided into tokens using method: split

Class: string

SYNTAX:

public string[ ] split(String regex, int limit) or
public string[ ] split(String regex)

Parameters: regex: the delimiting regular expression

Limit: the result threshold which means how many strings to be returned

Return value: It returns the array of strings computed by splitting this string around matches of the given regular expression.

Special CHARACTER removal

Removing special characters ( ) " [ ] ' ; \ / : . ! @ # $ % ^ & * { } | ? < >

Removing Numbers 0-9

We use String replaceall( ) method in Java.

Syntax:

public String replaceAll(string regex, String replacement)

Example:

String str = "hello +$ % ^ & * { } | ? < >world[ ] ' ; \ / : . !";

str = str.replaceAll("+$ % ^ & * { } | ? < >world [ ] ' ; \ / : . !;");

Output: hello world

STOPWORDS:

Given a list of stopwords and a plain text document, output will be set of words remaining after removing stopwords from document. In our work we consider 176 words as stopwords which are called when required.

Syntax:

public static Boolean is StopWord(String word, String[ ] stopwords)

STEMMING:

Stemming is mainly used to reduce word forms to proper class. The most efficient algorithm used in stemming is The Porter stemming algorithm[13] (or 'Porter stemmer') is a process for removing the commoner morphological and inflexional endings from words in English. Its main use is as part of a term normalization process that is usually done when setting up Information Retrieval systems.

Algorithm:

Step1a: Replace plurals

Example: Stresses by stress, gaps by gap, cries by cri

Step1b: Deletes ed, edly, ing, ingly suffixes

Example: Fished - fish, repeatedly – repeat, predating – predicate, interestingly –interest etc.

Step1c: Replaces y by i

Example: Happy – happi, sky – sky.

Step2: Replaces ization, ational, iveness, fullness etc by ate,tion,ive,full respectively

Example: Renationalization – relation, relational – relate, conditional – condition, effectiveness – effective, hopefulness – hopeful.

Step3: Replaces icate, alize, iciti, ical by ic, al, ic, ic and deletes active, ful, ness

Example: Triplicate-triplic, formalize- formal, electricity- electric, electrical- electrical, formative-form, hopeful-hope, goodness-good.

Step4: Deletes al, ance, ence, er etc

Example: Revival – reviv, allowance- allow, inference- infer, airliner- airlin

Step5a: Deletes e

Example: Probate-probat, rate-rate

Step6: Maps double suffix to single suffix

Example: Control-control, roll- rol

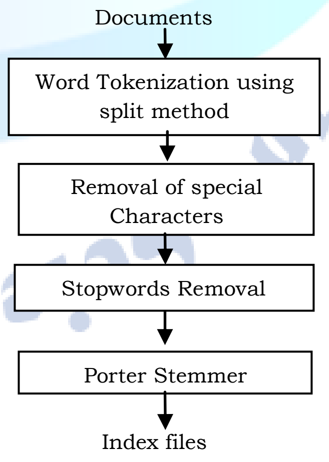
Documents
Word Tokenization using split method
Removal of special Characters
Stopwords Removal
Porter Stemmer
Index files

Fig 3: creating index file

2. Segmentation, Chunking, POS Tagging

For the process of segmentation, chunking and POS Tagging we are using tool known as jtextro which performs all the operations.

JTEXTPRO:

java text processor is essential processing steps for developing applications in NLP use Jtextpro developed by Xuan-Hieu Phan. A Java-based Text Processing toolkit that currently includes important processing steps for natural language/text processing as follows:

Sentence boundary detection

Sentence boundary detection is done using maximum entropy classifier train on WSJ corpus

Word tokenization

Tokenization is a process of breaking sentences into tokens

Syntax:

publicPENNTokenizer<word>newPENNTokenizer(Reader r)

Constructs a new PENNTokenizer that treats carriage returns as a normal whitespace

Parameters:

r- Reader whose contents will be tokenized

Returns:

A PENNTokenizer that tokenizes a strean of objects of type word

Input: tokenizer divides sentences into words known as tokens

Output: tokenizer divides sentences into words known as tokens

Part-of-Speech Tagging

Parts of speech (POS) tagging is assigning each word in a sentence to the parts of speech that it belongs to. POS tagging uses conditional random fields. Pos tagging guidelines as per university of Pennsylvania

Input: a string of words and a tagset

Output: a single best tag for each word

Example: Input: current account

Output: current JJ account NN

Phrase Chunking

Text chunking subsumes a range of tasks. It is the simplest of finding 'Noun groups'. More ambitious systems may add additional chunk types, such as verb group, or may seek a complete partition of sentences into chunks of different types. Chunking also uses conditional random fields. Chunks tags are made up of two parts

First part:

B- Marks the beginning of the chunk

I-Marks the continuation of a chunk

E- Marks the end of the chunk

Second part:

NP- noun chunk

VP- Verb Chunk

Example:

Word	POS tagging	Chunking
He	PRP	B-NP
Reckons	VBZ	B-VP
Current	JJ	I-NP
Account	NN	I-NP
Deficit	NN	I-NP
Will	MD	B-VP
Narrow	VB	I-VP
To	TO	B-PP
Only	RB	B-NP
#	#	I-NP
1.8	CD	I-NP
Billon	CD	I-NP

Table.1: POS tagging and chunking of the words

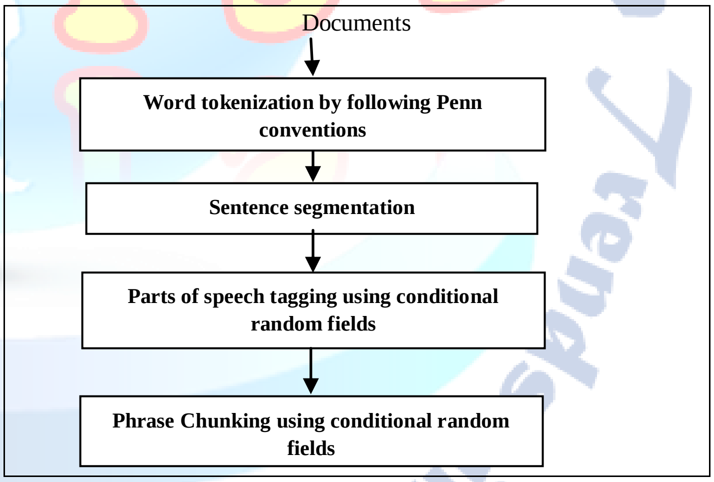
Documents
Word tokenization by following Penn conventions
Sentence segmentation
Parts of speech tagging using conditional random fields
Phrase Chunking using conditional random fields

Fig 4: segmentation, pos tagging, chunking

3. Neighborhood Construction

Given a specified document d_0, neighborhood construction aims to find a few nearest neighbors for the document from a text corpus or on the Web. The k neighbor documents $d_1, d_2, ..., d_k$ and the specified document d_0 build the expanded document set $D = \{d_1, d_2, ..., d_k\}$ for d_0, which can be considered as the expanded knowledge context for document d_0.The neighbor documents can be obtained by using the technique of document similarity search. Document similarity search is to find documents similar to a query document in a text corpus and return a ranked list of similar documents to users. The effectiveness of document similarity search relies on the function for evaluating the similarity between two documents. In this study, we use the widely-used cosine measure to evaluate document similarity.

Hence, the similarity $sim_{doc}(d_i, d_j)$, between documents d_i and d_j, can be defined as the normalized inner product of the two term vectors $\vec{d}_i$ and $\vec{d}_j$:

$$sim_{doc}(d_i, d_j) = \frac{\vec{d}_i . \vec{d}_j}{\|\vec{d}_i\| \times \|\vec{d}_j\|} \quad (3)$$

In the experiments, we simply use the cosine measure to compute the pair wise similarity value between the specified document d_0 and the documents in the corpus, and then choose k documents (different from d_0) with the largest similarity values as the nearest neighbors for d_0. Finally, there are totally $k+1$ documents in the expanded document set. For the document set $D = \{d_0, d_1, d_3, \ldots\ldots, d_k\}$, the pair wise cosine similarity values between documents are calculated and recorded for later use. The efficiency of document similarity search can be significantly improved by adopting some index structure in the implemented system, such as K-D-B tree, R-tree, SS-tree, SR-tree and X-tree (Böhm & Berchtold, [14]).

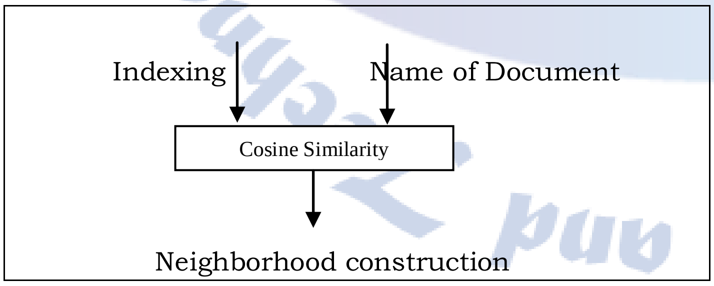

Fig 5: Construction of neighborhood

4. Keyphrase Extraction

Keyphrase extraction is done at two phases

- Neighborhood level word evaluation
- Document level keyphrase extraction

Neighborhood-Level Word Evaluation

Like the PageRank algorithm [15], the graph-based ranking algorithm employed in this study is essentially a way of deciding the importance of a vertex within a graph based on global information recursively drawn from the entire graph. The basic idea is that of "voting" or "recommendation" between the vertices. A link between two vertices is considered as a vote cast from one vertex to the other vertex. The score associated with a vertex is determined by the votes that are cast for it, and the score of the vertices casting these votes. Formally, given the expanded document set D, let $G = (V, E)$ be an undirected graph to reflect the relationships between words in the document set. V is the set of vertices and each vertex is a candidate word in the document set. Because not all words in the documents are good indicators of keyphrases, the words added to the graph are restricted with syntactic filters, i.e., only the words with a certain part of speech are added. As in Mihalcea and Tarau [16], the documents are tagged by a POS tagger, and only the nouns and adjectives are added into the vertex set. E is the set of edges, which is a subset of $v \times v$. Each edge e_{ij} in E is associated with an affinity weight $aff(v_i, v_j)$ between words v_i and v_j. The weight is computed based on the co-occurrence relation between the two words, controlled by the distance between word occurrences. The co-occurrence relation can express cohesion relationships between words. Two vertices are connected if the corresponding words co-occur at least once within a window of maximum w words, where w can be set anywhere from 2 to 20 words. The affinity weight $aff(v_i, v_j)$ is simply set to be the count of the controlled co-occurrences between the words v_i and v_j in the whole document set as follows:

$$aff(v_i, v_j) = \sum_{d_s = D} sim_{doc}(d_0, d_p) \times count_{d_s}(v_i, v_j) \quad (4)$$

Where $count_{d_s}(v_i, v_j)$ is the count of the controlled co-occurrences between words v_i and v_j in document d_p, and $sim_{doc}(d_0, d_p)$ is the similarity factor to reflect the confidence value for using document d_p $(0 \le p \le k)$ in the expanded document set. The graph is built based on the whole document set and it can reflect the global information in the neighborhood, which is called

Global Affinity Graph. We use an affinity matrix M to describe G with each entry corresponding to the weight of an edge in the graph. $M = (M_{ij})_{|v|\times|v|}$ is defined as follows:

$$M_{ij} = \begin{cases} aff(v_i, v_j); if\ (v_i links v_j)\&\&(i \neq j) \\ 0: otherwise \end{cases} \qquad (5)$$

Then M is normalized to $\tilde{M}$ as follows to make the sum of each row equal to 1:

$$\tilde{M}_{ij} = \begin{cases} M_{ij} \Big/ \sum_{j=1}^{|V|} M_{ij}, if \sum_{j=1}^{|V|} M_{ij} \neq 0 \\ 0, otherwise \end{cases} \qquad (6)$$

Based on the global affinity graph G, the saliency score $WordScore(v_i)$ for word v_i can be deduced from those of all other words linked with it and it can be formulated in a recursive form as in the PageRank algorithm:

$$WordScore(v_i) = \mu . \sum_{\forall j \neq i} WordScore(v_j) . \tilde{M}_{j,i} + \frac{(1-\mu)}{|V|} \qquad (7)$$

And the matrix form is:

$$\vec{\lambda} = \mu \tilde{M}^T \vec{\lambda} + \frac{(1-\mu)}{|V|} \vec{e} \qquad (8)$$

Where $\vec{\lambda} = [WordScore(v_i)]_{|V|\times 1}$ is the vector of word saliency scores. $\vec{e}$ is a vector with all elements equaling to 1. μ is the damping factor usually set to 0.85, as in the PageRank algorithm. The above process can be considered as a Markov chain by taking the words as the states and the corresponding transition matrix is given by $\mu \tilde{M}^T + \frac{(1-\mu)}{|V|} \vec{e}$. The stationary probability distribution of each state is obtained by the principal eigenvector of the transition matrix. For implementation, the initial scores of all words are set to 1 and the iteration algorithm in Equation (4.6) is adopted to compute the new scores of the words. Usually the convergence of the iteration algorithm is achieved when the difference between the scores computed at two successive iterations for any words falls below a given threshold (0.0001 in this study).

Document-Level Keyphrase Extraction

After the scores of all candidate words in the document set have been computed, candidate phrases (either single-word or multi-word) are selected and evaluated for the specified document d_0. The candidate words (i.e. nouns and adjectives) of d_0, which is a subset of V, are marked in the text of document d_0, and sequences of adjacent candidate words are collapsed into a multi-word phrase. The phrases ending with an adjective is not allowed, and only the phrases ending with a noun are collected as candidate phrases for the document. For instance, in the following sentence: "Mad/JJ cow/NN disease/NN has/VBZ killed/VBN 10,000/CD cattle/NNS", the candidate phrases are "Mad cow disease" and "cattle". The score of a candidate phrase p_i is computed by summing the neighborhood-level saliency scores of the words contained in the phrase.

$$PhraseScore(p_i) = \sum_{v_j \in p_i} WordScore(v_j) \qquad (9)$$

All the candidate phrases in document d_0 are ranked in decreasing order of the phrase scores and the top m phrases are selected as the keyphrases of d_0. m ranges from 1 to 20 in this study.

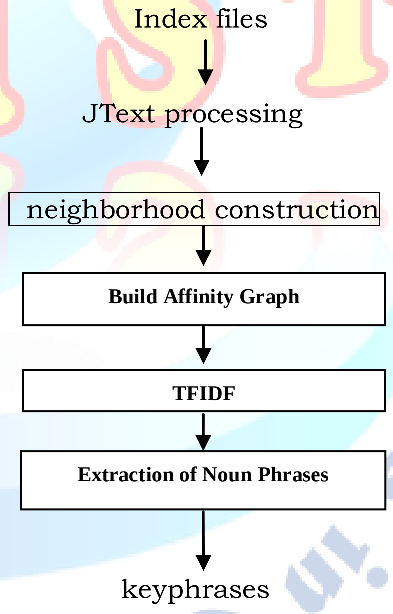

Fig 6: Extraction of keyphrases

VII. RESULT ANALYSIS

System takes plain documents as input undergoes through various techniques to extract keywords and constructs the neighborhood documents for the given document and then the keywords are extracted at neighborhood level and then at document level.

	Document Name	No. of words in document

Maximum words containing document	Sample18	1060
Minimum words containing document	Sample13	151
Average no. of words in all documents		345

Table 2: Test Data

DOCUMENT INDEXING

System takes a documents as input and it undergoes various stages such as tokenization, stopwords removal, stemming, term frequency (TF) through which the size of document will be reduced. Table 5.2 provides a detail frequency of the words before and after indexing of first ten documents. We can observe that number of words reduces after indexing.

Document Name	No. of words before indexing	No. of words after indexing	Term frequency	
			Most frequent word	Frequency
Sample	398	137	tax	11
Sample1	417	161	lokpal	8
Sample2	308	113	court	7
Sample3	469	142	govern	15
Sample4	360	121	will	7
Sample5	439	162	fodder	15
Sample6	347	145	modi	8
Sample7	214	87	lal	7
Sample8	109	47	justic	5
Sample9	369	141	farmer	10

Table 3: Document Indexing

NEIGHBORHOOD CONSTRUCTION

Neighborhood construction provides details about document in a text corpus and ranked list of similar documents. It takes input from index files and jtext processing output and by using cosine similarity method neighborhood is calculated. In Table 5.3 we consider sample document as considered main document and the remaining documents are the neighbor documents for sample document and arranged as per their similarity to sample document.

S.No	Document Name	Document Similarity
1	Sample	1.0
2	Sample10	0.40289894
3	Sample3	0.33210042
4	Sample4	0.29905915
5	Sample9	0.26591668
6	Sample1	0.24555373
7	Sample6	0.2197027
8	Sample5	0.14047898
9	Sample7	0.11404948
10	Sample2	0.11210743
11	Sample8	0.08503368

Table 4: neighborhood construction

KEYPHRASE EXTRACTION

News article: “The Hindu” News Paper Article related to implementation of GST in India

Top ten extracted Keyphrases: implementation gst reduction corporate tax rate, corporate tax rate per cent, rid exemption corporate tax side, hope goods services tax bill, parliament approval gst bill ongoing winter session, goods services tax, adversarial rent-seeking tax exercise, new indirect tax regime, revolutionary transformation taxes india since independence, minister state finance jayant sinha.

News article: “The Hindu” News Paper Article related to allegations made by Former Aam Admi Party Leader Prashant Bhushan related implementation of Lokpal NewDelhi in India

Top ten extracted Keyphrases: provisions jan lokpal draft anna movement, appointment removal processes lokpal, member activist-lawyer prashant bhushan Saturday, certain provisions draft bill, present presser mr. bhushan, strong lokpal body, lokpal act, delhi chief minister arvind kejriwal, dissident aap mla pankaj pushkar.

News article: “The Hindu” News Paper Article related to Supreme Court Judge on Law Minister Sadananda Gowda

Top ten extracted Keyphrases: plots hsr layout bangalore violation lease-cum-sale agreement mr. gowda, karnataka high court mr. gowda day, supreme court karnataka high court order, supremecourt high court, mr. gowda, spite clout mr. sadananda gowda, clean chit building law violation case, mr. gowda.

News article: “The Hindu” News Paper Article related to comments made by Rajanath Singh on Economic Growth.

Top ten extracted Keyphrases: several significant steps result domestic investors foreign investors, hot favourite destination foreign investors, many essential commodities prices, control economic growth double digit, dig economic policies previous upa government, destination foreign investors, sense confidence among investors, prices essential commodities, economic condition country, prices many goods.

News article: "The Hindu" News Paper Article related to current development scenario in India.

Top ten extracted Keyphrases: rivers across india national waterways current winter session, rivers across india national waterways, movement goods passengers across country, inland waterways, charge road transport highways, dry ports, large number national highways, many four-lane roads highways, ports, shipping minister nitin gadkari.

VIII. CONCLUSION

According to experiments conducted on several documents and keyphrases we consider top 10 keyphrases are most suitable keyphrases.

REFERENCES

[1] "*Single Document Keyphrase Extraction Using Neighborhood Knowledge*" by Xiaojun Wan and Jianguo Xiao submitted to Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (2008).

[2] "*Learning Algorithms for Keyphrase Extraction*" by Peter D. Turney, Submitted to Information Retrieval — INRT 34-99 on October 4, 1999.

[3] "*A Ranking Approach to Keyphrase Extraction*" by Xin Jianga, Yunhua Hub, Hang Lib , Microsoft Research Technical Report July 31, 2009.

[4] "*Improved Automatic Keyword Extraction Given More Linguistic Knowledge*" by Anette Hulth, Department of Computer and Systems Sciences, Stockholm University Sweden hulth@dsv.su.se

[5] "*Unsupervised Approaches for Automatic Keyword Extraction Using Meeting Transcripts*" by FeifanLiu, DeanaPennell, FeiLiu and YangLiu submitted to Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL.

[6] "*Automatic Keyword Extraction From Any Text Document Using N-gram Rigid Collocation*" by Bidyut Das, Subhajit Pal, Suman Kr. Mondal, Dipankar Dalui, Saikat Kumar Shome.

[7] "*The Contact Finder agent: Answering bulletin board questions with referrals*" by Bruce Krulwich and Chad Burkey: Center for Strategic Technology Research Andersen Consulting 3773 Willow Drive, Northbrook, IL, 60062.

[8] "*Compound key word generation from document databases using a hierarchical clustering ART model*" by Muñoz, A., Intelligent Data Analysis, Amsterdam: Elsevier. (1996).

[9] "*Exporting phrases: A statistical analysis of topical language*" by Steier, A. M., and Belew, R. K., Second Symposium on Document Analysis and Information Retrieval, PP.: 179-190, 1993.

[10] "*Domain-specific keyphrase extraction*" by Frank, E., Paynter, G.W., Witten, I.H., Gutwin, C., and Nevill-Manning, C.G., Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99), PP.: 668-673. California: Morgan Kaufmann, 1999.

[11] "*A statistical learning approach to automatic indexing of controlled index terms*" by Leung, C.-H., and Kan, W.-K., Journal of the American Society for Information Science, PP: 55-66, 1997.

[12] "*Extraction of index words from manuals*" by Nakagawa, H. RIAO 97 Conference Proceedings: Computer-Assisted Information Searching on Internet, PP. 598-611. Montreal, Canada, (1997).

[13] "*An algorithm for suffix stripping*" by M.F.Porter 1980

[14] "*Searching in high-dimensional spaces index structures for improving the performance of multimedia databases*" by Böhm, C., and Berchtold, S. 2001. ACM Computing Surveys, PP: 322-373.

[15] "*The page rank citation ranking: Bringing order to the web*" by L.; Brin, S.; Motwani, R.; and Winograd, T. Technical report, 1998, Stanford Digital Libraries.

[16] *TextRank: Bringing order into texts*" by Mihalcea, R., and Tarau, P., In Proceedings of EMNLP2004.