

Multi Document Extractive Summarization Based On Word Sequences

R.KOWSALYA , R.PRIYA and P.NITHIYA

UG Scholar , Sri Venkateswara College Of Engineering, Sriperumbudur ,Tamil Nadu,India

UG Scholar, Sri Venkateswara College Of Engineering, Sriperumbudur,Tamil Nadu,India

Asst.Prof , Sri Venkateswara College Of Engineering, Sriperumbudur,Tamil Nadu,India

Abstract

In this paper we propose to produce an extractive summary for given set of documents based on word sequence models by extracting **Maximal frequent sequences** from the given text. The main problem for generating an extractive automatic text summary is to detect the most relevant information in the source document. To overcome this problem we have successfully employed the word sequence information from the self-text for detecting the candidate text fragments for composing the summary. To compose the effective summarization we have used mfs technique to extract the detect the most important terms in the source document and Normalized Google dissimilarity distance for sentence clustering . This simple representation not only diminishes domain and language dependency but also enhances the summarization performance.

Keywords: *Multidocument summarization ,Extractive summarization, maximal frequent sequence, sentence clustering, normalized Google dissimilarity.*

1.Introduction

Typical information retrieval (IR) systems have two steps[2]: the first is to find documents based on the user's query, and the second is to rank relevant documents and present them to users based on their relevance to the query. Then the users have to read all of these documents. The problem is that these docs are much relevant and reading them all is time-consuming and unnecessary.

Multi-document summarization aims at extracting major information from multiple documents and has become a hot topic in NLP. A summary can be loosely defined as a text that is produced from one or more texts that conveys important information in the original text(s), and that is no longer than half of the original text(s) and usually significantly less than that.

The text summarization tasks can be classified into single-document and multi document summarization .In single-document summarization, the summary of only one document is to be built, while multi document summarization the summary of a whole collection of documents .

An extractive summary, is composed with a selection of sentences (or phrases, paragraphs, etc.) from the original text, usually presented to the user in the same order—i.e., a copy of the source text with most sentences omitted. An extractive summarization method only decides, for each sentence, whether or not it will be included in the summary.

The Maximal Frequent Sequences(MFS)[2] are attractive for extractive text summarization since it is not necessary to define the gram size (n), it means, the length of each MFS is determined by the self text. Moreover, the set of all extracted MFS s is a compact representation all frequent word sequences, reducing in this way the dimensionality in a vector space model.

Multi document summarization can be classified into three categories according to the way that summaries are created: term extraction, sentence clustering and sentence extraction.

We will assume that the units of selection are sentences (these could be, say, phrases or paragraphs). Thus final goal of the extractive summarization process is sentence selection. One of the ways to select the appropriate sentences is to assign some numerical measure of usefulness of a sentence for the summary and then select the best ones; the process of assigning these usefulness weights is called sentence weighting.

2.Requirements for multi document summarization

There are two types of situations in which multi document summarization would be useful: (1) the user is faced with a collection of dis-similar documents and wishes to assess the information landscape contained in the collection, or (2) there is a collection of topically related documents, extracted from a larger more diverse collection as the result of a query, or a topically-cohesive cluster.
Following is a list of requirements for multi-document summarization:

2.1 clustering:
The ability to cluster similar documents and passages to find related information.

2.2 coverage:
The ability to find and extract the main points across documents.

2.3 anti-redundancy:
The ability to minimize redundancy between passages in the summary

2.4 summary cohesion criteria:
The ability to combine text passages in a useful manner for the reader. This may include:
2.4.1document ordering: All text segments of highest ranking document, then all segments from the next highest ranking document, etc.

2.4.2 rank ordering: present the most relevant and diverse information first. So that the reader gets the maximal information content even if they stop reading the summary.

2.4.3topic-cohesion: Group together the passages by topic clustering using passage similarity criteria and present the information by the cluster centroid passage rank.

2.4.4 time line ordering: Text passages ordered based on the occurrence of events in time.

2.4.5 coherence: Summaries generated should be readable and relevant to the user.

2.4.6 context: Include sufficient context so that the summary is understandable to the reader.

2.4.7 Identification of source inconsistencies: Articles often have errors (such as billion reported as million, etc.); multi-document summarization must be able to recognize and report source inconsistencies.

2.4.8 summary updates: A new multi-document summary must take into account previous summaries in generating new summaries. In such cases, the system needs to be able to track and categorize events.

2.5 effective user interfaces:

2.5.1 Attributability: The user needs to be able to easily access the source of a given passage. This could be the single document summary.

2.5.2 Relationship: The user needs to view related passages to the text passage shown, which can highlight source inconsistencies.

2.5.3 Source Selection: The user needs to be able to, select or eliminate various sources. For example, the user may want to eliminate information from some less reliable foreign news reporting sources.

2.5.4 Context: The user needs to be able to zoom in on the context surrounding the chosen passages.

2.5.5 Redirection: The user should be able to highlight certain parts of the synthetic summary and give a command to the system indicating that these parts are to be weighted heavily and that other parts are to be given a lesser weight.

3. Proposed work

Commonly, an extractive summarization approach achieves the term selection, term weighting, sentence weighting and sentence selection steps. However, the strategy of sentence selection step is reduced to simply to

take the weightiest sentences. Although, this strategy could work well for the first ranked sentence, the strategy could convey that similar

sentences to the first one tend to be ranked after the first one; producing redundant sentences for the summary. This problem affects negatively in recall measure. For avoiding this problem, we employ the unsupervised learning algorithm of for automatically detecting the groups of similar sentences from which is selected the most representative sentence; reducing in this way the redundancy in the summary.

In this section, we describe the general steps of the proposed approach[2].

3.1 Preprocessing

3.1.1 Removal of stop words:

Stop words are frequently occurring, insignificant words that appear in a database record, article or web page. Stop words are an application dependent. They apply to the particular database or application (e.g.: searching, summarization). It is commonly assumed that words, which are not members of the noun-verb adjective classes, should be on stop words lists. When a document is summarized by sentence extraction method we assign weights to all the keywords or tokens in the input document. The process of doing such stop word elimination results in better summary generation. Since we eliminate these stop words unwanted sentences would never climb higher up the order . Single characters, common two-character and three-character words, frequently repeated words are typically included in the stop word list to maximize performance of summarization process.

3.1.2 Applying porter stemming algorithm:

Truncation, also called stemming, is a technique that allows us to search for various word endings and spellings simultaneously. Stemming algorithms are used in many types of language processing and text analysis systems, and are also widely used in summarization, information retrieval and database search systems. A stemmer is a program determines a stem form of a given word. In other words, generates the morphological root of the word. Terms with a common stem will usually have similar meanings. For the example shown below the common root word is 'IMPROV'. IMPROVE, IMPROVED, IMPROVEMENTS.

The suffix stripping process will reduce the total number of terms in the IR system, and hence reduce the size and complexity of the data in the system, which is advantageous.

3.2 Term selection:

A maximal frequent word sequences is a gram (the size is not restricted) that it is frequent in text, but it is not contained (as subsequence) in other frequent gram. In this case, for considering a gram as frequent it is necessary to establish a frequency threshold.

3.2.1 Proposed Algorithm

Our algorithm to extract the maximal frequent sequences from a given text belongs to pattern-growth methods class[5]; because it uses a bottom-up strategy without candidate generation. The main idea consists in to generate only all the distinct pairs of items from the text, i.e. the 2-sequences, and do not lose the relation between them, in order to allow the growth of the sequences. The input data of the algorithm are a text (T) and a β threshold. The proposed algorithm has three phases[5].

These phases are as follows:

Phase 1: Get the alphabet from the text. The algorithm gets an id for each different item (chars or words) from the text.

Phase 2: Construct an array structure for text representation (fig. 1). The algorithm constructs an array structure from the text T. Each element of the array contains two id's corresponding to a distinct pair (t_i,t_{i+1}), the frequency of this pair and a list of the positions where this pair appears in the text . Each position node of the positions list contains the position where the pair appears, together with the next-index of the following pair in each position. The phase 2 works as follows: for each item t_i get the index of the pair (t_i,t_{i+1}) in the array and in this position add a Position node at the end of the list of positions. Increase the frequency (Freq) of this pair and link this position node with the previously added position node in order to build the NextPos list, which stores the text representation.

Phase 3 : Find all MFS (fig. 2). For each element i of the NextPos list, check if it has a frequency ≥ β, in order to determine if this pair can become a possible maximal sequence PMS. If frequency ≥ β then grow forward all the elements in the NextPos list w.r.t i. If after this growth there is (in the NextPost list) a number of elements ≥ β, then the PMS can grow. When the PMS cannot grow it is added to the MFS list if only if the PMS is not a subsequence of any previously stored MFS, and all the MFS that are subsequence of the PMS are deleted from the MFS list. For our algorithm each PMS can be classified as a PMS with and without cycles. A cycle is detected when the first pair is repeated if it happens, the cycle function is used to get the PMS. The cycle function guarantees the mutually excluded property. If the PMS obtained from cycle function can grow, then it is treated as a PMS without cycles, because it could grow.

Phase 2:Algorithm to construct the array structure

Input: A text T **Output**:The array structure
For all the pairs[t_i,t_{i+1}] € T **do**
PositionNode.Pos ←index←array[t_i,t_{i+1}]; //if [t_i,t_{i+1}] it is not in array,add it
array[index].Positions←**New** PositionNode //new node
array[index].Freq ← array[index].Freq+1; //increase the frequency
array[LastIndex].Positions.NextIndex←index; //keep the index of the pair
array[LastIndex].Positions.NextPos←PositionNode //link the new node
LastIndex←index; //keep the last index
End-for

Fig.1.Algorithm to construct the array structure(phase 2)

Phase 3:Algorithm to find all MFS

Input: Array from phase 2, β support **Output**:MFS list
Actual ←1 //index of the array where it is the first element of NextPos List
While Actual ≠0 **do**
If Array[Actual].Frequency≥ β, then //if the pair has frequency ≥β
temporal← Copy_list(array[Actual].Postions) //create a similar list
PMS←Array[Actual].Id_1 +Array[Actual].Id_2 //initial elements of the PMS
Pos←Array[Actual].NextIndex //The first time Pos←1
While Pos ≠ 0 **do**
temporal ← Get common nodes((temporal.Pos+1),(Array[Pos].Positions.Pos))
if |temporal|≥β ,then //if temporal has a number of nodes ≥ than β
if Pos=Array,then there is a cycle,
PMS← Cycle(β,temporal,array,Actual,Pos) //call to Cycle function
if the PMS cannot grow then exit from the while
else PMS←PMS+Array[Pos].Id_2 //expand the PMS
Pos←Array[Actual].NextIndex
end-while
delete all the MFS € PMS
if (PMS€ MFS) then MFS ←Add(PMS)
Actual←array[Actual].NextIndex
End-while

Fig. 2. Algorithm to find all MFS (phase 3)

Cycle function : **Algorithm to find the PMS with cycles**
Input:β support,temporal,array,Actual,aux; **Output:**A PMS

CycleSize←Array[aux].Pos-Array[Actual].Pos; //size of the cycles
Intervals← From temporal find the intervals of groups of cycles
ActualGrpSize← Size of the interval where[Array[Actual].Pos-Array[aux].Pos] € Intervals
while ActualGrpSize ≥ 2 **do**
For each Interval get the frequency w.r.t ActualGrpSize
If ∑ frequencies ≥β then PMS ←T[Array[Actual].Pos]+..+T[Array[Actual].Pos+ActualGrpSize]
If ActualGrpSize was not decremented then the PMS can grow
Temporal ←Rebuild temporal with(end of Intervals-1) in which the frequency=1
Return(PMS,temporal);
end-for
end-while

The frequency is calculated as follows:
Used-Periods ← Ceiling (ActualGrpSize/cycleSize);
Period←GrpSize analyzed/CycleSize
If Used-periods.$_{remainder}$ =Period.$_{remainder}$ and period.$_{remainder}$>0 then
Period←Period+1
Frequency=Period/Used-Periods

Fig. 3. Algorithm for finding a PMS with cycles

Cycle function (fig. 3). Using the size of the cycle (number of elements between the first and the repeated pair) find all the groups of occurrences of the cycle in order to build a list of intervals with the beginning and end of such positions. Using this list of intervals it is possible to find the longest PMS. Given the size of the interval, this function tests in decreasing way (because we search the longest PMS) how many PMS are contained in each interval, therefore the sum of this local frequency becomes the total frequency that must be ≥ β. In such case, the PMS has as size the size of the interval that can appear β times into the text. If the size of the interval was not decremented then it is a PMS that can grow.

3.3Term Weighting:

3.3.1.Boolean Weighting (BOOL): It is the easiest way to weight a term. It models the presence or absence of a term in the document, defined as[4]:

W_i (tj)={1 if the term tj appears in document i.

{0 other case ……(1)

3.3.2Term Frequency (TF) : This weighting takes into account that a term that occurs in a document can better reflect the contents of document than a term that occurs less frequent. Therefore, the weighting TF assigns a greater relevance to terms with greater frequency and consists in evaluating the number of times the term appears in the document.

$W_i(t_j)=f_{ij}$,where f ij is the frequency of the term j in document i. ……. (2)

3.3.3.Inverse Document Frequency (IDF):
The problem of TF weighting in IR is that, when a term appears in almost all the documents in the collection; this term is useless for discriminating relevant documents. For example, the stop-word *and* could have a high TF, but it is useless for discriminating the relevant documents since tends to appear in most of the documents. IDF is defined as[4]:

$W_i(t_j)=\log(N/n_j)$ ….(3)

where *N* is the number of documents in the collection and n_j is the number of documents where the term *j* appears.

3.3.4.TF-IDF:
The problem of IDF weighting in IR is that it is not possible distinguish between two documents with the same vocabulary (list of different

words), even thought if the term is more frequent in a document. TF-IDF weighting gives more relevance to the terms that are less frequent in the collection but more frequent into the document.

$W_i(t_j) = f_{ij} \times \log(N/n_j)$ …(4)

3.4Sentence clustering

Data clustering is the process of identifying natural groupings or clusters within multidimensional data based on some similarity measure. Clustering is a fundamental process in many different disciplines such as text mining, pattern recognition, IR etc. In our method[1], a sentence Si is represented as sequence of words, $S_i=(t_1,t_2, \ldots ,t_{mi})$, instead of the bag of words, where mi is the number of words in a sentence S_i.

In this section we present a method to measure dissimilarity between sentences using the normalized google distance (NGD). NGD takes advantage of the number of hits returned by Google to compute the semantic distance between concepts. The concepts are represented with their labels which are fed to the Google search engine as search terms. First, using the NGD we define the global and local dissimilarity measure between terms [6] the NGD is nonnegative and does not satisfy the triangle inequality, i.e. hence isn't distance and consequently in the further it we shall name dissimilarity measure). According to definition NGD the global dissimilarity measure between terms t_k and t_l also is defined by the formula:

$$NGD^{global}(t_k,t_l)=\frac{\max\{\log f_k^{global},\log f_l^{global}\}-\log f_{kl}^{global}}{\log N_{google}-\min\{\log f_k^{global},\log f_l^{global}\}} \quad \ldots(5)$$

where f_k^{global} is the number of web pages containing the search term t_k and f_{kl}^{global} denotes the number of web pages containing both terms t_k and t_l, N_{google} is the number of web pages indexed by Google.

The following are the main properties of the NGD [6]:

(1) The range of the NGD is in 0 and 1;If $t_k = t_l$ or if If $t_k \neq t_l$ but frequency

(2) $f_k^{global} = f_l^{global} = f_{kl}^{global} > 0$

then $NGD^{global}(t_k,t_l)=0$.That is, the semantics of t_k and t_l in the Google sense is the same. If frequency $f_k^{global}=0$,then for every term t_k, we have $f_{kl}^{global}=0$, and the $NGD^{global}(t_k,t_l)=\infty/\infty$, which we take to be 1 by definition. If frequency $f_l^{global} \neq 0$ and $f_{kl}^{global}=0$,we take $NGD^{global}(t_k,t_l)=1$.

(3) $NGD^{global}(t_k,t_l)=0$.for every t_k. For every pair t_k and t_l, we have $NGD^{global}(t_k,t_l)=NGD^{global}(t_l,t_k)$
It is symmetric.

Using the formula Eq(5) we define a global dissimilarity measure between sentences S_i and S_j as follows:

$$diss_{NGD}^{global}(S_i,S_j)=\frac{\sum_{t_k\in S_i}\sum_{t_l\in S_j} NGD^{global}(t_k,t_l)}{m_i m_j} \quad \ldots(6)$$

From the properties of NGD follows, that:
(1) the range of the $diss_{NGD}^{global}(S_i,S_j)$ is in 0 and 1;
(2) If $t_k = t_l$ or if $t_k \neq t_l$ but frequency $f_k^{global}=f_l^{global}=f_{kl}^{global}>0$ then then $diss_{NGD}^{global}(S_i,S_j)=0$ and

(3) $diss_{NGD}^{global}(S_i,S_j)=0$ for every Si. Dissimilarity measure between sentences is exchangeable in that $diss_{NGD}^{global}(S_i,S_j)=diss_{NGD}^{global}(S_j,S_i)$ for every pair Si and Sj.

Similarly, we define the local dissimilarity measure between sentences Si and Sj:

$$diss_{NGD}^{local}(S_i,S_j)=\frac{\sum_{t_k\in S_i}\sum_{t_l\in S_j} NGD^{local}(t_k,t_l)}{m_i m_j} \quad \ldots(7)$$

Where

$$NGD^{local}(t_k,t_l)=\frac{\max\{\log f_k^{local},\log f_l^{local}\}-\log f_{kl}^{local}}{\log N_{google}-\min\{\log f_k^{local},\log f_l^{local}\}} \quad \ldots(8)$$

is the local dissimilarity measure between terms $t_k$and $t_l$which f_k^{local} denotes the number of sentences in a document D, containing the term t_k, and f_{kl}^{local} denotes the number of sentences containing both terms t_k and t_l If the number of sentences n =1, then we have

$f_i^{global} = f_i^{global} = f_{ic}^{global}$ and the $diss_{local}(S_i, S_j) = \begin{cases} 0 \\ 1 \end{cases}$, which we take to be 0 by definition. Thus, the overall sentence dissimilarity is defined as a product of global and local dissimilarity measures:

$$diss_{sent}(S_i, S_j) = diss_{sent}^{global}(S_i, S_j) . diss_{local}^{local}(S_i, S_j)$$

..(9)

3.5Sentence Extraction:

Extractive summarization works by choosing a subset of the sentences in the original document. This process can be viewed as identifying the most salient sentences in a cluster that give the necessary and sufficient amount of information related to main content of the cluster. In a cluster of related sentences, many of the sentences are expected to be somewhat similar to each other since they are all about the same topic. The approach, proposed in papers [7], is to assess the centrality of each sentence in a cluster and extract the most important ones to include in the summary. In centroid-based summarization, the sentences that contain more words from the centroid of the cluster are considered as central. Centrality of a sentence is often defined in terms of the centrality of the words that it contains. In this section we use other criterion to assess sentence salience, proposed in paper (Pavan & Pelillo, 2007). Let Cp be nonempty cluster and $S_i \in C_p$. Then the average weighted degree of S i with respect to cluster Cp is defined as

$$awdeg_{c_p}(s_i) = \frac{1}{|c_p|} \sum_{s_j \in c_p} diss_{sent}(S_i, S_j)$$

…..(10)

Observe that $awdeg_{s_i}(s_i)=0$ for any $s_i \in c_p$. Moreover, if $s_i \in c_p$, we define:

$$\phi_{c_p}(s_i) = diss_{sent}(S_i, S_j) - awdeg_{c_p}(s_i)$$

….(11)

From $awdeg_{s_i}(s_i)=0$ follows that $\phi_{c_p}(s_i, s_j) = diss_{sent}(S_i, S_j)$ for all $s_i, s_j \in c_p$, with i ≠ j. Intuitively, $\phi_{c_p}(s_i)$ measures the relative measure between sentences s_i and s_j with respect to the average measure between s_i and its neighbors in cluster c_p. Note that $\phi_{c_p}(s_i, s_j)$ can be either positive or negative.Thus the weight of sentence s_i with respect to cluster c_p will be defined by the following recursive formula as

$$w_{c_p}(s_i) = \begin{cases} 1 & if\ |c_p| = 1 \\ \sum_{s_j \in c_p \setminus \{s_i\}} \phi_{c_p \setminus \{s_i\}}(s_i, s_j) w_{c_p \setminus \{s_i\}}(s_j) & \text{otherwise} \end{cases}$$

…(12)

Note that

$$W_{c_p \setminus \{s_i\}}(s_i) = W_{c_p \setminus \{s_j\}}(s_j) = diss_{sent}(s_i, s_j)$$

for all $s_i, s_j \in c_p$ (i≠j).Intuitively, $W_{c_p}(s_i)$ gives us a measure of the overall (relative) dissimilarity measure between sentence s_i and the sentences of $c_p \{s_i\}$ with respect to the overall measure among the sentences in $c_p \{s_i\}$. Finally, as to selection of sentences to generate a summary, in each cluster sentences are ranked in reversed order of their score and the top ranked sentences are selected for in the extractive summary.

4. Conclusion

In this paper an efficient extractive summary generating algorithm is proposed for multiple documents. And also this algorithm improves accuracy of the summary when comparing with previous works done on this topic. **MFS** technique is used to extract the sentences from the documents that need to be in the summary. **NGD** technique is used for clustering the sentences from given multiple documents. And finally the most representative sentences are extracted from the clusters to compose the summary. Applications of multi document summarization include automatic construction of summaries of news articles or email messages for sending them to mobile devices as SMS; summarization of web pages to be shown on the screen of a mobile device, among many others.

References:

[1] Ramiz M. Aliguliyev , "A New Sentence Similarity Measure And Sentence Based Extractive Technique For Automatic Text Summarization", Expert Systems with Applications36 pp.7764–7772.2009.

[2] René Arnulfo García-Hernández, Yulia Ledeneva ,"Word Sequence Models for Single Text Summarization", in Second International Conferences on Advances in Computer-Human Interaction,2009,.

[3] Villatoro-Tello, E., Villaseñor-Pineda, L., Montes-y-Gómez, M, "Using Word Sequences for Text Summarization", TSD, LNAI Springer 2006.

[4] Ledeneva Yulia, Gelbukh Alexander, René Arnulfo García-Hernández, "Terms Derived from Frequent Sequences for Extractive Text Summarization", CICLing'2008. LNCS vol. 4919 Springer-Verlag, pp. 593-604, 2008.

[5]Rene A.Garcia-Hernandez,Jose Fco.Martinez-Trinidal and Jesus Airel Carrasco-Ochoa, "A Fast Algorithm To Find All the Maximal Frequent Sequences in A Text", Lecture Notes in Computer Science,2004,Volume 3287,Progress in Pattern Recognition,Image Analysis and Applications,Pages 305-320.

[6] R.L. Cilibrasi, P.M.B. Vitanyi, "The Google Similarity Distance",IEEE Transaction on Knowledge and Data Engineering ,March 2007 (vol. 19 no. 3), pp. 370-383.

[7] Radev,D.R,Jing,H.,Stys,M.,&Tam,D. (2004), "Centroid Based Summarization of Multiple documents".Information Processing and Management,40,919-938.

[8] Ramiz M. Aliguliyev , "A new sentence similarity measure and sentence based extractive technique for automatic text summarization", in Expert Systems with Applications ,pp.7764–7772, May 2009.

[9] Aliguliyev, R. M. "Automatic Document Summarization by Sentence Extraction" in Journal of Computational Technologies , pp.5–15,2007.

P.Ni

Kowsalya R: She is pursuing her B.Tech degree in Information Technology (Sri Venkateswara College of Engineering,Tamil Nadu).

Priya R: She is pursuing her B.Tech degree in Information Technology (Sri Venkateswara College of Engineering,Tamil Nadu).

Nith

Nithiya P: She obtained her M.E degree in Computer Science Engineering and obtained her B.E degree in Computer Science .She is presently working in Sri Venkateswara College of Engineering as Assistant Professor (Department of Information Technology).She has presented a paper titled "Development of semantic based information retrieval using wordnet approach", in Computer and Network Technology (ICCNT), 2010 Second International Conference .Her area of interest and research is in the field of Natural Language Processing.