J.Ramachandra, J.Ravi kumar,Y.V.Sricharan,Y.Venkata Sreekar / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 2, Issue4, July-August 2012, pp.2272-2277

Uncertainty Based Text Summarization

J.Ramachandra[1] J.Ravi kumar[2]
Y.V.Sricharan[3] Y.Venkata Sreekar[4]
[1] 2nd Year M.Tech, CREC, Tirupati
[2] Professor of Department of CSE, CREC, Tirupati

Abstract

Effective extraction of query relevant information present within the webpages is a nontrivial task. QTS, task by filtering and aggregating important query relevant sentences distributed across the webpages. This system captures the contextual relationships among sentences of these webpages and represents them as an "integrated graph". These relationships are exploited and several subgraphs of integrated graph which consist of sentences that are highly relevant to the query and that are highly related to each other are constructed. These subgraphs are ranked by the scoring model. The subgraph with highest rank which is rich in query relevant information is returned as a query specific summary. QTS is domain independent and doesn't use any linguistic processing, making it a flexible and general approach that can be applied to unrestricted set of articles found on WWW. Experimental results prove the strength of QTS.Very little work is reported on query specific multiple document summarization.The quality of summaries generated by QTS is better than MEAD(one of the popular summarizers).

Introduction

with the increased usage of computers huge amounts of data is being added to web constantly. This data is stored either on personal computers or data servers accessible through Internet. The increased disk space and popularity of Internet has resulted in vast repositories of data available at fingertips of a computer user resulting in "Information Overload" problem. The World Wide Web has become the largest source of information with heterogeneous collections of data. A user in need of some information is lost in the overwhelming amounts of data.

Search engines try to organize web dynamically by identifying, retrieving and presenting web pages relevant to users search query. Clustering search engines like cluster go a step further to organize the retrieved search results list into categories and allow user to do a focused search.

A web user seeking information on a broad topic will either visit a search engine and type in a query or browse through the web directory. In both these cases, the number of web pages that are retrieved to satisfy user's need is very high. Moreover, information pertaining to a query might be distributed across several sources. So, it is a difficult thing for a user to sift through all these documents and find the information that it needs. Neither web directories nor search engines are helpful in this matter. Hence it would be very useful to have a system which could filter and aggregate information relevant to user's query from various sources and present it as a digest or a summary. This summary would help in getting an overall understanding of the query.

Automatic Query Specific Multi-document Summarization is the process of filtering important relevant information from the input set of documents and presenting the concise version of that documents to the user. QTS fulfills this objective by generating a summary that is specific to the given query on a set of documents. Here we focus on building a coherent and highly responsive multi-document summary which is complete. This process poses significant challenges like maintaining intelligibility, coherence and non-redundancy.

Intelligibility/responsiveness it is the property that determines if the summary satisfies user's needs or not while Coherence determines the readability and information flow. As user's query in the context of a search engine is small (typically 2 or 3 words), we need to appropriately select important relevant sentences. Some sentences may not contain query terms but can still be relevant. Not selecting these sentences will affect the intelligibility of summary. Moreover, selecting sentences while summarizing long documents without considering the context in which it appears will result in an incoherent text. Specific to multi-document summarization, there are problems of redundancy in input text and the ordering of selected sentences in the summary.

Summarization process proceeds in three stages - source representation, sentence filtering and ordering. In QTS system, the focus is on generating summaries by a novel way of representing the input documents, identifying important sentences which are not redundant and presenting them as summary. The intention is to represent the contextual dependencies present between sentences as a graph by connecting a sentence to another sentence if they are contextually similar and highlight how these relationships can be exploited to score and select sentences.

IG is built to reflect inter and intra document contextual similarities present between sentences of the input documents. It is highly probable that the neighborhood of a node in this graph contains sentences that have similar content. So, from each node in this graph by exploiting its neighborhood, a summary graph called SGraph is built which will contain query relevant information.

This approach is domain independent and doesn't use any linguistic processing, making it a flexible and general approach that can be applied to unrestricted set of articles. We have implemented QTS and it works in two phases. First phase deals with creation of IG and necessary indexes that are query independent and can be done offline.

Second phase deals with generation of summary relevant to the user's query from the IG. First phase is a time consuming process especially for sets of documents with large number of sentences. So, this system can be used on a desktop machine in a digital library or in an intranet kind of an environment where documents can be clustered and graphs can be created .We present experimental results that prove the strength of QTS.

Related work

Several clustering based approaches were tried where similar sentences are clustered and a representative sentence of each cluster is chosen as a summary. MEAD [2] is a centroid based multi-document generic summarizer. It follows cluster based approach that uses features like cluster centroids, position etc., to summarize documents.

Recently, graph based models are being used to represent text. They use measures like degree centrality [3] and eigenvector centrality [4] to rank sentences. Inspired by PageRank [5], these methods build a network of sentences and then determine the importance of each sentence based on its connectivity with other sentences. Highly ranked sentences form a summary.

QTS proposes a novel algorithm that does statistical processing to exploit the dependencies between sentences and generate a summary by balancing query responsiveness in it.

System models

URL Extraction And Sentence Extraction

Sentence extraction: A html file is given as input and paragraphs are extracted from this html file. These paragraphs are divided into sentences using delimiters like ".", "!", "?" followed by a gap.

Stemming

Stemming is the process for reducing inflected (or sometimes derived) words to their stem, base or root form – generally a written word form. The stem need not be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root. The process of stemming, often called conflation, is useful in search engines for query expansion or indexing and other natural language processing problems.

In this project we used Porter Stemmer Algorithm. The Porter stemming [7] algorithm (or 'Porter stemmer') is a process for removing the commoner morphological 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.

THE PORTER STEMMER ALGORITHM

A consonant in a word is a letter other than A, E, I, O or U and other than Y preceded by a consonant. (The fact that the term `consonant' is defined to some extent in terms of itself does not make it ambiguous.) So in TOY the consonants are T and Y and in SYZYGY they are S, Z and G. If a letter is not a consonant it is a vowel.

A consonant will be denoted by c and a vowel by v. A group of consonants of length greater than zero will be denoted by C, and a group of vowels of length greater than zero will be denoted by V. Any word, or part of a word, therefore has one of the four forms: CVCV ... C, CVCV ... V, VCVC ... C, VCVC ... V. These may all be represented by the single form [C] VCVC ... [V]. Here the square brackets denote arbitrary presence of their contents. Using (VC) {m} to denote VC which is repeated m times, this may again be written as [C](VC) {m}[V]. "m" will be called the measure of any word or word part when represented in this form. The case m = 0 covers the null word as follows:

m=0 TR, EE, TREE, Y, BY.
m=1 TROUBLE, OATS, TREES, IVY.
m=2 TROUBLES, PRIVATE, OATEN, **ORRERY.**

The rules for removing a suffix will be given in the form (Condition) S1 -> S2.This means that if a word ends with the suffix S and the stem before S1 satisfies the given condition, S1 is replaced by S2. The condition is usually given in terms of m, *e.g.*, (m > 1) EMENT -> .Here S1 is "EMENT" and S2 is null. This would map REPLACEMENT to REPLAC, since REPLAC is a word part for which m = 2. The "condition" part may also contain the following:

*S - the stem ends with S (and similarly for the other letters).

v - the stem contains a vowel.

*d - the stem ends with a double consonant (*e.g.*, -TT, -SS).

*o - the stem ends with cvc, where the second c is not W, X or Y

(*e.g.*, – WIL,-HOP).

Summarization Framework

Contextual Graph

All the non-textual elements like images, tables etc are removed from the document .We use "." as a delimiter to segment the document into sentences and this document is represented as a

contextual graph with sentences as nodes and similarity between them as edges.

Definition 1 Contextual Graph(CG): A contextual graph for a document d is defined as a weighted graph, CG(d) = (V(d),E(d)) where V (d) is a finite set of nodes where each node is a sentence in the document d and E(d) is a finite set of edges where an edge $e_{ij} \in E(d)$ is incident on nodes i, $j \in V(d)$ and is assigned a weight reflecting the degree of similarity between nodes i and j. An edge exists only if its weight exceeds a threshold μ. Edges connecting adjacent sentences in a document are retained, irrespective of the threshold.

An edge weight $w(e_{ij})$ represents contextual similarity between sentences s_i and s_j . It is computed using cosine similarity measure. The weight of each term *t* is calculated using $tf_t * isf_t$ where tf_t is the term frequency and isf_t is inverse sentential frequency *i.e.*, $log(\ n/(n_t+1))$, where *n* is the total number of sentences in the document and n_t is number of sentences containing the term *t* in the graph under consideration. Stop words are removed and remaining words are stemmed before computing these weights. The edges that are having very low similarity value i.e. with an edge weight below a threshold μ(=0.001) are discarded. These edges and their weights reflect the degree of coherence in the summary.

Integrated Graph Generation

The input is the set of documents that are related to the particular topic that we have to search in multi-document summarization. Let χ=(D,T) be a set of documents D on a topic T. For each document in this set χ are combined incrementally which forms a graph called as integrated graph. As these documents are related to a topic T, they will be similar and may contain some redundant sentences *i.e.*, repeating sentences. These redundant sentences are identified and only one of them is placed in the integrated graph. Then the similarities between documents are identified by establishing edges across nodes of different documents and the edge weights of IG are calculated. Thus the integrated graph reflects inter as well as intra-document similarity relationships present in document set χ. Algorithm for integrated graph construction is given in later sections.

Whenever a query related to the particular topic T is given, relevancy scores of each node are computed with respect to each query term. During this process, sentences that are not related to query directly (by having terms), but are relevant are to be considered which can be called as supporting sentences. To handle this, centrality based query biased sentence weights are computed that not only consider the local information i.e., whether the node contains the query terms, but also global information like the similarity with its adjacent nodes. A mixture model is also used to define the importance of a node with respect to a query term in two aspects: the relevance of sentence to the query term and the kind of neighbors it is connected to. Initially each node is initialized to query similarity weight and then these weights are spread to their neighbors via the weighted graph IG. This process is repeated until the weights come to a steady state. Following this idea, the node weights for each node with respect to each query term $q_i \in Q$ where $Q = \{q_1,q_2,...,q_t\}$ are computed using the following equation 1.

$$wq(s) = d\frac{sim(s,q)}{\sum_{m\in N} sim(m,q)} + (1-d)\sum_{v\in adj(s)} \frac{sim(s,v)}{\sum_{u\in adj(v)} sim(u,v)} wq(v)$$

……**(1)**

where $w_q(s)$ is node weight of node s with respect to query term *q*, *d* is bias factor, *N* is number of nodes and $sim(s_i,s_j)$ is cosine similarity between sentences s_i and s_j. First part of equation computes relevancy of nodes to the Query and second part considers neighbors' node weights. The bias factor *d* gives trade-off between the set parts and is determined empirically. For higher values of *d*, more importance is given to the similarity of node to the query when compared to the similarity between neighbors. The denominators in both terms are for normalization. When a query Q is given to the system, each word is assigned weight based on $tf*isf$ metric and node weights for each node with respect to each query term are calculated. Intuitively a node will have a high score value if:

1) It has information relevant to the query and
2) It is connected to similar context nodes which share query relevant information.

If a node doesn't have any query term but is linked to nodes having it, then the neighbor weights are propagated in proportion to the edge weight such that it gets a weight greater than zero. Thus high node weight indicates a highly relevant node present in a highly relevant context and is used to indicate the richness of query specific information in the node.

CTree AND SGraph

For each query word, the neighborhood of each node in IG is explored and a tree rooted at each node is constructed from the explored graph. These trees are called as contextual trees (CTrees).

Definition 2

Contextual Tree (CTree) *: A* $CTree_i = (N_i,E_i,r,q_i)$ *is* defined as a quadruple where N_i and E_i are set of nodes and edges respectively. q_i is i^{th} term in the query. It is rooted at r with at least one of the nodes having the query term q_i. Number of children for each node is at most b (beam width). It has at most (1+ bd) nodes where d is the maximum depth. CTree is

empty if there is no node with query term q_i with in depth d.

CTrees corresponding to each query term that are rooted at a particular node, are merged to form a summary graph (SGraph) which is defined as follows:

Definition 3

Summary Graph (SGraph): For each node r in IG, if there are t query terms, we construct a summary graph SGraph = (N', E', Q') where N' and E' are union
of the set of nodes and edges of $CTree_i$ rooted at r respectively and $Q = \{q_1,q_2,...,q_t\}$.

Scoring Model: CTrees formed from each node in IG are assigned a score that reflects the degree of coherence and information richness in the tree.

Definition 4

CTreeScore: Given an integrated graph IG and a query term q, score of the CTree q rooted at node r is calculated as

$$CTreeScore_q = \beta w_q(r) + \sum_{\substack{(u,v)\in CTree_q \\ u \text{ is parent of } v}} \left[\frac{\alpha w(e_{u,v}) + \beta w_q(v)}{\left(\sqrt{level(v)}\right)} \right] \quad \text{.....(2)}$$

$$\propto = \frac{a}{b} * 1.5 \quad \text{.....(3)}$$

Here *a* is average of top three node weights among the neighbors of *u* excluding parent of *u* and *b* is maximum edge weight among nodes incident on *u*. The SGraphs formed from each node by merging CTrees for all query terms are ranked using following equation and the highest ranked graph is retained as summary.

Definition 5

SGraphScore: Given an integrated graph IG and a query $Q = \{q_1,q_2,...,q_t\}$, score of the SGraph SG is calculated as

$$SGraphScore = \frac{1}{\sqrt{size(SG)}} \sum_{q\in Q} CTreeScore_q \quad \text{.....(4)}$$

The function *size*(*SG*) is defined as number of nodes in graph. Using both edge weights representing contextual similarity and node weights representing query relevance for selecting a node connected to root node, has never been tried before. The summary graph construction is a novel approach which effectively achieves informativeness in a summary.

Summarization Methodology

Based on the scoring model presented in the above section, we design efficient algorithms to automatically generate query biased summaries from text.

Integrated Graph Construction

Integrated graph represents the relationships present among sentences of the input set. We assume that the longest document contains more number of sub topics than any other document and its CG is chosen as a base graph and is added to IG which is empty initially. The documents in the input set are ordered in decreasing order of their size and CG's of each document in the ordered list is added incrementally to IG . There are two important issues that need to be addressed in multi-document summarization.

Redundant sentences are identified as those sentences which have similarity that exceeds threshold λ. This similarity is computed using cosine similarity and λ =0.7 is sufficient in most of the cases. During the construction of IG, if the sentence in consideration is found to be highly similar to any sentence of a document other than document being considered in IG, then it is discarded. Otherwise it is added as a new node and is connected to existing nodes with which its similarity is above the threshold μ.

Sentence ordering in summary affects readability. For this purpose, an encoding strategy is followed where an "id" is assigned to each node in IG such that there is information flow in summary when nodes are put in increasing order of their ids

Encoding Strategy

Initially all nodes in the base graph are assigned ids as follows. The i^{th} sentence is assigned $(i - 1) * \eta$ as *id*. This interval η is used to insert all the nodes from other documents that are closer to *i* (*i.e.*, the inserted node has maximum edge weight with *i* among all nodes adjacent to it). The sentences in an interval are ordered in decreasing order of their edge weights with *i*. When a new node is added to IG, an id is assigned to it. Pseudo code for IG construction is given in Algorithm 1.

Algorithm 1 Integrated Graph Construction

1: **Input**: Contextual Graphs CG_i in the decreasing order of number of nodes
2: **Output**: Integrated graph IG
3: Integrated Graph $IG = CG_0$ {//base graph}
4: Set *id* of each node in IG as described in IG Construction
5: $i = 1$
6: **while** i <= number of *CG's* **do**
7: **for** each node $n \in CG_i$ considered in the document order **do**
8: **if** parent (*n*) precedes *n* in the i^{th} document then {//parent is the maximum weighted adjacent node in CG}
9: Let p = node representing parent (n) in IG

10: **if** there is no neighbour x of p such that *sim (n, x)* > λ **then**
11: for all y in IG, if *sim (n, y)* > μ then add an edge *(n, y)*
12: Set *id* of n as described in IG Construction
13**: end if**
14: **else if** there is no node x in IG such that *sim (n, x)* > λ **then**
15: for all y in IG, if *sim (n, y)* > μ then add an edge*(n, y)*
16: Set *id* of n as described in IG Construction
17: **end if**
18: **end for**
19: i + +
20: **end while**

If the input document set is singleton set, then the integrated graph is equivalent to the contextual graph of that document. Addition of any new document to the set can be reflected in IG by adding its CG as described above and the edge weights are updated accordingly. The integrated graph for the set of documents can also be computed offline and stored. When a query is posed on a document set, its IG can be loaded into memory and CTrees and SGraphs can be constructed as described above.

CTree Construction

The neighborhood of a node is explored and prominent nodes in it are included in CTree rooted at *r*. This exploration is done in breadth-first fashion. Only *b* (beam width) prominent nodes are considered for further exploration at each level. The prominence of a node *j* is determined by taking the weight of the edge connecting *j* to its parent *i* and its node score with respect to the query term q into consideration. It is computed as $(\alpha w(e_{ij}) + \beta w_q(j))$. These two factors specify the contribution of the node to the coherence of the summary and the amount of query related information. α is the scaling factor defined in Equation 3. This scaling brings edge weights into the range of node weights and β determines trade-off between coherence and importance of query relevant information. The exploration from selected prominent nodes (at most *b*) is continued until a level which has a node containing a query term (anchor node) or maximum depth *d* is reached. All nodes along the path from root to anchor node, along with their siblings are added to the CTree. When query Term is not found till depth *d* then CTree for that query term remains empty. If a root node has the query term then root and its adjacent "*b*" nodes are added to CTree and no further exploration is done.

SGraph Construction

The CTrees of the individual terms are merged to form an SGraph. The SGraph at a node contains all nodes and edges that appear in any one of the C Trees rooted at that node. With this, completeness is ensured as CTrees of all query terms are merged to form an SGraph and also as we are merging CTrees rooted at a node, we will have interconnected set of sentences in the summary and hence coherence is preserved. The SGraphs thus formed are ranked based on the score computed as given in Equation 4. Sentences from the highly ranked SGraph are returned as summary.

Experimental Results

In the experiments, QTS was compared with two query specific systems - baseline and MEAD. Our baseline system generates summaries by considering only centrality based node weights as per equation 1 using incremental graph, without generating CTrees and SGraphs. Nodes which have high weights are included in summary. Second system, MEAD is a publicly available feature-based multidocument summarization toolkit. It computes a score for each sentence from a cluster of related documents, as a linear combination of several features. For our experiments, we used centroid score, position and cosine similarity with query as features with 1,1,10 as their weights respectively in MEAD system. Maximal Marginal Relevance(MMR) reranker provided in the MEAD toolkit

Was used for redundancy removal with a similarity threshold as 0.6. Equal number of sentences as generated by QTS were extracted from the above two systems.

QTS used four criteria to evaluate the generated summaries. They are non-redundacy, responsiveness, coherence ad overall performance. Summaries generated by three systems QTS, baseline nd MEAD were evaluated by

A group of 10 volunteers. They were given a set of instructions defining the task and criteria and were asked to rate each summary on a scale of 1(bad) to 10(good) for each criteria without actually seeing the original documents. An average of these ratings for each query was computed and mean of them was calculated. The graph shows that QTS performs better when compared to other systems.On the whole QTS performed better than others in terms of user satisfaction.

Conclusion

A novel framework for multi-document summarization system that generates a coherent and intelligible summary. We propose notion of an integrated graph that represents inherent structure present in a set of related documents by removing redundant sentences. Our system generates query term specific contextual trees (CTrees) which are merged to form query specific summary graph (SGraph). We have introduced an ordering strategy to order sentences in summary using integrated graph structure. This process of computation has indeed improved the quality of the summary. We

experimentally show that our approach is feasible and is generating user satisfiable summaries.

References

1. M. Sravanthi, C. R. Chowdary, and P. S. Kumar. QTS: A query specific text summarization system. In Proceedings of the 21st International FLAIRS Conference, pages 219–224, Florida, USA, may 2008. AAAI Press
2. Radev, D. R.; Jing, H.; and Budzikowska, M. 2000.Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In NAACL-ANLP 2000 Workshop on Automatic summarization,21–30. Morristown, NJ, USA: Association for Computational Linguistics.
3. Salton, G.; Singhal, A.; Mitra, M.; and Buckley, C. 1997.Automatic text structuring and summarization. Inf. Process. Manage. 33(2):193–207
4. Erkan, G., and Radev, D. R. 2004. LexPageRank: Prestige in multi-document text summarization. In EMNLP.
5. Fisher, S.; Roark, B.; Yang, J.; and Hersh, B.2005. OGI/OHSU Baseline Query directed Multidocument Summarization System for DUC-2005. Proceedings of the Document Understanding Conference (DUC).John M. Conroy, J. D. S., and Stewart, J. G. 2005.CLASSY query-based multi-document summarization.Proceedings of the Document Understanding Conference (DUC).
7. http://tartarus.org/~martin/PorterStemmer/