TOPIC-DRIVEN DOCUMENT SUMMARIZATION USING ONTOLOGY KNOWLEDGE

by

SONU SWAIKA

(Under the Direction of Krzysztof J. Kochut)

ABSTRACT

Topic-driven summarization aims at extracting a summary from a single document or a document collection based on a given topic. This thesis presents an extractive, ontology-based approach to topic-driven summarization. We make use of an ontology formed out of the information present in Wikipedia. Given a document and one or more topic terms/phrases, our summarization system generates a topic-related summary. To produce a good summary which contains information related to the topic, it is important to understand the topic. We first expand the initial topic by using the Wikipedia ontology. The document is represented as a graph using entities from the Wikipedia ontology. A Spreading activation algorithm is applied to find all nodes in the document graph that are semantically related to the expanded topic terms. This determines the relative importance of nodes by assigning a weight to each node. These weights are used to decide which sentences should be included in the extractive summary.

INDEX WORDS: Semantic Web, Extractive Summarization, Topic Expansion, Ontology

TOPIC-DRIVEN DOCUMENT SUMMARIZATION USING ONTOLOGY KNOWLEDGE

by

SONU SWAIKA

B.E., Birla Institute of Technology & Science, Pilani, India, 2007

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillments of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2010

TOPIC-DRIVEN DOCUMENT SUMMARIZATION USING ONTOLOGY KNOWLEDGE

by

SONU SWAIKA

Major Professor :	Krzysztof J. Kochut
Committee:	John A. Miller Thiab Taha

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
August 2010

TABLE OF CONTENTS

CHAPTER

CHAPTER 1

INTRODUCTION

Due to the rapid growth of the Internet, there has been an explosion in the number of documents present on the World Wide Web. As a result, it is often difficult to find the exact information from such enormous collection of documents. Typically, users skim through several documents to find the information they are searching for. Summaries can help the users in achieving this objective.

A summary is a shorter version of the original document. It is a brief description of the original document outlining the most important information present in the document. Therefore, a reader need not read the full text but can only read the summary to find and gather information. It is easier and faster for users to read and browse the summary as compared to the original documents.

Summaries may be produced from a single document or multiple documents. Multi-document summarization extracts important information from multiple documents that are about the same topic. A multi-document summarization system needs to handle some additional issues like redundant information from multiple documents, coherence in the summary etc.

Single Document vs Multi-Document Summarization

Although many of the methods used in single-document summarization can also be used in multi-document summarization, there are some major differences:

- In multi-document summarization, it is likely that the information contained in different documents may overlap. Hence, steps should be taken to eliminate redundant information from a multi-document summary.
- If a document set contains information of a temporal nature (for example, a collection of news articles about a particular event) such that some articles give more complete information compared to others, multi-document summary of such articles should take this temporal information into account by allowing the more recent and complete information to be a part of the summary.
- The Compression ratio (ratio between the size of the summary and the size of the document set) for multi-document summarization drops as the number of documents increases. Document sets containing hundreds of documents may have a compression ratio of 1% or even less than that. Summarization becomes exceedingly difficult with such low compression ratio.

Applications of Document Summarization

A single-document summarization is useful to summarize research documents. While conducting research, we often have to browse a large collection of potentially useful documents and decide if a document is useful or not. Many of these documents may not contain an abstract, and therefore we need to read or skim through the full-length document. Moreover, an abstract usually gives an overview of what the document is talking about and may not contain the most important information present in the document. Hence, if we can automatically produce a summary of such documents, researchers can just read the summary to get an idea about the most important points present in a document and decide whether that document is useful or not.

Multi-document summarization is useful for news aggregation websites. Most online news services collect articles from different news agencies. They form a cluster of related articles and present them to the reader. For example, news related to business is presented under ‘Business’ cluster. News stories present in a cluster can have overlap in their contents. Therefore, a reader might have to read the same content multiple times. Wouldn’t it be nice to have a short summary of all these news articles that does not contain any redundant information? Multi-document summarization can be used to generate such short unified summary.

Extraction of a single summary from multiple documents can also be used in Internet search engines. Even though a search engine may give relevant results, there is usually a long list of documents retrieved. As a result, the user needs to check the retrieved documents one by one. Instead, if we could present a small summary of the most relevant documents, the user might find the desired information inside the summary. “Ideally, multi-document summaries should contain the key shared relevant information among all the documents only once, plus other information unique to some of the individual documents that are directly relevant to the user’s query” [1].

Classification of summaries

A summarization system could be either extractive or abstractive.

- Extractive summarization selects sentences from the original document(s) and concatenates them to form a summary.
- Abstractive summarization produces an abstract containing new sentences which together concisely describe the information present in the document(s).

Another classification of summarization systems is based on whether the summary is generic or topic-driven:

- Generic summarization highlights the most important information in a document or a collection of documents.
- Topic-driven summarization is concerned with summarizing document(s) with a concentration toward a particular external request (i.e. query, question, topic, etc.), or focus [2].

We present here an extractive, ontology-based approach to topic-focused summarization. We make use of an ontology formed out of the information present in Wikipedia. Given a document and one or more topic terms/phrases, our summarization system generates a topic-driven summary.

It is often difficult to express a topic using a small number of terms/phrases. Therefore, prior to summarization, the topic formulation module is used to formulate the topic. The topic graph generated by this module is used for summarization.

The terms in the document are matched against the entities present in the ontology to form a document graph. The document graph construction is one of the most important step in our summarization system. Graph algorithms such as spreading activation and HITS are used to assign weights to the entities in the document graph. After the execution of graph algorithms, the sentences in the original document are scored by adding the weights of the entities. Highly ranked sentences are then chosen to form a summary of the desired length.

The rest of the thesis is organized as follows: Chapter 2 presents the background on this thesis explaining the basic concepts. The related work for summarization is presented in Chapter 3. Chapter 4 describes the method for topic formulation. Chapter 5 presents an ontology-based topic-focused summarization method and a prototype summarization software system. Chapter 6 includes the experiments conducted and evaluation of the results obtained. Chapter 7 presents the conclusions and the future work.

CHAPTER 2

BACKGROUND

2.1 Summarization

Radev et al. [10] define a summary 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”. A summary may be produced from a single document or multiple documents. It should preserve important information and should be short.

As discussed earlier, a summary could be either generic or topic-driven. Generic summarization highlights the most important information in a document or a collection of documents. Topic-driven summarization on the other hand is concerned with summarizing document(s) with a concentration toward a particular external request (i.e. query, question, topic, etc.), or focus. For example, consider a document that talks about the TV series “Lost”. If a user wants to get an idea of what the document says about the character Jack Shephard, he/she would want a summary that focuses only on Jack Shephard. In our work, we focus on extractive topic-driven summarization.

When a summary is generated automatically, there is usually a loss of coherence and readability. This is especially true in case of extractive summarization since the sentences are picked from different parts of the document and combined to form a summary. Therefore, in case of extractive summarization relevance becomes more important than coherence. Though the summary jumps abruptly from talking about one

thing to something completely different in the next sentence, the sentences are relevant and are therefore chosen to be a part of the final summary.

2.2 Ontology

The term “ontology” comes from the field of philosophy. Wikipedia [3] describes an ontology as “the philosophical study of the nature of being, existence or reality in general, as well as of the basic categories of being and their relations”. In computer and information science, Tom Gruber [4] defines ontology as an “explicit specification of a conceptualization” which is, in turn, "the objects, concepts, and other entities that are presumed to exist in some area of interest and the relationships that hold among them." Ontologies generally describe individuals (instances), classes (concepts), attributes, and relations . An ontology can be represented in the form of a graph where nodes are entities and the edges are relationship between entities.

Ontologies are used in Semantic Web, Artificial Intelligence, biomedical informatics etc. as a form of knowledge representation in general or in some particular domain. An ontology which describes general entities and is not tied to a particular domain is known as a foundation ontology or upper ontology. Examples of standardized upper ontologies include Dublin Core (DC), OpenCyc, GFO, SUMO etc. Dublin Core ontology is a specification for the description of digital media such as text, image, video, sound and even composite media like web pages. The DC metadata element set is a standard in the fields of library and computer science and is widely used to describe digital materials. It consists of concepts such as ‘title’ , ‘creator’, ‘publisher’ etc.

A domain ontology represents the meaning of terms as they apply to a specific domain. For example, the word ‘card’ could mean “playing card” in the domain of card game or it could mean “video card” in the domain of computer hardware. Building domain-specific ontologies requires the assistance of a domain expert. A lot of ontologies specific to biological topics can be found at Open Biomedical Ontology website. Gene Ontology is a domain-specific ontology that contains information about genes.

An ontology language is a formal language used to construct ontologies. It allows us to encode knowledge in general or in a specific domain. It also includes reasoning rules in order to process that knowledge. Ontology languages are usually declarative languages, and are based on either first-order logic or on description logic.

Resource Description Framework (RDF) is a standard model for data interchange on the Web. RDF Schema (RDFS) is a language for RDF vocabulary sharing. Though RDFS provides some support for ontology specification, it is not very expressive.The Web Ontology Language (OWL) is an ontology language that builds on RDF and RDFS and is much more expressive i.e. it can be used to express complex relationship and constraints between entities. Classes, properties and individuals in OWL are defined as RDF resources and are uniquely identified by a Uniform Resource Identifier (URI). There are ontology libraries which provide lists or directories of ontologies with search facility. SchemaWeb [5] is a directory of ontologies in RDFS, OWL and DAML+OIL. Protege Ontology Library [6] contains a set of OWL ontologies and ontologies in other formats like RDFS, DAML+OIL etc.

2.3 Wikipedia and Wikepedia derived Ontology

A wiki is a website that allows the creation and editing of interlinked web pages using a WYSIWYG text editor on a web browser. Wikipedia (a combination of the word wiki and encyclopedia) is a free, multi-lingual encyclopedia. It has more than 3.2 million articles in English. It is a huge source of encyclopedic knowledge that is highly interconnected and categorized. The articles are written collaboratively by volunteers around the world. It is an unrestricted wiki, so anyone can create a new article and most of the articles can be edited by anyone. Wikipedia articles cover wide range of topics including geography and places, thought and philosophy, biographies and persons etc. A Wikipedia article is almost immediately created for any new event, discovery or invention. Around 1,300 articles are added to the Wikipedia knowledge base everyday as of August 2009 [7]. There are so many people using WIkipedia that erroneous information is often caught quickly. Hence, Wikipedia is an up-to-date source of information.

Recently, encyclopedic knowledge in Wikipedia is being used to perform a variety of text analysis tasks. Wikipedia articles mostly contain unstructured data in the form of free text, but also include structured information in the form of wiki markup like infobox templates, category information etc. We can extract the structured information from Wikipedia and form an ontology. The ontology formed from Wikipedia would be generic in nature since Wikipedia contains articles on varied topics and covers a wide area of general knowledge.

The DBpedia project [8] extracts structured information from Wikipedia and forms an ontology. Each article in Wikipedia is assigned a Unique Resource Identifier (URI) and is represented as an entity in the ontology. All Wikipedia articles have a title, and is

represented in the ontology using the property rdfs:label. Similarly, other information like disambiguation, page links etc are extracted from a Wikipedia article and represented using properties.

```
{{Infobox actor
| birthdate = {{birth date and age|mf=yes|1970|10|8}}
| birthname = Matthew Paige Damon
| birthplace = [[Cambridge, Massachusetts|Cambridge]], [[Massachusetts]], United States
| spouse ={{marriage|Luciana Barroso|2005}}
| occupation = Actor, screenwriter, producer
```

Figure 1: Infobox Matt Damon

A Wikipedia article might contain an infobox that describes the most important information about that article. Info-boxes are present on the top right-hand side of a Wikipedia article. Information present in info-boxes are structured in the form of attribute-value pairs. Figure 1 describes an excerpt of the wiki markup describing an infobox about Matt Damon. DBpedia extracts information from an infobox by using two approaches in parallel: A generic approach which provides wide coverage and a mapping-based approach which gives high data quality.

Our system uses an ontology which is formed from the English Wikipedia dump dated 2010-02-11. We use the ontology construction process by Maciej Janik in the LSDIS lab at University of Georgia [9], [17]. It uses a slightly modified approach as compared to DBpedia ontology creation. It differs in the way templates are extracted and how the literal values are assigned.

As discussed above, DBpedia does special handling of infobox templates while other templates present in a page do not receive such special handling. A general template might contain links to other entities. In DBpedia, for each general template, a new entity is created which further links to other entities. In Wikipedia ontology, entities present in the template are directly linked from the page, thereby eliminating the need of intermediate entities.

In our Wikipedia ontology, separate properties exist to distinguish between direct names, disambiguation and redirections. In Wikipedia articles, disambiguation is done by adding contextual information to the article name. For example, ‘orange’ and ‘orange (fruit)’. In documents, context information is not provided in parenthesis since readers can perform the disambiguation based on the content. Therefore, in our ontology, we remove contextual information to get shorter names. The shorter names are then added as an alias name of a different property to distinguish it from its full name.

CHAPTER 3

RELATED WORK

Although summarization has gained interest in the research community in recent years, it is not a novel task. In 1958, Hans Peter Luhn [11] used term frequency to identify important words in a text. The sentences in the text were scored based on the presence of these important words, and high ranked sentences were then selected to be part of the summary. In 1969, Edmundson [13] proposed that term weights can also depend on their location in the text or their presence in a cue phrase list. For example, indicator phrases like 'in conclusion', 'it is important' etc. signal important sentences in a text, and are therefore given higher weights. In 1961, Rath et al. [12] studied that the most important evaluation issue related to automatic summarization is that of human agreement since different people can choose different sentences for their summaries.

Since mid 1990's, new approaches have emerged in the area of automatic text summarization. Some of the approaches used these days are corpus-based, discourse-analysis based and knowledge-rich methods. These approaches build on existing methodologies, tools and resources developed over the past decades. In 2001, the Document Understanding Conference (DUC) was established to further the progress in summarization and enable researchers to participate in large-scale experiments [14]. DUC provides a wealth of data for text summarization research. DUC discontinued the single-document summarization task after 2002. Only multi-document summarization systems have been participating in the DUC conference from 2003 onwards. Since 2008, DUC has moved to Text Analysis Conference (TAC).

Since our work focuses on topic-driven summarization, we will discuss some topic expansion methods followed by some text summarization methods.

3.1 Overview of Topic Expansion

Topic expansion, also known as topic formulation, refers to adding new meaningful terms to the initial topic. Due to the ambiguity of natural language, it is difficult to express an information concept using a single term. Therefore, it is required to formulate a topic by adding contextual information to the initial topic. The topic expansion method could be either automatic, manual or user-assisted.

Topic expansion using relevance feedback

Relevance feedback is a well known technique to expand an initial topic by adding words from identified relevant documents [19]. The user enters an initial topic and is presented with a list of ranked documents. From this list, the user needs to select the most relevant documents. New terms from the relevant documents are added to the initial topic. When selecting new terms, several factors like sample size of relevant documents, weighting of new terms, how many new terms to include etc. should be taken into account. This method requires user assistance. An alternative approach could be pseudo relevance feedback in which the first few ranked documents are assumed to be relevant.

Corpus dependent topic expansion

Clustering and term co-occurrence are two popular methods that use the document collection to obtain context for a given topic. Term co-occurrence refers to two or more terms that are present close to each other in the document. In term clustering, clustering

algorithms are applied to the document collection. This might form several clusters. Each cluster contains similar documents. If a topic maps to one or more clusters, terms from that cluster(s) are used to expand the topic. If the document collection is small, then the quality of the clusters formed would be poor and hence the topic expansion would not give desired results [20]. Topic expansion using corpus dependent knowledge models is not suited for dynamic document collections since the knowledge models would have to be constantly updated. Therefore, this method is effective only for static collections as the knowledge models could be pre-generated.

Corpus independent topic expansion

The relevance feedback and corpus dependent methods require a document collection to expand the topic. They both require sufficient relevant documents and moreover the documents should contain terms related to the topic. These are some of the major drawbacks of corpus dependent models. Corpus independent models on the other hand do not use a document collection. Instead, they rely on external knowledge in the form of a thesaurus or an ontology.

Ontologies used in corpus independent methods could be either general or domain-specific. Some examples of general ontologies are WordNet [15], EuroWordNet and Cyc. There are several domain specific ontologies in medical and legal domain.

Suomela and Kekalainen (2005) compare topic formulation with the use of CIRI ontology (Concept based Information Retrieval Interface) and without using an ontology. The topic expansion resulted in higher number of search terms but their work requires that the user should be well familiar with the ontology. Also, the results show that the CIRI interface is complex to use.

In our system, we use corpus independent topic expansion that uses Wikipedia ontology to determine semantic relationship between words or phrases.

3.2 Overview of Summarization Methods

Using Lexical Chains for Text Summarization

Barzilay & Elhadad [16] use Wordnet to form lexical chains. Lexical chains represent lexical cohesion between two words or a sequence of related words. The original text is first segmented followed by the construction of lexical chains. The strong lexical chains are then selected, and the sentences which are part of the strongest chains are extracted to form the summary.

A contextual query expansion approach by term clustering for robust text summarization

Amini & Usunier [18] use clustering for query expansion. Term clusters are formed by using an unsupervised algorithm, and the topic is expanded with respect to one or more of these clusters. Once the topic is expanded, summary generation requires four steps:

- The expanded topic is compared against each sentence in the document collection. Marcu's alignment technique is used to select relevant sentences and approximately 3/4 of the sentences are discarded at this step.
- The remaining sentences are then scored based on 4 heuristic features.
- The first draft of the summary contains 10 highest scoring sentences
- A post-processing step is applied to remaining sentences to generate a final summary of around 250 words.

This method uses corpus dependent model to expand the topic internally. We, on the other hand, use corpus independent topic expansion that uses the Wikipedia ontology.

Topic-based Summarization and Maximal Marginal Relevance (MMR)

Maximal Marginal Relevance (MMR) was introduced by Carbonell and Goldstein in 1998. MMR aims to increase the relevance of the sentences to the query, while maintaining minimum redundancy in the summary. With MMR, sentences and documents could be represented as vectors obtained by Vector Space Model (VSM) or Latent Semantic Analysis (LSA).

Consider Q as user query and R as a ranked list of documents. We incrementally select documents from R and add them to S. So, S is the set of already selected documents, and R/S is the set of unselected documents. For each document Di from R/S, its marginal relevance is computed as:

$$MR(Di) = \lambda \cos(Di, Q) - (1 - \lambda) \max \cos (Di, Dj) \text{ where } Dj \in S$$

where λ is a parameter between 0 and 1 that controls the relative importance given to relevance versus redundancy. The similarity between vectors is measured by computing the cosine of the angle between these two vectors. The cosine similarity between two vectors A and B is calculated as follows:

$$\cos(A,B) = \frac{A \cdot B}{|A|\,|B|}$$

Here, "." denotes the dot-product of the two vectors.

The document achieving the highest marginal relevance is selected from R/S and added to the set S. The above procedure is applied repeatedly until the relevance threshold is reached or the maximum number of documents are selected.

It was experimentally found that dynamically choosing the value of λ is more effective than keeping it fixed. A small value of λ (0.3) will result in selection of documents

containing new information while a higher value will focus more on relevant documents. In the context of summarization, documents can be first divided into sentences. MMR algorithm is then applied to select sentences. Once the sentences are selected, they are reordered and presented as summary. Thus, MMR can be very effective in generating topical summaries.

Extractive Summarization Based on the Document Semantic Graph

This system was developed by Jurij Leskovec, Natasa Milic-Frayling and Marko Grobelnik at the Microsoft Research Center. [24] is an extractive summarization system that builds a semantic graph of the document and runs ranking algorithms on it.

A semantic graph is created by performing deep syntactic analysis of the document to extract the triples of the form subject-predicate-object and represent them as a connected graph where the subjects and objects are nodes of the graph and predicates are edges between them. Support Vector Learning is used to learn a model which is used to generate another semantic graph that would contain triples that have a high probability of being part of the summary if a human were to create it. The Support Vector Learning uses some human-generated summaries to learn the model. The semantic graph obtained is then used as an input to the graph-based ranking algorithms such as HITS and PageRank to measure node importance values. Sentence salience is calculated by picking those sentences that have highest weights when their corresponding set of entity weights are summed.

Our system also uses a semantic graph of triples formed out of the document. The difference is in the manner in which the semantic graph has been created. In this

system, deep syntactic analysis is performed to extract the triples from the document. We, on the other hand, extract triples from an existing ontology for the thematic graph formation. To overcome false positives from syntactic analysis, the Support Vector Machine or SVM classifier is used to build a model. Our system is completely unsupervised in this respect.

A Semantic Free-text Summarization System Using Ontology Knowledge [23]

This system generates a summary of the medical documents based on the user query. A semantically connected concept network is created for the original document using the Unified Medical Language System [32] or UMLS ontology. This network contains only those terms which appear in the ontology. The topic is also formulated using UMLS or WordNet ontology. The distance between each sentence and the revised topic is calculated. The sentences having the least distance values are selected as candidate sentences. Intra-pair distances are calculated for the candidate sentences and divided into groups based on some threshold value. Highest-ranked sentences are then selected from each group and put together as the final summary. Though our system is quite similar in the use of an ontology, we use graph-based methods to calculate importance of sentences as opposed to finding sentence distance from the user-query.

Multi-document summarization by graph search and matching [22]

Mani and Bloedorn (1997) [25] describe a topic driven method for summarizing similarities and differences in a pair of related documents using a graph representation for text. Words, phrases, and proper names in the text are represented as nodes in the graph.

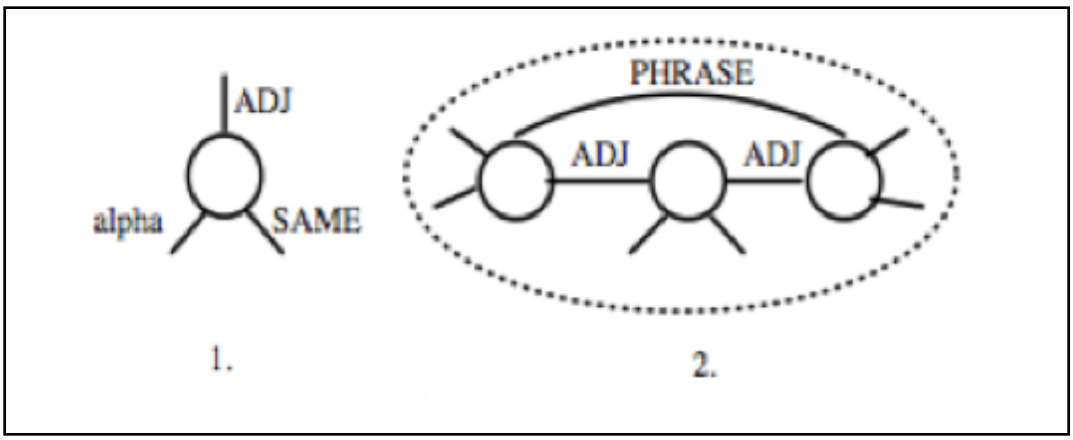

Figure 2: Graph Representation (Figure from Mani and Bloedorn, 1997)

As shown in Figure 2, a node can have three kinds of links: ADJ, SAME and alpha. Adjacency links (ADJ) link to adjacent words in the text, SAME links link to other occurrences of the same word in the text, and ALPHA links link to semantic relationships. The semantic relationships are found using WordNet and NetOwl. A PHRASE link ties together sequence of adjacent nodes that belong to the same phrase.

Stemming algorithms are applied to the topic terms to reduce them to their base form. The stems of the topic terms are then identified in the document graph, and these nodes become the entry points into the graph. Spreading activation algorithm is then applied to this graph which aims to find all the nodes that are semantically related to the entry nodes. For ADJ links, neighboring weights is an exponentially decreasing function of activating node weight and distance of the path between them.

For a pair of document graphs, common nodes are the ones that have the same stem or are synonyms. On the other hand, difference nodes are the ones that are not common. For each sentence, weights of common nodes are added to compute a score

and weight of difference nodes are added to get another score. These scores reflect the presence of common and difference nodes respectively in that sentence.

Once the spreading activation algorithm is over, sentences with higher common scores and higher difference scores are chosen. These similarities and differences in a pair of related documents could be used to generate a summary.

Topic-Driven Multi-Document Summarization with Encyclopedic Knowledge and Spreading Activation [25]

In this paper, the topic is first preprocessed using Stanford parser [27] to extract named entities (NEs). Atomic elements in text are classified into predefined categories such as names of organizations, persons, locations etc. After preprocessing, the topic is expanded using the following two methods:

Topic Expansion with Wikipedia - The NEs are matched with Wikipedia articles. The text of the hyperlinks in the first paragraph of these articles are treated as related concepts.

Topic Expansion with WordNet - The nouns and NEs from the topic are expanded with hypernyms, hyponyms and antonyms in WordNet. If a word or NE has multiple senses, the sense(s) which have the highest overlap with the query is chosen.

A large document graph is built that covers the entire document collection. The document graph consists of nodes corresponding to words and NEs in the documents. The edges in the graph correspond to grammatical dependency relations between nodes. The Spreading Activation algorithm is used to assign weights to the edges. The PageRank algorithm is then run on this graph with weighted edges. The PageRank algorithm assigns weights to the nodes. Finally, the sentences are scored based on the

presences of topic words, topic expanded words or top ranked nodes. Similar to the methods above, sentences with highest score are selected one by one until the length of the summary reaches the desired length.

CHAPTER 4

TOPIC FORMULATION

In chapters 2 and 3, we gave some background about topic formulation, summarization, Wikipedia and ontology. Given a document and one or more topic terms/phrases, our summarization system generates a topic-driven summary. As mentioned before, it is difficult to express an information concept using one or more topic terms/phrases. Therefore, prior to summarization, the topic formulation module is used to formulate the topic. The topic graph generated by this module is used for summarization as discussed in the next chapter. In this chapter, we present our methodology for topic formulation. Section 4.1 describes BRAHMS which is a fast main-memory RDF/S storage, capable of storing, accessing and querying large ontologies [28]. Section 4.2 describes our topic formulation module.

4.1 BRAHMS

"BRAHMS was created as a framework for testing graph search algorithms for SemDis project, where there is a need for storage that can execute graph search algorithm quite fast on big RDF data" [28]. Storage systems with database backend are much slower because of disk or database access. Therefore, to get fast access and querying, the only solution is to use a memory based model. RDF/S storage systems like Jena, Seasame or Redland can be used in memory-based model but it works well only with small ontologies. Therefore, BRAHMS was designed as a fast main-memory based RDF/S storage system that can handle large ontologies. It is implemented in C++ for

efficient performance. C++ also gives a good degree of control over memory management.

BRAHMS loads the data into memory by loading a memory snapshot file. Therefore, creating a snapshot is a required step for using data in BRAHMS. We use a program called snapshotCreator [28] to create a memory snapshot file from ontology files. The snapshotCreator program loads the ontology files and parses them into internal memory structures. The snapshot is then incrementally saved to disk.

BRAHMS is based on read-only memory model since it helps in optimizing memory usage. Also, in a read-only model the indexes can be created all at once. However, if the ontology changes, BRAHMS must be stopped and restarted with an updated memory snapshot created from the new ontology.

As discussed earlier, an ontology can have instances (individuals), concepts (classes/ schema) etc. Therefore, BRAHMS uses different resources to represent instances, schema etc. Some examples of resources/nodes in BRAHMS are:

- *Instance Node* - concrete instance in an ontology
- *Literal* - literal value associated only with instance nodes
- *Schema Class* - class instance in schema
- *Schema Property* - property defined in schema

There are different statement types that interlink nodes with each other. Some examples of statements in BRAHMS are:

- *Instance statement* - statements of the form [instance node] [schema property] [instance node]
- *Literal statement* - statements of the form [instance node] [schema property] [literal]
- *Schema Class Statement* - statements of the form [schema class] [schema property] [schema class]

In our system, we use BRAHMS as the backend RDF storage for the Wikipedia ontology. The Wikipedia ontology construction was described in chapter 2 and it uses English Wikipedia dump from 2010-02-11. We use snapshotCreator to create a snapshot from the ontology. The snapshot created was over 14 GB in size. Once the snapshot is loaded, we use BRAHMS API [28] to access Wikipedia ontology.

4.2 Topic Formulation Module

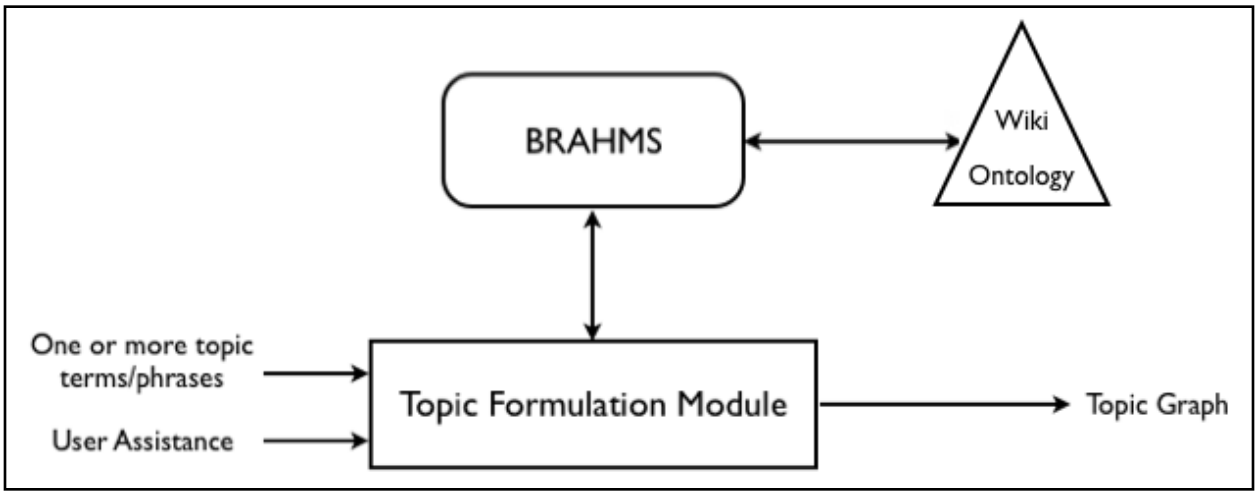

Figure 3: Topic Formulation Block Diagram

As shown in Figure 3, the topic formulation module takes as input one or more topic terms/phrases and gives a topic graph as an output. The encyclopedic knowledge in Wikipedia is used to formulate the topic. As discussed above, we use BRAHMS API to

access the knowledge present in Wikipedia ontology. User assistance may also be required to formulate the topic.

Let us look at an example that will formulate the topic for following topic terms/phrases: *“Tennis”, “Roger Federer”, “Leander”,”US Open”*

The term “*Leander*” is ambiguous since it could mean Leander Paes (tennis player), Leander Starr Jameson (a British colonial statesman) etc. Based on other topic terms, our module is able to detect that by Leander, the user probably means “Leander Paes”. Therefore, our module reconfirms it from the user by asking a question that does the user mean Leander Paes? This is an example where user assistance is required. After the user enters ‘yes’, the following topic graph is generated.

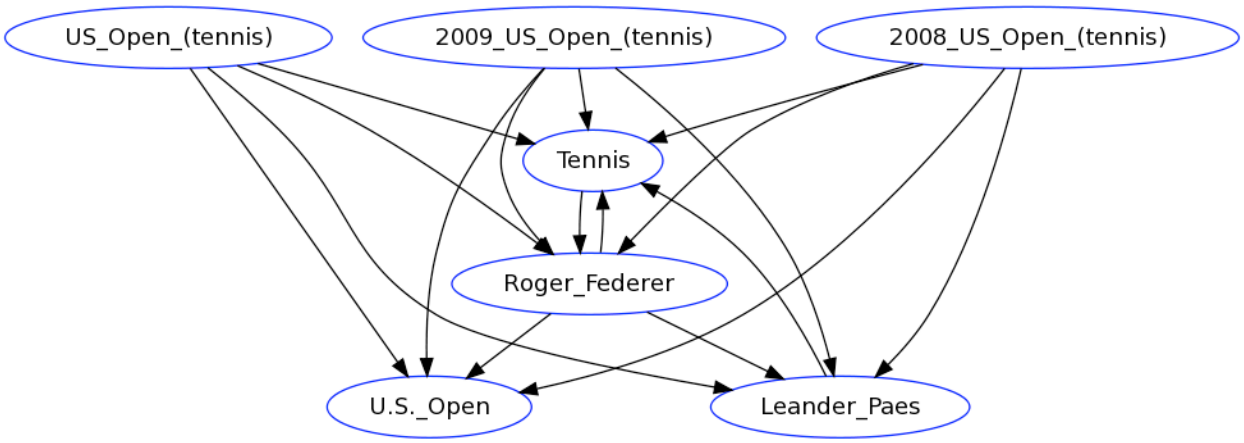

Figure 4: Topic graph for the phrases *“Tennis”, “Roger Federer”, “Leander”,”US Open”*

The edges between entities represent the relationship/links between those entities in Wikipedia ontology. These links could be anonymous (href links) or it could be the named links (for example, links extracted from infobox templates).

In Figure 4, we can see that other than the original four topic phrases, three more entities (US Open (tennis), 2009 US Open (tennis), 2008 US Open (tennis)) have been added to the topic graph. As shown in the figure, these entities are strongly connected to all the topic phrases.

Below is a step-by-step procedure to formulate the topic.

The named entities represented by the topic terms/phrases are identified by matching the topic terms with the names of the entities present in the ontology. Entity names in an ontology are usually values of a certain property. In our topic formulation module, we use three such properties.

• lsdis:wiki_name

• lsdis:wiki_redirect_name

• lsdis:wiki_disambiguation_name

The first step is to match the topic terms with the entities using the “wiki_name” property. These entities are added to the list of direct entities. In Wikipedia, a search for a specific phrase may be redirected to a page with a title that is more common and more popular name for the search phrase. For example, a search for “USA‟ takes you to a redirection page for “United States‟. If the topic terms match with the entities using the “wiki_redirect_name” property, these entities are also added to the list of direct entities. If at this stage, the number of direct entities is zero, the user is asked to re-enter the topic phrases. In the above example, “*Tennis*”, “*Roger Federer*” and “*US Open*” are examples of direct entities as these entities were identified by matching either “wiki_name” or “wiki_redirect_name” property.

In Wikipedia articles, disambiguation is done by adding contextual information to the article name. For example, the fruit orange is denoted as ‘orange (fruit)’ since orange could mean orange color, Orange city in California etc. In documents, context information is not provided in parenthesis since readers can perform the disambiguation based on the content. Therefore, a topic phrase like orange could match multiple entities using the “wiki_disambiguation_name” property. These entities are put into a list of ambiguous entities. User assistance is required to disambiguate entities at this stage. There might be a lot of entities in the ambiguous list, therefore instead of getting user assistance for each one of those entities, we use the relationship and neighbor information between direct and ambiguous entities to guess which ambiguous entities might be relevant.

In the second step, we use the relationship information between direct and ambiguous entities. In the above example, “*Leander*” might match ten ambiguous entities of which “*Leander Paes*” is one. If there exists a direct relationship between “*Leander Paes*” and any of the direct entities, we ask the user that by “Leander”, do they mean “*Leander Paes*”? Therefore, instead of asking the user ten times, we us the relationship information to determine that “*Leander Paes*” might be relevant to the user query. If the user enters yes, the entity is added to the list of direct entities. If not, we move on to the next step where we use the common neighbors between the direct entities and the list of ambiguous entities.

In the third step, for each ambiguous entity, we find the number of common neighbors it shares with the direct entities. The ambiguous entity with the maximum number of common neighbors is assumed to be probably relevant and we confirm it from the user.

If the user enters yes, we add this ambiguous entity to the list of direct entity and proceed until all the topic terms are disambiguated.

At the end of the above steps, we have matched all our topic phrases with the entities. In above example, we got entities like *“Tennis”, “Roger Federer”, “Leander Paes”, ”US Open”* for the topic phrases *“Tennis”, “Roger Federer”, “Leander”, ”US Open”.*

The next step is to further expand the topic by adding some extra entities which might be relevant to the topic but is not given in the topic phrases. We expand our topic by adding entities which are common to all the direct entities. Entities added at this stage are given lower weight as compared to the entities added above. In the above example, there were originally four topic phrases. We added three more entities (US Open (tennis), 2009 US Open (tennis), 2008 US Open (tennis)) which are common to all the topic phrases. We could expand our topic graph to include new entities linked to only few of the topic phrases. Therefore, experimentally we can specify the number of entities we want in our topic graph. However, a small number of entities in the topic graph will give more accurate results.

CHAPTER 5

TOPIC-FOCUSED SUMMARIZATION

In this chapter, we present our extractive topic-driven summarization system. Topic-driven summarization is concerned with summarizing document(s) with a concentration toward a particular external request (i.e. query, question, topic, etc.), or focus.

We use a knowledge-rich approach to summarization. The knowledge comes from the Wikipedia ontology discussed in chapter 2. The original document and the ontology are used to generate the document graph. Graph algorithms like spreading activation and HITS are used to assign weights to the entities in the document graph. Both document graph construction and graph algorithms make use of the topic graph discussed in chapter 4. After the execution of graph algorithms, the sentences in the original document are scored by adding the weights of the entities in the sentence. Highly ranked sentences are then chosen to form a summary of the desired length.

Section 5.1 describes the document graph construction which is one of the most important step in our summarization system.

5.1 Document Graph construction

A document graph represents the entities in a document as the nodes of a semantic graph and the relationship between those nodes as edges in the graph. The wikipedia ontology is used to identify entities in a document and relationship between those entities. Similar to topic formulation, this module also uses Brahms API to access the knowledge present in Wikipedia ontology.

The document graph construction process has 4 different phases as shown below.

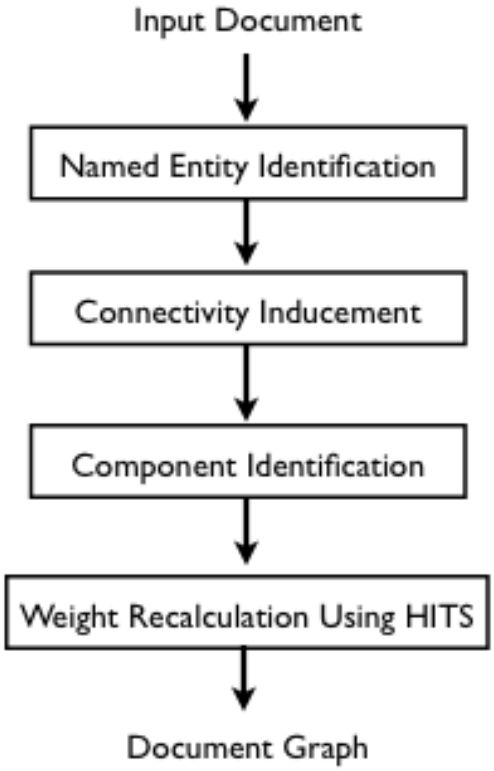

Figure 5: Document Graph construction steps

Named Entity Identification

The named entity identification process in a document is similar to the named entity identification process in topic terms/phrases as described in Chapter 4. The named entities present in a document are identified by matching the words or phrases in the document with the names of the entities present in the ontology. Entity names in an ontology are usually values of a certain property. These entities form the nodes of the initial document graph. Each node is assigned an initial weight according to the given formula:

$$w = 1 - 1 / (1 + \sum_{i=1..n} (p_i * s_i))$$

where,

w is the initial weight of the node

n is the total number of matches for that node in the document

p_i is the weight of the property that connects the literal value with the entity

s_j is the similarity measure between the match. This similarity measure takes into account the stemming or stop-word removal. If neither is used, the similarity measure is set to 1.

Thus, we can see that the weight of a node is calculated based on three factors: n, p_i and s_i. Also, if an entity is present in the topic graph, it is assigned the same weight as given in the topic graph.

If an entity occurs too often in a document, then it suggests that the entity is perhaps important and hence the entity should be given more weight. However, if an entity occurs too often, the formula given above would prevent the weight of the entity from increasing drastically.

As discussed in chapter 4, if an entity match is found using “wiki_name” property, then the weight of the entity would be more as compared to the “wiki_redirect_name” property. The property names used for named entity recognition are: “wiki_name”, “wiki_redirect_name”, “wiki_redirect_name_short”, “wiki_name_short”, “wiki_disambiguation_name” , “wiki_disambiguation_name_short”. These names are in the decreasing order of their confidence levels. We already explained redirection in

chapter 4. Entities found via redirection links are given lower weights when compared to a direct match for a search phrase. A phrase might match ambiguous entities. At this stage, all the matched entities are added to the document graph.

Connectivity inducement

Nodes in the semantic graph are connected via edges. The edges represent the relationship between entities in the Wikipedia ontology. The weights are given to the edges based on the importance of the relationship in the ontology schema. The names of the relationship in the decreasing order of their importance are wikipedia:infobox_*, wikipedia:template_*, wikipedia_href etc.

Component identification

During the named entity recognition phase, a phrase in the document might match several entities and we add all the entities to the semantic graph. As a result, the graph may have several connected components. The component(s) containing the entities from the topic graph are chosen for further processing. This is done because in a topical summarization, we are interested in extracting the summary of the document from the point of view of a particular topic. The rest of the components contain entities not related to the topic phrases, and therefore can be discarded at this stage.

Weight Recalculation Using HITS

For each component chosen in the previous step, we apply the Hubs and Authorities algorithm (HITS) to propagate and recalculate the weights of the entities. The HITS algorithm helps in establishing the authoritative entities. At this stage, our document graph is complete and can be used by our summarization module.

The document graph is saved as an XML document which contains the nodes and relationship between them. For each node, it provides information about the positions of occurrence of the matching term/phrase within the document. A weight is assigned to each node to denote its importance in the document. If a node is a topic node, then it may be assigned a higher weight closer to one or maybe one.

Given below is an example of part of a document graph. The first part of the XML contains the nodes of the graph. In this example, we can see that the XML contains the nodes like “Aerospace”, “International Space Station”, “NASA” and “1000 Miles (Grinspoon song)”. For each node, the “from” and “to” attributes in the match element is the range of occurrence of the matching phrase within the document. As discussed earlier, a topic phrase might appear in different components. So, the final document graph can have nodes belonging to different components. In this example, the node “ 1000 Miles (Grinspoon song)” belongs to the component number 10 while all the other nodes belong to the component number 4.

The second part of the XML contains all the edges in the graph. An edge represents the relationship in the form of a triple: subject, predicate, object. Most of the links in Wikipedia have no meaning associated with them and we call them anonymous href links. The first edge in this example represents an anonymous href link between Aerospace and NASA. If the information is present in the structured form like in infobox templates, the nodes involved in the relationship are represented in the XML with a non-anonymous link. For example, the link “wiki_template_infobox_space_station_publisher” connects International Space Station with NASA implying that in a wikipedia article, these two entities are connected in an infobox.

```
<node resource="http://dbpedia.org/resource/Aerospace" weight="0.061505" componentID="4">
    <match from="2039" to="2047"/>
</node>

<node resource="http://dbpedia.org/resource/International_Space_Station" weight="0.019615"
componentID="4">
    <match from="143" to="169"/>
    <match from="430" to="446"/>
</node>

<node resource="http://dbpedia.org/resource/1000_Miles_(Grinspoon_song)" weight="0.00345"
componentID="10">
    <match from="522" to="526"/>
</node>

<node resource="http://dbpedia.org/resource/NASA" weight="0.172967" componentID="4">
    <match from="36" to="39"/>
    <match from="541" to="544"/>
    <match from="1467" to="1470"/>
</node>

<edge>
    <subject resource="http://dbpedia.org/resource/Aerospace"/>
    <property resource="lsdis:wiki_href"/>
    <object resource="http://dbpedia.org/resource/NASA"/>
  </edge>

<edge>
    <subject resource="http://dbpedia.org/resource/International_Space_Station"/>
    <property resource="lsdis:wiki_template_infobox_space_station_publisher"/>
    <object resource="http://dbpedia.org/resource/NASA"/>
</edge>
```

Figure 6: Document graph XML

5.2 Link Analysis Algorithms

In our work, we use two graph algorithms: Spreading Activation and Hubs and Authorities (HITS) algorithm. Given below is the brief description of these algorithms.

Spreading Activation

Spreading activation is a method for searching associative networks, neural networks or semantic networks [30]. It is a graph-based algorithm in which a set of nodes are set as source nodes. The source nodes are given a weight or some activation value and is usually a real number. The activation is then iteratively propagated or spread to other nodes linked to the source nodes. As the activation propagates through the network, the weight decays according to a decay factor. We applied spreading activation algorithm to our document graph.

Hubs and Authorities (HITS)

Hyperlink-Induced Topic Search (HITS) [31], [21] also known as Hubs and Authorities is a famous graph-based algorithm to determine node importance. It was developed by Jon Kleinberg to rate web pages on the internet. The algorithm measures the *authority* and hub values of each node in the graph. The authority and hub values are defined in terms of each other. The authority value of a node is the sum of the scaled hub values of the nodes that points to it. The authority value increases if the number of incoming edges increase or if the hub value of the incoming nodes increase. The hub value of a node is calculated as the sum of the scaled authority values of the nodes it points to. The hub value increases if the number of outgoing edges increase or if the authority value of the nodes it points to increase. A high hub value node is important because it

connects to high-authority nodes in the graph. Similarly, a high authority node is important because nodes with high-hub values connect to it.

Given below are the steps of the Hubs and Authorities algorithm:

- Start with each node having a hub score and authority score of 1.
- Update authority value of each node by adding hub values of nodes that points to it.
- Update hub value of each node by adding the authority value of the nodes it points to.
- Normalize the values by dividing each Hub score by the sum of squares of all Hub scores, and dividing each authority score by the sum of squares of all authority scores.
- Repeat from the second step as necessary.

5.3 System Architecture

Figure 7 shows the architecture of our summarization system. The various modules are discussed in detail later in this section.

The document graph construction module has already been described above. We modified the thematic graph implementation [9], [17] to get the document graph. This module is written in C++. The Sentence Boundary Detector used in our system was created by Scott Piao from the Dept. of Linguistics, University of Lancaster in UK [29]. It is written in Java programming language and is a reliable system to detect the sentence boundary in a text document. The Document Graph Weight module, Sentence Weight Calculator module and the Summary Sentence Selector module was developed for our topic summarization system. These three modules are implemented in Java programming language.

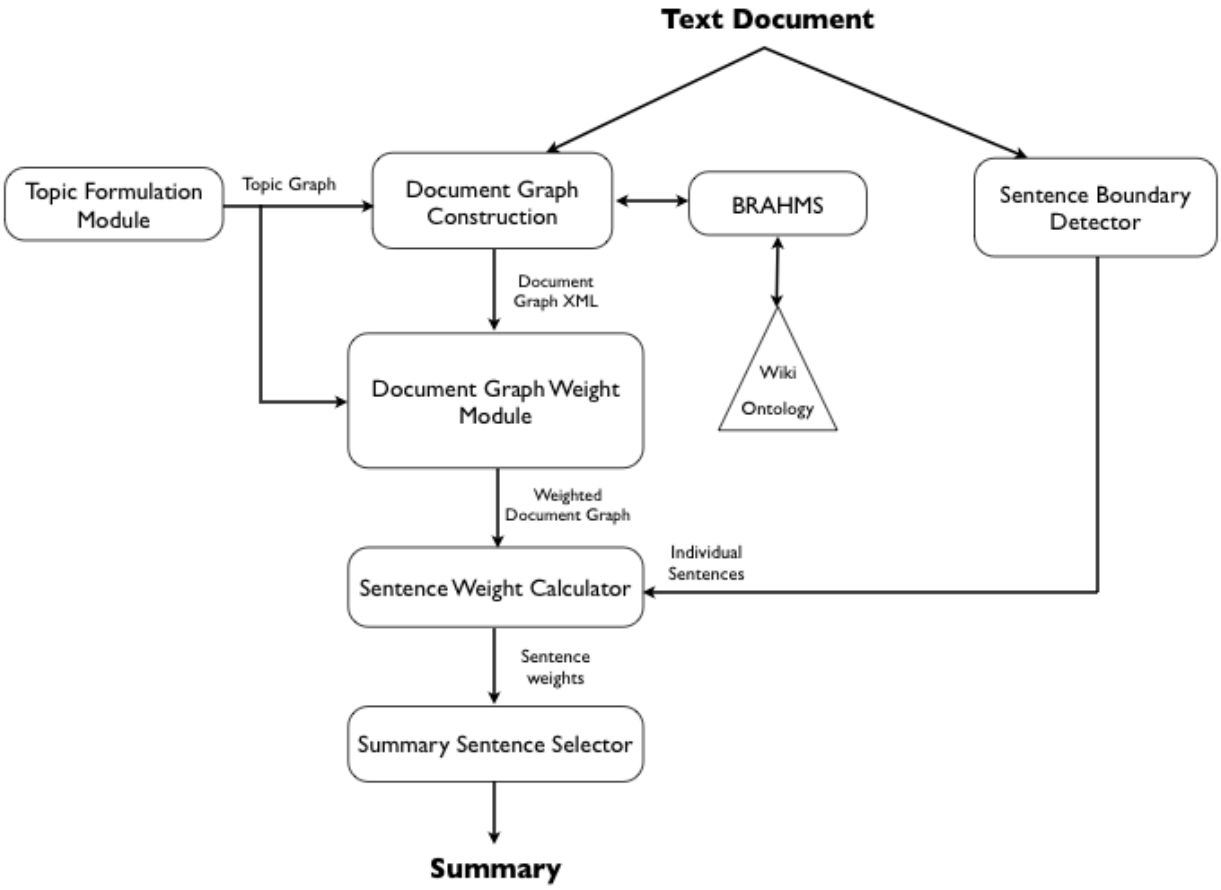

Figure 7: Summarization System Architecture

Document Graph Weight Module

The document graph weight module takes as an input the document XML from the document graph constructor module. The document XML is parsed using a DOM parser and stored in the memory in a graph data structure with nodes and edges. In this document graph, the nodes are given final weights using two different methods:

• Spreading Activation method

• Spreading Activation and HITS method

As described before, the document XML also contains a weight associated with each node. In Spreading Activation method, we do not make use of this weight. In Spreading Activation and HITS method, we use the node weights given in XML.

Spreading Activation method

From document XML, we create a document graph with all the nodes [1...N]. Each node is given an activation value A[i] which is a real number in the range [0.0...1.0]. A Link [i,j] connects source node [i] with target node [j] and has a weight W [i,j] associated with it. In our method, we have given a weight of 0.2 to all the links.

The algorithm takes the following parameters:

- Decay factor D
- Firing threshold F
- Maximum nodes fired per cycle MAX
- Number of cycles NUM

Below are the steps of the algorithm [30]:

1. Initialize the graph setting all activation values A [i] to zero. Set topic nodes to an initial activation value greater than the firing threshold F. A typical initial value is 1.0. In our system, the direct nodes are given an initial activation value of 1.0 and indirect topic nodes are given an activation value of 0.7.
2. For each unfired node [i] in the graph having an activation value A [i] greater than the node firing threshold F:
3. For each Link [i , j] connecting the source node [i] with target node [j], adjust A [j] = A [j] + (A [i] * W [i, j] * D) where D is the decay factor.

4. If a target node receives an adjustment to its activation value so that it would exceed 1.0, then set its new activation value to 1.0. Likewise maintain 0.0 as a lower bound on the target node's activation value should it receive an adjustment to below 0.0.
5. Once a node has fired it may not fire again. Nodes receiving a new activation value that exceeds the firing threshold F are marked for firing on the next spreading activation cycle. The maximum number of nodes that can be fired per cycle is MAX.
6. The procedure terminates when the maximum number of cycles NUM is reached.

At the end of this algorithm, the final activation value of each node is regarded as the final weight of that node and this weight is used by the sentence weight calculator module. We have used the spreading activation Java library which is part of Texai, an knowledge-based open source project to create artificial intelligence [26].

Spreading Activation and HITS method

This method is similar to the method above except for step 1 which is given below.

1. Initialize the graph setting all activation values A [i] to the weight of the nodes given in document XML. The direct topic nodes and indirect topic nodes, however, are given an initial weight of 1.0 and 0.7 respectively. As we have seen before, we get the weights in document XML by applying the HITS algorithm. The rest of the steps of this method is same as described above. Since, this method uses the weights from HITS algorithm and also uses the Spreading Activation algorithm , we call it the Spreading Activation and HITS method.

Sentence Boundary Detector

We use the Sentence Boundary Detector by Scott Piao from University of Lancaster [29]. It takes a text document as an input and breaks down the text into individual sentences. The system was evaluated on Genia corpus and it gave good results. The precision of the system was 0.997352 and the recall was 0.995093. It is also available online at http://text0.mib.man.ac.uk:8080/scottpiao/sent_detector.

The sentences are stored in the memory in a data structure that can also store other metadata information like sentence number, weight of the sentence, list of nodes in the graph that matched with the words/phrases in that sentence etc. The sentence boundary detector is an important component in an extractive summarization system. If the sentences are not split correctly, the final summary formed will contain improper sentences, and therefore will not make any sense. Hence, for a good quality extractive summarization system, the sentence boundary detector should be reliable.

Sentence Weight Calculator

The sentences detected by the sentence boundary detector are given weights during this step. For each sentence, we add the weights of all the nodes that appear in that sentence. If a sentence has a higher weight, it suggests that the sentence is important. Let's look at an example:

"NASA on Friday cleared space shuttle Discovery for launch on April 5 on one of its final cargo runs to the International Space Station before the fleet is retired later this year".

This sentence has two nodes from the document graph above: "NASA" and "International Space Station". We add the weights of these nodes i.e. 0.019615 + 0.172967 = 0.192582. Therefore, the weight of this sentence is 0.192582. Similarly, we

find the weight of all the sentences and the sentences with a higher weight are considered as important sentences with respect to the topic and the document.

Summary Sentence Selector

Once all the sentences are weighted, we choose the sentences that will be a part of the summary. The sentences are first arranged in the decreasing order of their weights. Sentences with the higher weights are then chosen one by one and added to the summary until the word limit for the summary is reached. The chosen sentences are then arranged in the order of occurrence in the original document to form the final summary.

CHAPTER 6

EVALUATION AND EXPERIMENTS

We conducted three experiments to evaluate the effectiveness of our topic driven summarization system. The first two experiments below are based on user evaluation of our system. In these experiments, we asked a set of 7 users to evaluate the topic formulation and the summarization module of our system. Four of them were graduate students and the remaining three were software professionals. The experiments were conducted on the users' own system by remotely logging in to our servers. In the first experiment, the users were given a brief overview of the topic formulation module, including some examples of topic terms and the output of the topic formulation module for the same. Similarly, in the second experiment, the users were given an overview of the summarization module including some examples.

6.1 Topic Formulation Experiment

In this experiment, we asked a group of 7 users to use our topic formulation module. Each user was asked to enter topic terms into the system and mark the entities returned by the system as relevant or not relevant. The system returned the top 20 entities. The users were asked to record their topic terms along with their evaluation of the entities returned for 5 different runs of the system. Further, they were asked to input a different set of topic terms in each run of the system. Appendix A contains User 1's evaluation of the module.

For each user, we calculate the percentage of relevant entities as shown in Table 1.

Table 1: Percentage of relevant entities for each user

	Percentage of relevant entities
User 1	61%
User 2	71%
User 3	59%
User 4	63%
User 5	53%
User 6	57%
User 7	65%

We took the average across all users which came up to 61.29% relevant entities.

Based on the feedback received from the users, we make the following preliminary observations:

1. The system returned more relevant results if more than one topic terms were entered by the user. This is because we are returning the top 20 common neighbors of the topic terms entered by the user. For a single topic term, we are looking at only the neighbors. When there more than one topic term, we look at the common neighbors which are generally low in number and is well connected to the topic terms.

2. An average taken across all the users' ratings showed that the system returned 61.29% relevant entities on the average. We consider this as a positive feedback from the users especially since the 20 entities that the user rated did not include the direct entities. These direct entity matches are automatically considered to be relevant.

6.2 Summarization Experiment I

In this experiment, the same 7 users used our document summarization module to generate topic-driven summary of documents. 28 documents were pre-selected for this experiment and each user was provided a set of 4 documents. For each document, the users were asked to first read the document to understand what it was about and then use our system to generate a topic-driven summary. To generate the topic-driven summaries, the users entered topic terms into our system. Once a summary was generated, the users were asked to rate it on a scale of 1 to 10. Since our summarization system is extractive, the users were asked to rate the summary based on its relevance to the topic and not on the coherence of the summary.

Appendix B gives an example of a topic-driven summary for a sample input document. Table 2 below shows the average rating by each user to the summaries generated for that user.

Table 2: Average rating of topic-driven summaries for each user

	Average Rating
User 1	7
User 2	6.875
User 3	6.55
User 4	5.5
User 5	6
User 6	6.75
User 7	7

Below are some observations made based on the above experiment:

1. The average rating given by the users to the summaries was 6.525. The length of the summary generated was 25% of the original document or 200 words, whichever is less. In the summaries that received the highest user rating, we observed that the document contained a lot of information about the topic entered by the user.
2. In some cases, the document just did not contain enough information related to the entered topic to generate a good summary. Not surprisingly, the summaries generated for these documents received lower user rating.

6.3 Summarization Experiment II

The goal of this experiment is to ensure that the summary generated by our system is not biased by location of sentences in the document, and also that the sentences picked up to be in the summary are indeed relevant. In this experiment, we selected a set of 10 documents. We generated a topic-driven summary for each document. We then combined each document with another unrelated document and generated the summary for the same topic again. We found that the summary generated still contained only the sentences from the first document regardless of the order in which the documents were combined.

The topic-driven summary of a single document is shown in Appendix B. Appendix C contains the topic-driven summary when the document from Appendix B is combined with another document. The summary generated in both the cases is almost the same, differing only by one or two sentences.

Comparison with other systems

In 2001, Document Understanding Conferences (DUC) [14] was established to further progress in summarization. DUC is sponsored by the Advanced Research and Development Activity (ARDA), and is run by the National Institute of Standards and Technology (NIST) to aid researchers in large-scale evaluation of summarization systems. One of the task in DUC competition is topic-driven summarization.

NIST provides three main evaluations of each system in DUC. The automated evaluation is done using ROUGE [33], which compares the system generated summary with the four manually written summaries from NIST evaluators. The manual evaluation is done based on responsiveness and linguistic quality of the summary.

Responsiveness (also called content) captures the amount of information in the summary that is relevant to the topic. The summaries were rated on responsiveness based on a 5-point scale. The average topic content score for systems in DUC 2007 was 2.6276 (52.552%). The UNC-CH system [34] achieved an average topic content score of 2.9556 (59.112%), which placed the system at 7th out of 32.

The initial evaluation result of our system as shown in the experiments above indicate responsiveness of 65.25%. In context of the DUC results described in the above paragraph, our initial results are encouraging. We plan to investigate ways to improve our system as described in the next chapter.

CHAPTER 7

CONCLUSION AND FUTURE WORK

Conclusion

The presented research was about knowledge-based topic driven summarization. The knowledge comes from an ontology created from Wikipedia. Wikipedia articles contain relevant related concepts that can be used to identify entities present in the topic and the document. We can also identify the relationship between entities using Wikipedia. However, Wikipedia has certain inherent limitations. Although Wikipedia contains a lot of articles, the quality and scope of the articles may differ for different topics. Also, most of the relationships between entities are anonymous which does not give any information about how strongly the entities are related. Therefore, in our topic formulation module, we might choose entities to be included in our topic graph which are perhaps not strongly related to the topic terms. If we know the domain of the documents that need to be summarized, specific domain ontologies can be used to form the topic and the document graph. This would greatly enhance the quality of our summarization system.

Future Work

In this section, we discuss how our work can be improved in the future. Also, we discuss some of the possible extensions to our system.

Improving Topic Formulation Module

Our Topic Formulation module uses the Wikipedia-based ontology to formulate the topic. As described earlier, we use common neighbors to disambiguate ambiguous entities. We also ask for user assistance to disambiguate entities. We can minimize user

involvement in disambiguating topic words/phrases by using other metrics, such as Wikipedia category information along with the common neighbors. Also, in our current system, users might not get the exact entities for the topic terms. This is because we are not using property names like “wiki_redirect_name_short”, “wiki_name_short”, “wiki_disambiguation_name_short” in our topic formulation module. In our future work, we can use all property names for entity recognition. This would result in a lot of entity matches. All these entities should be disambiguated with minimum user assistance.

Automatic Evaluation

We have used human subjects to evaluate our system. This greatly increases the expense of the evaluation. Also, human evaluation is not easily repeatable. In our future work, we would like to automate the process of evaluating our system. Typically, in an automated system there is a reference summary, against which the machine output is compared. Currently, we are using the Reuters data for which the reference summary is not available. We will need to find the evaluation corpus for which the topic-based summaries are available. DUC data has topic-based summaries but their topics are usually one or two sentences in length as compared to the topic words/phrases used in our system. We could also create the “ideal” summaries on our own. These summaries could be used as the reference summaries which can be compared against system generated summaries using automated methods like fixed word n-grams etc. for evaluation.

User Interfaces

We can improve our current system by building interactive graphical user interfaces so that users can effectively use our topic-driven summarization. This summarization

system could also be combined with a search system which would enable user to search for a document and then summarize its contents. It would help the user to browse and explore large document sets.

Multi-Document Summarization

We could extend our system to generate topic-driven summary for multiple documents. In a multi-document summary, steps should be taken to eliminate redundant information. As discussed earlier, MMR can be very effective in generating multi-document topical summaries. MMR aims to increase the relevance of the sentences to the query, while maintaining minimum redundancy in the summary.

REFERENCES

[1] Goldstein, J., Mittal, V., Carbonell, J., and Kantrowitz, M. 2000. Multi-document summarization by sentence extraction.

[2] Israel, Q. L., Han, H., and Song, I. 2010. Focused multi-document summarization: human summarization activity vs. automated systems techniques.

[3] Ontology (computer science) – http://www.wikipedia.org

[4] Ontology definition by Tom Gruber - http://tomgruber.org/writing/ontology-definition-2007.htm

[5] SchemaWeb - http://www.schemaweb.info/

[6] Protege Ontology Library - http://protegewiki.stanford.edu/wiki/Protege_Ontology_Library

[7] Wikipedia Statistics - http://stats.wikimedia.org/EN/TablesArticlesTotal.htm

[8] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Soren Auer, Christian Becker, Richard Cyganiak, Sebastian Hellmann. DBpedia - A Crystallization Point for the Web of Data

[9] Maciej Janik, Krys J. Kochut, "Wikipedia in Action: Ontological Knowledge in Text Categorization," icsc, pp.268-275, 2008 IEEE International Conference on Semantic Computing, 2008.

[10] Radev, D. R., Hovy, E., and McKeown, K. (2002). Introduction to the special issue on summarization. Computational Linguistics., 28(4):399–408.

[11] Luhn, H. P. (1958). The Automatic Creation of Literature Abstracts. IBM Journal of Research and Development, 2(2), 159-165.

[12] Rath, G., A. Resnick & T. Savage (1961). The formation of abstracts by the selection of sentences. American Documentation, 12(2):139–143.

[13] Edmundson, H. (1969). New methods in automatic extracting. Journal of the Association for Computing Machinery, 16(2):264–285.

[14] Document Understanding Conference - http://www-nlpir.nist.gov/projects/duc/intro.html

[15] Fellbaum, C. (Ed.) (1998). WordNet: An Electronic Lexical Database. Cambridge, Mass.: MIT Press.

[16] Barzilay, R. & M. Elhadad (1999). Using lexical chains for text summarization. In I. Mani & M. T. May- bury (Eds.), Advances in Automatic Text Summarization, pp. 111–121. Cambridge, Mass.: MIT Press.

[17] Maciej Janik, Krys Kochut. Training-less Ontology-based Text Categorization. 2008

[18] Amini, M. R. & N. Usunier (2007). A contextual query expansion approach by term clustering for robust text summarization. In Proc. of DUC-07.

[19] Salton, G., & Mcgill, J. (1983). Introduction to modern information retrieval. New York: Mcgraw Hill.

[20] Lesk, M. (1969). Word–word associations in document retrieval systems. American Documentation, 20(1), 27–38.

[21] HITS algorithm - http://en.wikipedia.org/wiki/HITS_algorithm

[22] Mani, I. and Bloedorn, E. (1997). Multi-document summarization by graph search and matching. In AAAI/IAAI, pages 622–628.

[23] Rakesh Verma, Ping Chen, Wei Lu. A Semantic Free-text Summarization System Using Ontology Knowledge, 2008

[24] Jurij Leskovec, Natasa Milic-Frayling, Marko Grobelnik. Extracting Summary Sentences Based on the Document Semantic Graph, 2005

[25] Vivi Nastase, Topic-driven multi-document summarization with encyclopedic knowledge and spreading activation, Proceedings of the Conference on Empirical Methods in Natural Language Processing, October 25-27, 2008.

[26] Texai project - http://www.texai.org/blog/?page_id=44

[27] Stanford Parser - http://nlp.stanford.edu/software/lex-parser.shtml

[28] BRAHMS - http://lsdis.cs.uga.edu/projects/semdis/brahms/

[29] Sentence and Paragraph Breaker - http://text0.mib.man.ac.uk:8080/scottpiao/sent_detector, Scott Piao

[30] Spreading Activation algorithm - http://en.wikipedia.org/wiki/Spreading_activation

[31] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment, Journal of the ACM, 1999

[32] Unified Medical Language System - http://www.nlm.nih.gov/research/umls/

[33] ROUGE - http://berouge.com/default.aspx

[34] UNC-CH at DUC 2007: Query Expansion, Lexical Simplification and Sentence Selection Strategies for Multi-Document Summarization

APPENDIX A

This appendix contains User 1's evaluation of the Topic Formulation Module. The evaluation includes 5 runs of the module. For each run, the output shown below consists of the topic terms entered by the user, followed by the direct entities matched by the system, followed by user-assistance, if required and finally, the list of 20 entities returned by the system to expand the topic.

TOPIC ---> "Aperture" "Final cut" "IMovie"

IMovie
Aperture

Did you mean Final Cut Studio? YES

☐	List of software products	0
☑	Proprietary software	1
☑	Mac OS X	1
☐	Independent film	0
☐	HDV	0
☑	Apple Inc.	1
☑	Digital camera	1
☑	QuickTime	1
☑	GarageBand	1
☑	Final Cut Pro	1
☐	Non-linear editing system	0
☑	Windows Movie Maker	1
☐	VirtualDub	0
☑	IMovie	1
☐	CineFX	0
☐	AVCHD	0
☑	Comparison of video editing software	1
☑	List of video editing software	1
☐	Sony Vegas	0
☑	List of Macintosh software	1
		12

TOPIC ---> "United states" "Iraq" "War"

War
Iraq
United States

☐	Italy	0
☐	Soviet Union	0
☐	Cold War	0
☑	Nuclear weapon	1
☑	Violence	1
☐	World War I	0
☑	Soviet war in Afghanistan	1
☐	Sweden	0
☑	Feb 1, 2003	1
☑	United States armed forces	1
☐	2000s (decade)	0
☐	Slavery	0
☑	Democratic Party (United States)	1
☐	Aug 1, 2003	0
☑	Capital punishment	1
☐	Pepper spray	0
☐	Philippines	0
☑	United Nations	1
☐	Jan 1, 2006	0
☐	Propaganda	0
		8

TOPIC ---> "Beatles" “Beach Boys"

	The Beach Boys	
	The Beatles	
☑	Rock music	1
☑	Folk rock	1
☑	EMI	1
☐	Allmusic	0
☑	Punk rock	1
☐	Extended play	0
☑	Billboard (magazine)	1
☑	Rolling Stone	1
☑	Psychedelic pop	1
☑	Billboard Hot 100	1
☑	UK Singles Chart	1
☐	NME	0
☐	Time (magazine)	0
☑	2001 in music	1
☑	2003 in music	1
☑	Backing vocalist	1
☑	Lead vocalist	1
☐	A-side and B-side	0
☑	Symphonic rock	1
☑	1966	1
		15

TOPIC ---> "APR" "Credit Card"

Credit card

Did you mean Annual percentage rate? YES

☐	United Kingdom	0
☐	United States	0
☑	Credit card	1
☑	Mortgage	1
☑	Credit (finance)	1
☑	Interest	1
☑	Loan	1
☑	Interest rate	1
☑	Outline of finance	1
☐	Office of Fair Trading	0
☑	Predatory lending	1
☑	Moneylender	1
☑	Truth in Lending Act	1
☑	Subprime lending	1
☑	Credit history	1
☑	Fee	1
☑	Payday loan	1
☐	Providian	0
☑	Credit card interest	1
☑	Credit card debt	1
		16

TOPIC ---> "Friends" “Lost” “Heroes"

Friends

Did you mean: Lost (TV series)? YES
Did you mean: Heroes (TV series)? YES

☐	Start date	0
☑	Entertainment Weekly	1
☐	Time (magazine)	0
☐	Theme music	0
☑	Television pilot	1
☑	Serial (radio and television)	1
☑	The Office (U.S. TV series)	1
☐	Middle East Broadcasting Center	0
☑	Golden Globe Award	1
☐	Seven Network	0
☑	Desperate Housewives	1
☑	Family Guy	1
☐	SF zwei	0
☐	Ensemble cast	0
☐	List of fictional companies	0
☑	List of American television series	1
☐	List of current child actors from the United State	0
☐	List of cliffhanger endings	0
☑	Television in the 2000s	1
☑	List of American television programs by date	1
		10

APPENDIX B

This appendix contains a sample topic-driven summary of a single document.

Input Document

Sweet cravings for dessert existed since the early beginnings of humanity. Now enjoyed by many, affluent society folk were the only ones who could afford such in the past, and “regular” people could afford desserts only on special occasions. Technological advances in sugar production allowed for more widespread distribution of it and more affordable prices for consumers. The increased sugar supply paved the way for people to enjoy desserts worldwide.

Thought to have originated from the custom of removing the aftertaste of a meal with a sweet taste, desserts leave a mouth sweet-tasting and revitalized. The confection derives its name from the French word desservir, which literally means “to clear the table.” Today there are literally thousands of dessert varieties. Some popular sweet desserts that have been around since ancient times include cakes, pies and ice cream.

The word “cake” comes from the Old Norse word kaka. The Oxford English Dictionary first documented the use of the word “cake” in the 13th century. Ancient cultures consumed a very different type of cake than we do today. The first cakes more closely resembled bread and honey, and nuts and dried fruit flavored them. The ancient Egyptians were the first people to use advanced baking methods. People in Medieval Europe baked fruitcakes and gingerbread as desserts because they preserved well.

Advances in baking technology and the availability of cooking ingredients aided cake development. Round cakes with icing that resemble what we eat today made their way to plates in the mid-17th century. The modern cake with ingredients like white flour and baking powder were first used in the 19th century.

The roots of pie as a dessert originated in the Neolithic Period beginning in 9500 B.C. Known as galettes, these free-form pies contained different grains and honey. Galettes baked over hot coals. Pastry originated as bakers added fruit, nuts and honey to bread dough to serve to the pharaoh (1304-1237 B.C.). Drawings of this practice decorate the tomb walls of King Ramses II. Pie-making techniques and ingredients often changed because of different conditions and ingredients.

The Pilgrim women brought their favorite family pie recipes to America to use as desserts. They used traditional pie fillings and incorporated berries and fruits that the Native Americans used as they adapted to the New World.

Colonial women started the tradition of using round pans for pies. This tradition came about to conserve ingredients. The women used flattened pies and then laid rolled out pastry over the top of the pan and cut off the corners. Pies became a part of American culture in the 1700s as pioneer women served pie as a dessert with every meal. As an evening meal, apple pie became a popular dessert with American children.

Ice cream originated around the 4th century B.C. The Roman emperor Nero (A.D. 37-68) ordered his minions to gather ice to combine with fruit toppings for his dessert. King Tang (A.D. 618-97) of Shang, China, developed an early form of ice cream by combining ice and milk. Traders likely brought ice cream from China back to Europe. Ice cream constantly evolved during history, and soon recipes developed for ices, sherbets, and milk ices to serve to the Italian and French aristocracy.

Ice cream recipes made their way into American history in the 1700s. Several important states people often served ice cream as a dessert. Notable figures such as George Washington, Thomas Jefferson and Dolly Madison served it to their guests. A London caterer ran an ad in a New York paper in 1774 announcing his intent to sell ice cream in his stores.

Topic: “Pie”

Summary:

Some popular sweet desserts that have been around since ancient times include cakes, pies and ice cream. The roots of pie as a dessert originated in the Neolithic Period beginning in 9500 B.C. Known as galettes, these free-form pies contained different grains and honey. Galettes baked over hot coals. Pastry originated as bakers added fruit, nuts and honey to bread dough to serve to the pharaoh (1304-1237 B.C.). They used traditional pie fillings and incorporated berries and fruits that the Native Americans used as they adapted to the New World. The women used flattened pies and then laid rolled out pastry over the top of the pan and cut off the corners. Pies became a part of American culture in the 1700s as pioneer women served pie as a dessert with every meal. As an evening meal, apple pie became a popular dessert with American children. Ice cream originated around the 4th century B.C. King Tang (A.D. 618-97) of Shang, China, developed an early form of ice cream by combining ice and milk. Notable figures such as George Washington, Thomas Jefferson and Dolly Madison served it to their guests. A London caterer ran an ad in a New York paper in 1774 announcing his intent to sell ice cream in his stores.

APPENDIX C

This appendix contains a sample topic-driven summary of two documents combined together.

Input Document

Sweet cravings for dessert existed since the early beginnings of humanity. Now enjoyed by many, affluent society folk were the only ones who could afford such in the past, and “regular” people could afford desserts only on special occasions. Technological advances in sugar production allowed for more widespread distribution of it and more affordable prices for consumers. The increased sugar supply paved the way for people to enjoy desserts worldwide.

Thought to have originated from the custom of removing the aftertaste of a meal with a sweet taste, desserts leave a mouth sweet-tasting and revitalized. The confection derives its name from the French word desservir, which literally means “to clear the table.” Today there are literally thousands of dessert varieties. Some popular sweet desserts that have been around since ancient times include cakes, pies and ice cream.

The word “cake” comes from the Old Norse word kaka. The Oxford English Dictionary first documented the use of the word “cake” in the 13th century. Ancient cultures consumed a very different type of cake than we do today. The first cakes more closely resembled bread and honey, and nuts and dried fruit flavored them. The ancient Egyptians were the first people to use advanced baking methods. People in Medieval Europe baked fruitcakes and gingerbread as desserts because they preserved well.

Advances in baking technology and the availability of cooking ingredients aided cake development. Round cakes with icing that resemble what we eat today made their way to plates in the mid-17th century. The modern cake with ingredients like white flour and baking powder were first used in the 19th century.

The roots of pie as a dessert originated in the Neolithic Period beginning in 9500 B.C. Known as galettes, these free-form pies contained different grains and honey. Galettes baked over hot coals. Pastry originated as bakers added fruit, nuts and honey to bread dough to serve to the pharaoh (1304-1237 B.C.). Drawings of this practice decorate the tomb walls of King Ramses II. Pie-making techniques and ingredients often changed because of different conditions and ingredients.

The Pilgrim women brought their favorite family pie recipes to America to use as desserts. They used traditional pie fillings and incorporated berries and fruits that the Native Americans used as they adapted to the New World.

Colonial women started the tradition of using round pans for pies. This tradition came about to conserve ingredients. The women used flattened pies and then laid rolled out pastry over the top of the pan and cut off the corners. Pies became a part of American culture in the 1700s as pioneer women served pie as a dessert with every meal. As an evening meal, apple pie became a popular dessert with American children.

Ice cream originated around the 4th century B.C. The Roman emperor Nero (A.D. 37-68) ordered his minions to gather ice to combine with fruit toppings for his dessert. King Tang (A.D. 618-97) of Shang, China, developed an early form of ice cream by combining ice and milk. Traders likely brought ice cream from China back to Europe. Ice cream constantly evolved during history, and soon recipes developed for ices, sherbets, and milk ices to serve to the Italian and French aristocracy.

Ice cream recipes made their way into American history in the 1700s. Several important states people often served ice cream as a dessert. Notable figures such as George Washington, Thomas Jefferson and Dolly Madison served it to their guests. A London caterer ran an ad in a New York paper in 1774 announcing his intent to sell ice cream in his stores.
Email was much older than the Internet itself. It was actually never invented, but evolved from very simple beginnings. Email messages were not sent, but left behind, much like leaving a note in someone else's desk. The message would be left behind in a location that would be easily seen by the user when logged in.

The MAILBOX was probably the first email system of this type. It was first used at the Massachusetts Institute of Technology in 1965. Another early program, SNDMSG, sent messages on the same computer. Therefore, before the Internet came, email messaging could only be done with the same computer.

Once the Internet was developed, making sure the message was sent to the right person became a problem. While the computers were able to recognize and "talk" with each other, identifying a specific person to receive the email message was much more complicated.

Addressing the email message was invented by Ray Tomlinson in 1972. Like many Internet inventors, Tomlinson worked for Bolt Beranek and Newman as an ARPANET contractor. He chose the @ symbol from the computer keyboard to denote sending

messages from one computer to another. Since then, for anyone using the Internet, addressing the email was as simple as nominating name-of-the-user@name-of-the-computer. The “nice hack” described by Internet pioneer Jon Postel lasted until this day.

The other features of email like sorting and labeling emails in folders came much later. Larry Roberts invented some email folders for his boss so he could sort his mail, a big advance. In 1975, John Vital developed some software to organize email. By 1976, email had really taken off, and commercial packages were available for users.

Connecting with the Internet cost a lot in those times. With connection charges costing many dollars a minute, it mattered to be able to prepare a reply without being connected to a telephone, and then get on the network to send it. Aside from the costs, problems such as keyboard functions such as delete keys and backspace keys did not work well, and lack of the “wrap around” text feature on the screen, vexed email users. Offline readers helped a lot.

Using offline readers allowed email users to store their email on their own personal computers, then read them, and prepare replies without actually being connected to the network. This setup is much similar to using an email client like Microsoft Outlook today.

Because the email’s features progressed from crude to sophisticated, standard protocols would have to be developed, especially for use in offline readers. One of the two standards, SMTP, or Simple Message Transfer Protocol, was the first important email protocol used. Although, it is still open to attacks from worms and viruses, the development of a stronger standard continues until this day.

The other standard is called POP, or Post Office Protocol. POP was an important standard that allowed different mail systems to work with each other. Both SMTP and POP were important configurations for email clients such as Microsoft Outlook to download email from the server, and to send email through the server using a desktop program.

Topic: “Pie”

Summary:

Some popular sweet desserts that have been around since ancient times include cakes, pies and ice cream. The roots of pie as a dessert originated in the Neolithic Period beginning in 9500 B.C. Known as galettes, these free-form pies contained different grains and honey. Galettes baked over hot coals. They used traditional pie fillings and

incorporated berries and fruits that the Native Americans used as they adapted to the New World. The women used flattened pies and then laid rolled out pastry over the top of the pan and cut off the corners. Pies became a part of American culture in the 1700s as pioneer women served pie as a dessert with every meal. As an evening meal, apple pie became a popular dessert with American children. Ice cream originated around the 4th century B.C. Notable figures such as George Washington, Thomas Jefferson and Dolly Madison served it to their guests. The “nice hack" described by Internet pioneer Jon Postel lasted until this day. Aside from the costs, problems such as keyboard functions such as delete keys and backspace keys did not work well, and lack of the “wrap around" text feature on the screen, vexed email users.