SENTENCE SIMILARITY BASED TEXT SUMMARIZATION USING CLUSTERS

A thesis submitted

In partial fulfilment of the requirements
For the degree of

MASTER OF TECHNOLOGY

In Computer Science & Engineering

by

SHIVAM MAURYA

(Enroll. No. 11104491898)

Under the Supervision of

Mr. Ramesh Vaishya
Sr. Lecturer
Department of Computer Science and Engineering
BBDNITM, Lucknow

to the

Faculty of Computer Science and Engineering

BABU BANARASI DAS UNIVERSITY

LUCKNOW

June 2013

DECLARATION

I hereby declare that this submission is my own work and that, to the best of my knowledge and i belief, it contains no material previously published or written by another person or material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Signature

Name: Shivam Maurya

Roll No. 1110449011

CERTIFICATE

It is certified that **Shivam Maurya (**Roll No. 1110449011) has carried out the research work presented in this thesis entitled **"Sentence Similarity Based Text Summarization Using Clusters"** in partial fulfillment of the requirement for the degree of **Master of Technology** from Babu Banarasi Das University, Lucknow under my supervision. The thesis embodies results of original work, and studies as are carried out by the student himself and the contents of the thesis do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Signature

Mr. Ramesh Vaishya
(Sr. Lecturer, Department of CSE)
BBDNITM**,** LUCKNOW.

Date:

ABSTRACT

Early work in the computational treatment of natural language focused on summarization, and machine translation. In my research I have concentrated on the area of summarization of documents in similar sentence summarization using clusters. This thesis presents my work on text similarity. This work enables the documentation of short units of text (usually sentences) that contain similar information even though they are written in similar sentence. I present my work on Similarity finder, a framework for text similarity computation that makes it easy to experiment with considerations for similarity computation and add support for similar sentence.

A detailed examination and evaluation of the system is performed using English data. I also apply the concept of sentence text similarity to summarization in two similar sentence systems. The first improves readability of English summaries of text by replacing machine translated sentences with highly similar English sentences when possible. The second is a novel summarization system that supports comparative analysis of documents. Sentence similarity clusters sentences to present information that is supported by both similar sentence summarization using clusters. Second, the system provides an analysis of how English documents similar by presenting information that is supported exclusively by documents in English language. This novel form of summarization is a first step at analyzing the similar in perspectives from news reported in English.

Sentences are then selected for inclusion in the summary depending upon their relative importance in the conceptual network. The sentences (nodes in graph) are then selected for inclusion in final summary based on relative importance of sentence in the graph and weighted sum of attached feature score. The user can find the document from their internet and analyze all to sort out the relevant information. Analyzing the text by reading all textual data is infeasible. So the technology of automatic document summarizer may provide a solution to information overload problems. We propose an extractive text summarization system. I proposed my work on sentence similarity based computation that helps to experiment for similar text computation. Extractive summarization text system choosing a subset of similar group from the text. Proposal work i used the part of speech, proper noun, verb, pronouns such as he, she, and they

etc. With the help of part of speech we find important sentence using statistical method like proper noun and sentence similarity system .It based on internet information that that contain picky sentence.

ACKNOWLEDGEMENT

The extensive endeavor that accompanies the successful completion of any task would not be complete without the expression of gratitude to the people who made it possible. I express my sincere thanks to our project guide **Mr. Ramesh Vaishya (Sr.Lecturer, Department of C.S.E)** Babu Banarasi Das University, Lucknow. I thank him for all his encouragement, valuable advices and suggestions throughout this thesis work.

I extend my whole-hearted thanks to all the staff of Computer science & Technology for providing all facilities, valuable suggestions and constant supervision for the completion of this work.

Last but not the least, I would like to acknowledge the ongoing support of my parents and my family members, whose patience and encouragement during these long days and night have been paramount in making this work a reality.

Shivam Maurya

(Roll No. 1110449011)

LIST OF TABLES

LIST OF FIGURES

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

There is a lot of text in the world. According to Global Reach's 2004 estimate, there are295.4 million English-speaking people with access to the internet, and 544.5 million non- English speaking people with access to the internet.1 The Internet Archive archives sites on the web, and has reached the size of approximately 1 petabyte of data and is currently growing at a rate of 20 terabytes per month.2. With such a large amount of text, English and non-English alike, it is difficult to alter and manage the information that people need. Information retrieval engines help people and access the information that they desire, but what should one do when there is too much information to readily handle?

Summarization is one important approach to managing the large amount of text that people must read. Summarization can reduce the amount of text people have to read to let them decide if a document is relevant to their information need. Since the inception of using computers to process written text, one of the first tasks undertaken was that of summarizing text by shortening a long document to present the document's content brief while preserving the underlying meaning [20]. Edmundson [1,3] proposed a method for weighting sentences using the keyword weighting proposed by Luhn, and added weights based on a list of cue-phrases indicating good and bad sentences, the words from titles and sub-titles, and the location of sentences.

In the mid-nineties statistical approaches to identifying sentences based on features, such as those used by Edmundson, began to appear, as well as well as linguistics-based approaches using discourse structure or more in-depth parsing of the text While the field of single document summarization has advanced considerably, early efforts focused mainly on monolingual text processing - English speaking people summarized English documents, Russian speaking people summarized Russian documents, Japanese speaking people summarized Japanese documents, and so on As progress was made in single document summarization, researchers began to study multi-document summarization. Given five or ten documents on the same event (e.g., multiple documents reporting on developments in

The same court case), the goal is to produce a short summary that gives an overview of all the documents. One approach to document summarization that has proven effective and gained popularity is similarity-based summarization.

The principle behind similarity-based summarization is that important information is repeated in different reports on the same event. Reporters for the New York Times and Los Angeles Times are going to both emphasize the same important facts in independently written articles on the same event. In a report about a specific trial, for instance, both reporters will state who the defendant and prosecutors are in the trial, and what charge the defendant is accused of. Identifying this repeated, important information is the approach taken in similarity based summarization systems. A similarity based summarization system identifies when sentences (or paragraphs, or clauses) state the same information. Sentences that are repeated many times across many documents are assumed to be more important than sentences that are not repeated, and a summary can be built by including information that has been repeated often. While most summarization systems are extractive, i.e., they take one of the sentences from the input documents verbatim and include it in a summary, some state-of-the-art summarization systems analyze the similar sentences and re-formulate a new sentence including only the specific similar information.

This thesis brings a similarity-based mostly extractive approach to multiple documents written in similar sentence. I present similarity finder, a framework I developed for identifying similar sentences within and between texts in multiple sentences. I have performed an evaluation of the system using English. I show the usefulness of my approach to similar text similarity for summarization tasks by presenting and evaluating two similar summarization systems. This thesis presents Similarity Finder, the system I developed as a framework for similar text similarity computation, examines the value of translation at similar levels and similar primitives for lingual similarity computation, and shows the implementation of a new summarization approach for similar document collections that shows both similarities and similarities between the documents across similar sentences.

1.1 Goals

There are two main goals for this thesis: to introduce my work in lingual text similarity, and to show that the system I built for the task can be used as the basis of similar document summarization system. Similarity finder, a system I developed for similar text similarity, is described and evaluated in a sentence and clustering-level evaluations of English text. Another contribution of the thesis is an approach to

similar document summarization that shows similarities in documents, as well as similar between them which use Similarity finder I first introduce related work that has been done on English text similarity in the Similarity finder system that forms the basis of my work on a similar version of Similarity finder called Similarity finder. Similarity finder was developed at Columbia University under the supervision of Judith Klavans and Kathy McKeown, and I have also worked on Similarity finder improvements and maintenance. I use this past work on Similarity finder to motivate that the approach Similarity finder takes is best suited for text similarity computation between small units of text (sentences or paragraphs) compared to the alternative of bag-of-words approaches used in information retrieval or document clustering.

In the Similarity finder approach, similar primitives, such as words that are nouns or words that are verbs are identified, and similarity is computed over all of these features. For to sentences, Similarity finder will compute how similar those sentences are based on each feature, and it combines all the similarities into a single similarity value representing the overall similarity of the two sentences. For example, in the following two sample sentences.

Sentence 1	The student ran the program.
Sentence 2	The athlete ran the race very quickly.

The noun primitives from Sentence 1 are (student, program) and from Sentence 2 are (athlete, race). The verb primitive in both sentences is (ran). Two features, verb similarity and noun similarity, are computed over the two primitive types, and while similarity is high over the verb feature they both share the same and only verb it is low over the noun feature. None of the nouns are the same.

My work on Similarity finder extends the approach taken in Similarity finder to enable text similarity computation. Similarity finder identifies primitives in text in similar text, and makes it easy to add support for new primitive types. Features are easy to define over different primitive types, allowing for experimentation in both primitive types, and features computed over the primitives. Similarity finder introduces a translation stage that maps primitives from one language to another to enable matching primitives across English texts without using full machine translation on the non-English documents. Similarity finder can identify similar text across languages using only simple techniques for primitive translation. Similarity finder is designed to make it straightforward to add support for new languages, and it has been tested with minimal modifications over text from. This thesis will show that Similarity finder, using simple techniques that can be quickly applied to other language, performs with high precision at identifying similar sentences across language.

The second main goal of this thesis is to present two summarization systems that have been built on top of the text similarity computation technology in Similarity

finder. The systems validate that Similarity finder is a useful similar text similarity computation engine. The first system takes a novel approach to improving the readability of English summaries of machine translated English data. It produces a summary of machine translated English text, and uses Similarity finder to identify English sentences that are similar to the machine translated summary sentences, replacing them if the two are similar enough.

Research questions that this thesis answers include

- ✓ Can a system automatically identify similar sentences across similar sentence?
- ✓ If so, at what levels should translation be used? At the word level? At the level of Noun phrases? Can translation at lower levels compete with full machine translation? At the sentence level?
- ✓ Can a system that identifies similar sentences across text be used for similar sentence summarization?
- ✓ Can sentence similarity be applied to improve summaries of machine translated text? Will cross-similar sentence similarity allow for the creation of summarization systems that present similar in perspective across languages by summarizing similarities and similar across the input documents?

Approaches to text similarity

English version of Similarity finder, a program for computing the similarity of English sentences and clustering them. Similarity finder was designed by other people at the Columbia University Natural Language Processing group, most notably Judith Klavans, Vasileios Hatzivassiloglou, and Melissa Holcombe. Similarity finder introduced the idea of using shallow linguistic features computed over the input text and a statistical model to combine those features into a similarity value between the sentences. Similarity finder also uses a clustering approach tuned to the task of clustering similar sentences. The shallow linguistic features encode information that can be derived by part-of-speech tagging or word lookup in lexical taxonomies such as word Net [22].
In my thesis I compared a comparison of the Similarity finder approach to text similarity and other text similarity measures as used in information retrieval, document clustering, or other natural language processing tasks. I have extended the approach exemplified by Similarity finder to similar sentence in Similarity finder, a similar re-implementation of Similarity finder. Similarity finder is described. Similarity finder is designed to allow for easy addition of new primitives and features for comparing text, and I present an in-depth description of the English support in Similarity finder. Chapter 4 presents an evaluation of how well Similarity finder is able to identify English sentences that are similar to other sentences. The usefulness

of Similarity finder as a cross-lingual text similarity computation system is verified by using it as the basis for two summarization systems.

Literature Survey

H.P Luhn is the father of information retrieval. In his pioneering work used simple statistical technique to develop an extractive text summarization system. Luhn used frequency of word distributions to identify important concepts, i.e. frequent words, in the text. As there could be uninformative words which are highly frequent (commonly known as stop words), he used upper and lower frequency bounds to look for informative frequent words. Then sentences were ranked according to the number of frequent words they contained. The criterion for sentence ranking was very simple and would read something like this-

If the text contains some words that are unusually frequent then the sentences containing those words are important. This quite simple technique which uses only high frequent words to calculate sentence ranking worked reasonably well and was modified by others to improve performance. Luhn provide a framework which can be used to measure various feature score for each text in the document. I used this approach with the weight of each term in the text instead of only frequency.

Edmund son's work exploiting cue phrases: Luhn's work was followed by H. P. Edmundson who explored the use of cue phrases, title words and location heuristic. Edmund son tried all the combinations and evaluated the system generated summaries with human produced extracts.

The disadvantage of previous work is that they provide summary by cumulative effect various key features like-

1- Sentence position
2- Title adhoc
3-Numerical data
4-Noun world

1.3 Similarity-based approaches to Document Summarization

Similarity based summarization approaches are not new in the area of summarization. Similarity based summarization is an accepted, well-respected approach to document summarization. While there are many summarization systems that use similarity-based approaches, they are typically applied to monolingual summarization systems. Similarity finder allows the approach to be applied to lingual document summarization systems. Similarity finder takes documents in multiple languages as input, and outputs similarity values for pairs of sentences within and across languages. Chapter 5 presents two systems that use the similarity values output by Similarity finder. One system summarizes machine translated text and

replaces sentences with very similar English sentences to improve the readability of the summary.

1.3.1 Highlighting Similarities between difference Data and text

A second summarization system using Similarity finder is novel in that it presents a summary in three parts that indicates both similarities and differences in the input. The CAPS system (Comparing and Contrasting Program for Summarization) described in Section 5.4 takes a cluster of English documents on the same topic as input. It generates a summary in three parts: information that is only present in the text, information that is only present in the English text and information that is supported by both the English text. While much previous work in summarization has been done on indicating similarities, very little work has been done on indicating differences between documents, or as in this case, groups of documents.

1.1 Contributions

1. Flexible framework for single lingual text similarity experimentation. I developed v, which supports rapid development of features for similarity computation for text, and support for different translation mechanisms over those primitives. This framework has allowed me to experiment with different combinations of primitives and translation methods as presented in

2. Experimentation with and evaluation of different levels of translation for single lingual text similarity identification. I have examined how translation can be used at different levels for lingual text similarity identification. I have compared full document translation using machine translation systems to primitive level translation that translates at the word level and translation of phrases extracted from the documents.

3. A focus on methods that is easily portable to new sentence. The main sentence pair presented in this thesis is English, but I have also used Similarity finder with very little engineering required to add support for that sentence. Using existing bilingual dictionaries for translation, or learning dictionaries from large collections of text and their translations (parallel corpora) allows one to quickly add support for similar text.

4. Investigating primitives for similarity, and translating primitives across sentences. An original contribution of this work is the investigation of primitives that are compatible across sentences for the similarity computation process and methods of translating those primitives. Similarity computation performed over primitives and their translations extracted from the native sentence is more easily extensible to sentence for which we do not already have a full machine translation system. For high precision tasks requiring identification of English sentences, translation at the primitive level performs better than similarity computation using machine translated

input documents. In this work, I investigate word-level primitives, and named entity based noun phrase primitives for similarity computation text in English. This work takes the first steps to identifying further primitives that may be helpful for cross-sentence similarity computation, and presents a framework for continued research in this area.

CHAPTER 2

SIMILARITY IN ENGLISH TEXTS: SIMILARITY FINDER

The concept of textual similarity is used in many applications that involve matching one text to another, such as information searching or retrieval, categorizing texts into pre-defined categories, filtering text, and text clustering. In these cases the similarity of a document is computed between a query, a category, a filter, or other documents. The work in this thesis is primarily concerned with text similarity at a lower granularity: typically the sentence or paragraph level.

Similarity finder is a system designed and implemented for identifying similar units of short text, either paragraphs or sentences, and clustering related sentences into themes that express the same information. Similarity finder has been used in multiple summarization and question-answering systems. This chapter describes the Similarity finder system as implemented for English. I build upon the work done on Similarity finder by re-implementing a version that performs similarity identification, Similarity finder, described in Chapter 3.

2.1 Related work in English text similarity

2.1.1 Information Retrieval

The concept of similarity is critical in the Information Retrieval field. The vector-based document model as popularized by Salton's SMART system [33] represents a document as a word vector, and queries are matched to similar documents in the document database via a similarity metric. The word-vector based document representation views documents as collections of words, without regard to the original word order, or syntactic function of the words; such systems do not have information about which words are nouns or verbs, or what words are the grammatical subject or object.

The task for information retrieval is to return a list of documents that are similar to a given query. Depending on the information retrieval system, the format of the query might be a document itself, a Boolean expression, a set of terms, and so on. In a standard vector-space information retrieval engine, the query document is mapped into the word vector space, and its distance to the other documents in the word-vector space is computed.

The similarity between the documents and the vector-space representation of the query is often calculated using a distance metric, such as the Euclidean distance, or

the cosine of the angles between the two vectors. The documents are then ranked on the basis of this similarity measure, and the list is returned to the end user.

In the task that I examine, there is no concept of a query to which all text units are compared. Instead, each text unit must be compared to every other text unit to compute similarity for the pair. The features that I compare similarity over are also context dependent while some of the primitives are similar to the vector-space model used in IR (simple overlap between words stems and tokens, for example), others features are more complex ,like features that require the two text units to have the same noun phrase followed by the same verb. Illustrates the differences between similarity determination and information retrieval.

The full text documents used in information retrieval system contrast with the text units used in similarity finder for similarity comparison, which are much shorter, being sentences or even clauses. This leads to a data sparsely problem. Since the documents are larger, they tend to use a more varied vocabulary, so there is a larger possibility for overlap with the query when examining specific text units there just is not as much text, and so a particular set of terms is more likely to be missing. Since there is much less data to deal with compared to the full text of documents, it is more important to use more evidence than just distances based on the word vectors of the documents. For this reason, similarity finder uses a variety of features built over different primitives, such as nouns or verbs that investigate similarity in a number of linguistically motivated areas.

2.1.2 Clustering Techniques

Similarity finder uses clustering in two ways:

- ✓ document clustering as a pre-input stage to similarity finder for identifying documents that are on the same topic

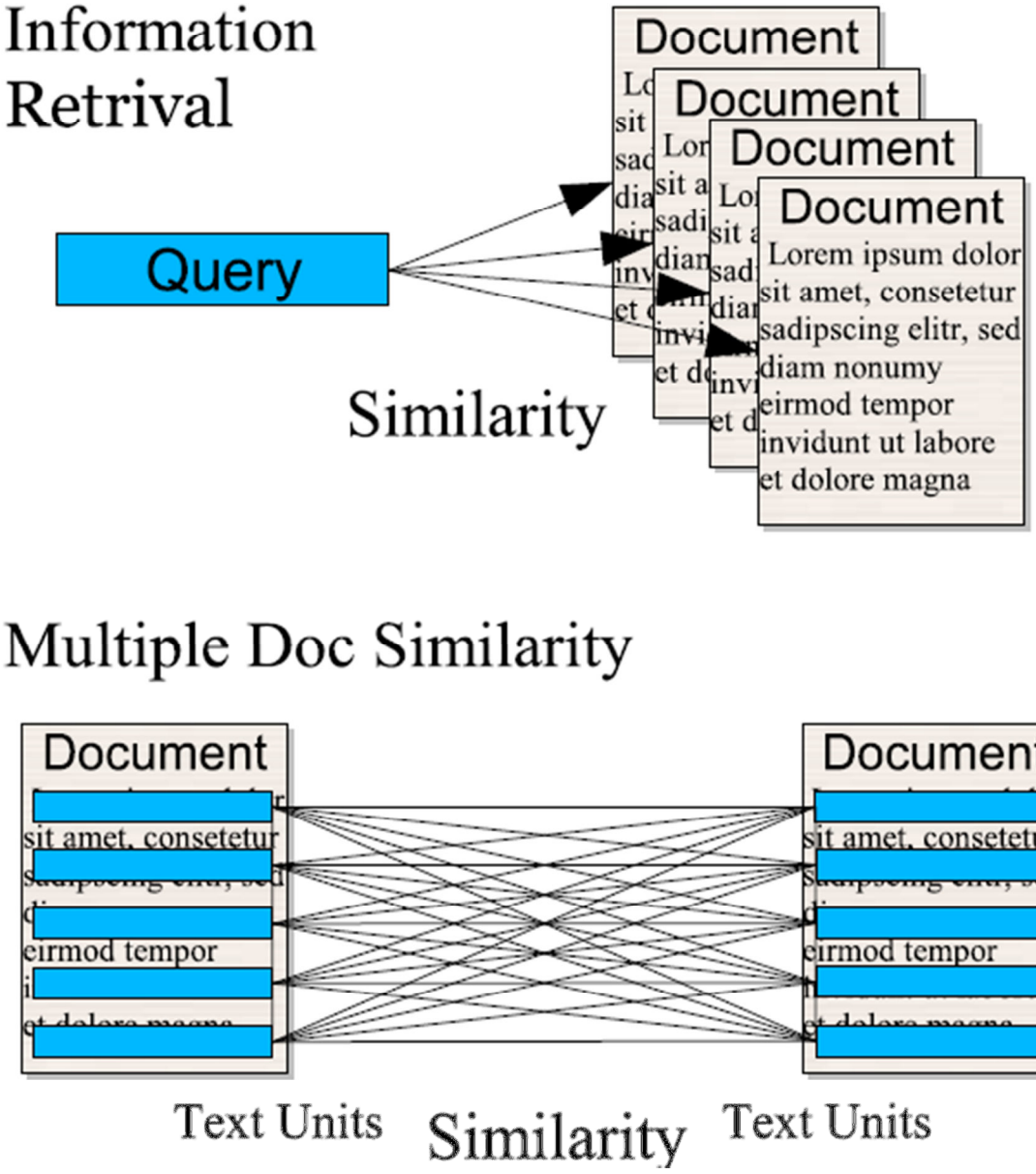


Figure 2.1: Comparison of IR to Multiple Document Similarity

- ✓ Clustering text units via their similarity to create the output text \themes" Cluster analysis is a general technique for multivariate analysis that assigns items to groups automatically based on a similarity computation. Cluster analysis has been applied to Information Retrieval to provide more efficient or more effective retrieval, and to structure large sets of retrieved documents. When applying clustering to text documents, the attributes over which the clustering is performed and their representation must be selected, and a clustering method and similarity measure must be chosen.

When applied to information retrieval, data sets are often very large, from hundreds to tens of thousands of documents, which necessitate an efficient representation for processing the documents. The documents are usually represented as word-space vectors.

2.1.2.1 Similarity measures - using term overlap

In a survey of document clustering techniques, Rasmussen 1992 [9] finds that the similarity measures used for clustering are easy to compute based on term counts, usually the Dice coefficient, Jaccard coefficient, or cosine coefficient. These measures are computed based on the term occurrences in the documents.

Dice coefficient:

$$S_{D_i,D_j} = \frac{2\sum_{k=1}^{L}(weight_{ik}weight_{jk})}{\sum_{k=1}^{L} weight_{ik}^2 + \sum_{k=1}^{L} weight_{jk}^2}$$

Where SDi, Dj is the Similarity of Document i compared to Document j, L is the total number of different words in the corpus, and weight ik is the weight of term k in documents i. The Dice coefficient takes into account the shared terms between two documents, and all of the separate occurrences of the terms in each of the documents. In Champollion, a system for statistical identification of collocation translations [8], the Dice coefficient is used as the similarity measure between collocations in different languages, since the Dice coefficient uses information on joint occurrences, and is not affected by cases where the term does not occur in either document.

Jaccard coefficient:

$$S_{D_i,D_j} = \frac{\sum_{k=1}^{L}(weight_{ik}weight_{jk})}{\sum_{k=1}^{L} weight_{ik}^2 + \sum_{k=1}^{L} weight_{jk}^2 - \sum_{k=1}^{L}(weight_{ik}weight_{jk})}$$

The Jaccard coefficient also takes into account the terms shared between two documents, but normalizes based on the union of the terms.

Cosine coefficient:

$$S_{D_i,D_j} = \frac{\sum_{k=1}^{L}(weight_{ik}weight_{jk})}{\sqrt{\sum_{k=1}^{L} weight_{ik}^2 \sum_{k=1}^{L} weight_{jk}^2}}$$

The cosine coefficient is commonly used in information retrieval applications, and measures the \angle" between two documents represented as word-space vectors. The calculation is quick to perform, and insensitive to the number of occurrences of terms in the document.

For efficiency reasons, these similarity measures are computed using only term overlap; usually concepts such as term order, predicate-argument structure, and so on are ignored. Due to the sparse nature of the data when using text units the size of a sentence or paragraph, term overlap alone is not sufficient for our task. Similarity finder uses multiple linguistically motivated complex features to compute a similarity measure. These complex features have been shown to improve clustering performance for our data when compared to using only term overlap, evaluating clustering performance on one of our test data sets. Previous work on document clustering has not shown any clear preference of similarity measure (Rasmussen 1992 [9],) although the three listed above are often used in information retrieval due to their ease of implementation and the property of normalizing for length.

2.1.2.2 Clustering methods

There are two general classes of clustering methods, hierarchical and non-hierarchical. When applying these methods to document clustering, especially for information retrieval, the algorithms used are honed for efficiency so large document sets can be clustered. What differentiates the methods used is how similarity between points (documents or clusters) is computed. In the single link method, the closest previously unlinked points are joined, when distance between two clusters is defined as the distance between the closest two points between the clusters. The complete link method merges clusters based on the sum of the distances between all pairs of documents in two clusters. The group average takes the average distance of all pairs of documents in the two clusters as the distance. Ward's method merges the clusters whose merge minimizes the increase in the total within-group variance.

Studies have been conducted to examine which clustering methods are best for clustering large document sets. Voorhees [36] compared the single link, complete link, and group average methods of hierarchical clustering on document collections of up to 12,684 documents, and found that complete link was most effective for larger collections with complete and group average link comparable for smaller

collections. El-Hamdouchi and Willet compared the same methods plus Ward's method on document sets of up to 2,361 documents, and found that the group average method was most effective for document clustering. Hatzivassiloglou et al. [10] examine single link, complete link, group average, and single pass clustering methods using linguistic features in the distance metric for document clustering. They found in tests using as many as 40,000 documents that group average was the best clustering method, and that inclusion of linguistic features improved overall performance.

2.2 English similarity finder

The similarity finder program was developed to identify short passages of text that are similar to each other from a set of multiple documents on the same topic. Similarity finder has been developed to work with text from the domain of edited news text, where sentences often constitute entire paragraphs. As a pre-processing stage to similarity finder, documents are often sentence segmented, but in the news domain it can be helpful for similarity finder to use paragraphs, rather than sentences, as the unit of text because a paragraph is more likely to contain background information (such as proper nouns) relevant to semantic comparison. Similarity finder uses many linguistically-motivated primitives for short-passage-level, either sentence or paragraph, similarity detection.

2.2.1 Similarity measure - Combining Linguistics and Machine Learning

Similarity finder identifies similar pieces of text by computing similarity over multiple features. There are two types of features, composite features, and unary features. All features are computed over primitives, syntactic, linguistic, or knowledge-based information units extracted from the sentences. Both composite and unary features are constructed over the primitives. Hatzivassiloglou et al.'s 2001 paper on similarity finder [14], illustrates some example primitives extracted by similarity finder through the use of two example similar paragraphs from the similarity finder training corpus. Typical types of primitives that are extracted by similarity finder include part-of-speech based primitives like all nouns, all verbs, or all adjectives. From the example, the verb primitives in the first sentence are (make, voice), and in the second sentence are (reject, mediate, say, invite, come, asses). While there are not any matches on the verb primitive type, there are matches on the noun and stemmed token primitive types, shown in the example in bold type. Unary features are feature that compare two sentences based on the overlap of a single primitive between the sentences, such as stemmed tokens or nouns. A unary feature over primitive p computes the similarity as the number of primitives of type p the two sentences

U.N. Human Rights Commissioner Mary Robinson made a landmark visit to Mexico at the government's invitation after voicing alarm last year of violence in the country's conflict-torn southern state of Chiapas.
Mexico's government last year rejected suggestions the United Nations might mediate in the long running Chiapas conflict, saying it could solve its own internal affairs. But it did invite Robinson and a special rapporteur on extrajudicial killings to come and assess human rights for themselves in the country.

Figure 2.2: Two similar paragraphs; the primitive features indicating similarities that are captured by similarity finder are highlighted in bold.

An OH-58 helicopter, carrying a crew of two, was on a routine training orientation when contact was lost at about 11:30 a.m. Saturday (9:30 p.m. EST Friday)
There were two people on board," said Bacon. \We lost radar contact with the helicopter About 9:15 EST (0215 GMT)."

Figure 2.3: A composite feature over word primitives, with the restriction that one primitive must be a noun and one must be a verb.

Share in common divided by the number of unique primitive's p in the two sentences. Unary features return a floating point similarity value in the range of 0{1. The more complex composite features return similarity values of either 1 or 0, and take two types of primitives.
The composite feature returns 1 if the two sentences both have instances of the primitives specified by the composite feature that match any restrictions on the composite feature(that the primitives appear in the same order, or are within a certain number of words from each other.) Figure 2.3 and Figure 2.4 illustrate two types of composite feature matches.

The paragraphs in Figure 2.2 have quite a few words in common, including government ,last, year, and country. They share several proper nouns: Robinson, Mexico, and Chiapas, which one might intuitively think should be weighted more for a match. Other similarities include words with the same stem, such as invitation and invite, and semantically related words such as killings and violence. Each of the

these matches between words with the same stems are examples of matches on the stemmed token primitive, while the matches

> Boris Yeltsin was hospitalized Monday with what doctor's suspect is pneumonia, the latest Sickness to beset the often ailing 68-year-old Russian president.
>
> Yeltsin has been hospitalized several times in the past three years, usually with respiratory Infections, including twice for pneumonia in 199 and 1998. The Kremlin tends to hospitalize The ailing president at the first sign of illness.

Figure 2.4: A pair of paragraphs that contain a composite match; a word match and Award Netmatch (highlighted in bold) occur within a window of five words, excluding Stopwords.

Between Mexico and Robinson are also matches on the Link IT noun phrase primitive, described in more detail in Section 2.2.1.1. The primitive features include several ways to define a match on a given word: considers matches involving identical words, as well as words that matched on their stem, as noun phrase heads ignoring modifiers, and as word Net [22] synonyms. The matches of primitive features can be further constrained by part of speech and combined to form composite features attempting to capture syntactic patterns where two primitive features have to match within a window of five words (not including stopwords). The composite features approximate in these manner syntactic relationships such as subject-verb or verb-object (see Figure 2.3, also from their paper). In other cases, a composite feature can serve as a more effective version of a single primitive feature. For example, Figure 2.4 illustrates a composite feature involving word Net primitives (i.e., words match if they share immediate hyponyms in word Net) and exact word match primitives. On its own, the word Net feature might introduce too much noise, but in conjunction with the exact word match feature it can be a useful indicator of similarity.

2.2.1.1 Identifying and Relating Noun Phrases: Link IT

One of the important features used in Similarity finder is the Link IT feature, which indicates Matches based on the heads of noun phrases. The motivation behind this primitive is my previous work using Link IT for document characterization, indexing, and browsing.

I developed Link IT as a document analysis and characterization system. Link IT identifies noun phrases in documents, and relates noun phrases within a document. I builtva grammar for noun phrase detection over part-of-speech tagged text for identification of noun phrases in documents, and a parser that builds links between the nouns phrases as they are extracted. Within a single document, noun phrases with the same head are linked together. Yarowsky [40] shows that with a single document, and often a single coherent collection, words tend to be used in the same sense, so

linking together instances of the noun phrases on the head brings together semantically related concepts. Presenting the list of related noun phrases can help to disambiguate the sense of the head by providing more contexts to the term.

The use of noun phrases as index terms leads to a high quality browsing interface ,as shown in [38] which describes Intel Index, a document browsing index for Digital Libraries built using the output of Link IT to enable browsing by noun phrases. Noun phrases have also been shown to be useful in two other NLP tasks which depend critically on similarity: information retrieval and document clustering. D. A. Evans and C. Zhai [8]examine the use of noun phrases as index terms in an information retrieval engine, and found that indexing based on components of complex noun phrases improves both precision and recall. Noun phrases and proper noun phrases were shown to have a significant benefit in improving performance of the document clustering system described in Hatzivassiloglouetal. 2000 [10]. These applications of noun phrases to similarity based tasks indicate that they are a useful area to focus on for lingual similarity detection.

2.2.1.3 Learning Method and Results

With such a large number of features available to Similarity finder to use, one would like to have a way to automatically choose those features that are most helpful for the similarity identification task. Some of the features may have high values for all sentences, including those which are not similar, while more useful features will have high values for similar sentences only. To determine which features are useful, a training set of similar and dissimilar sentences created, and a machine learning framework is used to identify which features are important over the training data.

A data set consisting of 10,535 manually marked pairs of paragraphs from the Reuters part of the 199 TDT pilot corpuses was developed. Each pair of paragraphs was judged by two human subjects, working separately. The subjects were asked to make a binary determination on whether the two paragraphs contained \common information". This was defined to be the case if the paragraphs referred to the same object and the object either
(a) Performed the same action in both paragraphs, or
(b) Was described in the same way
In both paragraphs. The subjects were then instructed to resolve each instance about which they had disagreed. In this and subsequent annotation experiments they found significant disagreements between the judges, and large variability in their rate of agreement (kappa statistics between 0.08 and 0.82). The disagreement was however significantly lower when the instructions were as specific as the version above and those annotators were able to resolve their differences and come with a single label of similar or not similar when they conferred after producing their individual judgments. The level of similarity that is represented in the training data and that Similarity finder to recover automatically is much more one grained than in a typical information retrieval application; going from topical similarity down to the level of propositional content similarity. This same training data is also re-used to train the English component of Similarity finder.

The first version of Similarity finder output binary similarity values for each pair of input sentences using a rule-based classifier learned from the training data over the features that Similarity finder computed for each sentence pair. Similarity finder used a classifier trained over both primitive and composite features using RIPPER [5]. RIPPER produces a set of ordered rules that can be used to judge any pair of paragraphs as similar or non-similar. Using three fold cross-validations over the training data, RIPPER included 11 of the 43 features in its final of rules and achieved 44.1% precision at 44.4% recall. The ten unary features were word overlap, proper noun overlap, Link IT overlap, verb overlap, noun overlap, adjective overlap, word Net overlap, word Net verb overlap, verb overlap, and stem overlap. One composite feature was selected, word Net collocation, which is a match between the WorldNet primitive and the word primitive (see [13] for more details on the various features). The selection of eleven features rather than just words validates the claim that more than word matching is needed for effective paragraph matching for summarization. The claim is also verified experimentally; the standard TF*IDF measure [34], which bases similarity on shared words weighted according to their frequency in each text unit and their rarity across text units, yielded 32.6% precision at 39.1% recall. They also measured the performance of a standard IR system on this task; the SMART system [2], which uses a modified TF*IDF approach, achieved 34.1% precision at 36.% recall.

21 of the 43 original features were normalized according to the matching primitives' IDF scores (the number of documents in the training collection they appear in). RIPPER selected none of those features, which suggests that TF*IDF is not an appropriate metric to use in evaluating similarity between small text units in a system such as ours. This observation makes sense given that in Similarity finder the collection of documents from which document frequency is calculated has been altered by topic and date. Thus, a primitive that would be rare in a large corpus could have an abnormally high frequency in the relatively small set of related documents on which Similarity finder operates.

The current version of Similarity finder, Similarity finder1.1, changed the machine learning approach to allow for values of similarity in the full range between 0 and 1 rather than the \yes"/\no" decisions that RIPPER supports. Such real-valued similarities enable the clustering component of Similarity finder to give higher weight to paragraph pairs that are more similar than others. Similarity finder1.1 uses a log-linear regression model to convert the evidence from the various features to a single similarity value. This is similar to a standard regression model (i.e., a weighted sum of the features) but properly accounts for the changes in the output variance as we go from the normal to the binomial distribution for a response between 0 and 1 [26].

A weighted sum of the input features is used as an intermediate predictor, ή which is related to the final response R via the logistic transformation

$$R = \frac{e^{\eta}}{1+e^{\eta}}.$$

Via an iterative process, stepwise refinement, the log-linear model automatically selects the input features that increase significantly the predictive capability of the model, thus avoiding overlearning. Their model selected input features, and resulted in a remarkable increase in performance over the RIPPER output (which itself offered significant improvement over standard IR methods), to 49.3% precision at 52.9% recall. The seven features selected are a sub-set of the features selected by RIPPER: six unary features, word stem overlap, noun overlap, verb overlap, adjective overlap, word Net overlap, proper noun overlap, and Link IT overlap. The single composite feature selected matches to the word Net primitive and a word primitive. As in the case of the RIPPER model, the automatic selection of multiple features in the log linear model validates the hypothesis that more than straightforward word matching is needed for effectively detecting similarity between small pieces of text. The focus on noun phrases, as seen by the selection of the Link IT feature, is also continued in this model.

2.2.2 Clustering Algorithm Tailored for Summarization

Once similarities between any two text units have been calculated, they are fed to a clustering algorithm that partitions the text units into clusters of closely related ones. Similarity finder clustering algorithm [14] departs from traditional IR algorithms, and is instead tailored to the summarization task's requirements. In Information Retrieval, hierarchical algorithms such as single-link, complete-link, and group wise-average, as well as online variants such as single pass are often used [9]. Compared to non-hierarchical techniques, such algorithms trade some of the quality of the produced clustering for speed [18], or are sometimes imposed because of additional requirements of the task (e.g., when documents must be processed sequentially as they arrive). For summarization, however, the distinctions between paragraphs are often one-grained, and there are usually much fewer related paragraphs to cluster than documents in an IR application.

Similarity finder uses a non-hierarchical clustering technique, the exchange method [Sp a85], which casts the clustering problem as an optimization task and seeks to minimize an objective function measuring the within-cluster dissimilarity in a partition P = fC1;C2; : : : ;Ckg

$$\Phi(P) = \sum_{i=1}^{k} \left(\frac{1}{|C_i|} \sum_{x,y \in C_i x \neq y} d(x,y) \right)$$

Where the dissimilarity d(x; y) is one minus the similarity between x and y. The algorithm proceeds by creating an initial partition of the text units that are to be clustered, and then looking for locally optimal moves and swaps of text units

between clusters that improve until convergence is achieved. Since it is a hill climbing method, the algorithm is called multiple times from randomly selected starting points, and the best overall configuration is selected as the final result.

The clustering method is further modified to address some of the characteristics of data sets in summarization applications. To reduce the number of paragraphs considered for clustering, an adjustable threshold is imposed on the similarity values, ignoring paragraph pairs for which their evidence of similarity is too weak. By adjusting this threshold, the system can be made to create small, high-quality clusters or large, noisy clusters as needed. Since every paragraph in that altered set is similar to at least another one, an additional constraint on the clustering algorithm to never produce singleton clusters is imposed.

Similarity finder also uses a heuristic for estimating the number of clusters for a given set of paragraphs. Since each cluster is subsequently transformed into a single sentence of the final summary, many small clusters would result in an overly lengthy summary while a few large clusters would result in a summary that omits important information. Similarity finder uses information on the number of links passing the similarity threshold between the clustered paragraphs, interpolating the number of clusters between the number of connected components in the corresponding graph (few clusters, for very dense graphs) and half of the number of paragraphs (lots of clusters, for very sparse graphs). In other words, the number of clusters c for a set of n text units in m connected components is determined as

$$c = m + \left(\frac{n}{2} - m\right)\left(1 - \frac{\log(L)}{\log(P)}\right)$$

$$P\ (= n(n-1)/2)$$

Where L is the observed number of links and p is the maximum possible Number of links. Similarity finder uses a non-linear interpolating function to account for the fact that, usually, L is less than or equal to P. The features selected for use with the log-linear regression model are word stem overlap, noun overlap, verb overlap, adjective overlap, word Net class overlap, proper noun overlap, and Link IT overlap. [14] presents further details as well as an evaluation of Similarity finder, and its application in two summarization systems.

2.3 A Flexible Framework for Similarity finder

I present work that uses Similarity finder with machine translated text as input. I also apply syntactic sentence simplification to English text that is used as input, thus

reducing sentence length and removing context. In both of these cases, Similarity finder is being used with input that is different from the sort of input used in its training, and so I made some modifications to the system to improve performance under these conditions.

Using Similarity finder to compute similarity between machine translate English sentences and English sentences, and details an altering step that I added which alters out sentences that are often not similar but that Similarity finder labels as similar when using syntactically simplified sentences and machine translated input. The alter removes sentence pairs with a cosine similarity below the threshold of 0.1, which has a 6% accuracy of identifying sentences that humans judged as not similar despite having a high Similarity finder similarity score.

Similarity finder presents the starting point for my original work in the area of lingual sentence similarity. Similarity finder, presented in full in Chapter 3, is a re-implementation of the ideas from Similarity finder, along with a framework that allows for easier addition of features and primitives, and a translation stage for relating primitives across languages. The ability to easily create new primitives is important for lingual similarity, as different languages can have vastly different computational resources available. With Similarity finder it is possible to define the primitives and features to use at run-time in a configuration, allowing one to use Similarity finder with different languages without modification of the program itself, which would not have been possible with Similarity finder.

CHAPTER 3

SIMILARITY BASED TEXTS SUMMARIZATION

Similarity finder is a re-implementation of the English version of Similarity finder. Focuses on adding support for computing similarity between multiple languages by making it easy to add new features and primitives. This chapter presents previous work in lingual text similarity, the approach I have taken to English language text similarity, the architecture of the Similarity finder system, and a description of the work required to add support for the language to Similarity finder.

3.1 Motivation

There are many applications of text similarity in Natural Language Processing. Approaches to multi-document summarization using text similarity have excelled at identifying content that is repeated and emphasized in the document set, and are able to take advantage of the identification of repetition to include important information and reduce redundancy in the summary. Text similarity measures have also been used in question answering systems, again to indicate importance via identifying repetition of text, and to reduce redundancy. Other opportunities for monolingual text similarity are for plagiarism detection and the detection of similar patent applications in an overburdened patent filing office. One area that has not seen much focus is lingual text similarity.

Similarity metrics would be useful is in machine translation. A good text similarity metric could be used as a scoring function for a statistical machine translation system, although Similarity finder in practice isn't designed for that sort of use. Given a foreign language string, and multiple generated translations, the text similarity metric could be used to prune non-similar translations, retaining similar ones for scoring via a language model of the target language. The core hypothesis of my similarity detection approach is that similarity between sentence-level units can be computed on the basis of easily extracted low-level primitives, without the need to explicitly model semantic sentence meaning. Extending this idea to similarity computation between languages, I hypothesize that similarity can be modeled by identifying simple lexical and syntactic primitives in the source and target languages, and by using translation at the level of the primitives to generate matches for the features used to compute the similarity score. This approach is attractive in that it allows for easy integration of foreign languages for which not many resources are available; if large parallel corpora are available, a statistical translation dictionary can be learned which achieves moderate performance.

The approach I have taken to similarity computation in Similarity finder is to:

- ✓ Identify and extract primitives basic units compared between sentences from the text
- ✓ Translate primitives between languages
- ✓ Compute features over extracted primitives
- ✓ Merge feature similarity values between sentences into a single, final similarity value for each sentence pair

Similarity finder identifies similar pieces of text by computing similarity over multiple features. All features are computed over primitives, syntactic, linguistic, or knowledge based information units extracted from the sentences. Examples of primitives are all nouns In a sentence, all verbs, all person names, or other sorts of information that can be identified automatically that might indicate similarity on some axis that can be separated from other axes. Primitives are extracted by modules that are loaded at runtime for each language, and features are defined over the extracted primitives. Both primitives and features are explained in more detail in Section 3.3.2 and Section 3.3.4.

Section 3.3 presents details about Similarity finder's architecture, and how the above steps are carried out, while section 3.4 is an in-depth discussion about adding English language support to Similarity finder and evaluation results. Support for other languages is discussed in section 3.5.

3.2 Related work in sentence similarity based text summarization

The English version of Similarity finder is the main innocence on Similarity finder, but is not included in this section as it is a monolingual system. Similarity finder takes the approach to text similarity introduced by English Similarity finder and modularizes the system to make it easier to add new features and primitives, as well as support for translation mechanisms between languages to allow for lingual similarity computation. This section focuses on other work in lingual text similarity.

3.2.1 Example based machine translation

Example based machine translation systems [6] became popular in the 1980's and 1990's, and introduced a new paradigm for machine translation: using similarity to previous translations to generate a new translation. In example based machine translation systems, an input source sentence is matched to other source sentences in a translation database via a similarity metric. The translation database typically contains short sentences or phrases in the source language, and aligned translations into the target language made by a professional translator or automatically through corpus alignment methods. The similarity metric typically involves part of speech tagging and low-level parsing or thesauri and other knowledge bases to identify possible synonyms or words substituting from a similar semantic or grammatical category. Exact matches between the source sentence and translation database improve the score, while matches on tokens with the same semantic or grammatical

category improve the score less so and a lack of match on a token decreases the score. Usually multiple matches to phrases are used to cover the entire source sentence to translate. Translation involves substituting the target language translation for each example matched for the source sentence, replacing words in each example that were not exact matches, and ordering and re-generating any connective text from the examples to cover the entire sentence.

There are many differences between example based machine translation system and Similarity finder. Similarity finder similarity metric is lingual; in example based machine translation systems, a source language sentence to be translated is matched to other source language sentences, while in Similarity finder similarity is computed between all the input sentences, some of which are in similar languages. Also, the examples in the translation database are often not full sentences as would be found in the news domain, but shorter sentence fragments. Similarity finder computes similarity between full sentences, and allows For a larger deviation in the structural similarity between the sentences, compared to example based machine translation which requires high syntactic similarity between the source and example for the translation of the example to be applicable.

The most related aspect of cross-language information retrieval to similarity finder is that of query translation and query {document similarity computation. In CLIR, short queries are translated into a language, and a similarity measure is computed between the query and documents. Similarity finder computes similarity between all sentences, not just a single query. Some of the same problems appear in both contexts, but since similarity finder deals with sentences, and not full documents, the problem of over-generalization when translating a term is not as severe. Additional terms added to a translated[5,3] query that have a different sense from the original query term are problematic because in large collections, it is likely that some document contains the spurious term. Similarity finder deals with shorter units of text and a spurious term are not as likely to appear. Similarity finder also uses bilingual lexicons for translation, and morphological analysis software or stemming to normalize words, and proper name identification and translation is implemented using BBN's identifiers. Similarity finder approach is more sophisticated; using Jaccard-like similarity over multiple features combined using a log-linear regression into a single similarity value, as compared to just doing a cosine vector-space distance in the term space, which is a common information retrieval approach. Similarity finder use of multiple features and ability to compute similarity for one-grained units of text set it apart from CLIR systems.

3.2.3 Statistical machine translation

Brown et al. [18] introduced a statistical machine translation system in the 1990's that has spurred a huge amount of research into purely statistical based machine translation. In this approach, language translation is viewed as the task of constructing a language model that estimates the probability of a given sentence S in the source language, and a translation model that estimates the probability of producing a target sentence T given a source sentence S. Translation is then cast as maximizing

$$\Pr(S,T) = \Pr(S) \times \Pr(T|S)$$

The cross-language similarity portion of similarity finder would fit well into this sort of framework for similarity identification, since it mirrors the translation task well.

Similarity finder does use results from statistical machine translation community by taking advantage of models for learning probabilistic dictionaries. In implementing the English portion of similarity finder, I use a dictionary learned from an IBM model 3 style translation probability models, which helped improve results over translation by dictionary lookup alone. A distortion model might also help improve similarity finder results at finding sentences that are translations of each other, however, since similarity finder is searching for similar sentences that might not be translations of each other, a distortion model might impose too many restrictions, giving similar, but structurally different sentences, low probabilities. Application of an IBM-style statistical model to intra-language similarity computation would be interesting as well, but faces the problem of training data. Given enough examples of sentences that are similar to each other, I think a statistical model that encodes the similarity of words such as shoot and attack would be very useful, although these sorts of relationships are also available by using primitives informed by word Net or other linguistic knowledge bases.

3.2.4 Sentence alignment cost functions

Parallel corpus sentence alignment is another area that implements a cross-lingual similarity function. The earliest approach, Gale and Church's program for bilingual sentence alignment [9], uses word length in characters as the main cost function between languages and dynamic programming to find the best alignment over sentences. Using even just a simple cost function as length resulted in surprisingly good results between French and English. More recent approaches such as [24] improve on the cost function using bilingual lexicons, or learning them on the way, and make improvements in adding linguistically derived information, such as statistical phrases or sub tree grammars.

Similarity finder does not perform the same task as sentence alignment because sentences are not assumed to map to another sentence in the target language; the approach of computing a cost for alignment and then maximizing the total cost to

match sentences to each other (or null) is not valid in this context. Similarity finder does make use of similar ideas though, especially in employing bilingual lexicons to anchor matches between the languages.

3.2.5 Lingual Phrase Translation

Malamud's method for discovering non-compositional compounds in parallel text [23] Takes a similar approach, but does not require a list of collocations in the source language. His method compares translation models that contain potential non-compositional compounds built up word-by-word from highly correlated terms in parallel corpora to translation models that do not contain the potential non-compositional compound, and chooses to include compounds that increase the predictive power of the translation model. This method is only capable of finding non-compositional compounds that are not translated word-for-word, and the compounds it finds translate as a unit, but might not be considered collocations in the source language.

3.3 Similarity finder Architecture

Similarity finder is designed to be modular system. Similarity finder identifies similar pieces of text by computing similarity over multiple features. There are two types of features, composite features, and unary features. All features are computed over primitives, syntactic, linguistic, or knowledge-based information units extracted from the sentences. Both composite and unary features are constructed over the primitives. The primitives used and features computed can be set at run-time, allowing for easy experimentation with different settings, and making it easy to add new features and primitives. Support for new languages is added to the system by developing modules conforming to interfaces for text pre-processing and primitive extraction for the language, and using existing dictionary-based translation methods, or adding other language-specific translation methods. As shown in

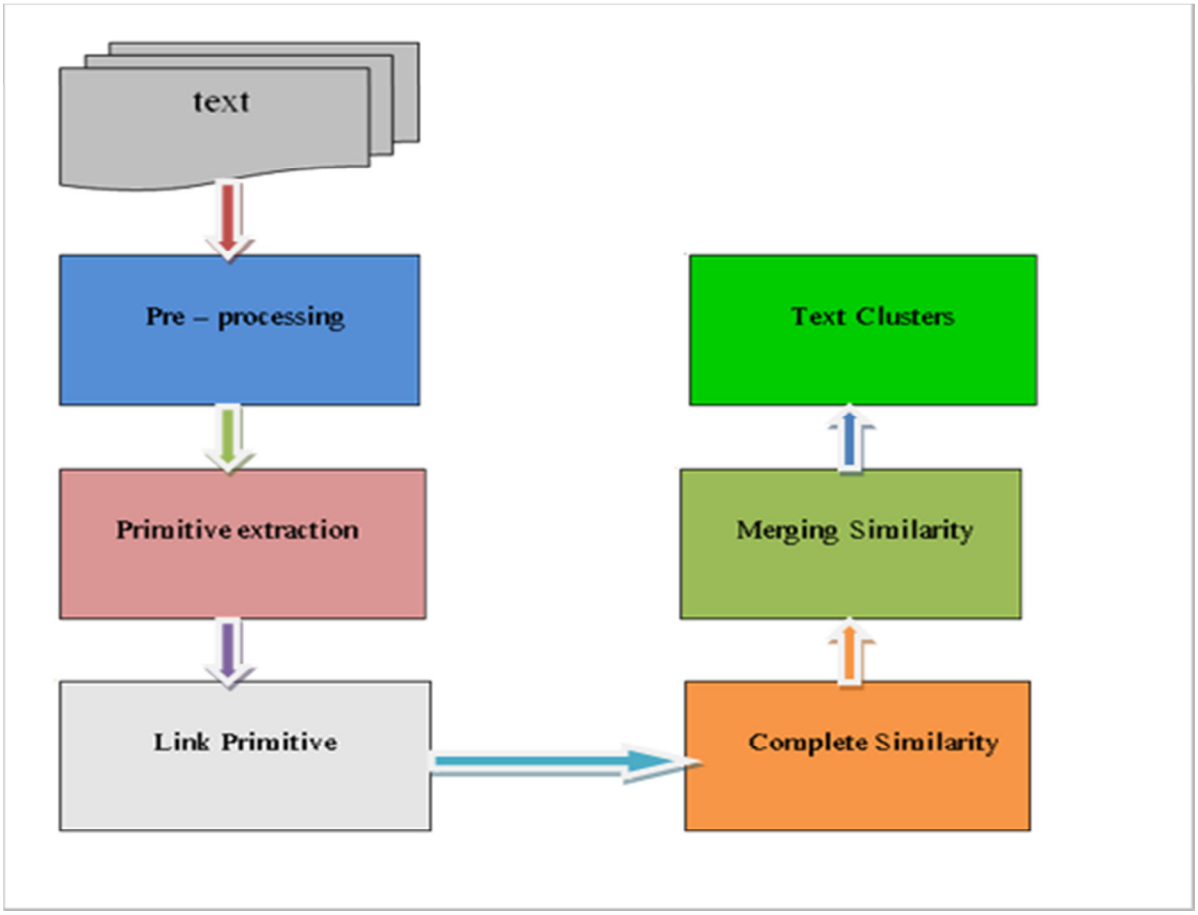


Figure 3.1: similarity finder Architecture.

3.3.1 Pre-processing

The first module is a pre-processing module, which prepares the input articles for processing. I have designed a language-independent API that abstracts the generalized pre-processing steps for the similarity discovery task. The steps in the pre-processing stage are to segment the text of the documents into units to compare for similarity, and to create alternative representations of the text, such as part of speech tagged versions, that will be used in later stages to extract primitives.

Similarity finder supports using different levels of granularity for similarity computation by segmenting the text into units using a user-specified segmentation class. I have focused on computing similarity at the sentence level, but similarity finder is not limited to processing sentences. Sentences offer a unit that can stand on their own, and while anaphoric reference can be a problem, the level of the sentence has been a good unit to work with for many applications. To support different text segmentation schemes, a user needs only to create a Java class that adheres to the sentence segmentation interface, and since these classes are loaded at runtime,

changing the type of segmentation used is very easy. I have implemented English sentence segmentation using simple regular-expressions based classes, and an interface to the MXTerminator1 [6] sentence segmentation program for English. The second part of the pre-processing stage is to create different representations of the text that will be used to extract primitives. The representations of the text react some form of mark-up or tagging that might be used in the primitive extraction phase to identify and extract primitives from the text. As with the other stages, classes are loaded at run-time to perform this task, making it easy to add new representations for a language. I have implemented English part-of-speech tagging, English and Japanese morphological processing, and English named entity recognition using existing tools in the similarity finder framework via this interface.

3.3.2 Primitive Extraction

In order to define similarity between two units, we need to identify the atomic elements used to compute similarity. These are called primitives. Primitives are general classes (for example, all stemmed words, all nouns, all noun phrases), while a particular instance of a primitive would be a specific word, or a specific noun phrase. Similarity between two units is computed using features over these primitives, which will be discussed shortly. The second stage identifies and extracts primitives for each unit. Primitive extractors are defined on a per-language basis using a plug-in architecture making it easy to add support for different languages by simply creating primitive extractors for that language. The primitive extraction, primitive linking, and similarity computation phases all interact with data structures that track which units contain which primitives on a per-language basis. These data structures allow us to select sets of text units that contain common primitives for comparison, while avoiding comparisons between text units that do not contain any primitives in common, and provide a central location at which to translate all of the primitive types that are seen in English.

There are ten primitive extractors implemented for English: all tokens, stemmed tokens, word Net classes, nouns, verbs, proper nouns, heads of noun phrases, adjectives, cardinals, and named entities. Token primitive extractors have also been implemented. The primitive extractor performs word segmentation using the main segment. Perl dictionary-based Chinese word segmentation program from the LDC. The Japanese primitive extractor first processes the text with Chosen [1], and then extracts the morphologically-analyzed text.

For example, for the sentence.

> The inspectors withdrew on Wednesday, a day after U.N. inspection chief Richard Butler told the U.N. Security Council that Iraq was not cooperating.

tokens	the inspectors withdrew on wednesday a day after u.n. inspection chief richard butler told security council that iraq was not cooperating
nouns	inspectors Wednesday day U.N. inspection chief Richard Butler Security Council Iraq
verbs	withdrew told cooperating
WordNet	(Iraq, Republic of Iraq, Al-Iraq, Irak) (examiner, inspector) (withdraw, retreat, pull away, draw back, recede, pull back, retire, move back) (Wednesday, Wed) (day, twenty-four hours, solar day, mean solar day) (subsequently, later, afterwards, afterward, after, later on) (United Nations, UN) (inspection, review) (head, chief, top dog) (state, say, tell) (security) (council) (collaborate, join forces, cooperate, get together)

And the following named entity primitives are extracted:

Named Entity	Category	Type
Wednesday	TIMEX	DATE
a day	TIMEX	DATE
U.N.	ENAMEX	ORGANIZATION
Richard Butler	ENAMEX	PERSON
U.N. Security Council	ENAMEX	ORGANIZATION
Iraq	ENAMEX	GPE

Primitive extractors operate over the original text of the unit, or use one of the representations created earlier, such as a named-entity of part-of-speech tagged version of the text. As each primitive is extracted, they are recorded in the text units, and entries are made in a per-language index (labeled Big Board in Figure 3.2) tracking which text units contain each primitive.

Features built over the primitives are used to compute how similar sentences are. For example, a pair of sentences will have five feature similarities computed that how

similar sentences are based on tokens, nouns, verbs, word Net, and named entity features. When all primitives have been extracted, Similarity finder relates primitives that mean the same thing across languages using a translation mechanism. Primitives within a language already track the units that contain the same primitive, and by using word Net primitives are identify.

3.3.3 Primitive Linking

Once all of the primitives have been extracted from the units, similarity finder collects lists of which units contain the same primitives. The final phase before features is computed over. The units are to determine which primitives from one language are translations of primitives in another language. In my application, I am concerned with finding translations from a non-English language into English, since I am working under the assumption that I will always have some English language input. Because of this, I focus on finding similarity from non-English to English text units. Extending similarity finder to search for links between a language and another non-English language would be quite easy as long as some translation facility existed for the language pair of interest. The translation facility does not have to be on the order of full machine translation; similarity finder has shown that translation using bilingual lexicons or learned probabilistic dictionaries can results in high-precision for cross lingual text similarity computation.

Similarity finder does not itself contain any mechanism for identifying and linking words that are synonymous. Within a single language, the choice of primitives is assumed to resolve problems of synonymy by extracting primitives that encapsulate that relationship, such as the word Net. Words that are synonyms will be mapped into the same word Net sunset, and thus match other word Net primitives for words in the same sunset.

The primitive linking phase is not a full translation phase. Since the goal is to use the translations to link to other potentially related primitives, I prefer to err on the side of opportunistically linking two primitives even if there might only be a tenuous relationship between them. Since there is at least one primitive for each token in a sentence, there are often a large number of primitives to compare between two sentences. Sentences that are similar usually have more than a single link between translated primitives due to additional links from other related words in the sentence. By making many links, even when the translation is tenuous, the additional matches from relevant words will help to reinforce similar sentences. Since our input consists of topically-clustered documents. Similarity finder supports some simple dictionary-based translation methods for linking primitives across languages. Similarity finder has support for three types of dictionary formats: a simple word to word format called the IDP dictionary format3, the edict format4 for Asian languages, and a simple probabilistic dictionary format for dictionaries learned from parallel corpora. Extending the dictionary support for other languages is quite simple by adhering to the generic dictionary interface.

3.3.4 Similarity Computation

Similarity between two units is computed on multiple features defined over the primitives identified for each unit. Before performing the actual comparison between the units, the units which should be compared are identified. Similarity finder uses an approach that avoids comparing units that will not be found to be similar. To collect units to compare, a primitive is chosen from the primitive-tracking data structure (Big Board for each language), and all units containing the primitive or a linked primitive are compared against each other. An N * N array, where N is the number of text units, tracks which units have been compared, ensuring that similarity is computed only once for each pair of units. A new primitive is selected, and the process is repeated until all primitives have been used for all languages. This approach only compares units that have a chance to be similar, while avoiding comparison between units that share no primitives in common. Units that have no primitives in common cannot be found to be similar by the similarity equation computed over the features, and will be skipped because they have no primitives in common, leaving The similarity comparison between two units is computed over multiple features defined on the primitives.
The most common feature is overlap between primitives of the same type. For example, if similarity finder has been set to extract token, verb and word Net primitives, three features that compare the overlap on each primitive could be set up. In that case, similarity finder would set up an N * N * 3 similarity matrix that tracks the similarity for each feature between pairs of Units. Each entry is computed as the number of primitives that are shared in common between the two units, divided by the total number of primitives in the two units, possibly normalized by the unit lengths. Primitives are weighted by the strength of the links between them if they are translations. Figure 3.4 shows how three primitives are linked between English sentences.

The similarity of two units, U1 and U2 with primitives P1 and P2, with the strength of a link between primitive P1a and P2b given as WP1a; P2b is defined as:

$$S_{U_1,U_2} = \frac{\sum_{a=1}^{|P_1|} \sum_{b=1}^{|P_2|} (W_{P_{1a},P_{2b}})}{|P_1 \cup P_2|}$$

Similarity finder also supports composite features, which compute a function that is either 0 or 1 depending on the state of two primitives between the units. A composite feature requires two primitives, such as verb and word Net primitives, and returns 1 if the two sentences both contain instances of the two specified primitives that match other criteria (the two must be within a certain distance of each other, and possibly react the same ordering, e.g., Verb, word Net and Verb, word Net.

F1: Sentence Position

We assume the first sentence of a paragraph is the most important. Therefore we rank a sentence in the paragraph according to their position. E.g. if there are 5 sentences in the paragraph then the 1st sentence have a score of 5/5, Then 2nd have score 4/5, 3rd have 3/5 and so on.

F2: Positive keyword in the sentence

Positive keyword is the keyword frequently included in the summary. It can be calculated as follows:

$$Score_{f2}(S) = \frac{1}{Length(s)} \sum_{i=1}^{n} tf_i * P$$

$$\text{Where} \quad P = \left| \frac{\text{No of Keywords in the sentence}}{\text{No of Keywords in the paragraph}} \right|$$

Tfi is the occurrence or frequency of it term in the sentence, which probably is a keyword.

F3: Sentence Relative Length

This feature is useful to filter out short sentences such as datelines and author name commonly found in news articles. The short sentences are not expected to belong in the summary. We use length of the sentences, which is the ratio of the number of word occurring in the sentence over number of word in the longest sentence in the document.

$$Score_{f3}(S) = \left| \frac{\text{No of words occurring in Sentence S}}{\text{No of words occurring in longest sentence}} \right|$$

F4: Sentence resemblance to title

It is the measure of vocabulary overlap between this sentence and the document title, generally the first sentence in the document is probably the title of the document. It is calculated as

$$Score_{f4}(S) = \left| \frac{\text{Keyword in S} \cap \text{Key word in title}}{\text{Keyword in S U Key word in title}} \right|$$

F5: Sentence inclusion of name entity (Proper noun)

Usually the sentence that contains more proper nouns is an important one and it is most probably included in the summary. Proper noun gives the literature of contents.

$$Score_{f5}(S) = \left| \frac{\text{Number of proper noun in S}}{\text{Length of S}} \right|$$

F6: Sentence inclusion of numerical data

Sentences that contain numerical data are more important than rest of sentences and are probably included in the summary.

$$Score_{f6}(S) = \left| \frac{\text{Number of numerical data in S}}{\text{Length of S}} \right|$$

F: Term Weight

The frequency of term occurrence within a document has often been used for calculating the importance of sentence. The score of sentence can be calculated as the sum of the score of word in the sentence. The score or weight wi of ith term or word can be calculated by traditional tf-idf.

$$Score_{f7}(S) = \frac{\sum_{i=1}^{i=k} W_i(S)}{\text{Max}\left(\sum_{i=1}^{i=k} W_i(S^N) \right)}$$

F8: Sentence similarity with other sentence

This feature measures the similarity between sentence S and each other sentences. It measures how much vocabulary overlap between this sentence and other sentences in the document. It is computed by cosine similarity measure with resulting between 0 and 1. The score of this feature for a sentence S is obtained by computing the ratio of

similarity of sentence S with each other sentence over the maximum similarity between two sentences.

$$Score_{f8}(S) = \frac{\sum Sim(S,S_j)}{MAX(\sum Sim(S_i,S_j))}$$

F9: Bushy path of the Sentence or node Sentence centrality

It has an overlapping vocal bury with several sentences it is defined as the number of links connected it to other sentences (node) on similarity graph. Highly busy node is linked to the number of other nodes. The busy path is calculated as follow:

$$Score_{f9}(S) = \frac{\#\text{ (branches connected to sentence (node) S)}}{\text{Max Degree in the scaled similarity graph}}$$

3.3.5 Merging Feature Similarity Values

The goal of similarity finder is to group textual units from multiple languages with similar meaning together. To do this, similarity finder uses a clustering algorithm over similarity values between the units. The clustering algorithm requires a single similarity value, but after the similarity computation stage, similarity is expressed over multiple features, so they must be merged into a single similarity value. This section deals with obtaining a single similarity value between units from the feature similarity values. Section 3.3.6 deals with clustering the units using the similarity values.

The similarity computation process used in similarity finder creates a similarity matrix between the units on several dimensions. For each of the primitives extracted from the units, a feature comparator is used to compare the similarity of the two units over that primitive. The similarity computation stage results in an N *N *F similarity matrix, where N is the number of textual units, and F is the number of features that were used during the run. Before clustering the units, the N *N * F feature similarity matrix is converted into an N *N matrix such that each element contains a single value expressing the total similarity between the two units.

The log-linear regression model, weights must be learned for the linear combination of the features. For the English version of similarity finder, a training set of similar textual units has been developed by human judges who made a similarity decision

over pairs of textual units. The lingual version of similarity finder requires the same sort of training data. Small Each of the annotators read all articles in the training document sets, containing English articles, and listed sentences in and English that expressed the same information. The amount of effort involved in this exercise was great, and since it would be very difficult to obtain a similar amount of training data used for English similarity finder, when training a model English similarity, I took a different approach: I used a sentence aligned parallel corpus for training examples. This alternative approach, which does not require manually annotated similarity training data.

3.3.5.1 Challenges for Lingual Feature Merging

While the above approach is tenable in the monolingual case where training data is available, there are additional problems in the lingual case. The features that are available for two textual units from different languages are usually different. For example, for English in similarity finder there are multiple primitives (part-of-speech based, stemmed tokens, word net classes, etc.) while only the token primitive has been implemented. When calculating the final similarity value between English and a text unit, the only feature that can be used is similarity as determined by overlap on tokens via dictionary lookup. As more primitives and more sophisticated primitive linking techniques are added, the number of features compatible between units in similar languages will change as well. Since similar languages will have different sets of compatible features, it is important to easily be able to switch feature merging models to suit the compatible primitives, and to be able to learn these models across languages.

To determine weights for the different combinations of language pairs, I perform a similar training step to learn the exponents for feature weighting as in the English case. This requires training data for the regression step, which is even more difficult to obtain than in the English monolingual case: the human judges have to be able to read and make similarity judgments over texts in all of the languages being clustered. Instead of tagging training data manually as was done for the English training data, I have taken a different approach and used data from the Machine Translation community. In Section 3.4.4 I detail the training data used for the English version of similarity finder, which is from the Multiple Translation English corpus from the LDC6. As with in the monolingual English case, the final similarity score is computed using a feature merging model that merges the feature similarity scores into a single similarity score. The training data for the feature merging model is generated in the same way as with the English case: similarity finder is run over the training data, creating feature similarity values for each training instance, using primitive translation as explained above to link primitives across languages. The sentences that are aligned in the parallel corpus or marked as similar in the case of manually annotated similarity training data) are marked as similar, other sentences are marked as not similar.

The similar sentences are transformed into a target value of 1 for the log-linear regression model, and 0 for not similar sentences. The log-linear regression then learns exponent values for the model to best approximate the target similarity value.

The clustering stage is unaffected by the lingual data, since it relies only upon the final similarity values.

3.3.6 Clustering

The process of clustering the textual units is a separate stage that uses the final similarity values computed. The clustering component uses the optimization-based method described in Hatzivassiloglou etal. 2001 [14]. The clustering method requires the number of output clusters to be specified, which is estimated for each input document set using the same estimation as the English version of similarity finder. The estimation is based on the similarity values between the textual units. The number of clusters c for a set of n textual units in m connected components is determined by-

$$c = m + \left(\frac{n}{2} - m\right)\left(1 - \frac{\log(L)}{\log(P)}\right)$$

$$P\ (= n(n-1)/2)$$

where L is the observed number of links between units based on their similarity being above a threshold, and P is the maximum possible number of links. A non-linear interpolating function is used to account for the fact that usually L is less than or equal to P. See Section 2.2.2 for more details.

3.4.2.1 Word feature matching

The basic primitive translation used is dictionary lookup in the Buck Walter morphological analyzer available from the LDC. A match is made between primitive and an English primitive if there is a non-stop word English translation in the Buck Walter lookup that matches the English primitive, with the strength of the match determined by the total number of English translations for the English word. Since each word may result in multiple analyses, and each analysis may contain multiple English glosses, the weight given to each English translation may be very small. No sense disambiguation is performed, so there may be spurious matches made. For an illustration of the translation method.

3.4.2.2 Using a probabilistic dictionary

The second translation method I use for English is lookup in a probabilistic translation dictionary. Similarity finder supports two translation dictionary formats, a simple word-to-word format, and a probabilistic format (see Section3.3.3 for supported dictionary formats.) The probabilistic format maps English tokens to

English tokens with probability for the likelihood of the translation. Describes the process used to learn the probabilistic dictionary used in similarity finder English word lookup. When using the probabilistic dictionary, primitive is looked up, and a link is made between the primitives for each target English token that exists. The strength of the link is assigned the probability of the translation from the dictionary. This translation method addresses one of the problems with the Buck Walter translation method; the probabilities in the dictionary assign links between likely translation pairs, and discount less-likely, but valid, translations.

3.4.2.3 Named entity feature matching

Named entity features are extracted from the text using BBN's Identity Finder for both English. A match is found between Identity Finder primitives using either dictionary lookup via the Buck Walter dictionary, or passing the entire named entity to a translation system.8 If using the machine translation system, the entire text of the translated named entity must match, otherwise, if there is at least one non-stop word overlap between the English and glosses for English word, a match is made. No disambiguation is performed, nor is locality of the text taken into account.

3.4.3 Learning a probabilistic English dictionary

To improve word translation I learned a probabilistic dictionary English using the GIZA software package [ON03] for statistical machine translation. I used the default settings for a model 3 alignment with the entire text of the English Parallel News Part 1 Corpus9 from the LDC. The corpus contains English news stories and their English translations LDC collected by the Ummah Press Service from January 2001 to September 2004. It totals 8,439 story pairs, 68,685 sentence pairs, words and 2.5M English words. LDC sentence-aligned the corpus, making it suitable to use for learning a translation dictionary. I generated the appropriate input for GIZA, ran GIZA, and used the resulting final word translation table to generate a dictionary that lists all words that were seen at least four times. Both the probabilistic dictionary and Buck Walter translation mechanisms have been used for various experiments reported.

3.4.4 Feature Merging Model Training Data

Similarity finder uses a log-linear model to generate the single similarity value between two sentences. In English case, a model must be learned that converts the two feature overlap values into a single similarity value. To do this, I require examples to use as training data for the regression analysis. Since using bilingual English annotators to mark sentences for similarity in a training corpus would be expensive and difficult to obtain, I used an existing corpus from the Machine Translation community of aligned translated sentences. The motivation is that sentences that are translations of each other are certainly similar to each other, and what is learned from training over this data should generalize to sentence pairs that, while not being exact translations of each other, are similar. The benefit of training

over data that is not the exact same as the target data that we plan to test with is that this type of training data is much more readily available.

I used the Multiple Translation corpus from the LDC as my training corpus. For each of the 141 English documents, I chose one of the manual English translations (the English translations labeled ahd, as those translations were generally accepted to be of the highest quality) and ran similarity finder over the pair of documents. This resulted in with training values for each of the sentence pairs that I then could use in training. Training data for the regression model was generated by marking each aligned English sentence pair as similar, and all other sentence pairs as not similar. The data was run though a general linearized model to retrieve exponents used to merge the feature values.

Table 3.1 and Table 3.2 shows the results from training feature merging models for both token and Identity Finder features, and just the token feature alone with different translations mechanisms. The tokens are translated using either lookup through the Buck Walter morphological analyzer, lookup in a learned probabilistic dictionary, or both. When using both resources for translation, English primitives are first looked up using the Buck Walter.

Thr	Token and IdentiFinder features					
esh	Buckwalter and Probabilistic		Probabilistic		Buckwalter	
old	**Precision**	**Recall**	Precision	Recall	Precision	Recall
0.1	**53**	**88**	43	80	36	76
0.2	**64**	**80**	57	68	47	57
0.3	**71**	**73**	63	60	54	44
0.4	**76**	**67**	69	54	55	33
0.5	**80**	**62**	73	46	58	26
0.6	**83**	**56**	77	40	60	21
0.7	**86**	**50**	80	34	65	17
0.8	**86**	**42**	86	27	66	12
0.9	**91**	**33**	89	19	68	7

Table 3.1: Feature merging model training results using token and Identity Finder features

With Buck Walter + Probabilistic, Probabilistic, and Buck Walter translation. system, and a link between the primitive and the target English translation are made. The English primitives are then looked up in the probabilistic dictionary, and additional links from the probabilistic dictionary are added. For different thresholds, the Precision and Recall training results for the similar class is given. During training, a test sentence pair is assigned a similar value if the similarity of the pair is

above the threshold. The best results are obtained using both token and Identity Finder features, using the combination of probabilistic and Buck Walter translation When using both the Token and Identity Finder features, in all cases using Probabilistic translation combined with Buck Walter translation resulted in improved precision and recall at every threshold over using just Probabilistic translation alone. The difference is statistically significant at the p = 0:05 value for both precision and recall using the paired Wilcoxon signed rank test. Similarly, Probabilistic translation alone outperforms using Buck Walter translation alone on the training data at every threshold, and is statistically significant at p = 0:01 for both precision and recall. Combining Buck Walter and probabilistic translation.

Thr	Token feature only					
esh	Buckwalter + Probabilistic		Probabilistic		Buckwalter	
old	Precision	Recall	Precision	Recall	Precision	Recall
0.1	53	88	43	79	37	75
0.2	63	80	57	68	48	55
0.3	69	72	64	60	53	43
0.4	74	67	69	54	58	35
0.5	77	60	74	46	60	27
0.6	70	55	77	41	62	21
0.7	84	49	81	43	65	17
0.8	89	41	86	27	66	13
0.9	91	30	90	18	68	7

Table 3.2: Feature merging model training results for token feature using Buck Walter and Probabilistic, Probabilistic, and Buck Walter translation.

Improves both precision and recall for training. Note that the source data used to learn the probabilities for the dictionary is different from the training data used here; the dictionary used data from 2001-2004 from the Ummah Press Service, while this training data is from 2001 from the AFP and Xinhua news services. The general genre and time frames do overlap, which means the dictionary is probably a good match for the data used. Using only the token feature, using both Probabilistic translation with Buck Walter translation outperformed using just Probabilistic translation alone for every threshold except for 0.6 in terms of precision, but always outperformed Probabilistic translation alone in terms of recall.

Probabilistic translation alone always outperformed Buck Walter translation alone. The differences in precision and recall are all statistically significantly greater at p = 0:05 using the paired Wilcoxon test. In either case, using both Probabilistic and Buck Walter translation provides the best performance. For all but one threshold (0.8) using the combination of the token and BBN Identity Finder features performs as

well or better than using tokens alone. The addition of the Identity Finder feature statistically signify frequently improved precision and recall over the token feature alone using probabilistic and Buck Walter translation according to the paired Wilcoxon test (p = 0:0593 for precision, p = 0:0099 for recall.) Adding more linguistically informed features has helped performance when looking at the training data, and as shown in section 4.2, also improves results when evaluated against unseen data.

3.4.5 Training Results

The learned model was added to similarity finder, was re-run over the training data. Each sentence was compared to the most similar English sentence as predicted by similarity finder89.00% of the sentences were correctly mapped back to their aligned counterpart. The average similarity of the most similar English sentence was 35.98%, but this rose to 3.51% when looking at only correctly mapped sentences (vs. 23.6% for incorrectly mapped sentences.) Figure 3.5 shows three examples of similar English sentences found by similarity finder. , machine translations of the sentences are provided in the blue boxes as a convenience to the reader, however similarity finder only uses the English sentences to perform the similarity computation.

3.5.1 Extracting article text from web pages

In order to work with similarity finder in similar languages, I needed to find a natural source of English and non-English news documents to work with. One source for such documents is the online news crawling and clustering component of Columbia News Blaster. I investigated methods for crawling and extracting article text in multiple languages, as well as clustering English and non-English text within the News Blaster framework. The following section discusses a new system Dave Elson and I developed for extracting the text of an article from crawled web pages that uses machine learning to enable support for English languages. One of the problems with using web news as a corpus is that we must be able to extract the \article text" from web pages in multiple languages. The article text is the portion of a web page that contains the actual news content of the page, as opposed to site navigation links, ads, layout information, etc. For example, a recent web page from the New York Times consisted of a total of 0,61 bytes, but the actual article text of the web page was only 6,88 bytes. The remaining 60k was extraneous formatting information, navigation links, advertisements, and so on.

I solved this problem by incorporating a new article extraction module that uses machine learning techniques to identify the article text. The new article extraction module parses HTML into blocks of text based on HTML markup and computes a set of features for each text block. 34 features are computed for each text block, based on simple surface characteristics of the text. For example, I use features such as the percentage of text that is punctuation, the number of HTML links in the block, the percentage of question marks, the number of characters in the text block, and so on. While the features are relatively language independent in that they can be

computed for any language, the values they take on for a particular language, or web site, vary.

3.5.2 Using simple document translation for lingual clustering

Once a suitable set of articles can be extracted from the web into text, it is necessary to cluster the articles into topics for use with similarity finder and lingual multi document summarization. The document clustering system that used in Columbia News- Blaster [10] has been trained on, and extensively tested with English. While it can cluster documents in other languages, our goal is to generate clusters with documents from English languages, so a baseline approach is to translate all non-English documents into English, and then cluster the translated documents. I take this approach, and further use different translation methods for clustering and summarization.

Since many documents are clustered, I use simple and fast techniques for glossing the input articles when possible. I have developed simple dictionary lookup glossing systems. While word sense disambiguation is important, my first implementations of glossing systems do not perform word sense disambiguation or other sophisticated disambiguation techniques. Documents that are used in a cluster are later translated with a higher-quality method (currently, an interface to SYSTRAN's system via Altavista's babel) For languages where we do not have a simple translation mechanism available, web interface to the SYSTRAN translation engine. The translated documents are then clustered as in the monolingual English version of NewsBlaster.

3.5. Lingual Clustering Evaluation

I supervised a Russian-bilingual project student, Larry Leftin, who applied my fast glossing translation system to Russian documents. We have performed an evaluation of the lingual clustering component using glossing techniques as discussed in Section 3.5.2 over Russian text by manually examining clusters from a small test data set. The data set is a crawl over news from two Russian news sites (http://www.izvestia.ru/, http://www.mn.ru/), and English news from CNN.com, for a total of 880 articles. After translating the Russian documents with our glossing system and clustering the English and translated Russian documents, 448 clusters are produced. Of those, clusters contained documents in both English and Russian. A hand-examination of the clusters showed that they were all high quality clusters i.e., the topics of the English documents were tightly related to the topics of the Russian translated documents. We also compared to clustering runs using documents with slightly different translation processes (various methods of trying to emphasize proper nouns in the translated Russian and original English text) but these variations on the translation did not perform as well as the original glossing scheme. We have not approached the task of looking at recall of the clustering, since even with this small data set, it would not be practical to examine the entire set by hand. The small number of lingual clusters does not sound unreasonable, since even with English-only runs of Columbia NewsBlaster, only a small number of clusters result from a

large data set (from out of 2,000 - 3,000 input documents, generally only 300 clusters altering requirements.)

Automated lingual article extraction and lingual document clustering is now a functional part of the lingual version of Columbia NewsBlaster. In the next section, I will detail similarity finder performance over Japanese training data collected from the web.

3.6 similarity finder Conclusion

In this chapter I presented similarity finder, a system for computing text similarity between text units (sentences, in this case) in languages. Similarity finder has been implemented to work with English. For English and, different translation mechanisms, feature sets, and feature merging models were explored, with the best performing combination yielding precision of 86% and recall of 50% at a threshold of 0. over the training data. Continuing with the work first started in the English version of similarity finder, computes overall similarity on the basis of multiple feature values defined over linguistically motivated primitive types instead of just a single function of shared terms. Similarity finder makes it easy to add new primitives for different languages, and allows for run-time definition of the set of features to use for similarity computation. The ease with which new primitives and features can be added allows for easy experimentation with features for similarity across languages. Existing natural language processing resources can easily be integrated into similarity finder, as shown by the integration of the English version of BBN's Identity Finder for a named entity primitive in similarity finder.

Similarity finder uses translation at the level of the primitives to for cross-lingual similarity computation. Performing translation at this level means that a full machine translation system for a language pair is not required. For languages that do not have a large amount of available tools available, similarity finder can be used in conjunction with a simple token based primitive extractor and a translation lexicon learned from a parallel corpus and still generate high precision output. Similarity finder is easily ported to other languages, and a strong implementation has been developed for English. Similarity finder to find similar sentences in English text, and compares to performance using similarity finder with machine translated text. Presents two summarization systems that use text similarity in novel applications of lingual multi-document summarization.

CHAPTER 4

SUMMARIZATION VIA SIMILARITY

With the large amount of text available on the web, summarization has become an important tool for managing information overload. While multi-document summarization of English text has become more common, less attention has been paid to producing English summaries of foreign language text. Yet, use of foreign language on the web is growing rapidly [10], and with growing globalization many news events are covered by many countries. In the face of the language diversity available on the web, it is more important to investigate techniques that can provide a summary of documents that end-users are not able to read. As much of the news that is internationally reported is also available in English, making use of the English documents for summarization in a lingual environment has become possible.

I have implemented two summarization systems:

- ✓ A system that builds a summary of the foreign language text, and replaces sentence in the summary with a similar sentence from the English text when possible.
- ✓ a system that uses sentence similarity to cluster all sentences, identifying sentence topics that only occur in one language or the other, and those which are present in both document sources.

The first system uses similar English text to improve the readability and comprehensibility of a summary primarily over the foreign language documents, while the second system indicates similarities and differences in the content between the foreign language and English text. The first system is tested using machine translated text, and English similarity finder to compute similarity. The second system is tested using machine translated text, and over translated text with similarity computed by similarity finder. Section 5.3 focuses on using text similarity to replace machine translated sentences with similar sentences from English text, while section 5.4 presents a system that uses sentence similarity to cluster sentences and present the information that differs between the English and foreign language documents.

4.1 Related work in sentence similar text document summarization

Previous work in lingual document summarization, such as the SUMMARIST system [15] extracts sentences from documents in a variety of languages, and translates the resulting summary. Chen and Lin [4] describe a system that combines multiple monolingual news clustering components, a lingual news clustering component, and news summarization component. Their system clusters news in each language into topics, then the lingual clustering component relates the clusters that are similar across languages. A summary is generated for each language based on scores from counts of terms from both languages. The system has been implemented for English, and an evaluation over six topics is presented. Our system differs by explicitly generating a summary in English using selection criteria from the non-English text.

Other work that use similarity-based approaches to summarization, such as the MEAD document summarization system [29] are related in their use of similarity to guide selection, but our work is original in using text originally from one language to guide selection exclusively on English text. The General summarization system [3] uses text similarity to identify \themes" in a document, and then builds a summary sentence from a theme by combining information from the similar sentences. Our application of text similarity is to improve grammaticality and comprehensibility by selecting similar content from English text, not to use similarity to identify important content, or merge information from similar sentences.

4.2 Summarizing Machine Translated text with Relevant English Text

In this section I present a lingual document summarizer that takes as input a set of multiple documents on a particular topic, some of which are English, and some of which are machine translations of documents into English. The summarizer produces an

Machine translated sentence: particular seven organizations Egyptian Organization for human rights today , Monday appealed to the Egyptian President Hosni Mubarak a cost-accounting the responsible for acts of torture which aimed villagers in upper Egypt during the investigation in the crimes killed in last August .

Similar English sentence: Seven Egyptian human rights groups appealed Sunday to President Hosni Mubarak to ensure that police officers they accuse of torturing hundreds of Christian Copts be brought to justice.

Figure 3.2: A system suggested replacement sentence for a machine translated English sentence

English summary of the foreign language documents. One of the problems with extracting sentences from the machine translated text directly is that they can be ungrammatical and difficult to understand. Moreover, removing context makes the resulting summary hard to comprehend. Figure 3.2 shows an example of an English sentence translated by IBM's statistical MT system from the DUC2004 corpus, and the English sentence that our system suggests as a replacement. I introduce a new method to summarize machine translated documents using text similarity to related English documents. The summary is built by identifying the sentences to extract from the translated text, and replacing the machine translated sentences from the summary with similar sentences from the related English text when a good replacement can be found. The idea is to match content in the non-English documents with content in the English documents, improving the grammaticality and comprehensibility of the text by using similar English sentences.

I present different models for summarization using replacement and show their effectiveness in improving summarization quality. In addition to different metrics and thresholds for similarity, I investigate the utility of syntactic sentence simplification on the replacement English text, and sentence chunking on the machine translated English text. I performed manual evaluation of whether replacements of machine translated sentences by similar English sentences improve a summary on a sentence-by-sentence basis, as well as an evaluation of a similarity-based summarization system using the automatic ROUGE [19] summary evaluation metric. I show that 68% of sentence replacements improve the resulting sum Mary, and that our similarity-based system outperforms a state-of-the-art document summarization system and first-sentence extraction baseline.

4.2.1 Summarization Approach

Our approach relies on first translating the input documents into English and then using similarity at the sentence level to identify similar sentences from the English documents. As long as the documents are on the same topic, this similarity based approach to lingual summarization is applicable. This thesis does not address the issue of obtaining on-topic document clusters; news clustering systems such as Google, or News demonstrate that this is feasible. The system architecture is:

1.Syntactically simplify sentences from related English documents, and possible chunk machine translated English sentences.

2. Produce a summary of the machine translated sentences using an existing sentence Extraction summarization system.

3. Compute similarity between the summary sentences and sentences from similar English documents.

4. Replace English sentences from summary with English sentences for those pairs with similarity over an empirically determined threshold.

Since the focus of this work is not extraction-based summarization, we used an existing state-of-the-art document summarization system, DEMS [35], to select the sentences for the similarity computation process.

4.2.1.1 Sentence Simplification

Since it is difficult to find sentences in the related English documents containing exactly the same information as the translated sentences, I hypothesize that it may be more effective to perform similarity computation at a clause or phrase level. I ran the English text through sentence simplification software [37] to reduce the English sentence length and complexity in the hope that each simplified sentence would express a single concept. The sentence simplification software breaks a long sentence into two separate sentences by removing embedded relative clauses from a sentence, and making a new sentence of the removed embedded relative clause. This allows a more grained matching between the English sentences, without including additional information from long, complex sentences that is not expressed in the English sentence.

For example, for the following English sentence-

1. Had decided Iraq last Saturday halt to deal with the United Nations Special Commission responsible disarmament Iraqi weapons of mass destruction. One similar English sentence found is:
2. Earlier, in Oman, Sultan Qaboos reportedly told Cohen that he opposed any unilateral U.S. strike against Iraq, which ended its cooperation with U.N. inspectors on Saturday.
That sentence simplifies to the following two sentences:

2a. Earlier, in Oman, Sultan Qaboos reportedly told Cohen that he opposed any unilateral U.S. strike against Iraq.
2b. Iraq ended its cooperation with U.N. inspectors on Saturday.

Using sentence simplification to break down the text allows us to match sentence 2b, without including 2a, which was not reported in the English sentence? I examined using two types of sentence simplification, syntactic and syntactic with pro- noun resolution, and compared them to not using any sort of simplification. To limit the number of systems evaluated in the manual evaluation, I determined settings to use based on results from automated summary evaluation. In all of our experiments, syntactic simplification performed about 3% better on ROUGE scores than simplification with pronoun resolution, or not performing any simplification. Simplification with pronoun resolution did not always beat un simplified text, possibly due to errors introduced by the pronoun resolution, which has a success rate of approximately 0%. I present results of the system using only syntactic simplification. Similarly, I performed experiments for splitting the machine translated text. Investigated two methods for splitting English text: one tags the text with TTT4 and splits on verb groups, copying the previous noun group and verb group to the start of the next sentence. The other splits on verb groups and \and",

\nor", \but", \yet" and \,", without performing the copying. In both cases, sentences with less than 3 tokens are altered from the output. The copying method was approximately 3% better on the manual evaluation below, so I omit results from the other chunking method.

4.2.1.2 Similarity Computation

Text similarity between the translated and relevant text is calculated using similarity finder [14]. Similarity finder is a tool for clustering text based on similarity computed over a variety of lexical and syntactic features. The features used in similarity finder are the overlap of word stems, nouns, adjectives, verbs, word Net [22] classes, noun phrase heads, and proper nouns. Each feature is computed as the number of items in common between the two sentences normalized by the sentence length. The final similarity value is assigned via a log-linear regression model that combines each of the features using values learned from a corpus of news text manually labeled for similarity. No modifications were made to similarity finder to compensate for using machine translated text as input, although the machine translated text is quite different from the news text used to train similarity finder.

4.2.1.3 System Implementation

Our summarization system can be run in multiple configurations.

1. Use DEMS to select English sentences, retain only sentences that have similar English sentences, replacing them with the single most similar English sentence. If the summary is too short (less than 600 bytes,) delete it, and build a new summary using all English sentences, sorted by similarity to English sentences, and replacing each one by the single most similar English sentence.

2. Use DEMS to select English sentences, replace only sentences above empirically determined threshold of 0.6 passing a cosine alter with similar English sentences, and retain non-replaced English sentences in the summary.

3. Use all English sentences, sort by decreasing similarity to English sentences, and replace each one by all English sentences above an empirically determined threshold of 0.6 that pass a cosine alter. Machine translated sentences are kept if they do not pass the threshold.

Configuration 1 uses DEMS to select sentences, and maximizes the number of replacements made by re-running without DEMS if not enough similar sentences are found to make a large enough summary. Configuration 2 also uses DEMS to select sentences, but retains any machine translated sentences for which no suitable sentence replacements are found. Configuration 3 focuses on maximizing replacements by not using DEMS for selection, and builds a summary by taking the most similar English sentences, using only similarity to English sentences to guide selection, removing any manually-constructed \intelligent" system from the selection task. All summaries are limited to 665 bytes since that was the size threshold that

was used for the DUC evaluation. An evaluation of the different configurations of the system using ROUGE scores is presented.

4.3.2 Evaluation

I performed evaluation at two levels: the sentence level to test the proposed sentence replacements of English sentences from similar English sentences, and the summary level to evaluate quality of the full summaries that include these sentence replacements. At the summary level, I used the automated system, ROUGE, for evaluation. It allowed us to make rough distinctions between different models for constructing the full summary. However, this would not tell us whether a particular English sentence was a good replacement for a translated one and thus, I used a more time-consuming, manual evaluation to quantify how well replacement worked.

4.3.2.1 Summary level evaluation

I evaluated the similarity-based summarization system using ROUGE, 5 a system for summary evaluation that compares system output to multiple reference summaries. I include results from two baseline systems: a first-sentence system, and runs of the DEMS system without replacement.

The first-sentence summarization baseline takes the first-sentence from each document in the set until the maximum of 665 bytes is reached. If the first-sentence was already included from each document in the set, the second sentence from each document is included in the summary, and so on. Two baseline summaries were generated; one for the relevant English documents only, and one for the IBM translated documents alone. The IBM translation baselines give us an idea of scores for summaries drawn from the same content as the reference summaries, while the relevant English baselines tell us how well summaries generated without any knowledge from the English text score. Our similarity-based system was run with simplified English sentences and full machine translated English sentences.

4.3.2.2 Summary level evaluation results

Table 3.3 lists the results using the ROUGE-L evaluation metric along with the results of the four baseline runs. The ROUGE-L score is a longest common substring score from the ROUGE system, which rates summaries based on n-gram overlap between the system summary and multiple reference summaries. Evaluations with ROUGE in the past have demonstrated that the score often fails to show statistical significance between scores for evaluated systems. In DUC04 on the lingual system task, the 95% confidence interval

Similarity System	ROUGE-L
System Config 1	0.25441
System Config 2	0.19348
System Config 3	0.21936
1st Sentence Baseline	
Related English	0.23973
IBM translations	0.22118
DEMS Baseline	
Related English	0.16197
IBM translation	0.21966

Table 3.3: Summary evaluation results.

split the 11 participating systems into two main groups; the bottom group containing three systems and the top group containing everyone else. One could argue for a third group containing the top system only, which was statistically significantly better than the bottom six systems when taking the 95% confidence interval into effect. It is not a surprise, then, that the results for the three versions of our system and the baselines also fall within the 95% confidence interval. As the only automated method for summarization, ROUGE is often, nonetheless, used to roughly rank different approaches. Even if the similarity-based systems do not beat the baselines by statistically significant margins, replacing the machine translated text with English text does improve the readability of the summary. The similarity-based summarization system in configuration 1 performs better than all the baselines, whether over the related English text, or the IBM machine translated text. By outperforming the first sentence baseline and DEMS on the machine translated text, I infer that the similarity system is able to choose sentences from the related English text that are relevant to the content summarized by the humans who read the manual translations of the English text. In contrast, simply running first sentence extraction and DEMS on the related English text does not perform as well; using the machine translated text to guide selection of related English sentences gives an improvement in performance over the related English baselines. The similarity-based system even outperforms DEMS when run over the manual translations.

Of the three system configurations, the first performs the best. In this evaluation, this configuration builds a summary using all English sentences and replaces them with the most similar English sentence because DEMS selection resulted in too few sentences. Using DEMS for selection in configuration 2 resulted in summaries containing mostly machine translated text, since few sentences pass the required threshold level and alters, but did not perform as well as the DEMS baseline since sentences were sorted by similarity, resulting in different sentences in the truncated

summary. Configuration 3 also contained some machine translated sentences, and did not perform as well as configuration 1, which only contained English text.

In Section 5.3, I presented a summarization system that summarizes machine translated text using the English sentences to guide selection of English sentences from a set of related articles. Syntactic sentence simplification on the related English text improves overall summarizer performance, and a hand evaluation of the sentence replacements show that 68% of the replacements improve the summary.

The results from the ROUGE metric show that the similarity based summarization approach outperform DEMS and the first-sentence extraction baseline. It is interesting that a state-of-the-art summarization system run over the relevant English articles performs worse than the similarity-based summarization systems run over the same data. This clearly demonstrates that the similarity-based selection system driven by the machine translations is able to select the good sentences from the relevant text.

In the process of performing our manual evaluation, often there was different content in the English texts, and finding similar content for some subset of the sentences was just not possible. This leads us to believe that a more useful approach to summarization for data of this kind is to separate out what is similar between the two document sources, and what is unique to each document source. Thus, I expand on the idea of summarizing two different sets of documents by looking at not just what is similar between them, but also what is different. Instead of just using the similarity values as is done here, I cluster the sentences, and identify sentence clusters that contain information exclusive to the English documents, information exclusive to the English documents, and information that is similar between the two. The clusters with similar sentences can be summarized using the approach in this thesis. For the other clusters, I am working on an approach to generate indicative summaries that point out the differences. Given that summaries that point out both similarities and differences are quite different from the model summaries currently used in DUC, I also develop strategies to evaluate these summaries, presented in Section 5.4.2.1.

4.4 Summarization that indicates similarities and differences in content

While Section 5.3 focused on using text similarity to find similar sentences and replacing them when possible, in many cases the documents from the two languages contain different information. The second model of summarization uses text similarity and sentence clustering to indicate both similar content between the two sources and content that differs between two sources of text in different sentence. In this case, our system works with English text.

In this section I introduce a second summarization system, CAPS (Compare And contrast Program for Summarization) that uses text similarity on the input text documents to generate clusters of sentences across languages that are similar to each

other, and identifies the source language documents from which the clusters draw their evidence. A summary produced by CAPS identifies facts that English and English sources agree on as well as explicit differences between the sources. Its three-part summary of an event identifies information reported in English sources alone, information reported by sources only, and information that appeared in both English and English sources. As with the work presented in Section 5.3, English text is first syntactically simplified, so CAPS can identify similarities and differences below the sentence level.

In addition to using similarity metrics to identify agreements and differences among articles, it also uses similarity to improve the quality of the summary from mixed sources over plain extraction systems by selecting English phrases to replace error full English translations. In the following sections I first describe the CAPS architecture, then present the similarity metrics that I use for clustering and for selection of phrases for the summary. Finally, I present an evaluation of our method which quantities both how well CAPS identifies content unique to or shared between different sources, and how well CAPS summaries capture important information. Our evaluation features the use of an automatic scoring mechanism that computes agreement in content units between pyramid representations [27] of the articles, separated by source. As before, I use English documents from the DUC 2004 lingual corpus [28] for the experiments here.

4.4.1 System Architecture

The input to the CAPS Summarizer is two sets of documents on an event. The input to CAPS can be:

- ✓ a set of un translated documents with a set of English documents, or
- ✓ A set of manual or machine translations of documents with a set of English documents.

When using English documents CAPS uses similarity finder to compute text similarity, or the English version of similarity finder when using manual or machine translations of the documents to compute the text similarity measure. As with replacement-based summarization, either syntactic sentence simplification software can be used to simplify the English text, or the original un simplified English text can be used. Figure 3.3 shows the CAPS system architecture, with 8 main phases: text simplification, similarity computation, clustering, cluster pruning, cluster language identification, cluster ranking, representative sentence selection, and summary generation. CAPS determine similarities and differences across sources by computing a similarity metric between each pair of simplified sentences. Clustering by this metric allows the identification of all sentence fragments that say roughly the same thing. As shown in Figure 3.3, CAPS first simplifies the input English sentences. It does not simplify the translated sentences because these sentences are often ungrammatical and it is difficult to break them into meaningful chunks. CAPS

then computes similarity between each pair of simplified sentences and cluster all sentences based on the resulting values.

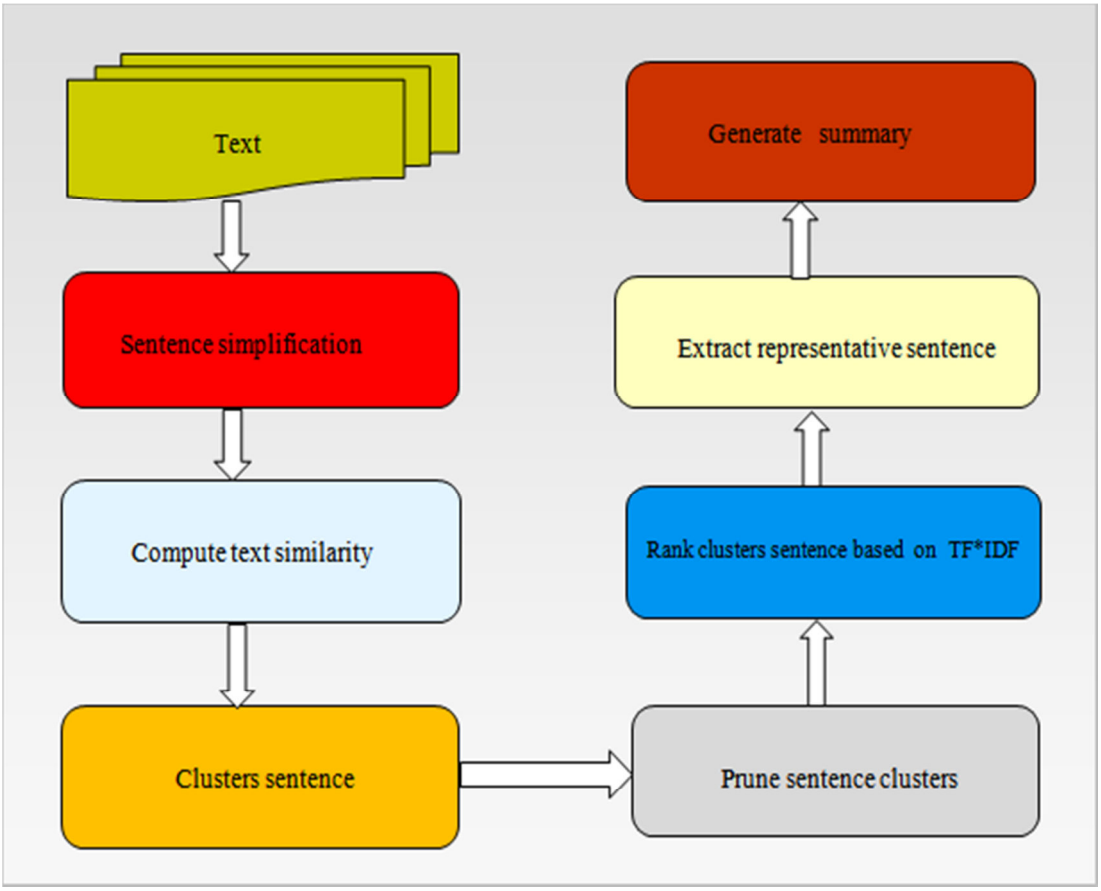


Figure 3.3: CAPS System Architecture

Next, sentence clusters are partitioned by source, resulting in multiple clusters of similar sentences from English sources, multiple clusters of sentences from English sources, and multiple clusters of sentences from English sources. Finally, I rank the sentences in each source partition using a TF*IDF score [32]. The ranking determines which clusters contribute to the summary (clusters below a threshold are not included) as well as the ordering of sentences. For each cluster, we extract are representative sentence (note that this may be only a portion of an input sentence) to form the summary. In this section, I describe each of these stages in more detail.

4.4.1.1 Sentence Simplification to Improve Clustering

As with the summarization system presented in Section 3.3, it is possible to performing syntactic sentence simplification on the input English text. I have previously performed experiments using both perform syntactic simplification and

not using simplification on the input English text, and show the results in Section 5.3.2.1. I opt to use syntactic sentence simplification with this system as well because it allows one to measure similarity at grain than would otherwise be possible. I use a sentence simplification system developed at Cambridge University [37] for the task. The generated summary often includes only a portion of the simplified sentence, thus saving space and improving accuracy. I use syntactic sentence simplification only instead of using syntactic simplification with pronoun resolution. The pronoun resolution phase included in the software sometimes makes anaphoric reference resolution errors, resulting in incorrect re-wordings of the text.

4.4.1.2 Text Similarity Computation

Text similarity between English sentences is computed using similarity finder, a program I developed which uses simple feature identification and translation at word and phrase levels to generate similarity scores between sentences across and within languages. Section 3.4 details the English version of similarity finder used in this work. Text similarity between manual and machine translated English documents and English is computed with similarity finder, an English-specific program for text similarity computation that similarity finder was modeled after. Similarity finder for English is presented in Chapter 2. In addition, I present a third baseline approach using the cosine distance for text similarity computation.

4.4.1.3 Sentence clustering and pruning

Sentence clustering uses the same clustering component described in Chapters 2 and 3. Each cluster represents a fact which can be added to the summary; each sentence in the generated summary corresponds to a single cluster. Since every sentence must be included in some cluster, individual clusters often contain some sentences that are not highly similar to others in the cluster. To ensure that our clusters contain sentences that are truly similar, I implemented a cluster pruning stage that removes sentences that are not very similar to other sentences in the cluster. I implemented the same cluster pruning algorithm described in [31]. This pruning step ensures that all sentences in a sentence cluster are similar to every other sentence in the cluster with a similarity above a given similarity threshold. I illustrate the procedure with the following example. For the cluster with these initial sentences:

P13 Sana'a 12-29 (AFP) - A Yemeni security official reported that Yemeni security forces killed three of the Western hostages who were held in Yemen, two Brits and an American, and managed to free 13 others when they attacked the place where they were detained. P36 London 12-29 (AFP) - British Foreign Secretary Robin Cook announced this evening, Tuesday, that the four Western hostages who were killed today in Yemen are three Brits and one Australian. P41 London 12-29 (AFP) - British Prime Minister Tony Blair announced today, Tuesday, that he was "shocked and hurried" about the killing of four Western hostages in Yemen, including at least three Brits. P51 London 12-30 (AFP) - One of the surviving hostages in Yemen, David Holmes, announced in a telephone call conducted with him from London by Agence France Presse that the hostages who died Tuesday in Yemen at the hands of

their kidnappers were killed during the attack by policemen, and not before the attack as Yemeni police asserted. P52 Holmes (64 years), who is still in Aden, regarded "that all reports that said that the criminals attacked the hostages (before the raid by security forces) do not agree with the developments of events. When the criminals found themselves threatened and realized that they may be defeated, they wanted to kill the hostages."P53 Aden's Security Chief, Brigadier General Mohammad Saleh Tareeq, had announced today, Wednesday, in the presence of some survivors who refused to speak to the press that "the hostage rescue operation started after the gang began killing the hostages, whereas they first killed three of the British hostages, which then forced the security forces to storm their location to prevent more bloodshed, and was consequently able to free the rest of the hostages." P58 In Yemen, three hostages were killed. P62 Authorities say it was the first time hostages had been killed in Yemen.

Based on the similarity values between the sentences in the cluster, those sentences that have values lower than the threshold are removed. In this example, sentences P51, P52, and P62 need to be removed. The final cluster is then:

P13 Sana'a 12-29 (AFP) - A Yemeni security official reported that Yemeni security forces killed three of the Western hostages who were held in Yemen, two Brits and an American, and managed to free 13 others when they attacked the place where they were detained. P36 London 12-29 (AFP) - British Foreign Secretary Robin Cook announced this evening, Tuesday, that the four Western hostages who were killed today in Yemen are three Brits and one Australian. P41 London 12-29 (AFP) - British Prime Minister Tony Blair announced today, Tuesday, that he was "shocked and hurried" about the killing of four Western hostages in Yemen, including at least three Brits. P53 Aden's Security Chief, Brigadier General Mohammad SalehTareeq, had announced today, Wednesday, in the presence of some survivors who refused to speak to the press that "the hostage rescue operation started after the gang began killing the hostages, whereas they first killed three of the British hostages, which then forced the security forces to storm their location to prevent more bloodshed, and was consequently able to free the rest of the hostages." P58 In Yemen, three hostages were killed.

The resulting cluster contains sentences that are much more similar to each other, which is important for my summarization strategy since I select a representative sentence from each cluster that is included in the summary. I do not want to make sentences that are not representative of the cluster available for inclusion in the summary.

4.4.1.4 Identifying cluster similar sentence

The final summary that I generate is in three parts:

- ✓ sentences available only in the similar text documents
- ✓ sentences available only in the English documents
- ✓ sentences available in both the similar text and English documents

After producing the sentence clusters, I partition them according to the sentence of the sentences in the cluster: English only. This ordering is important because it allows us to identify similar concepts across languages, and then partition them into concepts that are different: those that are unique to the documents, and the English documents, and concepts that are supported by English documents.

Note that these clusters are not known before-hand and are data driven, coming from the text similarity values directly.

4.4.1.5 Ranking clusters

Once the clusters are partitioned by language, CAPS must determine which clusters are most important and should be included in the summary. Typically, there will be many more clusters than cannot in a single summary; average input data set size is 263 words, with an average of 4050 words in clusters, and I am testing with 800 word summaries, 10% of the original text. In the default configuration, CAPS uses TF*IDF to rank the clusters; those clusters that contain words that are most unique to the current set of input documents are likely to present new, important information. For each of the three types of sentence clusters, English, the clusters are ranked according to a TF*IDF score [32].

The TF*IDF score for a cluster is the sum of all the term frequencies in the sentences in the cluster multiplied by the inverse document frequency of the terms to discount frequently occurring terms, normalized by the number of terms in the cluster. The inverse document frequencies are computed from a large corpus of AP and Reuter's news. CAPS have two other measures for ranking clusters: the number of unique sentences in each cluster, and the number of unique sentences in a cluster weighted by the TF*IDF score of the cluster. Experimentation over a single test document set showed that the TF*IDF score performed best of the three, and results from this thesis use that cluster ranking method.

When using A text in the input and text similarity computation phases, the text is translated into English after the clustering phase. TF*IDF counts are computed over the machine translated text. This is done because the ranking of clusters has to be done over, English, and mixed clusters, which presents a problem: how to rank the mixed clusters? For only clusters, a TF*IDF approach using IDF values from a large English corpus could be used, but it is unclear if direct application of TF*IDF to clusters with both languages and different IDF values for each languages would be applicable. As the English sentences need to be translated for presentation in an English summary anyway, and many of the sentences have been dropped through the clustering and pruning process, machine translation is performed at this step, and clusters are ranked with the machine translated versions of the sentences.

4.4.1.6 Sentence selection

The cluster ranking phase determines the order in which clusters should be included in the summary. Each cluster contains several (possibly simplified) sentences, but only one of these is selected to represent the cluster in the summary.

There are three methods implemented to select a specific sentence to represent the cluster:

1. The sentence most similar to all other sentences based on the computed similarity values.
2. A TF*IDF based ranking method that selects a sentence with the highest TF*IDF score.
3. A method that constructs a \centroid" sentence in a vector space model, and selects the most similar sentence to the centroid To compute a TF*IDF score for clusters with text in multiple languages, one must have a (preferably large) corpus to derive IDF values for terms in the respective languages. Experimentation over a test set showed that the first method performed best, so that is the method used in these experiments.

Only the set of unique sentences are evaluated for each cluster. In this sort of task, many of the input documents repeat text verbatim, as the documents are based on the same newswire (Associated Press, Reuters, etc.) report, or are updated versions of an earlier report. In order to avoid giving undue weight to a sentence that is repeated multiple times in a cluster, the unique sentences in each cluster are first identified. Unique sentences are identified using a simple hash function, removing leading and trailing white space.

Similarity based selection:

To select a sentence based on the text similarity values, first the set of unique sentences is determined as described above. For each unique sentence in the cluster, its average similarity to every other unique sentence in the cluster is computed. The unique sentence with the highest average similarity is then chosen to represent the cluster.

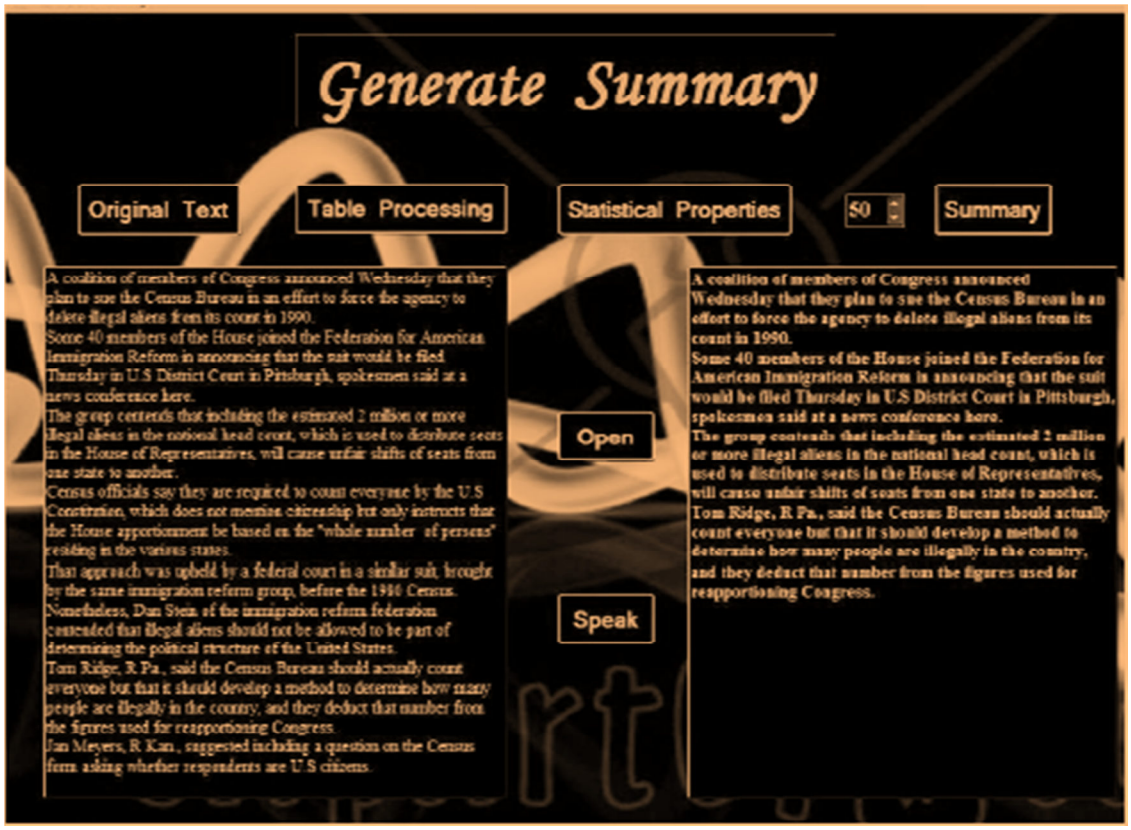
TF*IDF based selection:

Starting with the set of unique sentences, each sentence is scored using the same TF*IDF measure used for cluster ranking (see Section 5.4.1.5.) The frequency of each term in the sentence is computed, multiplied by the inverse document frequency for the term, and the score for the sentence is normalized by sentence length. The unique sentence with the highest TF*IDF score is selected to represent the cluster.

Centroid sentence selection:

The centroid measure for sentence selection first builds a simple vector-space model for all the unique sentences, and a model for the centroid sentence. The centroid sentence model is built by adding in the terms from all of the unique sentences in the cluster. The cosine distance between each unique sentence and the centroid sentence is computed, and the closest unique sentence is chosen to represent the cluster.

4.4.1. Summary generation



3.4 Summary generation

Once the clusters are ranked and a sentence has been selected to represent each cluster, the main remaining issue is how many sentences to select for each partition in English. There are two parameters that control summary generation: total summary word limit, and the number of sentences for each of the three partitions. The system takes sentences in proportions equal to the relative partition sizes. For example, if CAPS generates similar clusters, 24 English clusters, , then the ratio of sentences from each partition is 4 English. The smallest partition size is divided through the 3 partitions to determine the ratio. The total word count is divided among partitions using this ratio.

There are some measures which quantify the quality of summaries produced. It is classified into two types.

- ✓ Intrinsic evaluation is a method which measures the quality of the summary as output.
- ✓ Extrinsic evaluation is a method which measures the quality of output summary in the form of its assistance in another task.

Implementation

A network in general represents concepts as nodes and links between concepts as relations with weights indicating strength of the relations. The hidden or latent structure underlying raw data, a fully connected network, can be uncovered by preserving only critical links. The aim of a scaling algorithm is to prune a dense network in order to reveal the latent structure underlying the data which is not visible in the raw data. Such scaling is obtained by generating an induced sub graph. There are two link-reduction approaches: threshold-based and topology-based. In threshold-based approach elimination of a link is solely decided depending upon whether its weight exceeds some threshold. On the other hand, a topology-based approach eliminates a link considering topological properties of the network. Therefore a topology-based approach preserves intrinsic network properties reliably. We have used a threshold based approach with a threshold of 0.04 to discard branches among nodes that similarity less than 0.04.

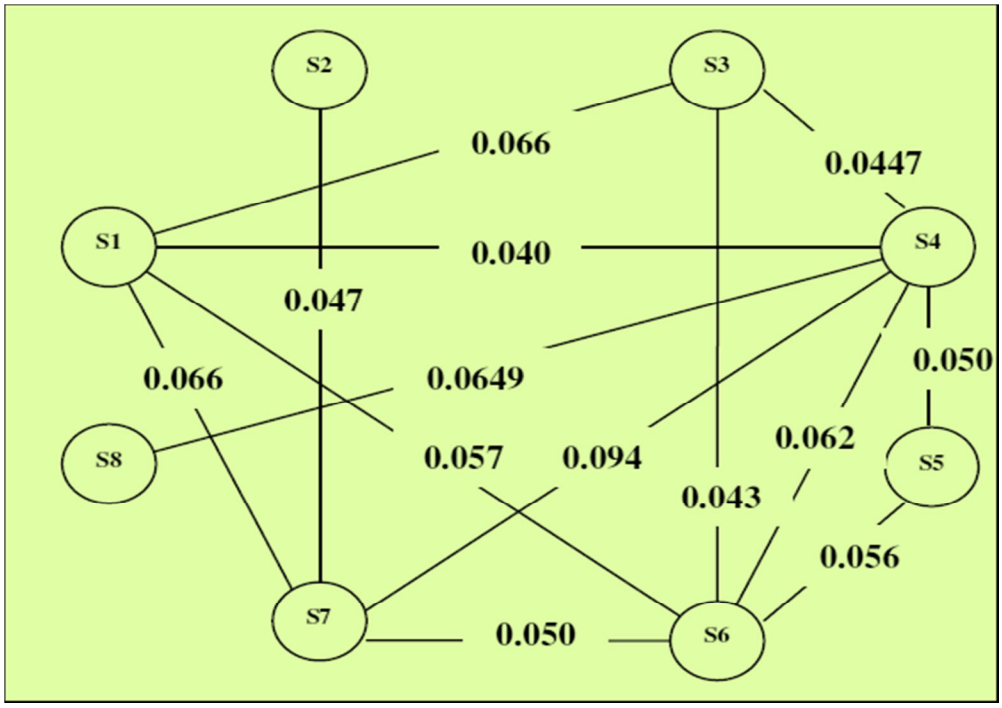


Figure 3.5: Scaled network graph with threshold of 0.04.

Feature Score and rank of the all sentences

All the sentences are ranked by calculating various feature score for all sentences and according to the compression rate they selected for inclusion in summary in descending order of their rank in the order of their appearance.

Sentnce No / Feature Score	1	2	3	4	5	6	7	8
Sentence Position	0.710	0.650	0.570	0.000	1.000	0.000	0.000	0.000
Positive Keyword	0.370	0.770	0.150	0.230	0.130	0.530	0.510	1.000
Sentence Relative Length	1.000	0.870	0.750	0.620	0.500	0.370	0.250	0.120
Proper Noun	0.780	0.850	0.970	0.950	0.560	0.680	1.000	0.430
Numerical Term	0.830	0.750	0.660	0.660	0.580	0.750	1.000	0.500
Term Weight	0.840	0.660	0.790	0.700	0.460	0.600	1.000	0.730
Similarity with others	0.714	0.889	1.000	0.898	0.618	0.647	0.994	0.476
Busy Path	0.667	0.167	0.500	1.000	0.333	0.833	0.667	0.167
Final Score	5.911	5.606	5.390	5.058	4.181	4.410	5.421	3.422
Rank	1	2	4	5	7	6	3	8

Table no 3.4 Feature Score and rank of the all sentences

4.4.2 Evaluation

The most common method to date for evaluating summaries is to compare automatically generated summaries against model summaries written by humans for the same input set Using different methods of comparison (e.g., [19], [28], [30]). Since there is no Corpus of model summaries that contrast differences between sources; I developed a new evaluation methodology that could answer two questions:

- ✓ Does the approach partition the information correctly? That is, are the facts identified for inclusion in the similar partition actually unique to only the documents? If our similarity matching is incorrect, it may miss a match of facts across language sources.
- ✓ Does the 3-part summary contain important information that should be included, regardless of source?

I use Summary Content Units (SCUs) [27] to characterize the content of the documents and the Pyramid method to make comparisons. The evaluation features four main parts: manual annotation of all input documents and the model summaries used in DUC to identify the content units, automatic construction of four pyramids of SCUs from the annotation (one for English, and mixed language SCUs and one for the entire document set regardless of language), comparison of the three partitions of system identified clusters against the source specific pyramids to answer question 1 above, and comparison of the facts in the 3-part summary against the full pyramid to answer question 2.

4.4.2.1 SCU Annotation

In the summarization experiments, I needed to come up with an evaluation methodology that can take into account summaries that indicate differences in information content between documents from different sentence. To do this, I first need to characterize the content of the documents in English, and determine what information is contained in both document sets, and what is exclusive to one set or the other. I have chosen to use Summary Content Units (SCUs) [27] to characterize the content of the documents, and evaluate the summaries output by the system.

The goal of SCU annotation is to identify sub-sentential content units that exist in the input documents. These SCUs are the facts that will serve as the basis for all comparisons. The SCU annotation aims at highlighting information the documents agree on. An SCU consists of a label and contributors. The label is a concise English sentence that states the semantic meaning of the content unit. The contributors are snippet(s) of text coming from the summaries that show the wording used in a specific summary to express the label. It is possible for an SCU to have a single contributor, in the case when only one of the analyzed summaries expresses the label of the SCU.

4.4.2.2 Characterizing English content by SCUs

This section deals with how the content differs from the English documents in the sets. Appendix A details the annotation process applied to the DUC sets. The and 10 English documents, as well as 4 human-written summaries for each set were marked by annotators as described to arrive at one large content pyramid for all 24 \documents" in the set. The large content pyramid was then automatically broken down into language-specific pyramids based on the language of the contributors in each SCU. An SCU that contains only contributors from English documents goes into the English pyramid, one that only has English contributors goes into the English pyramid, and SCUs that contain contributors from English documents are placed in the mixed pyramid.

Pyramid, while the Contributors column lists the total number of contributors for the set. Many SCUs have multiple contributors, with some SCUs having more than 30

contributors for a single SCU. The sets vary in terms of distribution of SCUs between the languages, but in general the English pyramid contains the most SCUs. There are two sets for each of mixed that both contain the largest number of SCUs. In six of the ten sets, the size of the mixed pyramid is greater than the size of the English pyramid. The partitioning of the manually annotated pyramids show that the majority of the time the English language documents more unique information than the English documents, but there is still information that is only reported in the documents and that has support from English documents.

4.4.2.3 Evaluating language partitions with SCUs

Once the SCU pyramids for a document set are created, they can be used to characterize the content of the English documents. The SCU pyramids reveal the information in each document set, and the weights of the SCUs indicate how frequently a particular SCU was mentioned in the documents. In general, more highly weighted SCUs indicate information that should be included in a summary. This section described how I have used the three different language pyramids to evaluate the CAPS summarizer output, both for how well it identified content particular to one language , and how well it chooses important content to include in the summary.

The following example shows how SCUs are weighted based on importance of a concept, and how the SCUs differ by language. This example is from a set about the explosion of a Pam-Am jet over Lockerbie, Scotland, the top three SCUs from the SCU annotation broken down by similar sentence.

Evaluating English clusters is done in the same manner as clusters; I collect the SCUs associated with each of the sentences in the cluster, and compute the SCU score and percentages of SCUs found in the cluster compared against the English-only SCU pyramid. Since the system can be run with syntactically simplified English text, I cannot just determine the SCUs for a sentence by reading the annotation. To determine the SCUs for the sentence, I first identify the longest match between the sentence being evaluated and the originally marked documents, and then read the SCUs for the matching portion. Since all of the sentences that are evaluated are either complete sentences that have been annotated, or simplified portions of marked sentences, this approach worked very well.

4.4.2.4 Importance evaluation

The overall summary content quality is evaluated using the Pyramid method for summary evaluation; the full 3-part summary is scored by comparing its content to the SCU pyramid constructed for all documents in the set as well as the four human model summaries. This pyramid encodes the importance of content units in the entire set; important SCUs will appear at the top of the pyramid and will be assigned a weight that corresponds to the number of times it appears in the input documents and model summaries. The pyramid score is computed by counting each SCU present in the system generated summary, multiplied by the weight of that SCU in the gold

standard pyramid. The intuitive description of a pyramid score is that the summary receives a score ranging 0 to 1, where the score is

$$s = \frac{\text{summary score}}{\text{Max pyramid score for summary}}$$

The score for the summary is simply the sum of the weights of each SCU in the summary. The max pyramid score for the summary is the maximum score one could construct given the scoring pyramid and the number of SCUs in the summary. E.g., for a summary with SCUs, the max score is the sum of the weights of the biggest SCUs.

I developed an automated technique to match summary sentences to the SCUs from the pyramid. For English sentences that have been syntactically simplified, I use a longest common substring matching algorithm to identify the original non-simplified sentence in the annotated data. The SCUs annotated for the simplified section of the sentence are then read from the annotation data. For sentences that have not been simplified, the SCUs can be read directly from the annotation because they are identical.

4.4.3 Results

Most of the summarization systems developed so far is for news articles. There are two major reasons for this: news articles are readily available in electronic format and also huge amount of news articles are produced every day. One interesting aspect of news articles is that they are written in such a way that usually most important parts are at the beginning of the text. So a very simple system that takes the required amount of leading text produces acceptable summaries. But this makes it very hard to develop methods that can produce better summaries.

4.4.3.1 Per-language Partition Evaluation

Table 5.3 shows the percentage of SCUs in each language pyramid that have a match in the representative sentences for the partition. This evaluates how well the similarity metric clusters text for each language, and is essentially the recall of SCUs for each language partition. Table 5.4 lists the SCU Pyramid scores of the three partitions using manually translated, machine translated, and UN translated English documents. This evaluates the importance of the sentences included in the language partition by the clustering algorithm and similarity metric. Note that these evaluations are over the representative sentence of all clusters in each partition, and not just the representative sentences in the summary.

Extractive summary baseline

As a baseline, I examined two approaches to summarizing the just the English portion of the DUC 2004 English {English data, which do not take advantage of the unique aspect of my system to summarize the similarities and differences between the English documents. Using two document selection strategies, I used DEMS [35], a state-of-the-art extractive summarization system to summarize English documents from the data sets. The two document selection strategies are:

1. Select all English documents and summarize.
2. Compute the centroid document of all (translated) input English documents, and select individual English documents with a cosine similarity of 0.0 or greater to the English centroid. If fewer than two documents have a similarity of 0.0 or greater, take the two most similar English documents.

Approach 1 is a baseline that examines how well summarizing all English documents performs, while approach 2 restricts the English documents to those that are similar to the English documents.

In both cases, the non-simplified versions of the English documents, the same versions used to generate the gold-standard testing data, are summarized using DEMS. The resulting summaries are evaluated in the same manner used for the English summaries. The chosen representative sentence might not contain as many SCUs as other sentences in the cluster.

4.4.3.2 Evaluating importance

To evaluate how well CAPS includes important information regardless of language, I score the entire 3-part summary against the merged SCU Pyramid for each document set, and compare to two baseline systems.

The baseline systems I compare to be:

1. Lead sentence extraction
2. Cosine system for similarity component for clustering component

The lead sentence extraction baseline extracts the first sentence from each document until the summary length limit is reached, including the second, third, etc. sentences if there is space. The first sentence baseline is very different from the CAPS system; I was unable to use it in the language-partition evaluation because such a system is not able to identify information that is only represented by one source or the other. It is a common baseline used in document summarization though, and so I compare to it in this part of the evaluation, which is a traditional summarization evaluation.

The cosine baseline uses a cosine metric for text similarity computation instead of Similarity finder in the CAPS framework. Table 3.5 shows average performance of CAPS and baseline systems over 10 different documents sets from the 2004 DUC

data. Since the pyramid sizes are different for different summaries, the average scores are computed as micro averages as before; the average is the total weight of all summary SCUs divided by the total of max SCU scores for each summary.

Run Identifier	Pyramid Score
Manual Translations (CAPS)	0.8571
Manual Translations 1st sent baseline	0.7844
Machine Translations (CAPS)	0.7940
Machine Translations Cosine baseline	0.7158
Machine Translations 1st sent baseline	0.7798
Untranslated (SFML Arabic–English model)	0.7487
Untranslated (SFML mixed model)	0.6360
English-only (all documents)	0.6373
English-only (similar documents)	0.6244

Table 3.5: Average SCU pyramid scores of CAPS and baseline systems of entire summary.

Baseline by including a representative first sentence as well as other sentences from sentence clusters that contain less frequently mentioned SCUs. When using machine translations, scores are predictably lower than using manual translations; however, the CAPS system still performs better than either of the two baselines. The similarity component in CAPS performs much better than a less sophisticated text similarity technique as shown by the cosine baseline run. Interestingly, the CAPS system run over machine translated text even performs better than the first sentence extraction baseline that uses manually translated sentences.

4.4.3.3 Example output

The following is an example of the summary for set d1018t, a set about the kidnap and rescue of western hostages in Yemen. This example is taken from a run using manually translated and syntactically simplified English, as those runs contain the most understandable, making it easier to see the differences in content between the English sources. The summary is an 805 word summary, 54% English, and 1% mixed. The original documents total 4,350 words, so the summary is about 18% of the original document size. On average, the 800 word summaries used for these sets

are 10% of the size of the original document set, but d1018t is smaller than the average document set.

4.4.4 Conclusions

I have presented a system for generating English summaries of a set of documents on the same event, where the documents are drawn from English sources. Unlike previous summarization systems, CAPS explicitly identifies agreements and differences between English sources. It uses sentence simplification and similarity scores to identify when the same facts are presented in two different sentences, and clustering to group together all sentences that report the same facts. I presented an evaluation methodology to measure accuracy of CAPS partitioning of similar facts by language and to score the importance of the 3-part summary content. The evaluation shows that our similarity metric outperforms a baseline metric for identifying clusters based on language, and performs almost as well using machine translated text as manual translations for identifying important content exclusive to English clusters. The CAPS summarization system outperforms cosine and first sentence baselines using machine translated text, and almost performs as well as a first sentence baseline using manually translated text.

Using Similarity finder, CAPS is able to use non-translated text as input, deferring translation until after sentences have been clustered, reducing the number of sentences that need to be translated. Using Similarity finder and translated input, CAPS out-performs all other methods for identifying information that is supported by English sources, a 0.826 micro-averaged SCU pyramid score, compared to the next best 0.641 using manually translated documents.

Similarity finder can be quickly ported to work with other language pairs, using a learned probabilistic dictionary and feature merging model from a parallel corpus. This quick portability using only a parallel corpus allows for quickly building a lingual summarization system based on CAPS with Similarity finder for a language that does not have a large infrastructure of natural language processing tools built up. While this version of CAPS does use machine translation to present the English sentences to the user in English, presenting the original language sentences to a bilingual analyst is possible, CAPS is able to reduce a large number of sentences from multiple documents down to a much smaller number of sentences that would be manageable for human translators to translate.

CHAPTER 5

CONCLUSION

This thesis presents my work in lingual text similarity, and its application to document text summarization. In this chapter, I will present my contributions to the field, limitations with the work described here, and future work.

5.1 Contributions

This thesis presents many contributions both in the field of summarization, and lingual text similarity computation. These contributions include:

- ✓ Development of flexible framework for experimenting with lingual text similarity in Similarity finder.
- ✓ Linguistically motivated primitives that are computed on a per-language basis.
- ✓ Support for computing similarity to languages with few natural language processing resources available by using learned bilingual translation lexicons.
- ✓ A summarization approach implemented in the CAPS system that identifies both Similarities and differences between documents in differ below the sentence level.
- ✓ CAPS summarization system is applicable to any language pair for which machine translation systems is available, or a lingual text similarity metric can be computed.
- ✓ Information that is supported by both languages is made easier to understand in the summary by selecting English sentences instead of machine translated English sentences for the summary.
- ✓ CAPS approach is applicable even without machine translation systems available to summarize English documents for lingual analysts.

5.1.1 Linguistically motivated primitives

Chapter 2 introduces previous work on English text similarity that forms the basis of my lingual text similarity research. This thesis presents Similarity finder, a system I developed that takes the ideas presented in English Similarity finder, in particular the idea of linguistically motivated features for comparing sentences on multiple axes. Using multiple axes for similarity allows the system to target more specific types of similarity than can be observed using just bag-of-words based approaches, and allows the easy integration of knowledge sources such as word Net [22] and grammatical information via part-of-speech based primitives.

5.1.2 A flexible framework for experimenting with lingual text similarity

Chapter 3 describes Similarity finder, my implementation of a cross-lingual text similarity computation system, and details my English implementation. Similarity finder computes similarity over text at the level of primitives, easily identifiable classes of text such as nouns, verbs, World Netsynsets, or named entities. The primitives are linguistically motivated and Similarity finder makes it easy to add and experiment with new primitives. Similarity across languages does not use full machine translation over the text, but is instead computed based on translation at the primitive level, where multiple translation approaches can be combined. In this work, I present results using two features for English similarity: token level primitives, and phrasal primitive's uses named entities.

Using a probabilistic dictionary is shown to improve results over using dictionary look up alone by increasing precision from 49.1% to 81.% when using both token and named entity features. The hypothesis that adding the named entity feature improves English could not be validated, as runs with the named entity feature did not statistically significantly improve precision, although it also did not statistically significantly reduce precision.

The best performing run of Similarity finder, using probabilistic and Buck Walter translation with tokens and named entity features, performed nearly as well considering precision only as the gold standard run of manually translated English text using Similarity finder : 81.%, compared to the manual run of 84.6%. Using Similarity finder has much better precision than machine translation with English Similarity finder, at 66.5%, although English Similarity finder does have much better recall.

Similarity finder is designed to make adding support for new languages easy. The approach taken in Similarity finder is applicable to \resource poor languages by using simple techniques for primitive identification, such as regular expression based tokenizes to identify token primitives for the language, and translation using a probabilistic bilingual translation lexicon learned from a parallel corpus. Similarity finder is able to use multiple translation methods. The best performing versions of Similarity finder use both a learned probabilistic Translation lexicon, and an existing

translation resource (glosses from the Buck Walter morphological analyzer [Buc02].) For languages without many resources, if a parallel corpus is available using only the learned probabilistic translation lexicon results in only a minor loss in precision compared to using both translation resources (.% precision vs. 80.0% precision with both.)

5.1.4 CAPS: Summarization that identifies similarities and differences across similar sentence

In the context of multi-document summarization, the test-bed application for sentences, precision is more important that recall. While missing some sentences is acceptable since many sentences will by necessity be pruned from the summary, and important content is assumed to be repeated, having poor clusters with non-relevant sentences is not acceptable. Chapter 5 presents results from two summarization systems. The first improves the understandability of a summary of machine translated documents by conditionally replacing machine translated summary sentences with highly similar English sentences when one exists. The second system, CAPS, is a novel use of lingual text similarity to build a summary that indicates to a user both what information is shared between two document sources and what information is specific to only one source or the other.
CAPS are evaluated using English and documents, and improve understandability by selecting English sentences for the summary for clusters with support from both English documents. CAPS also breaks down information below the sentence level by applying syntactic simplification to the English text. CAPS are evaluated using both machine translated text with English sentences text using sentences. The approach using sentences performs much better than using machine translation for identifying content shared between the English texts. It does not, however, perform as well at monolingual (English) content identification. CAPS receives an average 0.826 SCU pyramid score for mixed content with sentences, compared to 0.641 using manually translated English text, or 0.565 for machine translated text.

The sentences approach works very well for the cross-language English content identification task, validating one of the design goals of sentences. The summarization approach used in CAPS is also applicable to any languages for which a similarity metric can be computed between the English texts. Translation need only be performed as a presentation step; the foreign language sentences can be presented unmodified to bilingual users, taking advantage of a summarization strategy that does not require any linguistic knowledge beyond what is needed for similarity computation and cluster ranking. Even without translating the extracted sentences into English, the foreign language text can still be summarized and compared to English text, highlighting the similarities and differences between the two documents sources, which is a major contribution of this thesis.

5.2 Limitations

In the remainder of this chapter, I will present some limitations on the work in these thesis problems or limitations that came to light during the development and evaluation of the systems presented. Some of these limitations result from design decisions, others from practical considerations due to difficulty of implementations, lack of resources, etc., but that do not represent fundamental deficiencies in the approach. Section 6.3 presents further work to be done in this area that could extend the applicability and performance of the approach.

5.2.1 Experimentation with more English primitives

The language pair investigated in this work is English. One of the contributions of the original English sentences work was the use of multiple linguistically-motivated features used for similarity computation. The same approach is taken in sentences, but I only investigated two primitives: tokens and named entities. I decided to not perform morphological analysis at an early stage, but further work with English similarity should investigate using morphological analysis to break the tokens down into simple part of speech categories to use as additional primitives. Other more complex primitives would be an interesting area for further research as well, such as normalizing time expressions out to a format that would be comparable across documents and language, or primitives that make use of more knowledge-heavy linguistic resources, such as the corpora being produced by the Inter lingual Annotation of Lingual Text Corpora project [25].

5.2.2 Better translation for named entities

I use IBM's statistical English machine translation system to translate named entities when it is available; otherwise I try to match named entities based on translations of their component words. A much preferable approach would be to use a system such as Knight's transliteration system [4,7] for known named entities many named entities are known not to be in any lexicons, as it is an open class of words that is constantly being added to by the creation of new company names, or new celebrities with previously translated names entering into the news. For translation of general noun phrases, it would be interesting to try a system specific to noun phrase translation, such as the one described in Philipp Kohn's thesis work [17]. The primitive translation phase should also include more support for fuzzy matching and partial matches. sentences is not trying to detect only exact translations, but similar sentences which would benefit from a principled investigation of fuzzy matching for primitives across similar sentences.

5.2.3 Language feature sets and merging models

As explained in Section 3.3.5 sentences uses a single feature merging model when combining feature similarity values into a single similarity value. Sentences should

be extended to allow dynamically choosing the feature merging model to use based on the language of the text units being compared. For two English units, it should use a model trained specifically over English data, for units it should use a model trained with English data, and for English units, it should use a model trained using English translation data.

Currently, sentences can easily extract different sets of features for text units in different languages, but to simplify programming in this thesis I use the same features when computing similarity across similar and within similar sentences.

I have performed initial experiments combining both English training data and English training data in a single feature merging model, but this approach needs more work, since the additional English training data does not improve results for English similarity, and hurts English similarity results. The next section introduces these initial experiments.

5.2.3.1 Combining English training data

In Section 3.4.4 I train a model for English using the Multi-translation corpus. This training data is used to train a model that is used to compute similarity over English sentences, and English sentences. Intuitively, not having training data for the case and English cases would have a negative impact on similarity computation for English sentences. To improve the English results, I performed an experiment that adds the English training data to the English training data. I would also like to add similarity data, but I do not have a training set of similar sentences.

Threshold	Precision	Recall
0.1	32.88	58.24
0.2	47.97	40.25
0.3	54.02	29.55
0.4	57.96	22.72
0.5	60.81	18.84
0.6	64.85	15.63
0.7	67.72	12.67
0.8	72.06	9.65
0.9	74.62	6.37
0.95	82.02	4.79

Table 3.6: Feature merging model training results using token and Identity Finder features With Buck Walter and probabilistic translation using English training data.

Training with the English data results in lower precision and recall than training over just the English data. The English data includes many examples that are quite difficult to classify automatically; training the English version of sentences also results in lower scores than the English data, although due to additional features English version of sentences performs better than this run using only includes token and Identity Finder named entity features. For comparison, the model using only English training data has 86% precision and 50% recall at a threshold of 0., whereas the model with English training data has 6% precision and 13% recall. These results indicate that a single set of features and a single feature merging model are not appropriate in the lingual case. Future work should investigate adding feature sets on a per-language basis (this is already supported in sentences) with feature merging models that use the appropriate feature set merging model based on the languages of the sentences.

5.3 Future Work

This section explores other areas to be explored within the similarity finder framework for Lingual text similarity where I have not yet performed much work.

5.3.1 Further integration of statistical machine translation methods

Similarity finder uses a learned probabilistic translation lexicon using an IBM model 3 implementation. Further investigation of the integration of other statistical machine translation methods (distortion model, full decoder, etc.) would be useful.

A distortion model might help improve similarity finder s results at finding sentences that are translations of each other. However, since similarity finder is searching for similar sentences that might not be translations of each other conveying the exact same information, a distortion model might impose too many restrictions, giving similar, but structurally different sentences, low probabilities.

5.3.2 Noun Phrase Variant Identification

Noun phrase variant identification is an area where better translation methods would help. Given a feature that extracts noun phrases in one language, to properly match to a noun phrase in another language would require either a translation mechanism that produces an N-best list with all likely variants of a noun phrase, or a noun phrase variation system. This section describes some related work in noun phrase variant recognition, and early experiments I performed with similarity finder noun phrase variation in French and English. Initial results were not encouraging, and I believe a more in-depth investigation is required to see improvement based on these techniques.

5.3.2.1 Related Work on Noun Phrase Variation

One of the early areas of this thesis work was the investigation of using noun phrase variation to recognize different forms of noun phrases across documents and across languages. Noun phrase variation was used by Bourigault 1992 [1] for the identification of terminological units. Maximal length noun phrases were identified and parsed to identify likely terminological units due to the grammatical structure of the noun phrases. The resulting terminological units were then passed to a human expert for validation.

5.3.3 Sense disambiguation

When translating primitives, similarity finder does not perform any sense disambiguation in order to determine which sense of a primitive is most appropriate to choose for translation.

REFERENCE

[1] Didier Bourifault. Surface grammatical analysis for the extraction of terminological Noun phrases. In Proceedings of the 14th International Conference on Computational Linguistics, pages 9{981, 1992.

[2] Christopher Buckley. Implementation of the smart information retrieval system. Technical Report Technical Report 85-686, Cornell University, Ithaca, New York, 1985

[3] Regina Barzilay, Kathy McKeown, and Michael Elhadad. Information fusion in the context of multi-document summarization. In Proceedings of the 3th Association of Computational Linguistics, Maryland, June 1999.

[4] Hsin-Hsi Chen and Chuan-Jie Lin. A multilingual news summarizer. In Proceedings of the 18th International Conference on Computational Linguistics, pages 159{165, 2000.

[5] William W. Cohen. Learning trees and rules with set-valued features. In AAAI/IAAI, Vol. 1, pages 09{16, 1996.

[6] Jeffrey C. Reynar and Adwait Ratnaparkhi. A maximum entropy approach To identifying sentence boundaries. In Proceedings of the Fifth Conference on Applied Natural Language Processing, March 31{April 3 199.

[7] Nina Wacholder David Kirk Evans, Judith L. Klavans. Document processing with linkit, April 2000.

[8] David A. Evans and Chengxiang Zhai. Noun-phrase analysis in unrestricted text for information retrieval. In Proceedings of the ACL-96, 34th Annual Meeting of the Association for Computational Linguistics, pages 1{24, Santa Cruz, US, 1996.

[9] W.B. Frakes and R. Baeza-Yates, editors. Information Retrieval: Data Structures and Algorithms, pages 419{442. Prentice Hall, Englewood Clies, NJ, 1992.

[10] William A. Gale and Kenneth Ward Church. A program for aligning sentences in bilingual corpora. In Meeting of the Association for Computational Linguistics, pages 1{184, 1991.

[11] G. Grefenstette and J. Nioche. Estimation of English and non-English language use on the WWW. In Proceedings of RIAO'2000, Content-Based Multimedia Information Access, pages 23{246, Paris, 12{14 2000.

[12] Vasileois Hatzivassiloglou, Luis Gravano, and Ankineedu Maganti. An investigation of linguistic features and clustering algorithms for topical document clustering. In Proceedings of the 23rd ACM SIGIR Conference on Research and Development in Information Retrieval, 2000.

[13] Vasileios Hatzivassiloglou, Judith L. Klavans, and Eleazar Eskin. Detecting text similarity over short passages: Exploring linguistic feature combinations via machine learning. In Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, pages 203{212, College Park, Maryland, June 1999.

[14] V. Hatzivassiloglou, J. L. Klavans, M. Holcombe, R. Barzilay, M.Y. Kan, and K.R. McKeown. Similarity finder: A flexible clustering tool for summarization. In NAACL'01 Automatic Summarization Workshop, 2001.

[15] E.H. Hovy and Chin-Yew Lin. Automated text summarization in summarise. In I. Mani and M. Maybury, editors, Advances in Automated Text Summarization, chapter 8. MIT Press, 1999.

[16] Min-Yen Kan and Judith L. Klavans. Using librarian techniques in automatic text summarization for information retrieval. In Proceedings of the Joint Conference on Digital Libraries, pages 36{45, Portland, Oregon, USA, July 2002.

[17] Philipp Kohn. Noun Phrase Translation. PhD thesis, University of Southern California, 2003.

[18] Leonard Kaufman and Peter J. Rousseeuw. Finding Groups in Data: An Introduction to Cluster Analysis. Wiley, New York, 1990.

[19] Chin-Yew Lin and E.H. Hovy. Automatic evaluation of summaries using co-occurrence statistics. In Proceedings of 2003 Language Technology Conference (HLT-NAACL 2003), Edmonton, Canada, May 2003.

[20] H.P. Luhn. The automatic creation of literature abstracts. IBM Journal of research and development, 2(2), 1958.

[21] Daniel Marcu. From discourse structures to text summaries. In I. Mani and M. Maybury, editors, Proceedings of the ACL/EACL'9 Workshop on Intelligent Scalable Text Summarization, pages 82{88, Madrid, Spain, July 1999.

[22] George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. Introduction to word net: an on-line lexical database. International Journal of Lexicography, 4(3):235{244, 1990.

[23] I. Dan Melamed. Automatic discovery of non-compositional compounds in parallel data. In Claire Cardie and Ralph Weischedel, editors, Proceedings of the

Second Conference on Empirical Methods in Natural Language Processing, pages 9{108. Association for Computational Linguistics, Somerset, New Jersey, 199.

[24] I. Dan Melamed. A portable algorithm for mapping bitext correspondence. In Philip R. Cohen and Wolfgang Wahlster, editors, Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics, pages 305{312, Somerset, New Jersey, 199. Association for Computational Linguistics.

[25] Teruko Mitamura, Keith Miller, Bonnie Dorr, David Farwell, Nizar Habash, Stephen Helmreich, Eduard Hovy, Lori Levin, Owen Rambow, Florence Reeder, and Advaith Siddharthan. Semantic annotation for interlinguas representation of multilingual texts. In Language Resources and Evaluation Conference Work- shop: Beyond Named Entity Recognition - Semantic Labelling for NLP Tasks, Lisbon, Portugal, May 2004.

[26] Peter McCullagh and John A. Nelder. Generalized Linear Models. Chapman and Hall, London, second edition, 1989.

[27] Ani Nenkova and Rebecca Passonneau. Evaluating content selection in summarization: the pyramid method. In Proceedings of the Human Language Technology / North American chapter of the Association for Computational Linguistics conference, May 2004.

[28] Paul Over and J. Yen. An introduction to duc 2003 intrinsic evaluation of generic news text summarization systems. In Proceedings of the Document Understanding Conference, 2004. National Institute of Standards and Technology.

[29] Dragomir R. Radev, Hongyan Jing, and Malgorzata Budzikowska. Centroid based summarization of multiple documents: sentence extraction, utility-based evaluation and user studies. In Proceedings of ANLP/NAACL 2000 Workshop, pages 21{29, April 2000.

[30] D Radev, S. Teufel, H. Saggion, W. Lam, J. Blitzer, H. Qi, A. Elebi, D. Liu, and E. Drabek. Evaluation challenges in large-scale document summarization. In Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, Sapporo, Japan, May 2003.

[31] Advaith Siddharthan, Ani Nenkova, and McKeown Kathleen. Syntactic simplification for improving content selection in multi-document summarization. In Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004), 2004.

[32] G Salton. Automatic Information Organization and Retrieval. McGraw-Hill, New York, 1968.

[33] Gerald Salton. The SMART retrieval system - Experiments in automatic document processing. Prentice-Hall, Englewood Clies, New Jersey, 191.

[34] Gerard Salton and Chris Buckley. Term-weighting approaches in automatic text retrieval. Information Processing and Management, 24(5):513{523, 1988.

[35] Barry Schulman, Ani Nenkova, and Kathleen McKeown. Experiments in multi document summarization. In Proceedings of the Human Language Technology Conference, March 2002.

[36] E. M. Voorhees. The Effectiveness and Efficiency of Agglomerative Hierarchical Clustering in Document Retrieval. PhD thesis, Cornell University, 1986.

[37] Advaith Siddharthan. Resolving attachment and clause boundary ambiguities for simplifying relative clause constructs. In Proceedings of the Student Work- shop, 40th Meeting of the Association for Computational Linguistics (ACL'02), pages 60{65, Philadelphia, USA, 2002.

[38] Nina Wacholder, David Kirk Evans, and Judith L. Klavans. Automatic identification and organization of index terms for interactive browsing. In Proceedings of The First ACM+IEEE Joint Conference on Digital Libraries, pages 126{134, Roanoke, VA, 2001.

[39] Bonnie Glover Stalls and Kevin Knight. Translating names and technical terms in text. In Proceedings of the 1998 COLING-ACL, Montreal, Canada, 1998.

[40] David Yarowsky. One sense per collocation. In Proceedings of the ARPA Human Language Technology Workshop, pages 266{21, Princeton, NJ, 1993.

List of Publication

1. Shivam Maurya, Mohd. Saif Wajid, Ramesh Vaishya, Paper title “Sentence Similarity Based Text Summarization Using Clusters” . International Journal of Scientific and Engineering Research (IJSER) Volume 4, Issue 5(May 2013).

CURRICULUM VITAE

Career Objective

Utilize and enhance my skills by working in professional environment.

Knowledge Preview:

Operating Systems	:	Windows Family
Programming Languages	:	C, JAVA
Database	:	Oracle 10g, MySQL
Web Skills	:	HTML, DHTML
Office Suite	:	Microsoft Office (Word/Excel/PowerPoint)

Educational Qualification

Year	Degree/Certificate	Institute/school	%
2011-13	M.Tech (Computer Science) Pursuing	Babu Banarasi Das University, Lucknow	**71.72** **Upto III Semester**
2011	B.Tech (Computer Science)	Bhagwant Institute of Technology, Muzaffarnagar, UPTU	**67.40**
2006	Class XII	A.D.S.V.M.,Sitapur	**77.00**
2004	Class X	A.D.S.V.M.,Sitapur	**75.33**

Personal Information

Date of Birth	June /12/ 1990
Sex /Nationality	Male / Indian
Languages known	English & Hindi
Permanent Address	967-A Thomson Ganj Sitapur , U.P. – 261001
Mobile No.	+917376878629
E-mail	reverentshivam@gmail.com