Identifying Similarity in Text: Multi-Lingual Analysis for Summarization

David Kirk Evans

Submitted in partial fulfillment of the
requirements for the degree
of Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2005

ABSTRACT

Identifying Similarity in Text: Multi-Lingual Analysis for Summarization

David Kirk Evans

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 different languages. This thesis presents my work on multi-lingual text similarity. This work enables the identification of short units of text (usually sentences) that contain similar information even though they are written in different languages. I present my work on SimFinderML, a framework for multi-lingual text similarity computation that makes it easy to experiment with parameters for similarity computation and add support for other languages. An in-depth examination and evaluation of the system is performed using Arabic and English data. I also apply the concept of multi-lingual text similarity to summarization in two different systems. The first improves readability of English summaries of Arabic text by replacing machine translated Arabic sentences with highly similar English sentences when possible. The second is a novel summarization system that supports comparative analysis of Arabic and English documents in two ways. First, given Arabic and English documents that describe the same event, SimFinderML clusters sentences to present information that is supported by both the Arabic and English documents. Second, the system provides an analysis of how the Arabic and English documents differ by presenting information that is supported exclusively by documents in only one language. This novel form of summarization is a first step at analyzing the difference in perspectives from news reported in different languages.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to acknowledge, first and foremost, my advisor Judith L. Klavans, PhD. She has supported me all of these years, and I was able to work on multi-lingual summarization, which is what originally drew me to Columbia University. I am also indebted to Kathleen McKeown, PhD, who has had a profound impact on this work in its later stages.

This work has been funded in part by NSF grant number IRI-97-12069 and the DARPA TIDES program (contract N66001-00-1-8919). I would also like to thank Salim Roukos and Kishore Papineni for the use of their statistical Arabic machine translation system.

I am also extremely grateful to the Natural Language Processing group at Columbia University. There is not a nicer group of people around; I am indebted to the group for all the help proofreading papers, giving excellent feedback on practice talks, and most importantly taking part in evaluations.

I also have to thank my great office mates over the years. Eleazar Eskin for keeping me from working too hard, Carl Sable for pushing me to work harder and think of better palindromes, Smaranda Muresan for putting up with Carl and me for even just a few short months, and Michelle Galley for an even shorter time. More recently, and for a longer period of time, Elena Filatova for her sparkling conversation, and Sasha Blair-Goldensohn for reminding me to play basketball, even when I should be working on my thesis.

I thank Central Research Facilities for keeping the computers working, and in particular Dennis Shim and Mark Yeun for representing Computer Scientists everywhere at Columbia University's pickup basketball games.

I would like to acknowledge my three favorite Brooklyn/Manhattan based bands for keeping me sane through all the years and their great music. Bishop Allen for being great all around-guys and reminding me that things are what I make of them, the We Are Scientists for adding a hard edge to Science and Rock-and-Roll, and Say Hi To Your Mom for the most relevant album of 2004.

I also could not have done this without the support of my family and friends. My father, who himself has a PhD in Electrical Engineering, for inspiring me as a child to continue to study and to want to become an engineer myself. My mother for all the great support that only a mother could give, and for always being willing to chat about Dallas Mavericks basketball and not probing too deeply into whatever aspect of my research might be frustrating me at the time. I have also been blessed to have great friends in the area, Ron and Michelle who I met back in 5th and 6th grade, and whose marriage prompted me to grow up and get on with my life. Eric Han, who's research at Columbia University in the East Asian studies department has nothing to do with my research, but for helping me believe that I'll be able to end up in Japan doing research as well. And all the friends that I have made, many in the NLP research group here, and in particular Carl Sable, who has been a great help from a research and friendship point of view.

Dedicated to Judy Marie Evans and Gary Alan Evans, PhD.

Chapter 1

Introduction

There is a lot of text in the world. According to Global Reach's 2004 estimate, there are 295.4 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 filter and manage the information that people need. Information retrieval engines help people find 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 briefly while preserving the underlying meaning [Luh58]. Edmundson [Edm69] 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

[1]http://www.glreach.com/globstats/

[2]http://www.archive.org/about/faqs.php#9

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 [KPC95, Mar97, JM00]. 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 multi-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 [Bar03].

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

1.1 Goals

There are two main goals for this thesis: to introduce my work in multi-lingual text similarity, and to show that the system I built for the task can be used as the basis of a multi-lingual multi-document summarization system. SimFinderML, a system I developed for multi-lingual text similarity, is described and evaluated in a sentence and clustering-level evaluations of Arabic and English text. Another contribution of the thesis is an approach to multi-lingual multi-document summarization that shows similarities in documents in multiple languages, as well as differences between them which use SimFinder and SimFinderML. I first introduce related work that has been done on English text similarity in the Simfinder system that forms the basis of my work on a multi-lingual version of Simfinder called SimFinderML. Simfinder was developed at Columbia University under the supervision of Judith Klavans and Kathy McKeown, and I have also worked on Simfinder improvements and maintenance. I use this past work on Simfinder to motivate that the approach SimFinderML 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.

SimFinderML, like Simfinder, uses linguistically derived primitives in each language to define multiple axes for similarity measurement, translates at the level of these primitives instead of using full machine translation at the sentence level, and forms a final similarity value based on the similarity of the features. The traditional bag-of-words approaches treat text as unordered words and do not understand the grammatical roles of words, such as subjects or objects, or the part-of-speech roles of words, such as nouns or verbs.

In the Simfinder approach, different primitives, such as "words that are nouns" or "words that are verbs" are identified, and similarity is computed over all of these features. For two sentences, Simfinder will compute how similar those sentences are based on each feature, and it combines all the similarities into a single final 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 SimFinderML extends the approach taken in Simfinder to enable multi-lingual text similarity computation. SimFinderML identifies primitives in text in multiple languages, 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. SimFinderML introduces a translation stage that maps primitives from one language to another to enable matching primitives across languages without using full machine translation on the non-English documents. SimFinderML can identify similar text across languages using only simple techniques for primitive translation. SimFinderML is designed to make it straightforward to add support for new languages, and it has been tested with minimal modifications over languages from different families (French, Spanish, Chinese, Japanese, Arabic, and English.) This thesis will show that SimFinderML, using simple techniques that can be quickly applied to other languages, performs with high precision at identifying similar sentences across languages.

The second main goal of this thesis is to present two summarization systems that have been built on top of the multi-lingual text similarity computation technology in SimFinderML. The systems validate that SimFinderML is a useful multi-lingual text similarity computation engine. The first system takes a novel approach to improving the readability of English summaries of machine translated Arabic data. It produces a summary of machine translated Arabic text, and uses Simfinder to identify English sentences that are similar to the machine translated summary sentences, replacing them if the two are similar enough. Summaries of Arabic text can be greatly improved by substituting errorful Arabic text with fluent English text found in the English documents. The second system presents a new approach to multi-lingual multi-document summarization that highlights what information is shared between Arabic and English input documents, and what information is available in only the Arabic or English documents. This system is innovative in that it does not just present information that is similar between all source documents, as previous multi-document summarization systems have done. It takes the input documents as representatives of two different perspectives (Arabic and English in the implemented system) and also explicitly indicates information that is unique to one language or the other. That approach is not language dependent, and could work with any language pairs for which a machine translation system, or multi-lingual text similarity computation system such as SimFinderML exists. When using SimFinderML, even if full machine translation systems are not available, the English and non-English text can be summarized and presented to a bilingual analyst.

Research questions that this thesis answers include:

- Can a system automatically identify similar sentences across languages?
- 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 languages be used for multi-lingual multi-document summarization?
- Can sentence similarity be applied to improve summaries of machine translated text?
- Will cross-lingual sentence similarity allow for the creation of summarization systems that present differences in perspective across languages by summarizing similarities and differences across the input documents?

1.2 Approaches to text similarity

Chapter 2 introduces the English version of Simfinder, a program for computing the similarity of English sentences and clustering them. Simfinder was designed by other people at the Columbia University Natural Language Processing group, most notably Judith Klavans, Vasileios Hatzivassiloglou, and Melissa Holcombe. Simfinder 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. Simfinder 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 WordNet [MBF$^+$90]. Chapter 2 presents a comparison of the Simfinder 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 Simfinder to multiple languages in SimFinderML, a multi-lingual re-implementation of Simfinder. SimFinderML is described in Chapter 3. SimFinderML is designed to allow for easy addition of new primitives and features for comparing text, and I present an in-depth description of the Arabic support in SimFinderML. Chapter 4 presents an evaluation of how well SimFinderML is able to identify English sentences that are similar to Arabic sentences. The usefulness of SimFinderML as a cross-lingual text similarity computation system is verified by using it as the basis for two summarization systems, presented in Chapter 5.

1.3 Similarity-based approaches to Multi-Document Summarization

Similarity based summarization approaches are not new in the area of summarization. Similarity-based summarization is an accepted, well-respected approach to multi-document summarization. While there are many summarization systems that use similarity-based approaches, they are typically applied to monolingual summarization systems. SimFinderML allows the approach to be applied to multi-lingual multi-document summarization systems.

SimFinderML 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 SimFinderML. One system summarizes machine translated Arabic text and replaces Arabic sentences with very similar English sentences to improve the readability of the summary.

1.3.1 Highlighting Similarities and Differences between Foreign and Source Language Data

A second summarization system using SimFinderML 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 Arabic and English documents on the same topic as input. It generates a summary in three parts: information that is only present in the Arabic text, information that is only present in the English text, and information that is supported by both the Arabic and 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.

1.4 Contributions

Novel aspects of this research are:

1. **Flexible framework for multi-lingual text similarity experimentation.** I developed SimFinderML, which supports rapid development of features for similarity computation for different languages, 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 Chapter 4.

2. **Experimentation with and evaluation of different levels of translation for multi-lingual text similarity identification.** I have examined how translation can be used at different levels for multi-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 are easily portable to new languages.** The main language pair presented in this thesis is Arabic and English, but I have also used SimFinderML with French, Chinese, and Japanese with very little engineering required to add support for those languages. 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 new languages.

4. **Investigating primitives for similarity, and translating primitives across languages.** An original contribution of this work is the investigation of primitives that are compatible across languages for the similarity computation process and methods of translating those primitives. Similarity computation performed over primitives and their translations extracted from the native language is more easily extensible to languages for which we do not already have a full machine translation system. For high precision tasks requiring identification of Arabic–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 between Arabic and English. This work takes the first steps to identifying further primitives that may be helpful for cross-language similarity computation, and presents a framework for continued research in this area.

5. **Applying multi-lingual text similarity computation to multi-lingual multi-document summarization.** This thesis presents two summarization systems that use text multi-lingual similarity. The first uses similarity to replace machine translated sentences from Arabic documents with similar English sentences to improve readability of summaries in English.

6. **Using multi-lingual text similarity computation to show similarities and differences between information sources.** I present a second system that summarizes Arabic and English documents, indicating similarities and differences between

the Arabic and English sources. The ability to indicate differences between the document sources is a novel contribution, as previous work focused on identifying similarities between documents. This work leads the way for further research in active analysis of difference in perspectives between documents sets and languages, a boon for information analysts.

Chapter 2

Similarity in English Texts: Simfinder

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.

Simfinder is a system designed and implemented at Columbia University for identifying similar units of short text, either paragraphs and sentences, and clustering related sentences into "themes" that express the same information. Simfinder has been used in multiple summarization and question-answering systems [BME99, BGMS03, KK02]. This chapter describes the Simfinder system as implemented for English. I build upon the work done on Simfinder by re-implementing a version that performs cross-language similarity identification, SimFinderML, described in Chapter 3.

Section 2.1 discusses related work using textual similarity measures including information retrieval and document clustering. Information retrieval systems use similarity at a document level to identify documents that are similar to a user query. The granularity of the similarity judgment is more coarse than in the task of identifying specific sentence-to-sentence similarity which I present. Document clustering systems use a similarity metric over documents to create clusters containing similar documents.

Simfinder, unlike clustering methods that only use word overlap, uses multiple linguistically motivated features for similarity computation. One of the prominent features is noun phrase matches using LinkIT, described in Section 2.2.1.1, which I developed in my early work on document characterization. The features and the machine learning framework used to determine useful features for similarity computation are described in Section 2.2.1.

Like document clustering systems, Simfinder use the results of a similarity metric over text to cluster the sentences using a clustering method tailored for summarization, discussed in Section 2.2.2.

The system described in this chapter, English Simfinder, provides a basis for many of the techniques proposed in this thesis. Simfinder was originally implemented by Judith Klavans, Vasileios Hatzivassiloglou, Regina Barzilay, Eleazar Eskin, and Melissa Holcombe. I contributed the LinkIT primitive to Simfinder, and added a filtering step for use with machine translated text, but have not otherwise made major modifications to English Simfinder as described in this chapter.

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 [Sal71] 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 word 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. Figure 2.1 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 Simfinder for similarity comparison, which are much shorter, being sentences or even clauses. This leads to a data sparsity 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, Simfinder uses a variety of features built over different primitives, such as nouns or verbs, that investigate similarity in a number of linguistically motivated areas. A detailed description of Simfinder's approach is given in Section 2.2.

2.1.2 Clustering Techniques

Simfinder uses clustering in two ways:

- **document clustering** as a pre-input stage to Simfinder for identifying documents that are on the same topic

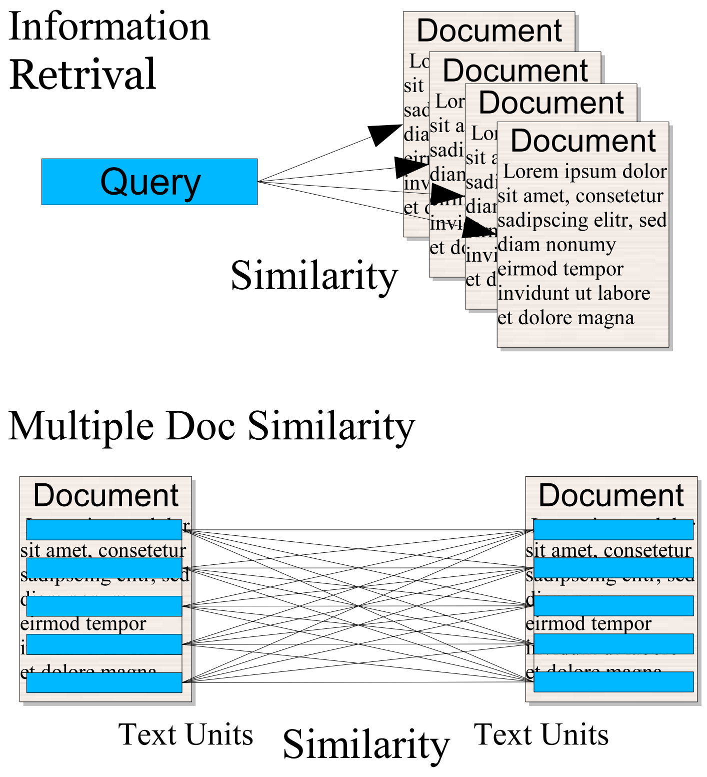

Information Retrival
Query
Document
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna
Similarity
Multiple Doc Similarity
Document
Document
Text Units
Similarity
Text Units

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, as discussed above in Section 2.1.1.

2.1.2.1 Similarity measures - using term overlap

In a survey of document clustering techniques, Rasmussen 1992 [FBY92] 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 S_{D_i,D_j} 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 document 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 [SMH96], 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. Simfinder 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 [FBY92],) 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, specifically 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 [EHW87] 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 [Voo86] 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 [EHW87] compared the same methods plus Ward's method on document sets of up to 27,361 documents, and found that the group average method was most effective for document clustering. Hatzivassiloglou et al. [HGM00] 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.

In a 2002 study of NewsBlaster, an online news clustering and summarization system [MBE$^+$02], document sets that were automatically clustered for summarization were as large as 60 documents in real-world data sets with an average of 18.7 sentences[1] per document, which is 1,122 sentences overall. Typical sizes for the sort of data sets that Simfinder and SimFinderML work with should therefore be able to cluster on the order of 1,000-5,000 units.

The task of clustering textual units in Simfinder is similar to the document clustering task, but the extremely small size of each text unit requires similarity measures that are based on more than just simple term overlap. The novel contribution of Simfinder is in computing the similarity measure between text units, described in Section 2.2.1, and the clustering method which has also been tailored to the sentence clustering task, described in

[1]Average of 18.7 sentences computed from 218 documents selected at random on 6/3/2002 with a sample standard deviation of 9.3 sentences.

Section 2.2.2.

2.2 English Simfinder

The Simfinder 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. Simfinder 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 Simfinder, documents are often sentence segmented, but in the news domain it can be helpful for Simfinder 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. Simfinder 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

Simfinder 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. Figure 2.2, from Hatzivassiloglou et al.'s 2001 paper on Simfinder [HKH$^+$01], illustrates some example primitives extracted by Simfinder through the use of two example similar paragraphs from the Simfinder training corpus. Typical types of primitives that are extracted by Simfinder 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 longrunning **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 similarity that are captured by Simfinder are highlighted in bold.

An OH-58 helicopter, carrying a crew of two, was on a routing 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 primitives 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 doctors 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 1997 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 a WordNet match (highlighted in bold) occur within a window of five words, excluding stopwords.

between Mexico and Robinson are also matches on the LinkIT 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: Simfinder considers matches involving identical words, as well as words that matched on their stem, as noun phrase heads ignoring modifiers, and as WordNet [MBF$^+$90] 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 this 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 WordNet primitives (i.e., words match if they share immediate hypernyms in WordNet) and exact word match primitives. On its own, the WordNet 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: LinkIT

One of the important features used in Simfinder is the LinkIT feature, which indicates matches based on the heads of noun phrases. The motivation behind this primitive is my previous work using LinkIT for document characterization, indexing, and browsing [DKE00, WEK01].

I developed LinkIT as a document analysis and characterization system. LinkIT identifies noun phrases in documents, and relates noun phrases within a document. I built a 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 [Yar93] 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 context to the term.

LinkIT identifies Simplex Noun Phrases [Wac98] in a document, and relates them together based on the head of the Simplex Noun Phrase. A simplex NP is a maximal NP with a common or proper noun as its head, where the NP may include premodifiers such as determiners and possessives but not post-nominal constituents such as prepositions or relativizers. Examples are *asbestos fiber* and *9.8 billion Kent cigarettes.* Simplex NPs can be contrasted with complex NPs such as *9.8 billion Kent cigarettes with the filters* where the head of the NP is followed by a preposition, or *9.8 billion Kent cigarettes sold by the company,* where the head is followed by a participial verb. Simplex NPs end at a conjunction, except for certain cases which coordinate adjectives to a noun phrase.

The use of noun phrases as index terms leads to a high quality browsing interface, as shown in [WEK01] which describes IntellIndex, a document browsing index for Digital Libraries built using the output of LinkIT 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 [EZ96] 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 Hatzivassiloglou et al. 2000 [HGM00]. These applications of noun phrases to similarity based tasks indicate that they are a useful area to focus on for multi-lingual similarity detection.

In Simfinder, the unary LinkIT feature has been chosen in the machine learning framework as an important feature that helps to identify similar sentences based on the training data. In SimFinderML I have continued the focus on noun phrases by implementing a named entity identification feature for Arabic and English using BBN's IdentiFinder [BBN04] system. Chapter 4 presents an evaluation of SimFinderML using the named entity feature, and shows that it improves precision in the cross-lingual Arabic–English case.

2.2.1.2 Other features

Simfinder uses other features besides LinkIT as well. The first version of Simfinder examined a variety of features over short units of text to see which ones were good predictors of similarity. In [HKE99], the authors identified 43 features that could be efficiently extracted from the text and that could plausibly help determine the semantic similarity of two short text units. The unary features examined are: word overlap, proper noun overlap, LinkIT overlap, verb overlap, verb class overlap, noun overlap, adjective overlap, WordNet overlap, and WordNet verb overlap. Composite features are: the same words in variants with a window from 1-4 words, composite features with the same word and verbs, composite features with the same word and nouns, a composite feature with the same word and a WordNet synonym, composite features with LinkIT and WordNet verbs, composite features with LinkIT and verbs, and word stem overlap. Most of these features had normalized versions as well that take into account the inverse document frequency scores of the matched words.

The input text is part-of-speech tagged, and primitives are built that restrict matches to certain part of speech tags. Nouns, proper nouns, verbs, and adjectives are examples of these sorts of primitives. The WordNet primitive matches on any words that share a WordNet synonym. The verb class primitive matches verbs that are in the same class based on output from Min-Yen Kan's program, Verber, which is based on Levin's verb classes [KK98, Lev93].

The next section discusses how the feature set for English Simfinder was chosen based on machine learning approaches, and discusses the model used to merge the feature similarity values into a final similarity value.

2.2.1.3 Learning Method and Results

With such a large number of features available to Simfinder 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 is 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 1997 TDT pilot corpus 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 that 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 Simfinder tries to recover automatically is much more fine-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 SimFinderML.

The first version of Simfinder output binary similarity values for each pair of input sentences using a rule-based classifier learned from the training data over the features that Simfinder computed for each sentence pair. Simfinder used a classifier trained over both primitive and composite features using RIPPER [Coh96]. RIPPER produces a set of ordered rules that can be used to judge any pair of paragraphs as similar or non-similar. Using threefold cross-validation over the training data, RIPPER included 11 of the 43 features in its final set of rules and achieved 44.1% precision at 44.4% recall. The ten unary features were word overlap, proper noun overlap, LinkIT overlap, verb overlap, noun overlap, adjective overlap, WordNet overlap, WordNet verb overlap, verb overlap, and stem overlap. One composite feature was selected, WordNet collocation, which is a match between the WordNet primitive and the word primitive (see [HKE99] 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 [SB88], 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 [Buc85], which uses a modified TF*IDF approach, achieved 34.1% precision at 36.7% 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 Simfinder the collection of documents from which document frequency is calculated has been filtered 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 Simfinder operates.

The current version of Simfinder, Simfinder 1.1, 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 Simfinder to give higher weight to paragraph pairs that are more similar than others. Simfinder 1.1 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 [MN89]. A weighted sum of the input features is used as an intermediate predictor,

η, 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 7 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, WordNet overlap, proper noun overlap, and LinkIT overlap. The single composite feature selected matches to the WordNet primitive and a word primitive. As in the case of the RIPPER model, the automatic selection of multiple features in the loglinear 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 LinkIT 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. Simfinder's clustering algorithm [HKH+01] 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 groupwise-average, as well as online variants such as single pass are often used [FBY92]. Compared to non-hierarchical techniques, such algorithms trade off some of the quality of the produced clustering for speed [KR90], 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 fine-grained, and there are usually much fewer related paragraphs to cluster than documents in an IR application.

Simfinder uses a non-hierarchical clustering technique, the exchange method [Spä85], which casts the clustering problem as an optimization task and seeks to minimize an objective function Φ measuring the within-cluster dissimilarity in a partition $P = \{C_1, C_2, \ldots, C_k\}$

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

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 Φ, until convergence is achieved. Since it is a hillclimbing 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 filtered set is similar to at least another one, an additional constraint on the clustering algorithm to never produce singleton clusters is imposed.

Simfinder 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. Simfinder 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

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

where L is the observed number of links and P $(= n(n-1)/2)$ is the maximum possible number of links. Simfinder uses a non-linear interpolating function to account for the fact that, usually, $L \ll P$. The features selected for use with the log-linear regression model are word stem overlap, noun overlap, verb overlap, adjective overlap, WordNet class overlap, proper noun overlap, and LinkIT overlap. [HKH$^+$01] presents further details as well as an evaluation of Simfinder, and its application in two summarization systems.

2.3 A Flexible Framework for Simfinder

In Chapter 4 and Chapter 5, I present work that uses Simfinder 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, Simfinder 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. Section 4.1.2 provides detail about using Simfinder to compute similarity between machine translated Arabic sentences and English sentences, and Section 4.1.3 details a filtering step that I added which filters out sentences that are often not similar but that Simfinder labels as similar when using syntactically simplified sentences and machine translated input. The filter removes sentence pairs with a cosine similarity below the threshold of 0.17, which has a 76% accuracy of identifying sentences that humans judged as not similar despite having a high Simfinder similarity score.

Simfinder presents the starting point for my original work in the area of multi-lingual sentence similarity. SimFinderML, presented in full in Chapter 3, is a re-implementation of the ideas from Simfinder, 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 multi-lingual similarity, as different languages can have vastly different computational resources available. With SimFinderML it is possible to define the primitives and features to use at run-time in a configuration file, allowing one to use SimFinderML with different languages without modification of the program itself, which would not have been possible with Simfinder.

Chapter 3

Similarity in Multi-Lingual Texts: SimFinderML

SimFinderML (SimFinder Multi-Lingual) is a re-implementation of the English version of Simfinder, discussed in Chapter 2, that 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 multi-lingual text similarity, the approach I have taken to multi-lingual text similarity, the architecture of the SimFinderML system, and a description of the work required to add support for the Arabic language to SimFinderML.

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. [RJB00, BME99] Text similarity measures have also been used in question answering systems, again to indicate importance via identifying repetition of text, and to reduce redundancy. [BGMS04] 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 multi-lingual text similarity.

This chapter presents SimFinderML, a system for multi-lingual text similarity computation, which addresses the need for cross-lingual text similarity computation. SimFinderML allows experimentation at different levels of translation for similarity computation, and allows one to leverage natural language processing resources available in the foreign languages. As a case study for usage, I present two summarization systems that use SimFinderML for multi-lingual multi-document summarization. In particular, the CAPS system uses multi-lingual text similarity to build summaries that indicate the similarities and differences between documents in English and Arabic; this is a very useful application for intelligence analysts for whom there is too much foreign language text to read even if it were translated into English.

Another area in which multi-lingual text similarity metrics would be useful is in machine translation. A good multi-lingual text similarity metric could be used as a scoring function for a statistical machine translation system, although SimFinderML 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.

I compare my approach to using machine translation on the full document, and using the English version of Simfinder to compute similarity for Arabic–English sentences. Section 4.2.5 and 4.2.6 present an evaluation of using SimFinderML to cluster Arabic and English sentences compared to using English Simfinder to cluster manually translated Arabic and English sentences. Using machine translation and the English version of Simfinder gives an average precision of 66.5% and recall of 51.1%, while SimFinderML achieves 81.7% precision and 14.4% recall. While the recall is much lower, the precision is much higher, and approaches the performance of the gold standard in this case — manually translated Arabic clustered with English using English Simfinder. In many applications, high precision is more important than high recall, particularly in the application of summarization presented in Chapter 5.

The approach I have taken to similarity computation in SimFinderML 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

SimFinderML 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 SimFinderML's architecture, and how the above steps are carried out, while section 3.4 is an in-depth discussion about adding Arabic language support to SimFinderML and evaluation results. Support for other languages is briefly discussed in section 3.5.

3.2 Related work in Multi-lingual text similarity

The English version of Simfinder is the main influence on SimFinderML, but is not included in this section as it is a monolingual system. It is described in Chapter 2. SimFinderML takes the approach to text similarity introduced by English Simfinder 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 multi-lingual similarity computation. This section focuses on other work in multi-lingual text similarity.

3.2.1 Example based machine translation

Example based machine translation systems [Som99] 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 SimFinderML. SimFinderML's similarity metric is multi-lingual; in example based machine translation systems, a source language sentence to be translated is matched to other source language sentences, while in SimFinderML similarity is computed between all the input sentences, some of which are in different 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. SimFinderML computes similarity between full sentences, and allows

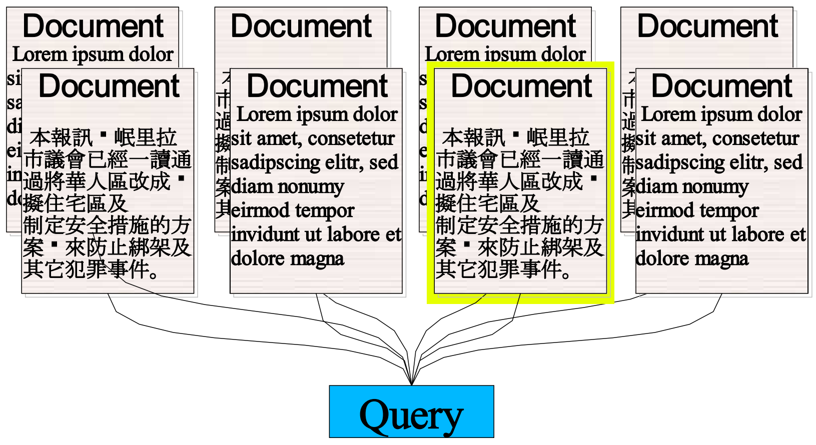

Document
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna
本報訊 · 岷里拉市議會已經一讀通過將華人區改成 · 擬住宅區及制定安全措施的方案 · 來防止綁架及其它犯罪事件。
Query

Figure 3.1: A CLIR query matching to one document from a collection of eight documents.

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.

3.2.2 Cross-Lingual Information Retrieval

Cross-language information retrieval (CLIR) is the task of searching documents in one language using queries from another language. The goal of Cross-Lingual Information Retrieval (CLIR) is to allow a user to search documents written in a language they do not understand using their native language. Figure 3.1 illustrates how a single query is used to find similar documents from a collection of documents in multiple languages. Cross-language information retrieval research has been undertaken in the past under the TREC CLIR track, and currently in the CLEF and NTCIR conferences [OD96, Che02, PHKJ01]. The most prevalent approach to cross-lingual information retrieval has been to translate the query into the target foreign language, and search the foreign language documents with the translated query using standard information retrieval techniques. Translation is often at the word level using general bilingual lexicons and specific domain-oriented lexicons, with translations from specific lexicons shown to improve performance when used in addition to general translations. Stemming and word normalization based on some sort of language-specific morphological analysis has been shown to improve performance. While word-level translation is generally used, it has been shown the phrase-based translation can improve results for some language pairs [CGJ01]. One problem area is translation of out-of-vocabulary terms, such as proper names, that are sometimes dealt with by transliteration or fuzzy-matching techniques.

Using general bilingual lexicons for translation can be problematic as many words have multiple senses with different meanings which makes the construction of a query in the target language difficult. Either all senses must be included, or specific senses of the translated words must be chosen. Methods which include alternate senses of translations, building "structured queries" that group translations of different senses of query terms as synonyms, are more effective than techniques which select only the most common sense of a translation, and can approach the effectiveness of monolingual information retrieval when coupled with specialized dictionaries [Pir98, SO00]. Many CLIR systems also use blind relevance feedback to expand query terms from the initial terms in the translated query by adding top terms from documents retrieved by a first retrieval run against the corpus. The full list of relevant documents is then returned by doing another retrieval using the original translated query augmented by the relevance feedback terms. Special attention is often paid to languages with compound nouns, such as German or Dutch, and translations into and out of these languages retain both compound translations and translations of each component of the compound.

The most related aspect of cross-language information retrieval to SimFinderML is that of query translation and query–document similarity computation. In CLIR, short queries are translated into a foreign language, and a similarity measure is computed between the query and documents. SimFinderML computes similarity between all sentences, not just a single query. Some of the same problems appear in both contexts, but since SimFinderML 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 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. SimFinderML deals with shorter units of text, and a spurious term is not as likely to appear. SimFinderML also uses bilingual lexicons for translation, and morphological analysis software (for Arabic and Japanese) or stemming to normalize words, and proper name identification and translation is implemented using BBN's IdentiFinder. SimFinderML's similarity 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. SimFinderML's use of multiple features and ability to compute similarity for fine-grained units of text set it apart from CLIR systems.

3.2.3 Statistical machine translation

Brown et al. [BCP+90] 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 SimFinderML would fit well into this sort of framework for similarity identification, since it mirrors the translation task well.

I have not adopted a completely statistical approach to similarity in SimFinderML. Such a system could model the similarity of two sentences given the feature similarity values between them as $\Pr(\text{sim}|F_1, F_2, \ldots, F_N)$. The approach taken in SimFinderML is to instead compute a predictor for similarity using a log linear regression model based on the feature values.

SimFinderML does use results from statistical machine translation community by taking advantage of models for learning probabilistic dictionaries. In implementing the Arabic–English portion of SimFinderML, I use a dictionary learned from an IBM model 3 style translation probability model, which helped improve results over translation by dictionary lookup alone. A distortion model might also help improve SimFinderML's results at finding sentences that are translations of each other, however, since SimFinderML 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 WordNet 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 [GC91], 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 [Mel97b] 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 subtree grammars.

SimFinderML 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. SimFinderML does make use of similar ideas though, especially in employing bilingual lexicons to anchor matches between the languages.

3.2.5 Bilingual Phrase Translation

Champollion [SMH96] is a system for translating text collocations from one language into another. Using a parallel corpus aligned at the sentence level, and a set of collocations from the target language, Champollion builds up translations of the collocation a word at a time. A final stage uses the target language corpus to determine the word order and if it is fixed or variable in the translated collocation. The accuracy of collocations translated in this manner was found to range from 65% to 78%.

Melamed's method for discovering non-compositional compounds in parallel text [Mel97a] 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.

The BICORD system [KT96] focuses on enriching definitions found in the Collins bilingual French—English and English—French machine readable dictionaries using evidence from a large English—French bilingual corpus. The system is able to return translations for an input word appropriate for a given sense that were not in the original dictionary, and also addresses the problem of single words resulting in multi-word translations.

SimFinderML could incorporate phrase translation by using one of the above systems above to identify phrases and their translations in the input documents as a primitive. SimFinderML itself does not incorporate any phrase identification or translation facilities, but these are easily added if such systems are available. For example, SimFinderML incorporates a named entity feature via BBN's IdentiFinder that identifies multi-word named entities in Arabic and English. I have not performed research on how to best translate these named entities, but instead take advantage of an Arabic–English machine translation system when available to match named entities across languages. In no such system is available, I use a component-based matching approach, although making use of a translation system specific to named entities would be preferable.

3.2.6 Proper noun phrase transliteration

Transliteration is the problem of translating terms and names between languages with different orthographical systems and sound inventories. The task is particularly important in translating news text between Japanese and English; in Japanese, many loan words and proper names from English are written in a phonetic alphabet (Katakana) that does not correspond to English spelling in a recoverable, 1-to-1 mapping. For example, the name "Charles Wang" would be written as チャールズ・ワング (Chya-ruzu Wangu). Collier and Hirakawa 1997 [CH97] describe a method to match Japanese Katakana strings to English strings by using a lookup table to map Katakana sequences to an intermediate representation. A dynamic programming algorithm is then used to compute the most likely matching English string for the intermediate representation. Their technique relies upon a high quality lookup table to map Katakana sequences to their intermediate representation, which was built by hand after inspecting over 200 examples. They show that their system performs better than two other systems when evaluated over 871 Katakana strings matching to 9742 potential English proper nouns.

Wan and Verspoor 1998 [WV98] describe a similar system for English-Chinese proper name transliteration. Their method is to transliterate words that can not be literally translated by first dividing the input words into syllables, normalizing the spelling, mapping from the syllables to pinyin, and mapping from pinyin to Chinese characters. Their system also relies on hand-built tables that maps English to an intermediate representation, and from that representation maps to the target language.

Knight and Grael 1997 [KG97] describe a method for Japanese-English transliteration built using a probabilistic model. The problem is broken down into five sub-problems, and they learn weighted finite state automata (FSA) to implement each stage. The first stage generates English word sequences, and is based on training data from a Wall Street Journal corpus, a name list, and a place-name gazetteer. They also have separate models for first or last names. The second stage converts English words into a phonetic representation, using the online CMU Pronunciation Dictionary as training data. The third stage converts the English phonetic representation into a Japanese phonetic representation. The authors created their own representation and training data from a Katakana-English glossary converted to the English and Japanese phonetic representations. They used EM training to generate the probabilities for the FSA that implemented the stage. The fourth stage is a small manually built finite state automata that converts from the Japanese phonetic representation to Katakana, and the fifth stage is a FSA that corrects for possible Optical Character Recognition (OCR) errors in the input. The five FSA are composed together to implement the complete system. The system is shown to perform very well when compared to native English speakers on the task of decoding Japanese Katakana back to the original English: out of 100 U.S. politician names, 64% were automatically decoded correctly, compared to 27% for the humans. One of the main benefits of this approach is that many of the components can be learned for a new language if suitable training data is available.

Stalls and Knight 1998 [SK98] show the flexibility of the above approach by porting it to Arabic-English name transliteration. The third and fourth stages are collapsed into a single stage that rewrites English phoneme sequences directly into Arabic writing, since there was no way to estimate the pronunciation of Arabic from the text.

SimFinderML would benefit greatly from integration of transliteration systems for language pairs that require it. It would be particularly useful for the current Arabic–English implementation, but unfortunately I did not have time to add a feature incorporating transliteration into the system.

3.3 SimFinderML Architecture

SimFinderML is designed to be a flexible, modular system. SimFinderML 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 3.2, there are six main modules in SimFinderML.

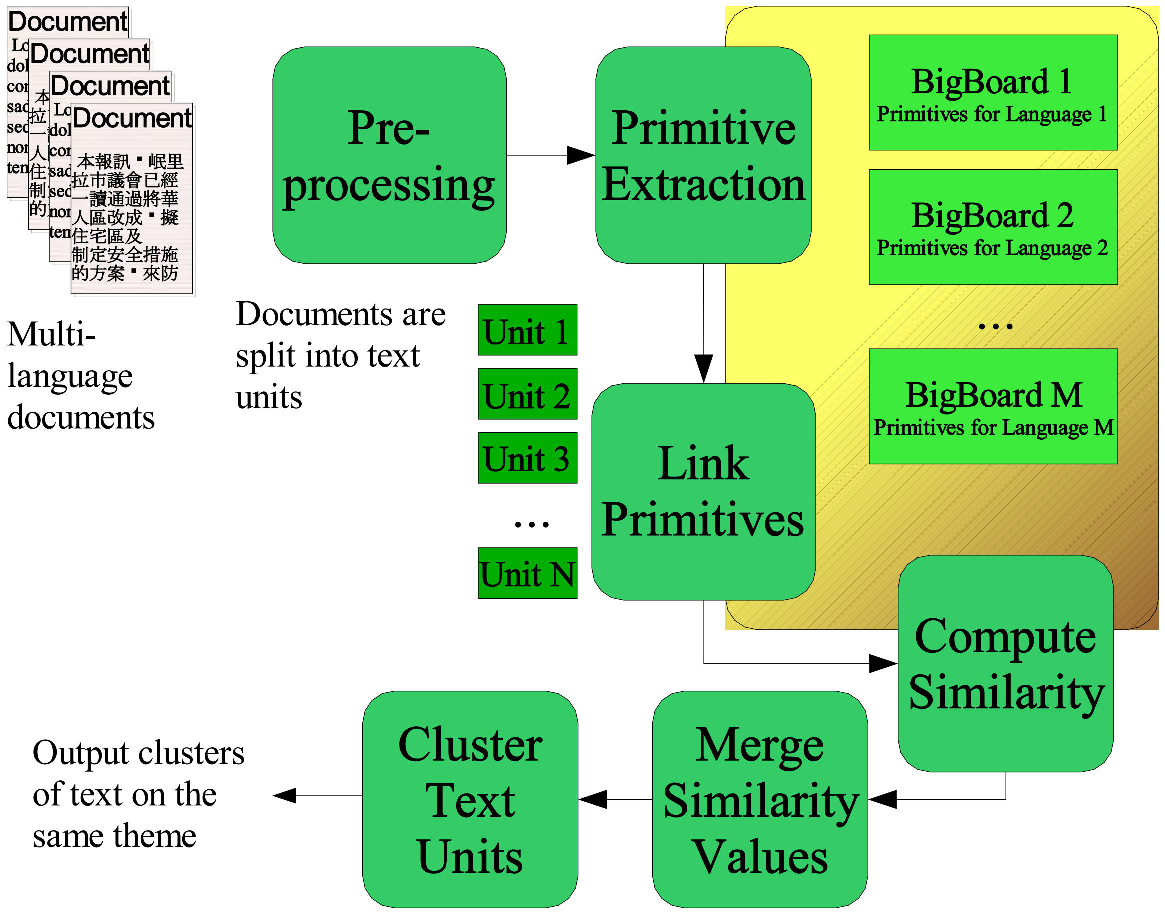

Document
本報訊・岷里拉市議會已經一讀通過將華人區改成・擬住宅區及制定安全措施的方案・來防
Multi-language documents
Pre-processing
Primitive Extraction
BigBoard 1
Primitives for Language 1
BigBoard 2
Primitives for Language 2
...
BigBoard M
Primitives for Language M
Documents are split into text units
Unit 1
Unit 2
Unit 3
...
Unit N
Link Primitives
Compute Similarity
Output clusters of text on the same theme
Cluster Text Units
Merge Similarity Values

Figure 3.2: SimFinderML 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.

SimFinderML 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 SimFinderML 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, Arabic, Chinese, and Japanese sentence segmentation using simple regular-expressions based classes, and an interface to the MXTerminator[1] [CRR97] 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 reflect 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, Arabic and Japanese morphological processing, and English and Arabic named entity recognition using existing tools[2] in the SimFinderML framework via this interface.

[1]ftp://ftp.cis.upenn.edu/pub/adwait/jmx/jmx.tar.gz

[2]BBN's IdentiFinder for English and Arabic respectively

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, WordNet classes, nouns, verbs, proper nouns, heads of noun phrases, adjectives, cardinals, and named entities. Token primitive extractors have also been implemented for Japanese, Chinese, and Arabic. The Chinese primitive extractor performs word segmentation using the mansegment.perl dictionary-based Chinese word segmentation program from the LDC. The Japanese primitive extractor first processes the text with Chasen [AM00], 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.

the following primitives are extracted:

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 BigBoard 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 reflect how similar sentences are based on tokens, nouns, verbs, WordNet, and named entity features.

When all primitives have been extracted, SimFinderML 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 WordNet primitives, for example, synonymy relationships not apparent at the token level can be identified.

3.3.3 Primitive Linking

Once all of the primitives have been extracted from the units, SimFinderML collects lists of which units contain the same primitives. The final phase before features are computed over the units is 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 SimFinderML 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; SimFinderML has shown that translation using bilingual lexicons or learned probabilistic dictionaries can results in high-precision for cross-lingual text similarity computation.

SimFinderML 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 WordNet feature described in Section 3.3.2. Words that are synonyms will be mapped into the same WordNet synset, and thus match other WordNet primitives for words in the same synset.

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, the assumption

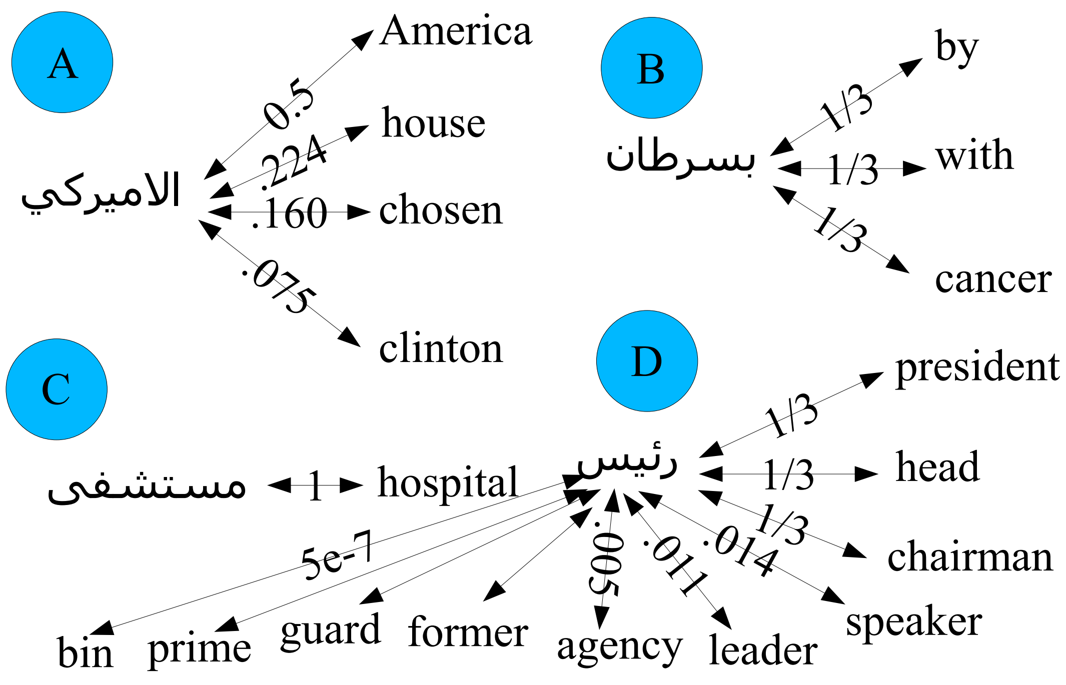


Figure 3.3: The primitive translation process

that words in the document set generally have the same sense can be made. Additionally, since translations of words from other languages often can help with sense disambiguation problems (see [DIS91]), a preference for high-recall linking seems to be justified.

SimFinderML supports some simple dictionary-based translation methods for linking primitives across languages. SimFinderML has support for three types of dictionary formats: a simple word to word format called the IDP dictionary format[3], the edict format[4] 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.

To enable usage of a particular dictionary at runtime, a setting is made in SimFinderML's configuration file that lists the dictionary to use for a given language pair, and what type of dictionary it is. During the primitive linking phase, the dictionary is loaded

[3]http://www.june29.com/IDP/IDPfileformat.html

[4]http://www.csse.monash.edu.au/~jwb/edict_doc.html

into memory[5], and SimFinderML queries the dictionary for translations of primitives in the source language. When a translation is found for which a target language primitive exists, a translational equivalence link between the source and target language primitives is made.

Each translational equivalence link has a strength, which represents SimFinderML's belief in the accuracy of the translation. The probabilistic dictionaries use a strength that is the same as the word translation probability, while the edict and IDP dictionaries always give the link between the primitives a strength of 1. In cases where a lookup returns more than one possible translation, the strength of the links made is evenly distributed between the target primitives. Figure 3.3 shows example links generated between Arabic and English token primitives using translations from both a probabilistic dictionary and from a morphological analysis program. Note that these are not probabilities; the strengths represent some quantitative level of belief of how good a translation is, and do not necessarily sum to 1.0 for all translations of a given primitive. The translation for token B, "hospital", contained only one entry in the Arabic morphological analysis program, so it has a strength of 1. The translations for token A come from both the morphological analysis program (American) and the probabilistic dictionary (house, chosen, and clinton.) Translations from the non-probabilistic dictionaries have a default weight of 1 unless there are multiple translations for the entry. The other translation from the morphological analysis program (the) was not included because it was filtered out as an unlikely translation candidate. Token D also includes multiple translations from the probabilistic dictionary, some with very low weights (not all weights are listed.)

As discussed in Section 3.4, SimFinderML also has language-specific translation mechanisms. These language specific mechanisms are not modularized like the rest of SimFinderML. With some simple coding, however, it is easy to add support for other languages and translation mechanisms. For Arabic, I have implemented translation based on the gloss output from the Buckwalter Morphological Analyzer, as well as an interface to a statistical Arabic machine translation system at IBM to translate named entities. The Arabic–English probabilistic dictionary uses the standard dictionary-based translation mechanisms though.

[5]There is also a setting to check for primitives as each translation from the dictionary is loaded, if the dictionary is too large to fit easily into main memory.

Figure 3.4 shows how translation occurs at the primitive level, and is not performed for each unit being compared. Translations for all primitives are computed first, only once for each primitive, and in the similarity computation stage those translations are made use of for each instance of a primitive. Additionally, Figure 3.4 illustrates the problem mentioned earlier of preferring recall over precision when linking related primitives. The Arabic token translated as “American” also has a link made to the English “Clinton” primitive, which erroneously introduces a small similarity value between the Arabic sentence (about King Hussein of Jordan’s cancer treatment in an American hospital) and President Clinton’s statement about the King’s influence. (Not all primitives and primitive translations are shown in this example.) However, due to the other relevant tokens in the English sentence, the similarity between this sentence and a truly similar sentence receives the highest score. Also note that in this example, cancer and American were both translated using lookup in the Buckwalter lexicon, giving them a weight of 1.0, while Clinton and hospital were translated by the learned probabilistic dictionary, and their weights reflect the strength of the relation from the dictionary.

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. SimFinderML 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 (BigBoard for each language), and all units containing the primitive or a linked primitive are compared against each other. An $N \times N$ 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 can not 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

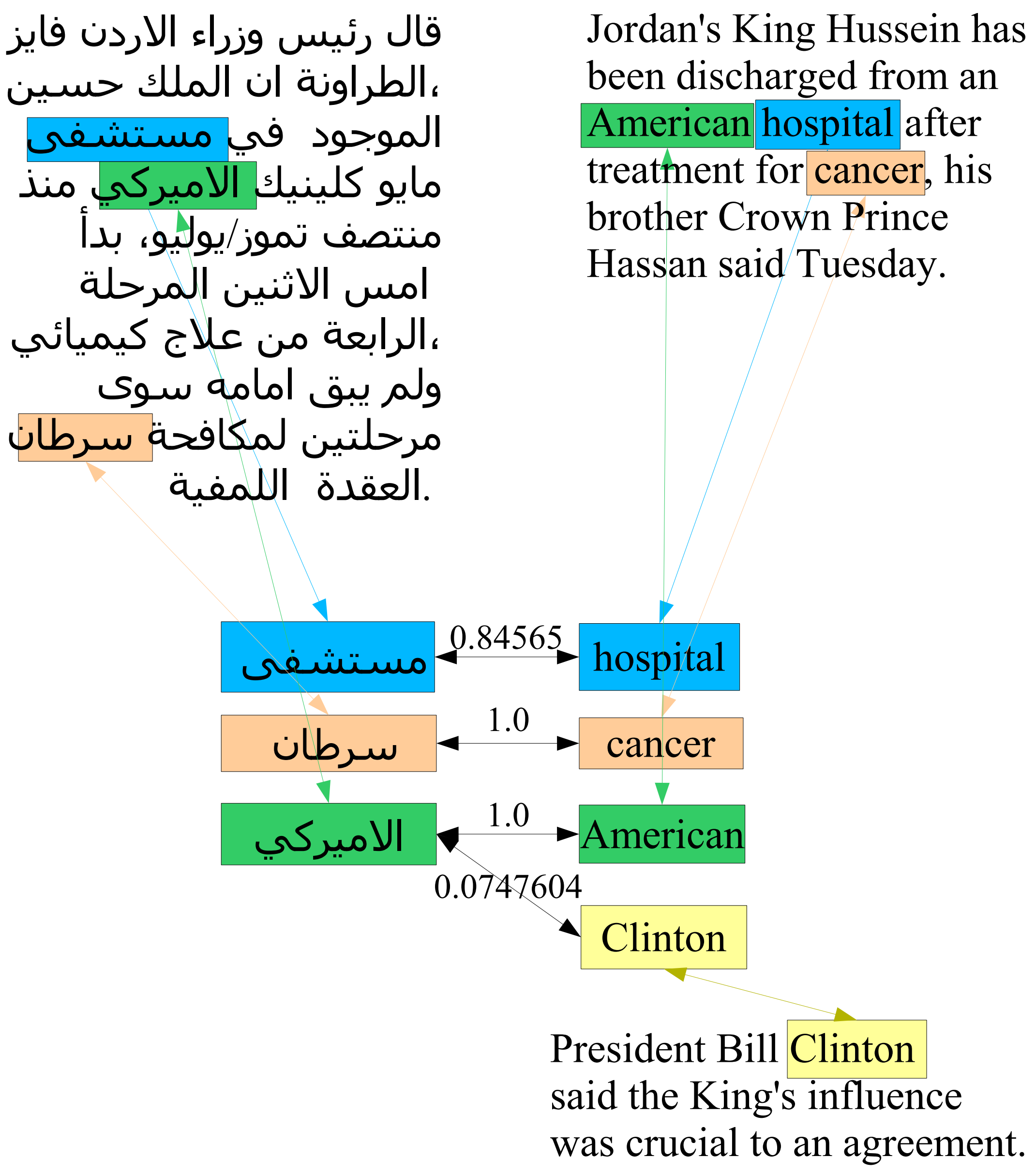

قال رئيس وزراء الاردن فايز الطراونة ان الملك حسين الموجود في مستشفى مايو كلينيك الاميركي منذ منتصف تموز/يوليو، بدأ امس الاثنين المرحلة الرابعة من علاج كيميائي، ولم يبق امامه سوى مرحلتين لمكافحة سرطان العقدة اللمفية.
Jordan's King Hussein has been discharged from an American hospital after treatment for cancer, his brother Crown Prince Hassan said Tuesday.
مستشفى
0.84565
hospital
سرطان
1.0
cancer
الاميركي
1.0
American
0.0747604
Clinton
President Bill Clinton said the King's influence was crucial to an agreement.

Figure 3.4: The primitive matching process using translations from a probabilistic learned dictionary

them with a default 0 similarity.

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 SimFinderML has been set to extract “token”, “verb”, and “WordNet” primitives, three features that compare the overlap on each primitive could be set up. In that case, SimFinderML would set up an $N \times N \times 3$ 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 an Arabic and English sentence.

The similarity of two units, U_1 and U_2 with primitives P_1 and P_2, with the strength of a link between primitive P_1a and P_2b given as $W_{P_{1a},P_{2b}}$ is defined as:

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

SimFinderML 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 WordNet 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 reflect the same ordering, e.g., Verb, WordNet and Verb, WordNet.)

3.3.5 Merging Feature Similarity Values

The goal of SimFinderML is to group textual units from multiple languages with similar meaning together. To do this, SimFinderML 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 SimFinderML 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 a $N \times N \times F$ 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 $N \times N \times F$ feature similarity matrix is converted into a $N \times N$ matrix such that each element contains a single value expressing the total similarity between the two units.

SimFinderML employs the same log-linear regression model used in English Simfinder [HKH$^+$01] to combine the evidence from multiple features into a single similarity value. A single similarity value between two units is computed using a logistic transform that weights contributions of each feature similarity value. In the log-linear regression model, a weighted sum of the input features is used as an intermediate predictor, η, which is related to the final response R via the logistic transformation $R = \frac{e^{\eta}}{1+e^{\eta}}$.

To use the log-linear regression model, weights must be learned for the linear combination of the features. For the English version of SimFinder, a training set of similar textual units has been developed by human judges who made a similarity decision over pairs of textual units. The multi-lingual version of SimFinderML requires the same sort of training data. Another Japanese speaking annotator and I marked a small set of documents in Japanese and English, labeling similar sentences (both English–English, Japanese–Japanese, and English–Japanese.) Each of the annotators read all articles in the training document sets, containing both Japanese and English articles, and listed sentences in both Japanese 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 Simfinder, when training a model for Arabic–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, is described in Section 3.4.4.

Once a training set of similar sentences has been determined, a set of feature values for the textual units is generated by running SimFinderML and computing all of the features over the pairs of textual units. The sets of feature values are then aligned to the similarity judgments, and a log-linear regression is performed that determines an exponent for each of the feature values to best synthesize a similarity score that matches the similarity judgments (either similar, 1, or not similar, 0.) Then the logistic transform can be used to output a similarity value from 0 (not similar at all) to 1 (highly similar) based on the feature similarity values and the learned feature weights.

3.3.5.1 Challenges for Multi-Lingual Feature Merging

While the above approach is tenable in the monolingual case where training data is available, there are additional problems in the multi-lingual case. The features that are available for two textual units from different languages are usually different. For example, for English in SimFinderML there are multiple primitives (part-of-speech based, stemmed tokens, word net classes, etc.) while only the token primitive has been implemented for Chinese. When calculating the final similarity value between an English and a Chinese 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, (as discussed in Section 3.3.3) the number of features compatible between units in different languages will change as well. Since different 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 Arabic–English version of SimFinderML, which is from the Multiple Translation Arabic corpus from the LDC[6]. As with in the monolingual English case,

[6]http://wave.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2003T18

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: SimFinderML 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 multi-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 as described in section 3.3.5. The clustering component uses the optimization-based method described in Hatzivassiloglou et al. 2001 [HKH$^+$01]. 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 Simfinder. 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 $c = m + \left(\frac{n}{2} - m\right)\left(1 - \frac{\log(L)}{\log(P)}\right)$ where L is the observed number of links between units based on their similarity being above a threshold, and P $(= n(n-1)/2)$ is the maximum possible number of links. A non-linear interpolating function is used to account for the fact that usually $L \ll P$. See Section 2.2.2 for more details.

3.4 Creating an Arabic–English version of SimFinderML

SimFinderML has been designed from the start to easily allow for adding support for non-English languages. SimFinderML currently has support for Arabic, Chinese, English, French, Japanese and Spanish. For Chinese and Japanese, simple classes for word segmentation and morphological processing were implemented, and support for bilingual dictionaries was added. The French and Spanish support was extremely easy to add, requiring no modification of the SimFinderML program at all; due to the dictionary support added for Chinese and Japanese, simply configuring the appropriate bilingual dictionaries and tokenizers at run-time is all that is required for basic support. I have spent the most time working on SimFinderML's support for Arabic, and in this section I document the features that I have implemented, the training that I have performed, and the results for Arabic–English similarity compared to full and partial translation.

3.4.1 Arabic-language features

I extract two features from Arabic-language text: *word* features, and *named entity* features.

Word features are extracted from the Arabic text using a simple regular expression that separates out words based on white space. I have chosen this simple approach over first applying morphological analysis and extracting, for example, word stems. In Arabic, words are a combination of consonant templates with (possibly elided) vowels. Morphological analysis on Arabic words typically leads to multiple possible interpretations for each word. I avoid making the decision about which interpretation to use by leaving the words in their unanalyzed form. The second reason I do not perform morphological analysis first is that the learned probabilistic dictionary that I use for translation does not contain morphologically analyzed tokens; it also uses the written form of the words directly from the text. Using this dictionary, some of the ambiguity is resolved based on the translation usage learned from the parallel corpus. There are two translation mechanisms that I use with Arabic tokens, the first uses the Buckwalter morphological analysis program, which often finds more than one acceptable parse for each token. If the text is pre-processed, I run the risk of choosing an incorrect parse, and losing that information for later use in the translation stage.

The named entity features are extracted from the text using the Arabic version of BBN's IdentiFinder^{TM}named entity extraction software. BBN's IdentiFinder finds 24 different types of named entities in three main categories: names, times, and numbers. The named entities that BBN IdentiFinder marks are extracted from the text and added to each Unit as a primitive in a class that keeps track of the kind of named entity that was found.

3.4.2 Arabic to English Translation Facilities

The similarity between two text Units is computed using a loglinear model to combine the values from multiple features into a single similarity value. When computing the similarity between different languages, a form of translation is used to try to generate matches between the primitives in each language. The form of translation can be specific to the type of Primitive.

3.4.2.1 Word feature matching

In Arabic, the basic primitive translation used is dictionary lookup in the Buckwalter morphological analyzer[7] available from the LDC. A match is made between an Arabic primitive and an English primitive if there is a non-stopword English translation in the Buckwalter lookup that matches the English primitive, with the strength of the match determined by the total number of English translations for the Arabic 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. See Figure 3.3 and the discussion in Section 3.3.4 for an illustration of the translation method.

3.4.2.2 Using a probabilistic dictionary

The second translation method I use for Arabic is lookup in a probabilistic translation dictionary. SimFinderML 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 Arabic tokens to English tokens with probability for the likelihood of the translation. Section 3.4.3 describes the process used to learn the probabilistic dictionary used in SimFinderML for Arabic–English word lookup. When using the probabilistic dictionary, an Arabic 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

[7]http://wave.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2002L49

with the Buckwalter 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 IdentiFinder$^{\text{TM}}$for both English and Arabic. A match is found between IdentiFinder primitives using either dictionary lookup via the Buckwalter 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-stopword overlap between the English and glosses for the Arabic word, a match is made. No disambiguation is performed, nor is locality of the text taken into account.

3.4.3 Learning a probabilistic Arabic–English dictionary

To improve word translation I learned a probabilistic dictionary between Arabic and 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 Arabic English Parallel News Part 1 Corpus[9] from the LDC. The corpus contains Arabic 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, 2M Arabic 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 files 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 Buckwalter translation mechanisms have been used for various experiments reported on in section 4.2.

[8]In this case, IBM's statistical Arabic machine translation system circa 2004.

[9]http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2004T18

3.4.4 Feature Merging Model Training Data

As explained in section 3.3.5, SimFinderML uses a log-linear model to generate the single similarity value between two sentences. In the Arabic and 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 Arabic–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 Arabic[10] corpus from the LDC as my training corpus. For each of the 141 Arabic 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 SimFinderML over the pair of documents. This resulted in 141 files 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 Arabic–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. The training uses 4-fold cross-validation to train over $\frac{3}{4}$ of the corpus and test over $\frac{1}{4}$ of the corpus, switching through all 4 slices for testing.

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

[10]http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2003T18

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 IdentiFinder features with Buckwalter + Probabilistic, Probabilistic, and Buckwalter translation.

system, and a link between the Arabic primitive and the target English translation is made. The Arabic 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 IdentiFinder features, using the combination of probabilistic and Buckwalter translation.

When using both the Token and IdentiFinder features, in all cases using Probabilistic translation combined with Buckwalter 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 Buckwalter translation alone on the training data at every threshold, and is statistically significant at $p = 0.01$ for both precision and recall. Combining Buckwalter 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 Buckwalter and Probabilistic, Probabilistic, and Buckwalter 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 Buckwalter 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 Buckwalter 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 Buckwalter translation provides the best performance. For all but one threshold (0.8) using the combination of the token and BBN

IdentiFinder features performs as well or better than using tokens alone. The addition of the IdentiFinder feature statistically significantly improved precision and recall over the token feature alone using probabilistic and Buckwalter 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

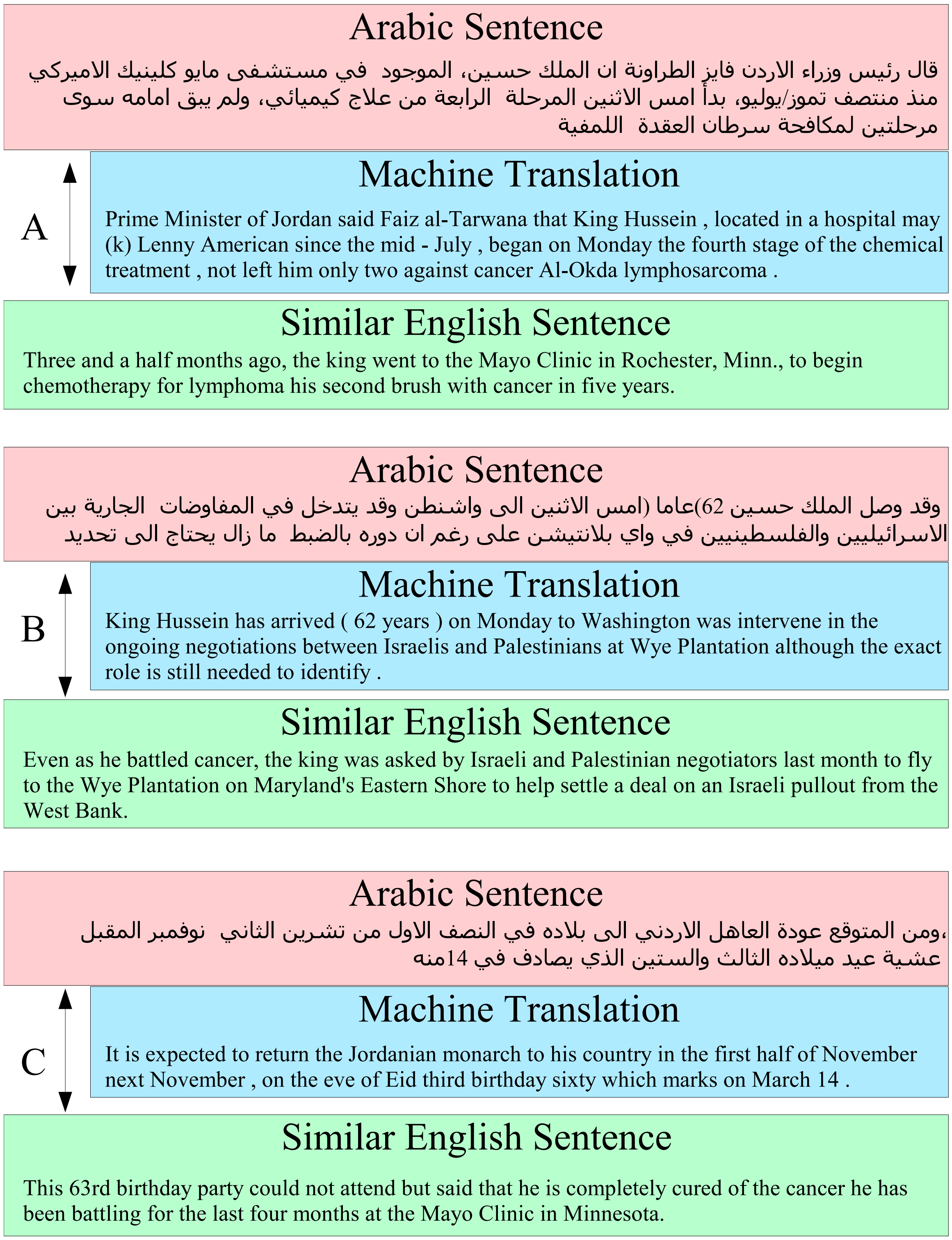
The learned model was added to SimFinderML, and SimFinderML was re-run over the training data. Each Arabic sentence was compared to the most similar English sentence as predicted by SimFinderML's. 89.00% of the sentences were correctly mapped back to their aligned counterpart. The average similarity of the most similar English sentence was 35.98%, but this rose to 37.51% when looking at only correctly mapped sentences (vs. 23.67% for incorrectly mapped sentences.)

Figure 3.5 shows three examples of similar Arabic and English sentences found by SimFinderML. In the figure, machine translations of the Arabic sentences are provided in the blue boxes as a convenience to the reader, however, SimFinderML only uses the Arabic and English sentences to perform the similarity computation.

3.5 Porting to other languages

The modular architecture of SimFinderML makes it very easy to add support for other languages. While I have not run extensive experiments with languages other than Arabic and English, I have used SimFinderML with other non-English languages and run small experiments with them. To date, I have run SimFinderML with the following languages:

1. English
2. Arabic
3. Chinese
4. Japanese
5. French



Arabic Sentence

قال رئيس وزراء الاردن فايز الطراونة ان الملك حسين، الموجود في مستشفى مايو كلينيك الاميركي منذ منتصف تموز/يوليو، بدأ امس الاثنين المرحلة الرابعة من علاج كيميائي، ولم يبق امامه سوى مرحلتين لمكافحة سرطان العقدة اللمفية

A

Machine Translation

Prime Minister of Jordan said Faiz al-Tarwana that King Hussein , located in a hospital may (k) Lenny American since the mid - July , began on Monday the fourth stage of the chemical treatment , not left him only two against cancer Al-Okda lymphosarcoma .

Similar English Sentence

Three and a half months ago, the king went to the Mayo Clinic in Rochester, Minn., to begin chemotherapy for lymphoma his second brush with cancer in five years.

Arabic Sentence

وقد وصل الملك حسين 62)عاما (امس الاثنين الى واشنطن وقد يتدخل في المفاوضات الجارية بين الاسرائيليين والفلسطينيين في واي بلانتيشن على رغم ان دوره بالضبط ما زال يحتاج الى تحديد

B

Machine Translation

King Hussein has arrived (62 years) on Monday to Washington was intervene in the ongoing negotiations between Israelis and Palestinians at Wye Plantation although the exact role is still needed to identify .

Similar English Sentence

Even as he battled cancer, the king was asked by Israeli and Palestinian negotiators last month to fly to the Wye Plantation on Maryland's Eastern Shore to help settle a deal on an Israeli pullout from the West Bank.

Arabic Sentence

ومن المتوقع عودة العاهل الاردني الى بلاده في النصف الاول من تشرين الثاني نوفمبر المقبل، عشية عيد ميلاده الثالث والستين الذي يصادف في 14منه

C

Machine Translation

It is expected to return the Jordanian monarch to his country in the first half of November next November , on the eve of Eid third birthday sixty which marks on March 14 .

Similar English Sentence

This 63rd birthday party could not attend but said that he is completely cured of the cancer he has been battling for the last four months at the Mayo Clinic in Minnesota.

Figure 3.5: Examples of similar Arabic–English sentences found by SimFinderML. Machine translations of the Arabic sentences are provided for the reader, but only the Arabic and English is used for the actual matching.

With each of these language pairs I used aligned parallel corpora as training data for the feature merging models, similar to the process described in Section 3.4.4. I created a French–English feature merging model using token and Fastr primitives based on a subsection of the sentence-aligned Hansards French-English corpus.

Some East Asian languages such as Chinese and Japanese do not use spaces to separate "words", so an additional pre-processing step is needed tokenize the text into word-like units. SimFinderML has a well-defined interface for text pre-processing, so I was able to write a simple tokenizer for Chinese that uses a Mandarin Chinese word segmentation program available from the LDC[11] for segmentation. Any segmentation system can easily be used by creating a wrapper class to call the system. A freely available Chinese–English dictionary in the Edict format, CEDICT[12] was added to the configuration without requiring any new code. To add support for a dictionary format that SimFinderML does not currently support is also quite easy, requiring only a simple class that can parse the dictionary and implement some simple lookup methods. With these minor changes, SimFinderML is able to run with Chinese and English text with token-level translation.

The Japanese language, similar to Chinese, also does not segment words using space. I implemented a class that calls Chasen[13], a Japanese morphological analysis program, to segment words and retrieve base forms of verbs and adjectives. The Japanese–English Edict[14] dictionary was added to the runtime configuration, and SimFinderML was able to support simple token-level translation. A Japanese–English bilingual annotator and I read 6 articles sets containing Japanese and English documents, and manually created clusters of sentences with similar meaning. Using the hand-selected sentences for similarity, I created a feature merging model using token translation with the Japanese Edict. This feature merging model was only created with a small number of example sentences, and a full implementation of Japanese–English SimFinderML should include more features and more training data – perhaps using an aligned corpus, as was done for Arabic (see Section 3.4.4.)

[11]http://www.ldc.upenn.edu/Projects/Chinese/segmenter/mansegment.perl

[12]http://www.mandarintools.com/cedict.html

[13]http://chasen.aist-nara.ac.jp/hiki/ChaSen/

[14]http://www.csse.monash.edu.au/~jwb/j_edict.html

To add support for French, I simply added a French–English dictionary for translation, and used the existing sentence and word segmentation systems implemented for English. One interesting feature added to SimFinderML for French and English is support for noun phrase variation. The Fastr system[Jac99] is a parser for term and variant recognition which can identify noun phrases and syntagmatic variants of noun phrases. Fastr is implemented for a variety of languages, including French and English. I added a new Fastr feature extractor to SimFinderML which parses documents using Fastr, which extracts noun phrases to a canonical form, relating variations of noun phrases together. Fastr is run over both English and French, and the primitive translation phase attempts to match Fastr primitives by compositional translation (similar to how IdentiFinder primitives are matched using glosses from the Buckwalter morphological analysis program in 3.4.2.3.)

3.5.1 Extracting article text from web pages

In order to work with SimFinderML in multiple 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 NewsBlaster. I investigated methods for crawling and extracting article text in multiple languages, as well as clustering English and non-English text within the NewsBlaster 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 multiple 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 70,671 bytes, but the actual article text of the web page was only 6,887 bytes. The remaining 60k was extraneous formatting information, navigation links, advertisements, and so on.

The previous approach to extracting article text in Columbia NewsBlaster used regular expressions that were hand-tailored to specific web sites. Adapting this approach to new web sites was difficult, since a human has to build regular expressions for each new site. Additionally, if the site layout changed, regular expressions would often need to be re-written. This approach is also difficult to adapt to foreign languages sites; in addition to requiring a human to write regular expressions, technical problems often arise when dealing with regular expressions in different character encodings.

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.

Training data for the system is generated using a GUI that allows a human to annotate text candidates with one of fives labels: “ArticleText”, “Title”, “Caption”, “Image”, or “Other”. The “ArticleText” label is associated with the actual text of the article which we wish to extract. At the same time, we try to determine document titles, image caption text, and image blocks in the same framework. The “Title” tag is used to annotate the title of an article, while “Image” and “Caption” are used to indicate images and their captions. Columbia NewsBlaster extracts and categorizes the images based on the caption text, and includes images in the summaries and an image browser. “Other” is a catch-all category for all other text blocks, such as links to related articles, navigation links, ads, and so on. The training data is used with the machine learning program Ripper [Coh96] to induce a hypothesis for categorizing text candidates according to the features. The article extractor module then uses the hypothesis to predict a category for each text block based on the features of the text block. This approach has been trained on web pages from sites in English, Russian, and Japanese as shown in Table 3.3, but has been used with sites in English, Russian, Japanese, Chinese, French, Spanish, German, Italian, Portuguese, and Korean.

Language	Training set	Precision	Recall
English	353	89.10%	90.70%
Russian	112	90.59%	95.06%
Russian	English Rules	37.66%	73.05%
Japanese	67	89.66%	100.00%
Japanese	English Rules	100.00%	20.00%

Table 3.3: Article extractor performance for detecting article text in three languages.

The English training set was composed of 353 articles, collected from 19 web sites. Using 10-fold cross-validation, the induced rules classify into the article text category with a precision of 89.1% and a recall of 90.7%. Performance over Russian data was similar, with a precision of 90.59% and recall of 95.06%. I also show performance using the English rules to extract news from the Russian data set to test the hypothesis that rules tailored for each language would improve performance. As expected, the English rules resulted in poor performance over the Russian data - precision was 37.66% and recall was 73.05%. Similarly, in Japanese recall fell from 100.00% to 20.00% when using the English rules. Precision remained high though, as many fewer article text blocks were extracted. Using rules learned from data for each language clearly improves performance over using a single ruleset (the English rules in this case.) Adding new sites to this system is easy; a human annotates web pages using the GUI, and a new categorization hypothesis is learned from the new training data.

The article extraction system uses a back-off approach to dynamically choose the ruleset to use for article extraction. Since using a ruleset for a specific language or web site improved performance, I implemented a back-off strategy to select the best ruleset to use. Rulesets are used in this order: specific to a website, specific to a language, and the default ruleset. One of the more practical aspects of this back-off approach is that it is very easy to re-train the system for a specific website: simply create training data just for the website, train a hypothesis for it, and drop it in to the system without making any changes to the other rulesets.

3.5.2 Using simple document translation for multilingual clustering

Once a suitable set of articles can be extracted from the web into text files, it is necessary to cluster the articles into topics for use with SimFinderML and multilingual multi-document summarization. The document clustering system that used in Columbia NewsBlaster [HGM00] 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 multiple 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 for Japanese and Russian. 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 babelfish.[15]) For languages where we do not have a simple translation mechanism available, we use the babelfish web interface to the SYSTRAN translation engine. The translated documents are then clustered as in the monolingual English version of NewsBlaster.

3.5.3 Multilingual 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 multilingual 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, 7 clusters contained documents in both English

[15]http://babelfish.altavista.com/

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 multilingual 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 fit post-filtering requirements.)

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

3.5.4 Japanese Performance

I tested SimFinderML's performance on Japanese to show that the same techniques and approaches used for English are applicable to other languages. To do this, I collected three sets of articles on different topics[16] and had a native Japanese-speaking judge[17] annotate the sentences in the article sets for similarity. Statistical models for combining feature values were generated from the training sets with 10% of the training examples held aside, to ensure some unseen cases in the test set.

Table 3.4 shows the precision, recall, and number of features used for four different models used for feature combination in Japanese sentence similarity identification. The features used are derived from the Chasen morphological analysis program that SimFinderML calls,

[16]5 articles on the February 2003 Space Shuttle Columbia explosion (63 sentences total), 5 articles on Secretary of State Colin Powell's February 6th, 2003 address to the UN (71 sentences total), and 5 articles on the Japanese government's response (44 sentences total).

[17]Not the author.

SFML Model and # of Features		Precision	Recall
random performance	–	5.02%	50.0%
tokens	1	42.1%	9.375%
nouns	1	52.1%	13.5%
proper nouns, common nouns	2	42.6%	7.3%
large model	9	53.3%	9.375%

Table 3.4: SimFinderML Performance for Japanese sentence similarity detection using different models.

as described in 3.5. The baseline of random performance (choosing similar or dissimilar with equal probability) is given as a point of reference, while the first model containing only token overlap is used as a baseline, achieving 42.1% precision and 9.375% recall. The model containing only nouns achieves 52.1% precision and 13.5% recall, while a model which differentiates between proper and common nouns curiously performs worse with 42.6% precision and 7.3% recall. The best performing model contains 9 features (tokens, all nouns, all verbs, independent verbs, proper nouns, common nouns, Japanese "suru" nouns, counter affixes, and cardinals), and has a recall of 53.3% precision and 9.375% recall.

3.6 SimFinderML Conclusion

In this chapter I presented SimFinderML, a system for computing text similarity between text units (sentences, in this case) in multiple languages. SimFinderML has been implemented to work with English, French, Chinese, Japanese, and Arabic. For English and Arabic text, 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 0.7 over the training data.

Continuing with the work first started in the English version of Simfinder, SimFinderML 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. SimFinderML 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 SimFinderML, as shown by the integration of the Arabic version of BBN's IdentiFinder for a named entity primitive in SimFinderML.

SimFinderML 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, SimFinderML 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. SimFinderML is easily ported to other languages, and a strong implementation has been developed for Arabic and English.

Chapter 4 uses SimFinderML to find similar sentences in Arabic and English text, and compares to performance using English Simfinder with machine translated text. Chapter 5 presents two summarization systems that use text similarity in novel applications of multi-lingual multi-document summarization.

Chapter 4

Finding Similar Arabic-English Sentences

In Chapters 2 and 3, I introduced two systems for computing text similarity, Simfinder and SimFinderML, for English monolingual similarity and multi-lingual similarity respectively. In this chapter, I compare how well these two systems perform for a task that is crucial in the similarity-based summarization systems that I have built. In Section 4.1 I examine the task of finding a similar English sentence for an Arabic sentence using machine translated text. I perform an evaluation that examines whether replacing machine translated Arabic sentences with automatically identified similar English sentences improves a summary of the Arabic text or not.

After I examine sentence similarity by evaluating individual sentence pairs for similarity, I evaluate at the level of sentence clusters formed using different similarity metrics. In Section 4.2 I examine different methods of similarity computation for Arabic–English sentence clustering. I perform the task using translation of varying sophistication, starting with full machine translation and English Simfinder, and moving to translation at a word-by-word level only with SimFinderML for those languages for which extensive translation tools are not available.

4.1 Sentence level evaluation

This section examines the task of finding English sentences that are similar to machine translated Arabic sentences. Section 3.2 presents related work in the area of multi-lingual similarity metrics, but this section focuses specifically on finding similar English sentences to machine translated sentences.

Assuming that machine translation tools are available for a given language pair, in this case Arabic and English, I examine whether tools can be developed that find similar sentences between Arabic and English text. The first step is to translate the Arabic sentences with a machine translation system, the second is to possibly syntactically simplify the English text, and the third is to compute similarity to the simplified English text. I present an evaluation of whether replacing the machine translated Arabic sentences with similar English sentences makes an improvement in the context of summary evaluation based on human judgments, and compare the results of using Simfinder to a cosine-similarity metric. I also investigate the effect of using sentence simplification software on the Arabic text for similarity detection.

In this section, I compare different approaches to identifying similar sentences between machine translated Arabic text and English text. I will test the hypothesis that:

1. Replacing machine translated Arabic sentences with similar English sentences in a summary improves the summary
2. Syntactic sentence simplification on English text improves similarity results
3. Chunking machine translated Arabic text improves similarity results
4. Simfinder computes more accurate similarity values for finding similar sentences than a cosine-based metric

Of the four hypotheses presented above, hypothesis 1, 3, and 4 are examined in this chapter, while results on hypothesis 2 are presented in Section 5.3. I report on experiments that use a summarization system to evaluate whether simplifying the input English text improves summaries over not using simplification at all, and which forms of simplification are most beneficial. The summaries are evaluated using ROUGE, a system that automatically scores system summaries based on multiple reference summaries. Due to the automatic nature of the evaluation, I am able to test both syntactic simplification with pronoun resolution and syntactic simplification without pronoun resolution compared to not using simplification at all. The results indicate that using syntactic simplification only is an improvement over not using simplification or simplification with pronoun resolution.

In this chapter I test hypotheses 1, 3, and 4 based on results from two user studies. As the user studies are labor intensive, requiring a human to make similarity judgments for each pair of sentences tested, I opt to always use syntactically simplified English sentences, which performs best in the summary evaluation described in Section 5.3 and intuitively should allow for matching information at a more fine-grained level.

4.1.1 Finding Similar English Sentences

This section details an evaluation of the process of identifying similar sentences between machine translated Arabic sentences and English sentences. The goal of this section is to show that when using machine translated text, one can identify an English sentence that is a good replacement for the Arabic sentence, in that it improves a summary of the Arabic documents, or we can alternatively identify when we do not have any good replacement sentences for the Arabic sentence. By replacing machine translated sentences with English sentences one runs the risk of introducing false information that is not a good representation of the content of the Arabic sentences. The experiments in this section are designed to see if one automated methods can be used to minimize these sorts of errors by identifying only sentences that make good replacements in the context of summarization.

I perform two evaluations that examine whether the English sentence improves upon, or detracts from the overall meaning and understandability of the machine translated sentence being replaced. To address hypothesis 1, whether replacing machine translated Arabic sentences with a similar English sentence improves a summary, the first evaluation examines IBM machine translated Arabic sentences replaced with sentences from syntactically simplified related English sentences. Two methods are used to compute similarity for the sentence

1. improves the summary without changing the meaning
2. improves the summary but changes the meaning
3. is no better or worse than before
4. degrades the summary without changing the meaning
5. degrades the summary and changes the meaning

Figure 4.1: Judgment scale used for sentence-level evaluation one and two.

replacements, Simfinder, and a simple cosine-based similarity metric. The cosine similarity metric is a standard technique used to compute similarity by creating word vectors for each sentence. A word vector is a vector of length N where N is the number of different tokens in the document set, and each entry in the vector is the count of how many times each type appeared in the sentence. The cosine similarity measure then computes similarity as the angle between the word-vectors for the two sentences in vector space. Human judges are asked to rate similar sentence pairs as computed by the Simfinder and cosine metrics on the five point scale shown in Figure 4.1 to determine whether suitable sentence replacements can be automatically made.

The second sentence evaluation addresses hypothesis 3, whether chunking Arabic text can improve replacement, and examines chunked IBM machine translated Arabic sentences and syntactically simplified related English sentences. The machine translated Arabic sentences are split using the TTT[1] tagging software, splitting on verb groups and copying the previous noun and verb group. In addition to the 1-5 scale, each sentence pair is labeled as to whether the replacement sentence is "related" or "not related" to the machine translated sentence, where related is an indicator that the sentences are on the same topic. The "related" or "not related" judgment is made to help identify primarily short sentences that result from the Arabic sentence chunks that erroneously receive high similarity values. I later use this feature to learn filters to improve performance over chunked Arabic text.

I use data from the Document Understanding Conference 2004 [OY04] corpus, which contains machine translated versions of Arabic documents, and English documents on the same topic. There are 24 document sets, each with approximately 10 Arabic and 10 En-

[1]http://www.ltg.ed.ac.uk/software/ttt/

Pair	2-class Kappa	2-class % Agreement
devans noemie	0.553808454155605	83%
devans smara2	0.466275659824047	73%
devans delson	0.187394103693663	60%
devans as372	0.395131086142322	70%
devans ani	0.276960048062481	65%

Table 4.1: Agreement between annotator devans and other annotators on sentence-replacement evaluation with full length machine-translated sentences.

glish documents in the set. The corpus is further described in Section 5.2. For the 24 document sets in the DUC2004 corpus, the Arabic documents in the set are summarized using DEMS [SNM02], and for each sentence in the summary, the top three most similar sentence replacements are evaluated by humans on the five point scale with reference to the understandability of the sentences, and the content with respect to the final summary. The scale is shown in Figure 4.1.

Six humans performed the evaluation using full machine translated sentences, with each sentence pair being marked by two evaluators. Average agreement between evaluators was 70% on whether replacement improved (category 1 and 2) or degraded the summary (category 3, 4, and 5), with a Kappa [Coh60] of 0.41. Table 4.1 shows agreement and Kappa scores for pairs of annotators on the two class, whether the replacement improves or degrades the summary, replacement. All examples were marked by the author (devans in Table 4.1) while the other five participants each labeled a portion of the data such that all sentences are labeled by two annotators. Both devans and as372 are familiar with the output of machine translation systems, while the other annotators do not regularly work with machine-translated text. Surprisingly, devans and as372 do not have the highest agreement among annotator pairs. Looking at the raw histogram, as372 rates similarly to noemie, while devans rates more similarly to ani or delson. Noemie rates most critically, followed by as372 and devans, while the other three raters use the “is no better or worse than before” rating more heavily than negative ratings. As the annotators were not able to achieve high agreement, for the second part of the experiment using chunked Arabic text, only devans annotated the data.

The next two sections examine how well Simfinder, the cosine similarity metric, and Simfinder with a cosine filter perform at identifying syntactically simplified English sentence replacements for chunked or unchunked machine translated Arabic sentences that improve the summary.

4.1.1.1 Chunking Machine Translated Arabic Text

To determine whether chunking machine translated Arabic text into smaller units would improve scores for finding replacement sentences in a summary, I first needed to develop an algorithm to chunk machine translated Arabic text into smaller units. Chunking the sentences well is hard because often machine translation system output is ungrammatical, and difficult to parse. Instead of using a full parser and splitting the Arabic sentences based on syntactic structure, I use methods that only required a part of speech tagger, which is more tolerant of a wide-range of input.

I investigate two Arabic chunking algorithms, both using the TTT tagging software to tag the Arabic text for verb and noun groups. The first method, called *copy-split*, splits each sentence on verb groups, and copies the previous noun group and verb group to the next sentence. The intuition is that copying over the previous verb group and noun group will keep the antecedent of a following verb group in the newly-created sentence fragment.

It splits the sentence:

> of the evidence to resolve the benefit presidential stressed the press Lebanese issued today that Prime Minister Rafic Hariri contacted yesterday evening by telephone the Imad Lahoud from Saudi Arabia to say to him " Sir President .

into 6 sentences:

> of the evidence to resolve
>
> the evidence to resolve the benefit presidential stressed
>
> the benefit stressed the press Lebanese issued today

the press Lebanese issued today that Prime Minister Rafic Hariri contacted yesterday

Prime Minister Rafic Hariri contacted yesterday evening by telephone the Imad Lahoud from Saudi Arabia to say

Saudi Arabia to say to him " Sir President .

The second chunking method, called *simple split*, simply splits a sentence on verb groups, certain conjunctions (and, not, but, yet), and the "," character. That method splits the same sentence into:

of the evidence to resolve

the benefit presidential stressed

the press Lebanese issued today

that Prime Minister Rafic Hariri contacted yesterday

evening by telephone the Imad Lahoud from Saudi Arabia to say

to him " Sir President .

Figure 4.2 shows the percentage of good sentence replacements as a function of similarity threshold for the two Arabic text chunking methods based on the human judgments. Good replacements are those sentences that humans judged to be in category 1 or 2 from Figure 4.1 (improves with summary without changing the meaning, or improves the summary but changes the meaning) in the manual evaluation described earlier. Since the *copy-split* chunking method performs better than, or equal to the simple chunking method in all cases, the following experiments use the copy split chunking method.

With the decision of which method of chunking Arabic text made, the next section presents results of a human evaluation done using Simfinder with full Arabic machine translated sentences, Simfinder with chunked Arabic sentences, and the cosine similarity metric.

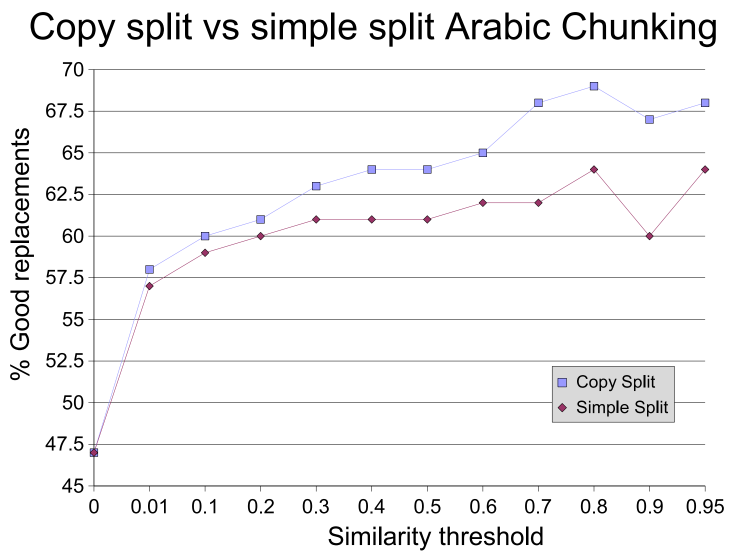


Figure 4.2: Percentage of good sentence replacements using copy-split chunking technique and simple-split chunking technique as a function of similarity threshold.

4.1.2 Sentence level evaluation results

There are two evaluations for which manual judgments are made, both using syntactically simplified English sentences. The first uses full Arabic machine translated sentences, and the second uses chunked Arabic machine translated sentences. For each document set, the ten English sentences with the highest similarity to an Arabic sentence are evaluated. The top three most similar Arabic sentences for each of the ten English sentences in the set are judged by humans on the 5 point scale presented previously.

I compute evaluation results by examining, for a given similarity threshold, how many proposed sentence replacements have a similarity higher than the threshold. Those sentences that are above the threshold and marked by evaluators with category 1 or 2 are marked as "Good" sentence replacements, while those from category 3, 4, or 5 are poor replacements.

Table 4.2 summarizes the results of the two evaluations (full Arabic sentences and chunked Arabic sentences) at a similarity threshold of 0.7. The threshold of 0.7 is used for two reasons. First, the similarity threshold of 0.65 was chosen as an optimal threshold for use with Simfinder over the English training data from the perspective of precision and

Arabic, Similarity type	% Good	# Sents
Full, Cosine	77%	24
Full, Simfinder	59%	227
Chunked, Simfinder	56%	294
Chunked, Simfinder with filter	62%	250
Full, cosine chunked similarity	71%	21
Full, Simfinder chunked similarity	**68%**	**151**

Table 4.2: Percentage of good sentence replacements for sentence-by-sentence evaluation at 0.7 similarity threshold.

recall. The region of 0.6–0.7 is the area with the best F-measure for Simfinder. Second, the precision and coverage for these experiments seems to be similar, as shown in Figure 4.4. Figure 4.4 shows the percentage of good sentence replacements and number of sentences replaced plotted against the similarity threshold using Simfinder with full Arabic sentences and chunked similarity values. The first two entries in the table are discussed in Section 4.1.2.1, the next two entries are discussed in Section 4.1.2.2, and the last two are discussed in Section 4.1.3.1.

Each entry in the table lists the type of text and similarity metric used. Full is full Arabic sentences, while chunked is chunked Arabic sentences, all with simplified English sentences. The cosine metric is a common measure used for text similarity in information retrieval and other areas. It measures the similarity of two items x and y as $\frac{\sum_{i=1}^{N} x_i \times y_i}{\sqrt{\sum_{i=1}^{N} x_i^2 + \sum_{i=1}^{N} y_i^2}}$, where the variable i ranges over all the terms in the vector space. Simfinder with filter is Simfinder run with a filter to remove certain sentences that Simfinder erroneously scores highly, and full with chunked similarity refers to scoring a full sentence using the scores of its component chunks. Both of these will be more fully explained later.

4.1.2.1 Full Arabic Sentences

The cosine similarity metric results in very few replaced sentences in the full machine translated Arabic sentence evaluation, only 24 sentences compared to Simfinder's 227. Figure 4.3

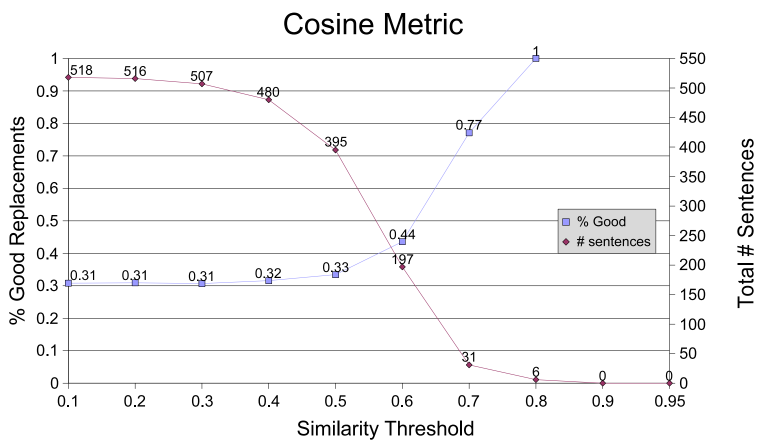


Figure 4.3: Number of good sentence replacements and total sentence replacements graphed vs. similarity threshold for cosine similarity metric using full machine translated Arabic sentences.

shows the percentage of good replacements and total number of sentences for the cosine similarity metric as a function of the similarity threshold. The cosine metric performs uniformly poorly for similarity thresholds below 0.5, at about 32% good replacements, and then improves to 70%–100% from 0.7–1.0, but only very few sentences are found at these levels. The cosine metric has fewer than 5% of the number of sentences that Simfinder finds from thresholds of 0.7 and above. While the percentage of good replacements shoots up rapidly past thresholds of 0.7, the number of sentences falls dramatically, and so the cosine similarity metric is not evaluated in the chunked Arabic sentence evaluation.

In the first evaluation, replacing full machine translated Arabic text with syntactically simplified related English text based on similarities computed by Simfinder, about 59% of the replacements are judged to improve the summary at a similarity threshold of 0.7. One can improve that percentage by increasing the threshold, but that reduces the total number of sentences that are replaced as shown in Figure 4.4. With more than half of the sentence replacements helping the summary, I am interested in the effect splitting the Arabic sentences might have; by focusing on smaller parts of the Arabic sentences, I hypothesized that I can find better matches to just that part of the sentence in the related English text.

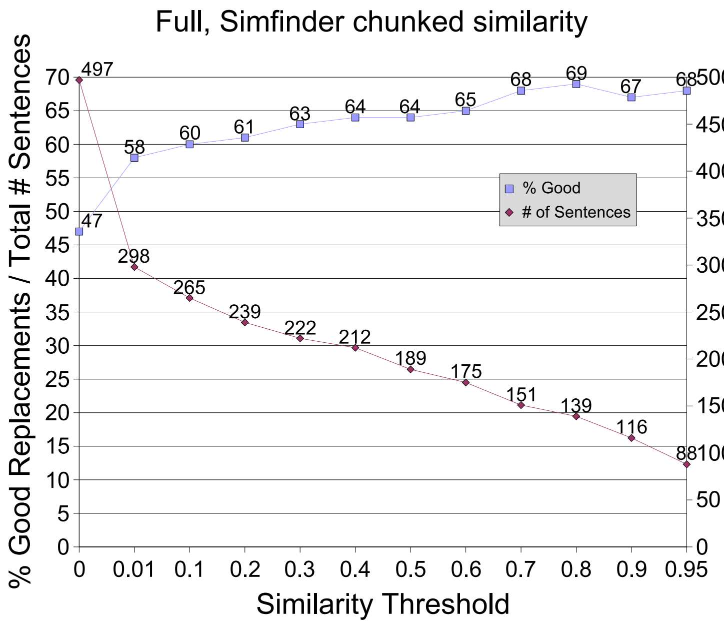


Figure 4.4: Percentage of good sentence replacements for full machine translated Arabic sentences with simplified English sentences using chunked Arabic similarity values.

4.1.2.2 Chunked Arabic Sentences

So far, I have presented the results of the first sentence-level evaluation using human similarity judgments to rate replacement of full Arabic machine translated sentences with similar English sentences. The results show that using Simfinder to compute similarity values used for identifying sentences for replacement improves a summary in 59% of the cases, validating hypothesis 1. Simfinder performs much better than a cosine-based metric, which doesn't identify enough similar sentences to be useful, validating hypothesis 2. Now, I will investigate hypothesis 3, which asks whether chunking the Arabic text into smaller units can improve results.

The initial results using chunked Arabic text with Simfinder similarity are not as good as using full Arabic sentences: 56% at a 0.7 similarity threshold. With shorter sentences, there is a problem of lack of context, which causes short sentences to have high similarity scores based on only one or two similar words. In this second evaluation, evaluators were asked to make a second judgment on whether the Arabic and English sentences were really related at all, as sometimes the short sentences, lacking context, were either incomprehensible or nonsensical. When using chunked Arabic text with Simfinder similarity scores, 26% of the sentences over the 0.7 similarity threshold are labeled as "not related" by evaluators. For example, the following sentences were labeled as "not related":

Arabic chunked sentence	Suggested English sentence
decline rapidly wind	British Prime Minister Tony Blair.
Rafah (Gaza Strip (11 $DATEMD 24 (AFP) - Egyptian plane landed	Its mission is over."
the presidential elections early be held in April .	" It will stop.

4.1.3 Error analysis

I performed a manual error analysis to determine what types of errors the similarity-based sentence replacement system was making. In the manual examination, 100 instances of sentences labeled as highly similar and 99 instances of sentences labeled with low similarity values are examined for errors. The sentences were automatically chosen such that many of the sentences are erroneously classified. I examined the sentence pairs and placed them into error classes, which led to the development of the filters presented in Section 4.1.3. Table 4.3 presents results of the error analysis for sentences labeled as "similar", while Table 4.4 presents results on the sentences labeled as "not similar".

In the analysis of sentences incorrectly labeled as "not similar" with high similarity, in approximately half of the sentence pairs an unidentifiable error was the source of the incorrect similarity value. In cases such as these, I was unable to assign a reason that would lead to the high similarity score assigned. Many such instances are possibly the result of using machine translated text, with an example sentence pair being:

1a. parliamentarians will be published today not that not confirmed whether include

and

1b. " We are not going to give up territory.

Error Type	Count
Unknown Error	52
Word overlap only	22
Number mismatch	10
General to specific mismatch	9
Main action verbs differ	3
Different main actors	2
Factual vs. opinion	1
Correct	1

Table 4.3: Categories of errors made in a manual analysis of 100 high-similarity sentences of machine translated Arabic text and simplified English.

Sentences 1a and 1b, with a similarity value of 0.78, are incorrectly labeled similar, but it isn't clear what leads to this error. The next most frequent type of error, word overlap only, occurs when a sentence pair contains one or more words in common which are not important enough to warrant similarity. These two error classes lead to the main filters developed in Section 4.1.3. Following these errors, sentence which have similar information but mismatches based on numbers (years, numbers dead, etc.) indicating different events follow at ten errors, while general to specific mismatches are next at nine. The error class "Different main actors" indicate sentences where the actors are different, but performing similar actions. I theorize that named entities are important in similarity identification, and in this examination of sentences labeled as highly similar, few of the errors are attributable to named entities in the sentences. In this error analysis, named entities are not present in incorrectly marked sentence pairs.

Table 4.4 presents the results of examining 99 sentence pairs with low similarity values that are actually similar. Of the low-similarity sentence pairs, many more are actually correct than with the high-similarity sentence pairs: 23 compared to 1. The sentence pairs were selected automatically from low-similarity sentence pairs – the majority of which are correctly labeled as "not similar" – by choosing low similarity sentence pairs marked as either

Error Type	Count
Unknown Error	29
Correct	23
External Inference	20
Important subjects not weighted highly enough	16
Context used to resolve pronoun	6
Numbers indicate match	5

Table 4.4: Categories of errors made in a manual analysis of 100 low-similarity sentences of machine translated Arabic text and simplified English.

"good replacements" for machine translated sentences, or sentence pairs no better or worse than the machine translated sentences. Many of the sentences, upon close examination, are correctly identified as being "not similar".

As with the high similarity sentences, many pairs are marked with "Unknown Error" as no concrete reason can be found for the low similarity, or when the sentences have many words in common, but are not actually similar – they could be in the "Correct" category but it is unknown why the similarity is low.

The most frequent explainable error type with low similarity sentences is that some form of external inference would be required to realize that the sentences are similar. For example, in the sentences:

2a. its resolution to cease cooperation with inspectors United Nations.

and

2b. Iraq crisis is over Iraq's decision Saturday to end all cooperation with U.N. disarmament experts.

some form of external inference is required to realize that Iraq is ceasing cooperation in sentence 2a, and that "inspectors United Nations" are the same as "U.N. disarmament experts". The next most frequent error class, important subjects not weighted highly enough, occurs when sentences have correct terms in common, but they are not weighted highly enough to make the sentences similar based on the statistical regression model. Twelve

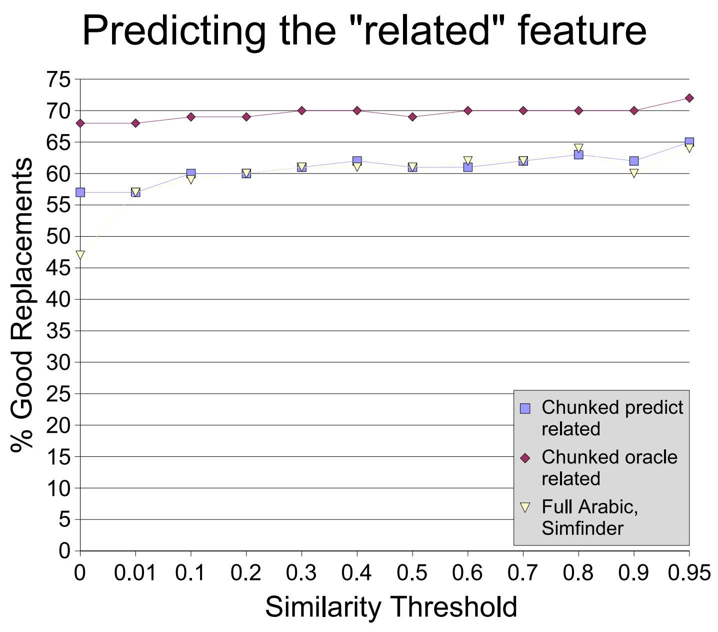


Figure 4.5: Percentage of good replacements by similarity threshold when predicting "related" feature.

of the sixteen instances have proper names in them. The proper names in the examined sentences are not always exact string matches, for example, "the Libyan leader Colonel Muammar Qaddafi" and "Libyan leader Moammar Gadhafi" – even within a single language better named entity resolution would help improve similarity results. When dealing with cross-language similarity, the problem is exacerbated due to difficulties in translating names across languages [KG97, SK98]. A similar error analysis could be performed across languages with a bilingual evaluator.

Filtering unrelated sentences Chunking the Arabic sentences removes more context from the sentences, causing similarity scores from the shorter chunks to not be as reliable as full sentences. To determine if this is a problem, I had the judges evaluate whether the shorter Arabic sentence chunks are actually related to the simplified English sentences or not in addition to the five-point scale used in the first evaluation of full-length Arabic sentences. Looking at the results for only sentences labeled "related", results were much better: 70% good replacements at the same threshold.

Figure 4.5 graphs the percentage of good replacement sentences as a function of the similarity threshold for three different similarity metrics: *chunked oracle related*, *chunked predict related*, and *full Arabic, Simfinder*. *Full Arabic, Simfinder* is the same metric presented in Table 4.2 using full Arabic sentences with Simfinder to compute similarity. *Chunked oracle related* shows the performance evaluated using only sentences that are known by an oracle to be "related" as marked by the evaluators. *Chunked predict related* predicts the related feature using a decision tree learned over the sentence data.

Figure 4.5 shows that with an oracle that knows which sentences are tagged with the "related" feature, the percentage of good replacements rises to the 70% level. Due to this dramatic increase, I investigate predicting the "related" feature given the sentences. I use a machine learning approach, computing a variety of features between the two sentences, and then use the WEKA machine learning toolkit [Gar95] to induce learners for the "related" class using 10-fold cross validation over all the data. I compute 22 features over the two sentences, including cosine similarity, jaccard similarity, length differential features, tf*idf differential features, longest common substring features, overlap on verbs, non-communicative verb overlap, and overlap on proper nouns. The best resulting classifier, a decisions stump, uses only the cosine feature, and has about 76% accuracy at predicting the "related" class. This compares to a baseline of 62% of just always choosing the "related" class. Integrating the "related" class prediction, sentences are only replaced if they are predicted to belong to the "related" class, and have a similarity above the similarity threshold. This improves results using chunked Arabic text to 62% at a 0.7 similarity threshold, as shown in Table 4.2. Predicting the "related" feature compares favorably with using Simfinder to compute similarity over non-chunked Arabic sentences, as shown by the *Full Arabic, Simfinder* and *Chunked predict related* lines in Figure 4.5. When predicting the related feature, the percentage of good replacements made when chunking Arabic sentences is about the same as with full sentences. Unfortunately, this does not help to validate hypothesis 3, that chunking Arabic text will help to improve results when replacing sentences. Chunking the Arabic text into smaller units reduces the context available for the sentences, and while many of the "unrelated" (often short) sentence pairs can be filtered out by predicting the "related" feature, there is no overall improvement by chunking the Arabic text.

4.1.3.1 Full Arabic sentences with chunked sentence similarity

The approach of chunking the Arabic text into smaller units has disappointing results; some of the Arabic chunks are so small that computing meaningful similarity separately was difficult. However, if we could combine the comparison of chunks within the context of the full sentence, perhaps this would give an improvement. Breaking the Arabic sentence into chunks better differentiates the similarity values. With longer sentences similarity values tend to increase because of the greater number of words, but by looking at smaller chunks, the incidental similarity decreases. An Arabic sentence that is not really very similar to an English sentence might have two chunks that taken together by chance have enough words in common to appear similar, but when broken into chunks do not have high individual similarity values. Figure 4.6 shows a simplified English sentence, E, that has a high Simfinder similarity value of 0.930 to the full machine translated Arabic sentence A. Unfortunately, sentences A and E are not very similar to each other; but perhaps inspection of the similarities of the chunks A1 and A2 can help reveal that the two sentences are not similar. A1 and A2, the two component chunks that sentence A is chunked into, have Simfinder similarities of 0.979 and 0.0038 respectively to E. The average similarity of A1 and A2 to E, 0.49115, is much lower than the similarity of A to E, thus indicating that the two sentences are not really very similar.

The final system shown in Table 4.2 replaces full machine translated Arabic sentences with syntactically simplified English sentences, but uses similarity values for the Arabic sentences from the chunks it contains as shown in Figure 4.6. When evaluating replacement of a full Arabic sentence by an English sentence, I retrieve the similarity values for each chunk in the Arabic sentence to the English sentence, yielding a set of similarity values for each Arabic sentence. I use the average similarity value of all the chunks, check whether the average similarity value is above the threshold, and perform replacement if so. This model avoided the problems caused by the very short Arabic chunks by choosing the average similarity score from all chunks and at the same time, it avoids some of the problems with

E: In an atmosphere of political tension, U.S. President Clinton met Sunday with Israeli Prime Minister Benjamin Netanyahu in a bid to put the troubled Wye River peace accord back on track.	
A: the status of American President Bill Clinton Sunday grave Israeli Prime Minister former Yitzhak who was assassinated three years ago by extremist of the Israeli block of the Wye agreement .	
A1: the status of American President Bill Clinton Sunday grave Israeli Prime Minister former Yitzhak who was assassinated	**A2**: former Yitzhak was assassinated three years ago by extremist of the Israeli block of the Wye agreement .

Figure 4.6: An example of an English sentence (E), and a full Arabic sentence (A) with high similarity to the English sentence, and the chunked components of the Arabic sentence (A1, A2) that have low similarity to the English sentence.

comparing overly long sentences where false matches are suggested for replacement simply due to the larger quantity of words. The full threshold–precision chart for this run is shown in Figure 4.4. Of all the approaches that cover a large number of sentences this one performs the best, with approximately 68% of the replacements being judged as improving the summary.

Using similarity values computed as a function of the similarity of sub-sections of the Arabic sentence allows comparisons of sentences by the propositions they contain and thus, shows an improvement over using similarity values from the entire sentence. While taking the average of the chunk similarity values performs best, taking the minimum or maximum of all the values also performs better than using similarity values from the full sentence. Using the similarity values of the chunked components of the full Arabic sentence increases precision above the levels obtainable using full sentences with Simfinder similarity values only, or chunked sentences with Simfinder and a filter.

In this section on sentence-level evaluation, I examined three main approaches to identifying similar syntactically simplified English sentences to machine translated Arabic text. First, I used Simfinder and compared to a cosine metric baseline, and found that Simfinder identifies many more possible replacement sentences, which improve a summary 59% of the time. I next examined chunking the Arabic sentences, and found that 62% of replacement sentences identified by Simfinder combined with a filter that eliminates un-related sentence pairs improve the summary. Finally, using full Arabic sentences with the similarity scores of the chunked components improves results to 68%.

In this section, I examined four main hypotheses:

1. Replacing machine translated Arabic sentences with similar English sentences in a summary improves the summary
2. Syntactic sentence simplification on English text improves similarity results
3. Chunking machine translated Arabic text improves similarity results
4. Simfinder computes more accurate similarity values for finding similar sentences than a cosine-based metric

Hypothesis 1 was validated via two manual evaluations that examine replacing machine translated Arabic sentences with English sentences, and found that in 68% of the cases, replacement improves the summary. Hypothesis 2 was not validated directly through manual examination of individual sentence replacements, but a larger-scale evaluation presented in Section 5.3 finds that syntactic simplification of English sentences improves results, and so this section uses syntactic simplification of the English text. Hypothesis 3 was not shown to be directly true, as chunking applied to the Arabic text is shown to degrade performance unless combined with a filter. Chunking Arabic text is shown to improve scores for full Arabic sentence replacement when the similarity scores are a function of the similarity scores of the component chunks of the Arabic sentences though. Hypothesis 4 was shown to be true as Simfinder consistently outperformed the cosine similarity metric, which while accurate at high thresholds, has such a low recall that it is not usable.

4.2 Clustering evaluation

The previous sections compared how well one can identify similar sentences using Simfinder, and whether replacing a machine translated Arabic sentence with a similar English sentence improves a summary or not. The second application of similarity for summarization that I use in summarization is clustering text. Clustering text is the process of computing similarity values over sentences, and then building clusters of sentences that are all similar to each other. These clusters are used to both judge how important a concept is based on repetition throughout the document set, and to reduce repetition by only including one sentence for each concept represented by a cluster. This section compares how well a run of Simfinder using machine translated Arabic text compares to a run using manually translated Arabic text for the clustering task.

The next sections evaluate SimFinderML's performance at finding English sentences similar to Arabic sentences in a clustering-based evaluation. The evaluation shown in section 4.1.2 requires human judgments to be made over the (machine translated) Arabic and English sentences. Due to a lack of bilingual Arabic–English speakers, that same evaluation was not performed for the SimFinderML runs. Without access to bilingual Arabic–English speakers to make the sentence level judgments, the evaluation could not be performed. Instead, a second clustering-based evaluation compares SimFinderML's performance to the performance of using full machine translation at the same task.

Since the desired application for SimFinderML is multi-lingual, multi-document sentence clustering, a proper evaluation of SimFinderML should answer the question "How well does SimFinderML build sentence clusters with both English and Arabic source documents?" To answer that question and estimate performance over data with similar text in both Arabic and English, I compare my Arabic–English system to the English version of Simfinder using a sentence-aligned corpus of manually translated documents. Comparing to the English version of Simfinder is a way to automatically evaluate over large amounts of data without manually hand-tagging the data. The English version of Simfinder has been well-tested over English data, and gives SimFinderML a good target to which it can aspire.

Figure 4.7 shows a cluster from DUC2004 set D1001T that has one manually translated Arabic sentence, and three English sentences. In the clustering evaluations, the clusters produced by English Simfinder over the manually translated Arabic sentences and syntactically simplified English sentences are used as a gold standard. Clusters produced by SimFinderML run over untranslated Arabic and syntactically simplified English are compared to

Arabic S1 Washington 12-16 (AFP) - The American Defense Department (the Pentagon) reported today, Wednesday, that the bombing runs against Iraq are being especially carried out via cruise missiles launched from American aircraft carriers and B52 bombers -.
English S2 Most of you saw some large explosions around Baghdad yesterday and many of you correctly surmised that those were caused by air launched cruise missiles launched from B-52 aircraft.
English S3 Navy ships also launched additional Tomahawk missiles at targets in Iraq.
English S4 Although I will not get into the specific numbers and types of cruise missiles launched to date, I can tell you that the total number of air launched and ship launched cruise missiles for this operation thus far now exceed the total number expended during all of Operation Desert Storm.

Figure 4.7: Example cluster of manually translated Arabic sentences and simplified English sentences.

the clusters from English Simfinder. The comparison is performed pairwise, checking that pairs of sentences in the English clusters are also in the same cluster in the SimFinderML clusters. In this example, sentence pairs **(S1,S2)**, **(S1,S3)**, **(S1,S4)**, **(S2,S3)**, **(S2,S4)**, and **(S3,S4)** are marked as similar in the English clustering. The same construction of similar sentences is done with the SimFinderML clusters, the Arabic sentence numbers are mapped to the aligned manually translated sentence numbers, and the similar sentences are compared between the two clustering runs. In the following sections, I present precision and recall numbers using the DUC 2004 corpus, and a method of checking clustering similarity using binary comparison between items in two clusterings.

This section tests the following hypotheses:

1. Computing similarity between Arabic and English sentences using SimFinderML can be as or more effective than performing machine translation on the Arabic text and using English Simfinder to compute similarity

2. Probabilistic translation lexicons in SimFinderML improve performance over dictionary lookup

3. Using Named Entity detection in SimFinderML improves Arabic–English similarity results over using only tokens for similarity computation

4.2.1 Aligning the DUC 2004 Corpus

I use the DUC 2004 corpus, described in Section 5.2, and align the sentences in the original Arabic documents to the sentences in the manual human translations into English. The sentences are aligned with Dan Melamed's GSA implementation of the SIMR algorithm [Mel97b]. To perform the alignment, a translation lexicon to make initial alignments is required. In typical alignment tasks, some initial anchor points are needed to perform the alignment. Often, numbers and punctuation are used. I created a translation lexicon that contained translations of the words in the Arabic text that contained only one target English gloss from the Buckwalter parse of the Arabic morpheme. Using a translation lexicon of this sort is also common, in addition to numbers and punctuation to create anchor points for the alignment process. The GSA program was run with default settings for other parameters.

For each Arabic document, GSA mapped the sentence numbers from the Arabic document to sentence numbers in the English translation. Not all sentences are mapped, and a single sentence might possibly be mapped to multiple sentences, although that is not very common with our data.

4.2.2 Using the Aligned Corpus for Evaluation

I use the aligned Arabic and human-translated portion of the DUC corpus to evaluate SimFinderML's Arabic–English similarity performance by comparing to Simfinder 1.1's clustering over English. In this evaluation, I compare the following cases:

- SimFinderML run over the DUC corpus with Arabic text and the provided relevant English articles
- English Simfinder run over the sentence-aligned human translations of the Arabic articles from the DUC corpus with the same relevant English articles.

To determine the effectiveness of the SimFinderML similarity computation compared to the similarity computation of English Simfinder, I evaluate whether sentences that are clustered together in the SimFinderML run are also clustered together in the English Simfinder run. In the following experiments, I run the Arabic–English version of SimFinderML with different feature sets over the Arabic documents with the relevant English documents, and cluster the sentences. I also run English Simfinder over the human-translated English versions of the Arabic documents with the relevant English documents, and cluster those sentences. For each sentence cluster in the Arabic run, I used the sentence maps generated in Section 4.2.1 to see whether each sentence pair in the cluster was also clustered together in the English Simfinder run.

To evaluate the performance, I cluster the text with SimFinderML at various thresholds. English Simfinder is typically run with a threshold of 0.65, a value which has been effective for a wide variety of inputs and general clustering needs. For SimFinderML, especially for the Arabic–English version, a suitable value for the threshold parameter has not been determined. For each run, I wrote a program to examine all sentence pairs that are deemed "similar" by their inclusion in the same cluster, map those sentence numbers to the corresponding English sentence numbers in the translated data, and then check to see if the sentence pair(s) (possibly multiple sentences since the Arabic–English mapping is not always 1 to 1) are included in the same cluster in the English data.

Note that the maximum precision and recall in this evaluation is not 100%; as shown in Section 4.2.4 a run of Simfinder with the same data averages about 84% precision and 87% recall in this evaluation.

Features and Translation type	Precision	Recall	F-Measure
Token feature, Buckwalter translation	58.5%	21.4%	31.4%
Token feature, probabilistic translation	77.7%	15.2%	25.5%
Token feature, probabilistic and Buckwalter translation	80.0%	14.8%	25.0%
Token and Named Entity features, Buckwalter translation	49.1%	22.6%	30.9%
Token and Named Entity features, probabilistic translation	75.0%	16.0%	26.3%
Token and Named Entity features, probabilistic and Buckwalter translation	81.7%	14.4%	24.4%

Table 4.5: Summary table showing performance at threshold 0.7 for a variety of translation mechanisms and features.

4.2.3 Evaluation Overview

This section presents results using different feature merging models and translation mechanisms for similarity computation.

Table 4.5 shows the average recall, precision, and f-measure of different feature sets with different translation mechanisms at a similarity threshold of 0.7. There are two forms of translation investigated in this section: Buckwalter translation and probabilistic translation. Using probabilistic translation greatly increases precision over using only Buckwalter translation, and using both Buckwalter and probabilistic translation together slightly increases precision with a slight degradation on recall.

English Simfinder uses seven features to compute similarity in English: similarity between two sentences in terms of stemmed tokens, proper nouns, noun phrase heads, verbs, adjectives, nouns, and WordNet categories. Each of these features is a different scale on which two sentences are measured, and the final similarity is a combination of the similarity of the different features. See Chapter 2 for further details. SimFinderML also uses multiple feature sets to compute similarity between text in different languages. In this section, two features are investigated for Arabic–English similarity: the token and named entity features. Adding the named entity feature tends to slightly increase recall with a slight dip in precision.

4.2.4 Full machine translated and English Simfinder

Before presenting the work on clustering non-translated Arabic text and English text, I will first present results from clustering manually translated Arabic text with English text. Using machine translated text presents an upper-bound for performance using resource-intensive machine translation systems. I also present the results of comparing clusters from two different runs of the clustering system using manually translated Arabic and English text. This presents an upper-bound on how well a system with perfect translation could expect to perform. Since the clustering process is an optimization-based search, it finds different clusterings given the same input data, resulting in non-perfect scores in this clustering-based evaluation even given the same input data.

To test how well Simfinder clusters machine translated Arabic and English text, I cluster the "best only" version of the IBM translated documents from the DUC data set with the relevant English text from the set. The clusters produced by the system are compared to clusters generated using manual translations of the Arabic documents with the relevant English text.

Figure 4.8 shows a graph of precision and recall of two runs of English Simfinder over the same set of manually translated Arabic and English data where the clustering of one set is compared to the other. To compute precision, I first transform the clusters for both runs into all pairs of sentences that are in the same cluster. If sentences **A**, **B**, and **C** are in the same cluster, then I note that pairs **A,B**, **B,C**, and **A,C** are clustered together. I compute precision by checking whether each pair clustered together in the system run is also clustered together in the manually translated English run. Recall is computed by checked how many of the sentence pairs clustered together in the manually translated run are also clustered together in the system run.

These runs use a threshold of 0.65, the default for English Simfinder. The average precision of one run of manually translated Arabic and English text evaluated against another run over the same text is 84.6% over all the runs, and average recall is 87.1%.

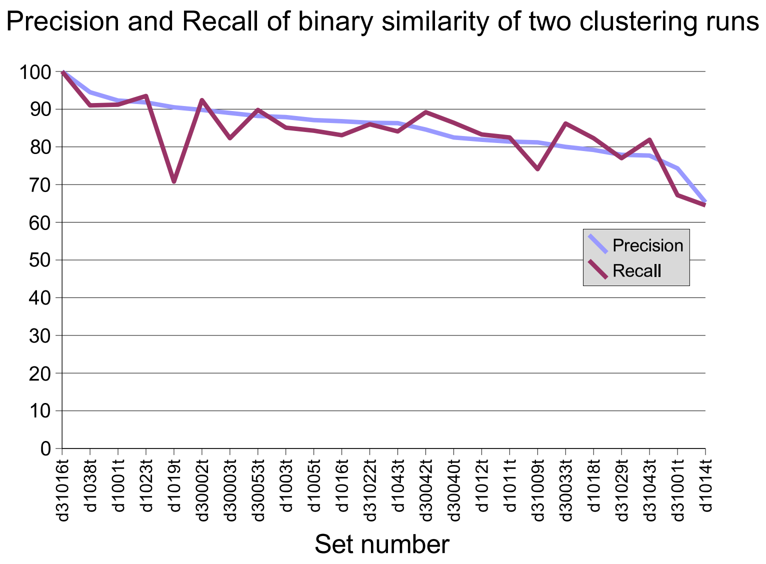


Figure 4.8: Graph of precision and recall of binary similarity values between two runs of Simfinder over the same manually translated Arabic and English data.

Results vary depending on the set, but in all but run d31016t, the output clusterings differ between the two runs. The machine translated runs should not be expected to perform better than approximately 85% precision or recall over the same data, as some amount of error can be attributed to the clustering and evaluation process.

Table 4.6 shows the results using machine translation of the Arabic text and English Simfinder at a variety of thresholds. The row for threshold 0.7 is boldfaced since other experiments in this chapter use the threshold of 0.7 as the threshold to compare results at. At 0.7 precision is 66.5%, and recall is 51.1%, which fall short of the upper-bound using manually translated Arabic and English presented above, 84.6% and 87.1%. While the results fall short of the performance of using non-translated text, they are still fairly good; 66.5% of the time two sentences clustered in the machine translated run were also clustered in the run with non-translated English. Using the similarity values between sentences in the cluster, which are not taken into account in this evaluation, sentences that are not highly similar can be ignored. In many applications, such as the summarization applications presented in Chapter 5, precision is more important than recall. As shown in Table 4.6,

Table 4.6: Precision and Recall between machine translated Arabic–English data and corresponding aligned manually translated Arabic–English data

Threshold	Avg. Precision	Avg. Recall	F-measure
0.1	0.105	0.605	0.178
0.2	0.197	0.578	0.293
0.3	0.302	0.582	0.398
0.4	0.393	0.564	0.463
0.5	0.506	0.563	0.533
0.6	0.590	0.553	0.571
0.7	**0.665**	**0.511**	**0.578**
0.8	0.714	0.437	0.542
0.9	0.767	0.337	0.468

precision can be increased at the cost of recall by changing the threshold. At a threshold of 0.9, precision is 76.7%, which is not bad compared to the level of manually translated Arabic and English presented as an upper bound, 84.6%. In the remainder of this chapter, a threshold of 0.7 is used as it maximizes the F-measure in this experiment, and is close to the English Simfinder default of 0.65.

4.2.5 SimFinderML Token feature

In Section 4.2 I presented an overview of how the clustering-based evaluation works, and in Section 4.2.4 I presented an upper-bound on performance using manually translated Arabic text, and an evaluation using machine translated Arabic text. In this section, I will present results using SimFinderML with untranslated Arabic text and a token-based feature for translation and similarity computation.

In cases where full machine translation is not available, SimFinderML can be used to perform translation at a word-by-word level. This section presents results using SimFinderML computing similarity between Arabic text and English text with one of two translation mechanisms: the output of a morphological processing system that contains a dictionary, and a probabilistic dictionary learned from a parallel aligned corpus.

As more and more text is becoming available in foreign languages, even when a bilingual dictionary is not available, it is possible to learn a probabilistic dictionary as used in these experiments. The probabilistic dictionary is shown to have good results, and existing bilingual dictionaries can be integrated to improve performance further. Integration of the learned probabilistic dictionary in addition to dictionary lookup improves results compared to lookup alone.

4.2.5.1 Arabic–English with word-level translation feature

I perform a series of experiments using only a single feature, word level translation, to compute similarity between the Arabic and English text. I have implemented two different translation mechanisms for token translation: Buckwalter translation, and probabilistic translation.

I run three experiments with the token feature:

1. Token feature with Buckwalter translation
2. Token feature with Probabilistic translation
3. Token feature with Buckwalter and Probabilistic translation

I will show that the integration of a probabilistic dictionary improves translation over using Buckwalter translation alone.

Buckwalter translation uses the Buckwalter Arabic morphological analyzer available from the LDC [Buc02] and extracts all translations for an Arabic token from the glosses for Arabic tokens that are successfully parsed by the morphological analysis program. A single token might have multiple possible interpretations, each one with multiple glosses. There is no information about likelihood of any particular translation, and SimFinderML does not perform any disambiguation to determine the correct sense of the translation. SimFinderML does weight each link between Arabic and English tokens found as $\frac{1}{\text{\# of translations}}$ so that translations from highly polysemous words are not weighted as heavily as words with few translations.

Probabilistic translation uses dictionaries of the type described in Section 3.4.3. These dictionaries are learned from large bilingual corpora and contain, for each token, a target translation with a translation probability. SimFinderML uses the translation probability to weight the strength of the link between the Arabic and English tokens when computing similarity for the token feature. The learned probabilistic dictionary might not have translations that are less common if they are not in the corpus, but it does contain information on which translations are more likely.

4.2.5.2 Results using token feature

Table 4.7 shows the average precision, recall, and F-measure for runs at different threshold levels using the token feature and Buckwalter translation. To use SimFinderML with a new translation method or set of features, a model must first be learned that contains information on how to best combine evidence from the features under the given translation method to compute similarity values. The only feature used in this section is the token feature, so a similarity value is computed between two sentences based on how many tokens they have in common, and the tokens are translated using the English glosses from the Buckwalter morphological analyzer. The final similarity value is then computed by a logistic transformation based on the trained model of the similarity value from the token feature.

To perform this run, I first learned a feature merging model that uses only the Arabic token feature with Buckwalter translation, as explained in Section 3.4.4. SimFinderML is then run over all the sets using the new feature merging model, and the resulting clusters are evaluated.

The second run, shown in Table 4.8, shows the results when using probabilistic translation over the token feature. Compared with using Buckwalter translation only, the precision is much higher, but the recall is much lower, resulting in a lower F-measure. Both of these differences are statistically significant using a signed paired Wilcoxon test with $p < 0.05$. This might be because the probabilistic dictionary reflects only translations seen in practical usage, while the Buckwalter glosses provide a wider range of possibly less-likely translations. Translations from the probabilistic dictionary are more likely to be correct, but do not offer as wide a range of coverage as the Buckwalter glosses. Note that the corpus used to learn the Arabic–English probabilistic dictionary is different from the DUC2004 data: the corpus covers news from January 2001 to 2004 from the Ummah Press Service, while the DUC data

Threshold	Avg. Precision	Avg. Recall	F-measure
0.2	0.575	0.261	0.359
0.3	0.475	0.234	0.314
0.4	0.448	0.259	0.328
0.5	0.503	0.251	0.335
0.6	0.543	0.231	0.324
0.7	**0.585**	**0.214**	**0.314**
0.8	0.606	0.191	0.290
0.9	0.653	0.173	0.273

Table 4.7: Precision and Recall between Arabic–English data and aligned English–English data using token feature only with Buckwalter translation.

Threshold	Average Precision	Average Recall	F-measure
0.2	0.745	0.183	0.294
0.3	0.743	0.174	0.282
0.4	0.699	0.163	0.264
0.5	0.736	0.154	0.255
0.6	0.753	0.154	0.256
0.7	**0.777**	**0.152**	**0.255**
0.8	0.772	0.164	0.271
0.9	0.790	0.166	0.275

Table 4.8: Precision and Recall between Arabic–English data and aligned English–English data using token feature only with probabilistic translation.

Threshold	Avg. Precision	Avg. Recall	F-measure
0.2	0.794	0.143	0.243
0.3	0.792	0.162	0.270
0.4	0.782	0.153	0.256
0.5	0.777	0.150	0.251
0.6	0.785	0.152	0.254
0.7	**0.800**	**0.148**	**0.250**
0.8	0.828	0.155	0.262
0.9	0.855	0.157	0.266

Table 4.9: Precision and Recall between Arabic–English data and aligned English–English data using token feature only with Buckwalter and probabilistic translation.

covers news from the Agence France Press Arabic Newswire in 1998 and 2000-2001. The mismatch in topics and names from those time periods might mean that common terms in the DUC data do not have representative translations in the learned dictionary.

It is interesting to note, however, that using only translations from a dictionary learned automatically from a parallel corpus results in high precision in the clustering task, close to the upper-bound precision seen when clustering the same English data over two runs (84.6%), and better than the precision observed when using full machine translation (66.5%). Precision is important for one of the tasks that I am interested in applying multi-lingual text similarity to: summarization. When clustering sentences across languages it is important that the clusters that are formed are of high quality (high precision) because the clusters will be used as the basis for sentences in a summary. Recall is less important, because not all sentences are needed for a summary; many are dropped. The sentences that are in the summary though, do need to be topically related to the sentences that they are clustered with, which puts an emphasis on precision in this clustering evaluation.

Table 4.9 shows results from using both Buckwalter and Probabilistic translation in conjunction. Compared to using Buckwalter translation alone, precision is much higher at every threshold: 80.0% vs. 77.7% at threshold 0.7. Recall is a bit lower, particularly at the lower thresholds. The increase in precision and decrease in recall compared to using only probabilistic translation is significant with $p < 0.05$ in both cases on a signed paired Wilcoxon test. That combining the two methods of translation improves precision is interesting; it shows that the two translation resources are bringing in different information to the process, and for applications that require improved precision, both translation mechanisms can be applied.

For languages where resources such as manually created bilingual translation dictionaries are available, they can easily be integrated into SimFinderML with other translation resources, such as probabilistic translation dictionaries that can be learned from parallel corpora. For tasks where precision is important, SimFinderML's outperforms full machine translation (80.0% precision for SimFinderML vs. 66.5% precision for full machine translation) in this clustering evaluation. For languages where the set of natural language processing tools available is limited but large parallel corpora are available, SimFinderML can be used with only a learned probabilistic translation lexicon with only a minimal loss of precision compared to using the probabilistic translation lexicon with an integrated bilingual dictionary (77.7% vs. 80.0%.)

4.2.6 Token and Named Entity features with SimFinderML

If additional natural language processing resources are available in the foreign language, SimFinderML allows for easy integration of primitives for the foreign language based on those resources. I have implemented a named entity primitive for Arabic using BBN's IdentiFinder^{TM}Arabic named entity tagging software. The following section reports results from runs that use both the Named Entity feature in addition to the token feature discussed in Section 4.2.5.1.

4.2.6.1 The Named Entity Feature in Arabic–English SimFinderML

I use BBN's IdentiFinder^{TM}version 3.2 in English and Arabic to detect named entities in the text. IdentiFinder tags 24 different types of named entities in three main categories: names, times, and numbers. The names category includes types for animal, contact info, disease, event, facility, game, geo-political entity, language, law, location, nationality, organization, person, plant, product, substance, and work of art. Times include dates and clock times, and numbers includes cardinal, money, percentage, and quantity. Input text is tagged by IdentiFinder, and then each identified named entity is inserted as a primitive for each language, which are matched across languages at the primitive translation stage.

I implemented two types of translation to match named entities across languages: matches are made using component word translations via lookup in the Buckwalter morphological analyzer gloss, or using full machine translation on the named entity via IBM's statistical machine translation system. The results reported here are using Buckwalter translation to match primitives, as IBM's translation server was unavailable at the time these experiments were run.

4.2.6.2 Results using Named Entity Feature

This section deals with a run of SimFinderML that uses two features: word-level translation, and named entities. Word-level translation is performed as above in Section 4.2.5.1. The named entity features are extracted using BBN's Arabic and English IdentiFinder version 3.2 [BBN04], and matches between Arabic and English named entities are made when more than one non-stopword token matches between the identified Arabic and English named entities. As before, experiments are performed that use Buckwalter translation, probabilistic translation, and Buckwalter translation with probabilistic translation.

Results for the first run using only Buckwalter translation are shown in Table 4.10. The precision values are statistically significantly less than those using only the token feature under a two-sided paired Wilcoxon test at $p < 0.05$, but the recall values are statistically significantly greater than the token-only recall values with $p < 0.05$. The addition of the named entity feature degrades precision, but improves recall when using Buckwalter translation.

Using probabilistic translation was shown to improve precision when using only the token feature, so using probabilistic translation might also improve precision when using the named entity feature in addition to the token feature. Table 4.11 shows the precision, recall, and F-measure for a run using both token and named entity features with probabilistic translation. The precision and recall results are not statistically significantly different between a run using only the token feature and probabilistic translation and tokens with

Threshold	Avg. Precision	Avg. Recall	F-measure
0.1	0.475	0.263	0.339
0.2	0.423	0.229	0.297
0.3	0.385	0.238	0.294
0.4	0.356	0.264	0.304
0.5	0.405	0.259	0.316
0.6	0.457	0.245	0.319
0.7	**0.491**	**0.226**	**0.309**
0.8	0.543	0.210	0.302
0.9	0.585	0.182	0.277

Table 4.10: Precision and Recall between Arabic–English data and corresponding aligned English–English data using token and IdentiFinder features with Buckwalter translation.

named entities and probabilistic translation under the signed Wilcoxon test. So, as with using only tokens, precision increases when using tokens and named entity features with probabilistic translation. While there is an improvement in recall at higher thresholds compared to the recall using only the token feature, it is not enough to be statistically significant across all thresholds.

When combining both Buckwalter and probabilistic translation, neither precision nor recall are statistically significantly different when adding the Named Entity feature compared to using only the token feature. Adding the named entity feature does not improve results over using just the token feature alone, but it does not hurt them either. In comparison to using only the token feature, adding the named entity feature helps to improve recall, with a slight impact on precision when using only Buckwalter translation. When using probabilistic translation, either alone or in conjunction with Buckwalter translation, the positive effect of the named entity feature on recall is not evident, but adding the named entity feature is not shown to hurt performance.

As with using the token feature alone, the different translation methods do have an impact when comparing runs that use both token and named entity features though; using probabilistic translation, precision increases greatly, but recall suffers. Using both proba-

Threshold	Avg. Precision	Avg. Recall	F-measure
0.2	0.795	0.194	0.311
0.3	0.740	0.171	0.278
0.4	0.693	0.158	0.258
0.5	0.753	0.151	0.252
0.6	0.744	0.142	0.238
0.7	0.750	0.160	0.263
0.8	0.759	0.165	0.270
0.9	0.780	0.175	0.273

Table 4.11: Precision and Recall between Arabic–English data and corresponding aligned English–English data using token and IdentiFinder features with probabilistic translation.

bilistic and Buckwalter translation improves precision at higher thresholds (although not statistically significant overall) compared to using just probabilistic translation alone, but there is a slight degradation in recall that is statistically significant at $p < 0.05$.

Precision when using the named entity and token features with Buckwalter and probabilistic translation approaches that of precision across two clustering runs of the same English data – 75.0% vs. 84.6%. This indicates that translation at the word and named entity level can result in clusters with Arabic and English sentences that are highly likely to be similar to each other.

4.3 Conclusions

This chapter presented results on two different evaluations about finding similar sentences between Arabic and English text. The first focused on a sentence-level evaluation, using human judgments to determine whether a machine translated Arabic sentence should be replaced by a similar English sentence in the context of a summary. The second evaluation compared methods of similarity computation for clustering (possibly machine translated) Arabic and English sentences. Manually translated Arabic sentences and English sentences clustered by English Simfinder are used as the gold standard data. Runs using full ma-

Threshold	Avg. Precision	Avg. Recall	F-measure
0.2	0.747	0.170	0.276
0.3	0.715	0.170	0.274
0.4	0.710	0.153	0.251
0.5	0.748	0.147	0.245
0.6	0.786	0.135	0.231
0.7	0.817	0.144	0.244
0.8	0.823	0.149	0.253
0.9	0.823	0.150	0.254

Table 4.12: Precision and Recall between Arabic–English data and corresponding aligned English–English data using token and IdentiFinder features with Buckwalter and probabilistic translation.

chine translation and similarity computation using English Simfinder, SimFinderML with untranslated Arabic text using only the token feature, and SimFinderML with untranslated Arabic text using token and named entity features were presented.

At the start of this chapter, I listed the hypotheses that I would test using empirical data. Section 4.1 dealt with using machine translated Arabic sentences and finding similar English sentences to replace the Arabic sentences in a summary. A task like this is useful if there are Arabic documents and English documents reporting on the same topic; the English can be used to improve the readability and understandability of a summary of the Arabic documents. Such a summarization system is presented in Section 5.3. In this chapter, I examined whether:

1. Replacing machine translated Arabic sentences with similar English sentences in a summary improves the summary
2. Chunking machine translated Arabic text improves similarity results
3. Simfinder computes more accurate similarity values for finding similar sentences than a cosine-based metric

The first hypothesis was validated when about 68% of replacement English sentences suggested by Simfinder using full machine translated Arabic sentences with similarity values from the component chunks improves summaries based on human judgments. Chunking the Arabic text into smaller units introduces a problem of lack of context, and was not shown to greatly improve results over full Arabic sentences (62% for chunked sentences with a filter vs. 59% for full sentences.) Computing a similarity score for a full sentence using the similarity scores of the sentence's component chunks does show a dramatic increase though – up to 68%. Simfinder was also shown to outperform a competing cosine-based metric for similarity computation. While the cosine-based metric has high precision at high thresholds, recall is particularly low, with only one tenth the number of sentences found as with Simfinder, rendering the metric unusable for the task. The cosine-based metric was integrated as a filter for Simfinder, improving performance in chunked similarity metric, and complementing Simfinder well.

The second evaluation showed how SimFinderML can be used with untranslated Arabic text to cluster Arabic and English sentences together. I tested the following hypotheses in that evaluation:

1. Computing similarity between Arabic and English sentences using SimFinderML can be as or more effective than performing machine translation on the Arabic text and using English Simfinder to compute similarity

2. Probabilistic translation lexicons in SimFinderML improve performance over dictionary lookup

3. Using named entity detection in SimFinderML improves Arabic–English similarity results over using only tokens for similarity computation

The second hypothesis was validated as using probabilistic translation alone or Buckwalter and probabilistic translation together improves precision over dictionary lookup alone – using both token and named entity features with Buckwalter and probabilistic translation improves precision from 49.1% with just Buckwalter translation to 81.7%. While the third hypothesis could not be validated, over multiple thresholds, runs with the named entity feature were not statistically significantly different from runs with only the token feature, the named entity feature does not hurt performance. In fact, adding support for a named entity feature improves recall (significantly using just Buckwalter translation) with minimal impact on precision. The best performing configuration at the threshold 0.7, empirically determined to be a good operating threshold, uses the token and named entity features with Buckwalter and probabilistic translation, has 81.7% precision compared to 80.0% precision using only the token feature with Buckwalter and probabilistic translation.

The first hypothesis is validated if we consider precision alone, as the best performing SimFinderML run, at 81.7% precision at a threshold of 0.7 is significantly better than the run using full machine translation, 66.5% precision also at a threshold of 0.7. In fact, the precision of Arabic–English clustering using the named entity and token features with probabilistic and Buckwalter translation is close to the upper-bound precision of clustering two runs of the same manually translated Arabic and English data: 84.6%. Precision is more important than recall in the context of multi-document summarization which these techniques are applied to in Chapter 5.

This chapter showed that the similarity metrics I have devised are able to find English sentences similar to machine translated Arabic sentences, and to determine when it is appropriate to replace them to improve readability. The utility of SimFinderML for computing similarity between sentences in different languages for a clustering task was also shown. SimFinderML's performance was shown to be better than using full machine translation with English Simfinder for tasks where precision is preferred over recall, and SimFinderML was shown to be almost as precise as clustering using manually translated Arabic.

Chapter 5

Multi-Lingual Multi-Document 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 [GN00], 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 multi-lingual environment has become possible.

I introduce the approach of using text similarity for multi-document summarization where there are two sources of documents: a foreign language source, and an English source. I have implemented two summarization systems:

- a system that builds a summary of the foreign language text, and replaces sentences in the summary with a similar sentence from the English text when possible, and
- 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 Simfinder to compute similarity. The second system is tested using machine translated text, and over untranslated Arabic text with similarity computed by SimFinderML. 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.

5.1 Related work in multi-lingual, multi-document summarization

Previous work in multi-lingual document summarization, such as the SUMMARIST system [HL99] extracts sentences from documents in a variety of languages, and translates the resulting summary. Chen and Lin [CL00] describe a system that combines multiple monolingual news clustering components, a multi-lingual news clustering component, and a news summarization component. Their system clusters news in each language into topics, then the multi-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 Chinese and 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 multi-document summarization system [RJB00] 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 MultiGen summarization system [BME99] 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.

5.2 DUC 2004 Arabic Corpus

Both Sections 5.3 and 5.4 use data from the DUC 2004 corpus for evaluation. I briefly describe the corpus here, and describe an evaluation method using this data in Section 5.4.2.1.

In 2004 the Document Understanding Conference (DUC) added a multi-lingual track to the tasks offered to participants. The multi-lingual track had participants create English summaries for machine translated Arabic documents. The DUC had the LDC collect documents from the TDT corpus on 24 topics in Arabic, which were then manually translated into English, as well as translated by two statistical machine translation system (IBM and ISI.) For each of the 24 topics, 10 Arabic documents were collected, and in addition, 10 English documents on the same topic were found. Four different subjects created 100-word manual summaries for each of the multi-document sets based on the manual translations of the Arabic documents which are used as model summaries. Ten word (headline) summaries and 100-word summaries were also created for each of the Arabic documents.

All of the Arabic documents were machine translated into English using both ISI's statistical translation system and IBM statistical translation system, in both "bestonly" and variant forms which list translation probabilities for different paths through the machine translation systems. In all cases, I've used only the "bestonly" variant of the translations, which had the overall highest score from each translation system.

5.3 Summarizing Machine Translated text with Relevant English Text

In this section I present a multi-lingual multi-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 Arabic 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 5.1: A system suggested replacement sentence for a machine translated Arabic 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 5.1 shows an example of an Arabic 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 Arabic text. I performed a 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 [LH03] summary evaluation metric. I show that 68% of sentence replacements improve the resulting summary, and that our similarity-based system outperforms a state-of-the-art multi-document summarization system and first-sentence extraction baseline.

5.3.1 Summarization Approach

Our approach relies on first translating the input documents (Arabic, in this work) 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 multi-lingual summarization is applicable. This thesis does not address the issue of obtaining on-topic document clusters; news clustering systems such as Google News[1], Columbia NewsBlaster[2], or News In Essence[3] demonstrate that this is feasible. The system architecture is:

1. Syntactically simplify sentences from related English documents, and possibly chunk machine translated Arabic 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 Arabic 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 multi-document summarization system, DEMS [SNM02], to select the sentences for the similarity computation process.

5.3.1.1 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

[1]http://news.google.com/

[2]http://newsblaster.cs.columbia.edu/

[3]http://NewsInEssence.com/

sentence simplification software [Sid02] 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 fine-grained matching between the Arabic and English sentences, without including additional information from long, complex sentences that is not expressed in the Arabic sentence.

For example, for the following Arabic 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 Arabic sentence.

I examined using two types of sentence simplification, *syntactic* and *syntactic with pronoun 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 unsimplified text, possibly due to errors introduced by the pronoun resolution, which has a success rate of approximately 70%. I present results of the system using only syntactic simplification.

Similarly, I performed experiments for splitting the machine translated Arabic text. I investigated two methods for splitting Arabic text: one tags the text with TTT[4] 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 filtered from the output. The copying method was approximately 3% better on the manual evaluation below, so I omit results from the other chunking method.

5.3.1.2 Similarity Computation

Text similarity between the translated and relevant text is calculated using Simfinder [HKH+01]. Simfinder is a tool for clustering text based on similarity computed over a variety of lexical and syntactic features. The features used in Simfinder are the overlap of word stems, nouns, adjectives, verbs, WordNet [MBF+90] 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 Simfinder to compensate for using machine translated text as input, although the machine translated text is quite different from the news text used to train Simfinder.

5.3.1.3 System Implementation

Our summarization system can be run in multiple configurations.

1. Use DEMS to select Arabic 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 Arabic sentences, sorted by similarity to English sentences, and replacing each one by the single most similar English sentence.

[4]http://www.ltg.ed.ac.uk/software/pos/

2. Use DEMS to select Arabic sentences, replace only sentences above empirically determined threshold of 0.6 passing a cosine filter with similar English sentences, retain non-replaced Arabic sentences in the summary.
3. Use all Arabic 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 filter. 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 Arabic 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 in Section 5.3.2.1.

5.3.2 Evaluation

I performed evaluation at two levels: the sentence level to test the proposed sentence replacements of Arabic 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.

I use the 2004 DUC corpus for both the sentence and summary level evaluations. The corpus contains 24 topics with relevant documents, some in English and some in Arabic, and machine translations of the Arabic documents into English from 2 different systems. As part of the corpus, each of the sets contains summaries by 4 human assessors who read manual translations of the Arabic documents. These 4 summaries are used as the reference models against which the automatic summaries are evaluated – note that the model summaries were created only with knowledge of the content from the Arabic documents, and not the English documents. In Section 5.3.2.2 I evaluate the similarity-based summarization system using ROUGE.

5.3.2.1 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 Arabic text score. Our similarity-based system was run with simplified English sentences and full machine translated Arabic sentences.

5.3.2.2 Summary level evaluation results

Table 5.1 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 multi-lingual system task, the 95% confidence interval

[5]version 1.2.1, -b 665

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 5.1: 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 out-performing 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 Arabic text. In contrast, simply running first sentence extraction and DEMS on the related English text does not perform as well; using the machine translated Arabic 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 Arabic 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 filters, 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 Arabic text using the Arabic 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 outperforms 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 Arabic and 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 Arabic 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.

5.4 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 languages. In this case, our system works with Arabic and 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 Arabic 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 Arabic sources only, and information that appeared in both English and Arabic 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 errorful Arabic 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 quantifies 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 a pyramid representation [NP04] of the articles, separated by source. As before, I use Arabic and English documents from the DUC 2004 multi-lingual corpus [OY04] for the experiments here.

5.4.1 System Architecture

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

- a set of untranslated Arabic documents with a set of English documents, or
- a set of manual or machine translations of Arabic documents with a set of English documents.

When using Arabic documents CAPS uses SimFinderML to compute text similarity, or the English version of Simfinder when using manual or machine translations of the Arabic 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 unsimplified English text can be used.

Figure 5.2 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 determines 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 5.2, CAPS first simplifies the input English sentences. It does not simplify the translated Arabic 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.

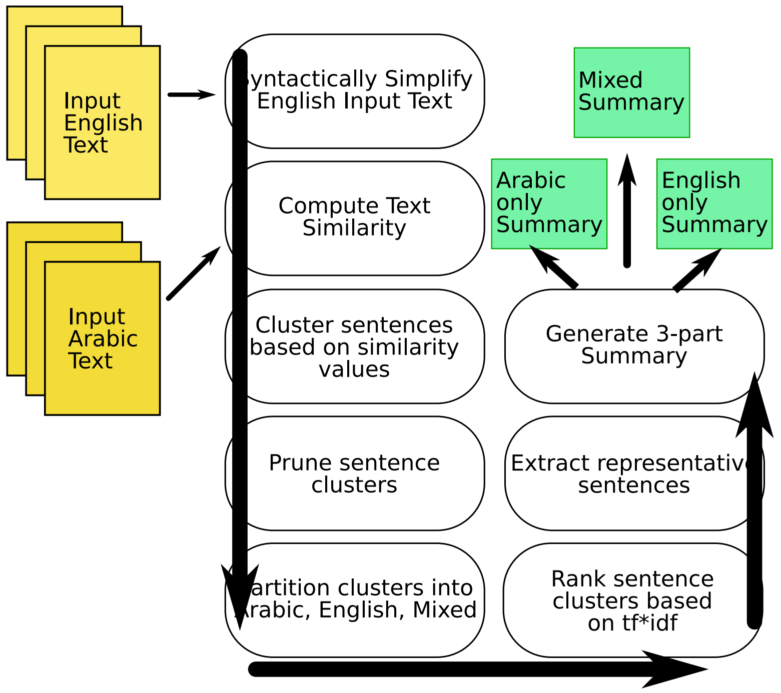

Input English Text
Input Arabic Text
yntactically Simplify English Input Text
Compute Text Similarity
Cluster sentences based on similarity values
Prune sentence clusters
artition clusters into rabic, English, Mixed
Rank sentence clusters based on tf*idf
Extract representativ sentences
Generate 3-part Summary
Arabic only Summary
Mixed Summary
English only Summary

Figure 5.2: 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 Arabic sources, and multiple clusters of sentences from both English and Arabic sources. Finally, I rank the sentences in each source partition (English, Arabic or mixed) using a TF*IDF score [Sal68]; 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 a 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.

5.4.1.1 Sentence Simplification to Improve Clustering

As with the summarization system presented in Section 5.3, 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 a finer grain than would otherwise be possible. I use a sentence simplification system developed at Cambridge University [Sid02] for the task. The generated summary often includes only a portion of the unsimplified sentence, thus saving space and improving accuracy. I opt to 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.

5.4.1.2 Text Similarity Computation

Text similarity between Arabic and English sentences is computed using SimFinderML, 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 Arabic–English version of SimFinderML used in this work. Text similarity between manual or machine translated Arabic documents and English is computed with Simfinder, an English-specific program for text similarity computation that SimFinderML was modeled after. Simfinder for English is presented in Chapter 2. In addition, I present a third baseline approach using the cosine distance for text similarity computation.

5.4.1.3 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 [SNK04]. 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 horrified" 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 horrified" about the killing of four Western hostages in Yemen, including at least three Brits.
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.

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.

5.4.1.4 Identifying cluster languages

The final summary that I generate is in three parts:

- sentences available only in the Arabic documents
- sentences available only in the English documents
- sentences available in both the Arabic and English documents

After producing the sentence clusters, I partition them according to the language of the sentences in the cluster: Arabic only, English only, or Mixed. 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 Arabic documents, and the English documents, and concepts that are supported by both Arabic and English documents.

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

5.4.1.5 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 can fit in a single summary; average input data set size is 7263 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, Arabic, English, and mixed, the clusters are ranked according to a TF*IDF score [Sal68]. 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 Reuters news.

CAPS has 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 Arabic text in the input and text similarity computation phases, the Arabic text is translated into English after the clustering phase. TF*IDF counts are computed over the machine translated Arabic text. This is done because the ranking of clusters has to be done over Arabic, English, and mixed clusters, which presents a problem: how to rank the Arabic and mixed clusters? For Arabic-only clusters, a TF*IDF approach using IDF values from a large Arabic 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 Arabic 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.

5.4.1.6 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.

In order to generate a fluent summary, CAPS draws from the English sources as much as possible. For summary sentences from clusters with only Arabic sentences, clearly nothing can be done to improve upon the machine translated Arabic. But when generating the summary from mixed English/Arabic clusters, CAPS uses English phrases in place of translated Arabic when the similarity value is above a learned threshold, a is the case for the pruned clusters. Section 4.1.2 shows that this method improved summary quality in 68% of the cases in a human study.

5.4.1.7 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 (English, Arabic, and mixed). 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 6 Arabic clusters, 24 English clusters, and 12 mixed clusters, then the ratio of sentences from each partition is **1 Arabic : 2 mixed : 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.

The summary is built by extracting the number of sentences specified by the ratios computed above, and cycling continuously until the word limit has been reached. Representative sentences are chosen based on the cluster rankings computed as explained previously.

5.4.2 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., [LH03], [OY04], [RTS$^+$03]). 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 Arabic partition actually unique to only the Arabic 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) [NP04] 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 Arabic, 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.

5.4.2.1 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 languages. To do this, I first need to characterize the content of the documents in Arabic and 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) [NP04] 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.

All 20 documents (10 Arabic and 10 English) and 4 summaries of 10 sets (a total of 240 documents) of the DUC data were annotated by volunteers in the Natural Language Processing group here at Columbia. Annotators marked SCUs in the English source and in the *manual translations* of the Arabic sources, which was available in the DUC dataset. Machine translations were too difficult for human annotators to understand and reliably mark in the SCU identification task.

5.4.2.2 Characterizing Arabic and English content by SCUs

This section deals with how the content differs from the Arabic and English documents in the sets. Appendix A details the annotation process applied to the DUC sets. The 10 Arabic 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 3 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 Arabic contributors goes into the Arabic pyramid, and SCUs that contain contributors from both Arabic and English documents are placed in the mixed pyramid.

Of the ten DUC sets that have been annotated, Table 5.2 lists the relative sizes of the language pyramids for each set. The SCUs column lists the number of SCUs in the

Set Number	English		Arabic		Mixed	
	SCUs	Contributors	SCUs	Contributors	SCUs	Contributors
d1003t	199	272	26	36	29	155
d1005t	21	31	28	28	4	86
d1011t	136	280	87	140	42	681
d1012t	213	281	43	53	17	176
d1018t	87	163	23	32	49	495
d30003t	44	80	3	7	19	145
d30040t	6	38	3	8	19	249
d30042t	106	180	44	60	60	474
d30053t	41	90	52	72	32	191
d31001t	6	14	15	25	22	184

Table 5.2: Sizes in number of SCUs / number of contributors for different pyramids based on language-breakdown of different DUC 2004 sets.

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 Arabic and 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 Arabic pyramid. The partitioning of the manually annotated pyramids show that the majority of the time the English language documents more unique information than the Arabic documents, but there is still information that is only reported in the Arabic documents, and that has support from both Arabic and English documents.

5.4.2.3 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 Arabic and 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 or both languages, 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 language are:

Mixed Arabic and English:

- SCU 14 weight 31: The crime in question is a bombing
- SCU 17 weight 24: The bombing took place in 1988
- SCU 36 weight 22: Anan expressed optimism about the negotiations with Al Kaddafy

English only:

- SCU 57 weight 6: Libya demands the two suspects will serve time in Dutch or Libyan prisons
- SCU 121 weight 5: Libyan media reported that Al Kaddafy had no authority to hand over the two suspects
- SCU 128 weight 5: Libyan media is controlled by the government

Arabic only:

- SCU 53 weight 6: Kofi Anan informed Madeleine Albright about the discussions with Al Kaddafy
- SCU 21 weight 4: The plane involved in the bombing was an American plane
- SCU 82 weight 3: Kofi Anan visits Algeria as part of his North African tour

The SCU ID is a unique identifier for the SCU, and the weight is the number of different contributors for the SCU from all documents. The mixed language partition contains the highest weighted SCUs, which give important basic details about the article set: there was a bombing in 1998, and Kofi Anan and Al Kaddafy are involved in negotiations about the event. Only the top three SCUs were included in this example, but other SCUs contain more information about the negotiations over the bombing suspects. The English language SCUs are about the bombing suspects, of interest to American audiences who desire prosecution, and that the Libyan media is state-controlled, which would not generally be stated for a Libyan audience. The Arabic SCUs state that the plane is an American plane, and information about Kofi Anan's visit to North Africa.

With the above SCU pyramids for each language partition (Arabic, English, mixed) as described above, to determine how well CAPS clusters and partitions sentences I compare the per-language partition output of CAPS to the SCU pyramids.

Given the ranked set of clusters of each type (Arabic, English, mixed) I compare the SCUs found in the sentences of each cluster to the manually annotated SCUs of each language-specific pyramid. The comparison can only be made directly when using the human translated Arabic text and non-simplified relevant English text; in the other cases either some approximation must be made to identify the SCUs in the text. When dealing with machine translated text, I identify SCUs by replacing the machine translated sentence with its manually translated counterpart. For the simplified English text, I use a dynamic programming algorithm for identifying the longest common substring between the annotated text and the simplified English sentence which finds the source sentence for the simplified sentence. The SCUs are then read from the source unsimplified sentence.

For example, for a cluster marked as "Arabic only", I first determine the SCUs that are associated with the sentences in the cluster. Since the SCU annotation has been performed over the manual translations of the Arabic documents, I need to use a sentence alignment mapping that maps machine translated sentences to their counterpart in the manual translation. For each sentence in the cluster, I map that sentence back to the sentence from the manual translations (which contains the SCU annotation) and identify which SCUs came from that sentence. I collect all SCUs for all of the sentences in the cluster, and compute the SCU score (a score that reflects the importance of the sentences based on the weighted SCUs in the pyramid) against the Arabic-only pyramid. *Note that this automatic technique assumes that all the information conveyed in the manually translated sentence is also conveyed by the machine translated sentence.* I then compute the percentage of the SCUs that occurred in the Arabic-only pyramid, which is basically recall of the "Arabic only" SCUs.

This process is repeated for the mixed-source clusters and for the English-only clusters (although, clearly, no alignment is needed for the English sentences.) I compared similarities produced by CAPS against a baseline using the cosine distance as a similarity metric.

Evaluating English clusters is done in the same manner as Arabic 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 can not just determine the SCUs for a sentence by reading the annotation file. 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 off 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.

If the English text is not first simplified, the SCUs are read off directly from the annotation file.

Mixed clusters are evaluated as described above, using SCU gathering techniques appropriate to the type of sentence in the cluster.

Aligning DUC Manual Translations and Machine Translations

In addition to aligning Arabic sentences with their manually translated counterparts in Section 4.2.1, I had to align the machine translated sentences with their manually translated counterparts for this evaluation. I performed the alignment using a modified Gale and Church algorithm that uses dynamic programming to find the best alignment of sentences based on costs derived from character lengths of the sentences being aligned [GC91].

5.4.2.4 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 7 SCUs, the max score is the sum of the weights of the 7 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 off directly from the annotation file because they are identical.

For machine translated text as input, I identify the manual sentence aligned to the machine translated sentence, and read the SCUs from the annotation file that the manually translated sentence was labeled for. For input given in Arabic, which is machine translated after the clustering and pruning phase, I have manually annotated those for SCU scores.

5.4.3 Results

5.4.3.1 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 untranslated Arabic 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 Arabic–English data, which do not take advantage of the unique aspect of my system to summarize the similarities and differences between the Arabic and English documents. Using two document selection strategies, I used DEMS [SNM02], 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 Arabic documents, and select individual English documents with a cosine similarity of 0.70 or greater to the Arabic centroid. If fewer than two documents have a similarity of 0.70 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 Arabic 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 Arabic–English summaries.

The run of CAPS using manually translated Arabic documents contained sentences that covered 25.88% of the SCUs in the Arabic SCU pyramid. Given that the summaries are approximately 10% of the input text, these are perfectly acceptable recall figures. A maximal score of 1.0 would be achieved if the extracted sentence segments contained every single SCU in the pyramid. This does not happen in practice though, since not all sentences in the input documents are in the clusters; sentences that are not highly similar to other sentences are dropped. Approximately 45% of the input text does not end up in a cluster, however, almost all of the input text was annotated (although some non-relevant phrases were not annotated at the annotators' discretion.) Also, only the representative sentence is

Run identifier	Arabic	English	Mixed
Manual (CAPS)	0.2588	0.2862	0.2387
Machine (CAPS)	0.1974	0.2659	0.1195
Machine cosine	0.1909	0.0798	0.0167
Untranslated (SFML Arabic–English model)	0.0542	0.1141	0.4076
Untranslated (SFML mixed model)	0.0174	0.0630	0.4249
English-only (all documents)	0.0000	0.1420	0.3875
English-only (similar documents)	0.0000	0.1561	0.3826

Table 5.3: Pyramid scores of representative sentences from every cluster scored against entire corresponding language SCU pyramid. These figures represent the recall of each partition.

output for each cluster, and the chosen representative sentence might not contain as many SCUs as other sentences in the cluster.

The first table answers the question "how many SCUs for the language partition were found?" while the second table answers the question "How important are the SCUs that were found?" for each language partition. For the set of clusters in each language partition I compute pyramid scores by comparison against the pyramid for that partition. Table 5.4 shows the micro-averaged Pyramid score normalized by the number of SCUs in the clusters for each language. The micro-average is the total weight of all cluster SCUs across all document sets divided by the total max of SCU scores across all sets. I use a micro-average instead of a macro-average (just averaging results from each set equally) because the sets are of different sizes. Micro-averaging weights large sets more than smaller sets. This normalized score indicates how important the SCUs the system covered are; a maximal score of 1.0 is achieved by choosing the highest weighted SCUs. Some SCUs are clearly less important than others, as illustrated by one of the low-weight SCUs from the Lockerbie set:

SCU 236 weight 1: Prince Philip is the queen's husband

The run of CAPS using manually translated Arabic documents performs the best at identifying material that is exclusive to either source, and shared between the two sources.

Run Identifier	Arabic	English	Mixed
Manual (CAPS)	0.7748	0.7881	0.6417
Machine (CAPS)	0.7521	0.7585	0.5765
Machine cosine	0.6519	0.5377	0.3615
Untranslated (SFML Arabic–English model)	0.4717	0.6936	0.8039
Untranslated (SFML mixed model)	0.7273	0.5422	0.8276
English-only (all documents)	0.0000	0.5982	0.7817
English-only (similar documents)	0.0000	0.6212	0.7641

Table 5.4: Micro-averaged SCU Pyramid scores of representative sentences from every cluster scored against corresponding language pyramid, normalized for number of SCUs. These figures represent the importance of the per-partition information extracted by the system.

The system has more difficulty in identifying content that is shared between the two languages, which is not surprising given the data; the annotation task was very difficult and the annotators used much world knowledge and inference in connecting the SCUs.

Using machine translated documents lowers performance (line 2, Table 5.4), particularly in the Mixed partition. The Mixed partition is difficult because there is considerably more English text than Arabic text in the document sets, and when the machine translated Arabic text is not similar enough to the English, it is dropped from sentence clusters.

The cosine text similarity baseline performs much worse than CAPS for the English and Mixed partitions, and slightly worse for Arabic. While it covers approximately the same number of Arabic SCUs, the SCUs that it chooses are much worse, as is reflected in the micro-average pyramid score. The CAPS run with machine translated documents performs almost as well as the run with manually translated documents for the Arabic and English partitions, and only drops off for the Mixed partition.

There are two runs that use untranslated text using SimFinderML. Both runs using SimFinderML (SFML in the table) use tokens and Named Entities as features, and both Buckwalter and probabilistic translation. The first run, Untranslated (SFML Arabic–English model), used a feature merging model that was trained using only examples of

Arabic sentences similar to English sentences (the training data in Section 3.4.4.) The second run, Untranslated (SFML mixed model), used a feature merging model that was trained with the Arabic–English similar sentences in the first run, and additional examples of English–English similar sentences (described in Section 6.2.3.1.)

The Arabic–English SFML model performs much better at identifying mixed Arabic–English content than any of the other runs, even performing better than the run with manually translated text. It does not perform well at identifying English–English similarity, although it does perform better than the cosine baseline. As the feature merging model did not contain any information on English–English similarity, this is not overly surprising. Similarly, it performs poorly for Arabic–Arabic similarity, not even performing as well as the cosine baseline. The SFML mixed model run contains English–English training data, which I anticipated would improve English performance, but that was not the case; mixed partition performance improved, but both Arabic and English performance got worse. This unexpected result could be attributed to the use of a single feature merging model for both languages; if the trends in the Arabic–English and English–English data for indicating similarity over the two features used are not the same, the conflicting data introduced by the English–English examples compared to the Arabic–English data could just degrade performance. This indicates that using separate feature sets and feature merging models within a language would be a big improvement for SimFinderML.

Both untranslated runs using SimFinderML perform very well in the micro-averaged SCU Pyramid scores — outperforming the manual run at 0.80 compared with 0.64. They both also outperform the cosine baseline for English, and the mixed model outperforms the cosine baseline for Arabic.

While SimFinderML does not perform well at Arabic–Arabic or English–English similarity, which is partly due to the training data used, it performs the best of any method at identifying inter-language similarity, one of the design goals for SimFinderML. Section 6.3 deals with future improvements that can be made to address this deficiency.

As expected, the English-only baseline does not receive any score for the Arabic language, as it uses none of the Arabic language documents. The coverage from Table 5.3 on English is good, better in both cases than the Untranslated (SFML) systems, and the

Run identifier	Arabic	English	Mixed
Manual (CAPS)	0.43	0.41	0.52
Machine (CAPS)	0.40	0.43	0.47
Machine cosine	0.46	0.47	0.24
Untranslated (SFML Arabic–English model)	0.17	0.44	0.31
Untranslated (SFML mixed model)	0.10	0.42	0.28
English-only (all documents)	0.00	0.34	0.32
English-only (similar documents)	0.00	0.36	0.32

Table 5.5: Precision scores of representative sentences from every cluster scored against corresponding language SCU pyramid

machine-translated cosine-based system. Summarization from only the English documents does not perform as well as the CAPS systems using either machine or manually-translated documents, which might be surprising. In the case of the extractive systems though, it is not informed by the similarity of the document sets; more the extracted sentences are in the “mixed” category, and not enough focus on information that is unique to the English language documents exists because the system does not know what information in the document set is shared with the Arabic documents. The English-only summaries do perform better than the CAPS systems on the “mixed” language partition, but not quite as well as the Untranslated (SFML) systems. Again, the Untranslated (SFML) systems strength is in their identification of mixed-language content. The CAPS system is flexible and emphasis of one language partition or another can be shifted based on the similarity metric used (SFML or English SimFinder over manually or machine translated text.)

Precision evaluation Table 5.5 shows the precision of the systems for the sentences selected for each language partition. Table 5.3 can be thought of as the recall for the pyramids; of all the SCUs in each language pyramid, how much of the pyramid was covered by the sentences that were selected for that language partition? Table 5.5 is the precision for the selected sentences: for each sentence that the system says belongs in one partition (Arabic, English, or Mixed) how accurate is the system in making that decision? The precision in this case is, for each language partition, how many sentences match to SCUs in that language partition, divided by the total number of sentences the system suggests. In some cases, not all sentences that the system suggests are matched to the SCU pyramid; for example, if the extracted sentence was not marked up by the human judges, then it will not match to any SCUs in the pyramids, and lower precision overall. Note also that it is difficult to score well under this metric: an SCU will be labeled as "mixed" if only one contributor out of many (possibly as many as 30 for highly weighted SCUs in this data) is from Arabic documents (for SCUs with mostly English contributors.) The distinctions that the judges make in the annotation are often very fine, and the system is not expected to perform as well as humans at the SCU annotation task.

The CAPS system using the cosine metric performs better than either CAPS systems with Simfinder for Arabic and English in the precision numbers from Table 5.5, but in the English case it has significantly lower coverage than either system with Simfinder (see Table 5.3.) When looking at the other systems performance over the precision numbers in Table 5.5, surprisingly the CAPS system with Manual translations performs worst of all the runs in accuracy at identifying English sentences. It does perform significantly better than any other system at identifying mixed Arabic and English content though. Both systems using untranslated Arabic text with SimFinderML are significantly worse than the systems using manually or machine translated Arabic text for Arabic similarity identification. As discussed before, this isn't surprising as the similarity models for SimFinderML were not trained with any Arabic–Arabic similarity data. Both SimFinderML systems perform about the same as the CAPS system with machine translated Arabic text at identifying sentences from English-only SCUs, and both beat the cosine system for mixed sentences. That the two SimFinderML systems can beat the cosine baseline for mixed Arabic–English sentence identification is encouraging, because they used Arabic–English training data. The addition of Arabic–Arabic training data should improve scores on the Arabic–Arabic similarity task.

The English-only extractive baseline runs performed the worst of all the systems at identifying sentences annotated as SCUs that are only supported by English documents. In the English-only runs, approximately a third of the sentences were in the English-only pyramid, a third in the Mixed pyramid, and a third were not annotated in the gold standard. The precision of the Mixed partition is better than the cosine-based system, and approximately the same as the Untranslated (SFML) systems, but not as good as either manually or machine translation-based systems.

5.4.3.2 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 are:

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 multi-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 Simfinder in the CAPS framework. Table 5.6 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.

When using the manual translations of the Arabic documents, the CAPS system performs much better than the first sentence extraction baseline. The first sentence extraction systems perform well on this data as the first sentence of the news articles tend to include the important information from the document set that is heavily weighted in the SCU pyramid. The CAPS system, however, performs better than the first sentence extraction

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 5.6: 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.

The untranslated runs using SimFinderML do not perform as well as the runs using full machine translation, however, the SimFinderML run using the Arabic–English model does perform better than the cosine baseline using machine translation. It does not perform as well as the first sentence baseline though. The SimFinderML runs do not identify as much English and Arabic specific clusters as the other runs, which hurts them in this evaluation, as much of the highly-weighted SCUs come from the English side of the documents. See Table 5.2 which lists relative sizes in the gold standard of SCUs from the language partitions. Across all ten sets, only two had the majority of the SCUs from the mixed language partition, which is what the SimFinderML-based runs excel at identifying.

Both of the English-only extractive summary baselines perform poorly, on par with the Untranslated (SFML mixed model) scores. The pyramid score is composed of all three languages, so both of the English-only systems miss out completely on the portion of the score that is composed of Arabic information. Since a large portion of the score is from the "mixed" pyramid though, both systems are able to make up for missing data from the Arabic pyramid by performing well on the English and Mixed portions.

5.4.3.3 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 Arabic and syntactically simplified English, as those runs contain the most understandable Arabic, making it easier to see the differences in content between the Arabic and English sources. The summary is an 805 word summary, 54% English, 29% Arabic, and 17% 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.

From Arabic sources only:

This is also the first time an Islamic group conducts a kidnapping operation in Yemen, where armed tribes usually conduct such operations.
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."
Police had earlier announced that the "Islamic Jihad" group kidnapped the Western tourists and demanded the release of its leader in addition to lifting the embargo imposed upon Iraq.
Some 150 foreigners were kidnapped in Yemen in the past few years, and they were released without shedding any blood.
London 12-29 (AFP) - British Prime Minister Tony Blair announced today, Tuesday, that he was "shocked and horrified" about the killing of four Western hostages in Yemen, including at least three Brits.
He explained that Al-Atwani was detained two weeks ago after armed clashes between individuals from the Islamic Jihad group and Yemeni security forces in the Abyan region, more than 400 km south of Sana'a.
In London, British Foreign Secretary Robin Cook announced this evening that the four Western tourists who were killed in Yemen are three Brits and one Australian.

From English sources only:

‘They said members of the group flogged men for selling and drinking an alcohol.’
The kidnapping occurred Monday near the southern town of Mawdiyah, about 200 kilometers (175 miles) south of the capital, San‘a.’
Yemeni officials on Tuesday started negotiations with Islamic extremists for the release of 16 Western tourists while security forces encircled the area where the militants are said to be holding the hostages.
Tribal leaders said about 10 gunmen ambushed a convoy of five vehicles carrying a group of 17 tourists traveling near the southern town of Mawdiyah, in the Abyan province.
‘Security officials in Abyan, also speaking on condition of anonymity, said al-Atwi’s arrest was part of a crackdown on some Islamic vigilantes.’
‘They said the kidnappers ambushed a convoy of five vehicles in which the tourists were traveling and opened fire on a number of policemen escorting them.’
The government two weeks ago arrested their leader.
‘During the melee, the British tour leader and a Yemeni guide escaped and raised alarm.’
‘The kidnappers are also demanding the release of another leader.’
The Foreign Office advised that British nationals traveling to Yemen should “keep in touch with developments” during their stay.
“We have made it clear that our top priority is the safe and swift return of the hostages,” the spokesman said on condition of anonymity.
These Islamic extremists had taken the law into their hands by enforcing strict Islamic rules on the population of southern Yemen.
Gov. Ahmad Ali Mohsen of Abyan province, where the kidnapping took place, is talking with leaders of the Al-Fadl tribe, to which the kidnappers belong, an official at the governor’s office said.
‘The kidnappers are demanding more schools, hospitals and telephone lines in their area.’
‘Yemen was once a haven for Islamic militant fugitives from other countries.
This Alcohol is forbidden in Islam.

(Summary sentences from English sources only continued)

‘Four Germans currently are being held by a tribe in northeast Yemen.’

‘The gunmen threatened to kill the hostages if the police didn’t back off, said one official.’

This happened when police stormed the hideout of some islamic militants.

‘The group also ran a military camp in southern Yemen, said the officials.’

In London, a Foreign Office spokesman said Britain was in touch with Yemeni authorities.

But I cannot say any more,” PA quoted the operator as saying.

One of these wounded hostages was a Briton.

‘The security sources did not identify him.’

‘The hostages are usually released unharmed.’

But the government has expelled many of them.

These Two others were seriously wounded in the clash.

From both Arabic and English sources:

Sana’a 12-29 (AFP) - A new tally obtained from a security source showed that six people, including three Western tourists, were killed and seven others were wounded, including two tourists, in the attack that was carried out today, Tuesday, by Yemeni security forces on the kidnappers of 16 foreign tourists.

The Yemeni security official announced that ”the kidnappers are members in the Islamic Jihad.”

Islamic militants kidnapped 16 Western tourists in southern Yemen Monday, including 12 Britons, two Americans and two Australians, security officials said.

The six dead are two British tourists, an American woman, a policeman and two kidnappers who belong to an Islamist group.

Two of the kidnappers were also killed.

This happened when Yemen security forces attacked the kidnappers.

‘The remaining tourists were two American women and two Australian men.’

5.4.4 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 and Arabic sources. Unlike previous summarization systems, CAPS explicitly identifies agreements and differences between English and Arabic 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 Arabic and 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 SimFinderML, CAPS is able to use non-translated Arabic text as input, deferring translation until after sentences have been clustered, reducing the number of sentences that need to be translated. Using SimFinderML and untranslated input, CAPS out-performs all other methods for identifying information that is supported by both Arabic and English sources, a 0.8276 micro-averaged SCU pyramid score, compared to the next best 0.6417 using manually translated Arabic documents.

SimFinderML 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 multi-lingual summarization system based on CAPS with SimFinderML 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 Arabic 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 6

Conclusions

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

6.1 Contributions

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

- Development of a flexible framework for experimenting with multi-lingual text similarity in SimFinderML.
- 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 different languages that goes below the sentence level.
- CAPS summarization system is applicable to any language pair for which machine translation systems are available, or a multi-lingual text similarity metric can be computed (e.g. SimFinderML.)

- Information that is supported by both languages is made easier to understand in the summary by selecting English sentences instead of machine translated Arabic sentences for the summary.
- CAPS approach is applicable even without machine translation systems available to summarize Arabic and English documents for bilingual analysts.

6.1.1 Linguistically motivated primitives

Chapter 2 introduces previous work on English text similarity that forms the basis of my multi-lingual text similarity research. This thesis presents SimFinderML, a system I developed that takes the ideas presented in English Simfinder, 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 WordNet [MBF$^+$90] and grammatical information via part-of-speech based primitives.

6.1.2 A flexible framework for experimenting with multi-lingual text similarity

Chapter 3 describes SimFinderML, my implementation of a cross-lingual text similarity computation system, and details my Arabic–English implementation. SimFinderML computes similarity over text at the level of primitives, easily identifiable classes of text such as nouns, verbs, WordNet synsets, or named entities. The primitives are linguistically motivated and SimFinderML 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 Arabic and English similarity: token level primitives, and phrasal primitives uses named entities.

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. Additionally, I tested the hypothesis that including a learned probabilistic translation lexicon improves performance over translation by dictionary lookup alone, and that using a named entity feature improves Arabic–English similarity.

Chapter 4 presents an evaluation of both English Simfinder and SimFinderML that investigates how well the English Simfinder and SimFinderML can identify similar sentences using translation at three different levels: word-by-word using SimFinderML, word-by-word and at the phrase level of extracted named entities, and using full machine translation with English Simfinder. The main SimFinderML hypothesis is validated when considering precision only: using SimFinderML with probabilistic and Buckwalter translation with token and named entity features resulted in a precision of 81.7% compared to a 84.6% using manually translated data, or 66.5% using machine translated data. Precision is more important than recall for the summarization tasks that use SimFinderML. The hypothesis is not validated for systems that also require high recall.

Using a probabilistic dictionary is shown to improve results over using dictionary lookup alone by increasing precision from 49.1% to 81.7% when using both token and named entity features. The hypothesis that adding the named entity feature improves Arabic–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 SimFinderML, using probabilistic and Buckwalter translation with tokens and named entity features, performed nearly as well considering precision only as the gold standard run of manually translated Arabic text using Simfinder: 81.7%, compared to the manual run of 84.6%. Using SimFinderML has much better precision than machine translation with English Simfinder, at 66.5%, although English Simfinder does have much better recall.

6.1.3 Multi-lingual text similarity for resource-poor languages

SimFinderML is designed to make adding support for new languages easy. The approach taken in SimFinderML is applicable to "resource poor" languages by using simple techniques for primitive identification, such as regular expression based tokenizers to identify token primitives for the language, and translation using a probabilistic bilingual translation lexicon learned from a parallel corpus. SimFinderML is able to use multiple translation methods. The best performing versions of SimFinderML use both a learned probabilistic translation lexicon, and an existing translation resource (glosses from the Buckwalter 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 (77.7% precision vs. 80.0% precision with both.)

6.1.4 CAPS: Summarization that identifies similarities and differences across languages

In the context of multi-document summarization, the test-bed application for SimFinderML, 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 Arabic 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 multi-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 is evaluated using English and Arabic language documents, and improves understandability by selecting English sentences for the summary for clusters with support from both English and Arabic documents. CAPS also breaks down information below the sentence level by applying syntactic simplification to the English text. CAPS is evaluated using both machine translated text with English Simfinder and Arabic text using SimFinderML. The approach using SimFinderML performs much better than using machine translation for identifying content shared between the Arabic and English text. It does not, however, perform as well at monolingual (English–English or Arabic–Arabic) content identification. CAPS receives an average 0.8276 SCU pyramid score for mixed content with SimFinderML, compared to 0.6417 using manually translated Arabic text, or 0.5765 for machine translated Arabic text. The SimFinderML approach works very well for the cross-language Arabic–English content identification task, validating one of the design goals of SimFinderML. The summarization approach used in CAPS is also applicable to any languages for which a similarity metric can be computed between the foreign language and English text. 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.

6.2 Limitations

In the remainder of this chapter, I will present some limitations on the work in this 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.

6.2.1 Experimentation with more Arabic primitives

The language pair investigated in this work is Arabic and English. One of the contributions of the original English Simfinder work was the use of multiple linguistically-motivated features used for similarity computation. The same approach is taken in SimFinderML, 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 Arabic and 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 languages, or primitives that make use of more knowledge-heavy linguistic resources, such as the corpora being produced by the Interlingual Annotation of Multi-Lingual Text Corpora project [MMD$^+$04].

6.2.2 Better translation for named entities

I use IBM's statistical Arabic–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 [KG97, SK98] 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 untranslated 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 Köhn's thesis work [Köh03]. The primitive translation phase should also include more support for fuzzy matching and partial matches. SimFinderML is not trying to detect only exact translations, but similar sentences which would benefit from a principled investigation of fuzzy matching for primitives across languages.

6.2.3 Per-language feature sets and merging models

As explained in Section 3.3.5 SimFinderML uses a single feature merging model when combining feature similarity values into a single similarity value. SimFinderML 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 Arabic units it should use a model trained with Arabic data, and for Arabic–English units, it should use a model trained using Arabic and

English translation data.

Currently, SimFinderML 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 languages and within languages.

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

6.2.3.1 Combining Arabic and English training data

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

Initial results from adding English training data to the Arabic–English training data lowered performance in both the English–English case and Arabic–English case. This indicates that just using a single set of features and feature merging model for both intra- and inter- language similarity computation is not the right direction.

The training results shown in Section 3.4.4 use a feature merging model trained using the correspondences between sentences in an aligned Arabic and English corpus. Since SimFinderML computes similarities between sentences within languages and across languages at the same time, ideally the training data for the feature merging model would use information from both intra and inter language similarity training data. I create a feature merging model using both the Arabic–English training data described in the section above,

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 6.1: Feature merging model training results using token and IdentiFinder features with Buckwalter and probabilistic translation using both English–English and Arabic–English training data.

and also the English–English training data described in Chapter 2. To combine the two sets of training data, the same feature similarity values are computed over both sets, and sentence pairs are noted as either “similar” or “not similar” based on the human judgments (English data) or alignment status (Arabic–English data.) Table 6.1 reports precision and recall for the training data that includes both training sets using Buckwalter and probabilistic translation with both token and IdentiFinder features, as that was the best performing combination from previous experiments.

Training with the English data results in lower precision and recall than training over just the Arabic–English data. The English–English data includes many examples that are quite difficult to classify automatically; training the English version of Simfinder also results in lower scores than the Arabic–English data, although due to additional features the English version of Simfinder performs better than this run using only includes token and IdentiFinder named entity features. For comparison, the model using only Arabic training data has 86% precision and 50% recall at a threshold of 0.7, whereas the model with Arabic and English training data has 67% precision and 13% recall. These results indicate that a single set of features and a single feature merging model are not appropriate in the multi-lingual case. Future work should investigate adding feature sets on a per-language basis (this is already supported in SimFinderML) with feature merging models that use the appropriate feature set merging model based on the languages of the sentences.

6.3 Future Work

This section explores other areas to be explored within the SimFinderML framework for multi-lingual text similarity where I have not yet performed much work.

6.3.1 Further integration of statistical machine translation methods

SimFinderML 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 SimFinderML's results at finding sentences that are translations of each other. However, since SimFinderML 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.

Adding a statistical machine translation decoder as a feature that estimates the probability of one sentence giving rise to another similar sentence in the same language is another area that deserves investigation. A large amount of training data of examples of English sentences re-written in different ways that convey almost the same information would be required to implement this kind of system, but mirrors the paraphrase identification task well.

6.3.2 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 SimFinderML and 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.

6.3.2.1 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 [Bou92] 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.

Jacquemin has used noun phrase variation techniques for automatic morphological processing [Jac97], and term normalization for text indexing tasks [Jac94, TKJ97, Jac99]. Jacquemin's variation framework is built on the FASTR system, which has implementations for French, English, Japanese, Spanish, and German versions. An initial experiment which integrated the French version of FASTR-identified noun phrases as a feature into SimFinderML did not improve results over a model that didn't use the FASTR noun-phrase feature. As I began to work more with Arabic, and since FASTR does not have an implementation available for Arabic, work on the detection and usage of noun phrase variants became less emphasized as the work continued. I would like to explore the identification of noun phrase variants more for intra-language similarity. At some level, noun phrase variation is dealt with in the translation stage for inter-language similarity; I would like to focus more efforts on statistical approaches for recognizing phrase translations.

6.3.3 Sense disambiguation

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

In some ways this might not be a large problem, as there is some natural disambiguation power inherent in the translation process; when comparing two sentences with words that have different senses, by matching against all the possibly translation only the senses that are relevant in the target sentence will have matches. In fact, there has been much work in the area of using bilingual corpora to perform monolingual word sense disambiguation based on the translations used in the corpus [DIS91, Dia03]. An interesting area of research would be to investigate whether adding a feature using word sense disambiguation software to SimFinderML would help improve results based on the more specific translation that could be done. Even more interesting would be to investigate whether SimFinderML could automatically determine the appropriate sense of a word for translation based on the links to other primitives in other languages, and which senses are compatible with an interpretation that takes the required sense from a target language sentence that is highly similar.

Appendix A

Detailed SCU Annotation Instructions

This appendix contains more details on the SCU Annotation task, and how it was adapted for document-level SCU annotation.

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 consist 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 or documents 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.

The definition of content unit is somewhat fluid – it can sometimes be a single word but it is never bigger than a sentence clause. Any event realized by a verb or a nominalized verb (e.g, "blow up" and "bombing" in the examples below) is a candidate SCU.

Example 1: The three sentences below come from four different summaries A, B, C and D.

> A: In 1992 the U. N. voted sanctions against Libya for its refusal to turn over the suspects.

> B: The United Nations imposed sanctions on Libya in 1992 because of their refusal to surrender the suspects.
>
> C: The U.N. imposed international air travel sanctions on Libya to force their extradition.
>
> D: Since 1992 Libya has been under U.N. sanctions in effect until the suspects are turned over to United States or Britain.

Among other information, all four sentences express the fact that "Libya was under U.N. sanctions" and this is the label for the SCU. The contributors are marked in brackets below (ignore SCU2 for now.)

> A: In 1992 [the U. N. voted sanctions against Libya]$_1$ [for its refusal to turn over the suspects.]$_2$
>
> B: [The United Nations imposed sanctions on Libya]$_1$ in 1992 [because of their refusal to surrender the suspects.]$_2$
>
> C: [The U.N. imposed]$_1$ international air travel sanctions on Libya [to force their extradition.]$_2$
>
> D: Since 1992 [Libya has been under U.N. sanctions]$_1$ [in effect until the suspects are turned over]$_2$ to United States or Britain.

Other information, such as when the sanctions where imposed, what specific sanctions were imposed, why they were imposed etc, will form their own SCUs. Identifying a main topic event in the summaries and asking yourself such questions as above about specifics will help you formulate labels and identify the SCU contributors. The contributors of an SCU need not share identical wording. For example in the sentences above, the SCU with label "The goal behind the sanctions is to make Libya surrender the suspects" is expressed by the text coindexed with "2". Sentence B differs in wording from the rest of the sentences, but the meaning is the same as that of the other contributors, expressing the fact that Libya does not want to surrender the suspects and the other nations involved want to force their extradition. (Note that this is an example of only two SCUs that will be derived from the sentences, the full analysis will lead to identifying more SCUs and will lead to complete bracketing of the sentences.)

The contributors are simply a part of the sentence–not all grammatical arguments necessary to reconstruct the label will be included in the contributor. This is ok, because the label will "bring in" any argument needed.

I have adapted SCU annotation to the annotation of documents as well as summaries. To determine if SCU annotation at the document level is repeatable and results in results in agreement between different annotators, myself and another Natural Language Processing PhD student annotated the same document set. We both annotated a set of Arabic documents (manually translated from Arabic into English), English documents, and summaries of the Arabic documents from the DUC 2004 corpus.

When annotating summaries for SCUs, there should not be repetition of information, assuming the summary is a good one. When annotating documents, some information is repeated multiple times. Repeated information in a document is added to a pre-existing SCU, which increases the weight for that SCU. Typically a lead sentence will give rise to one more SCUs, while later sentences might only elaborate on the dense information imparted in the lead sentence. The intuition behind marking multiple instances of the same information in a document is that, while the repetition might be fulfilling some linguistically-related reference role, most edited news also does not repeat unimportant information. Information that is repeated on a consistent basis should be important, and this importance should be reflected in the weight of the SCU it contributes to.

Once the SCU pyramids for a document set have been created, we have a method for characterizing the content of Arabic and English documents. The SCU pyramids for each document set 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, the more highly weighted SCUs should indicate information that we would like to include in a summary.

Bibliography

[AM00] Masayuki Asahara and Yuji Matsumoto. Extended models and tools for high-performance part-of-speech tagger. In *Proceedings of COLING 2000*, July 2000.

[Bar03] Regina Barzilay. *Information Fusion for Multidocument Summarization: Paraphrasing and Generation.* PhD thesis, Columbia University, 2003.

[BBN04] BBN. Bbn identifinder http://www.bbn.com/, 2004.

[BCP$^+$90] Peter F. Brown, John Cocke, Stephen Della Pietra, Vincent J. Della Pietra, Frederick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85, 1990.

[BGMS03] Sasha Blair-Goldensohn, Kathleen McKeown, and Andrew Hazen Schlaikjer. A hybrid approach for qa track definitional questions. In *12th Text Retrieval Conference (TREC 2003)*, Gaithersburg, MD, November 2003.

[BGMS04] Sasha Blair-Goldensohn, Kathleen R. McKeown, and Andrew Hazen Schlaikjer. *Answering Definitional Questions: A Hybrid Approach*, chapter 4. AAAI Press, 2004.

[BME99] Regina Barzilay, Kathy McKeown, and Michael Elhadad. Information fusion in the context of multi-document summarization. In *Proceedings of the 37th Association of Computational Linguistics*, Maryland, June 1999.

[Bou92] Didier Bourifault. Surface grammatical analysis for the extraction of terminological noun phrases. In *Proceedings of the 14th International Conference on Computational Linguistics*, pages 977–981, 1992.

[Buc85] Christopher Buckley. Implementation of the smart information retreival system. Technical Report Technical Report 85-686, Cornell University, Ithaca, New York, 1985.

[Buc02] T. Buckwalter. Buckwalter arabic morphological analyzer version 1.0, linguistic data consortium (ldc) catalog number ldc2002l49 and isbn 1-58563-257-0., 2002.

[CGJ01] Aitao Chen, Fred Gey, and Hailing Jiang. Alignment of english-chinese parallel corpora and its use in cross-language information retrieval. In *Proceedings of the 19th International Conference on Computer Processing of Oriental Languages*, Seoul, Korea, May 2001.

[CH97] N. Collier and H. Hirakawa. Acquisition of english-japanese proper nouns from noisy-parallel newswire articles using katakana matching. In *Proceedings of the Natural Language Pacific Rim Symposium (NLPRS-97)*, Phuket, Thailand, December 1997.

[Che02] Aitao Chen. Cross-language retrieval experiments at clef 2002. In Carol Peters, Martin Braschler, Julio Gonzalo, and Michael Kluck, editors, *CLEF*, volume 2785 of *Lecture Notes in Computer Science*, pages 28–48. Springer, 2002.

[CL00] 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.

[Coh60] J. Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20:37–46, 1960.

[Coh96] William W. Cohen. Learning trees and rules with set-valued features. In *AAAI/IAAI, Vol. 1*, pages 709–716, 1996.

[CRR97] 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 1997.

[Dia03] Mona Talat Diab. *Word Sense Disambiguation within a Multilingual Framework*. PhD thesis, University of Maryland, College Park, May 2003.

[DIS91] Ido Dagan, Alon Itai, and Ulrike Schwall. Two languages are more informative than one. In *Proceedings of the 29th conference on Association for Computational Linguistics*, pages 130–137. Association for Computational Linguistics, 1991.

[DKE00] Nina Wacholder David Kirk Evans, Judith L. Klavans. Document processing with linkit, April 2000.

[Edm69] H. P. Edmundson. New methods in automatic extracting. *Journal of the ACM*, 16(2):264–285, April 1969.

[EHW87] A. El-Hamdouchi and P. Willet. Techniques for the measurement of clustering tendency in document retrieval systems. *Information Science*, 13:361–365, 1987.

[EZ96] 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 17–24, Santa Cruz, US, 1996.

[FBY92] W.B. Frakes and R. Baeza-Yates, editors. *Information Retrieval: Data Structures and Algorithms*, pages 419–442. Prentice Hall, Englewood Cliffs, NJ, 1992.

[Gar95] S. R. Garner. Weka: The waikato environment for knowledge analysis, 1995.

[GC91] William A. Gale and Kenneth Ward Church. A program for aligning sentences in bilingual corpora. In *Meeting of the Association for Computational Linguistics*, pages 177–184, 1991.

[GN00] 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 237–246, Paris, 12–14 2000.

[HGM00] 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.

[HKE99] 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.

[HKH$^+$01] V. Hatzivassiloglou, J. L. Klavans, M. Holcombe, R. Barzilay, M.Y. Kan, and K.R. McKeown. Simfinder: A flexible clustering tool for summarization. In *NAACL'01 Automatic Summarization Workshop*, 2001.

[HL99] E.H. Hovy and Chin-Yew Lin. Automated text summarization in summarist. In I. Mani and M. Maybury, editors, *Advances in Automated Text Summarization*, chapter 8. MIT Press, 1999.

[Jac94] Christian Jacquemin. Fastr: a unification-based front-end to automatic indexing. In *Proceedings, Intelligent Multimedia Information Retrieval Systems and Management (RIAO'94)*, pages p. 34–47., 1994.

[Jac97] Christian Jacquemin. Guessing morphology from terms and corpora. In *Research and Development in Information Retrieval*, pages 156–165, 1997.

[Jac99] Christian Jacquemin. Syntagmatic and paradigmatic representations of term variation. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL'99)*, pages 341–348, 1999.

[JM00] Hongyan Jing and Kathy McKeown. Cut and paste based text summarization. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 178–185, 2000.

[KG97] Kevin Knight and Jonathan Graehl. Machine transliteration. 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 128–135, Somerset, New Jersey, 1997. Association for Computational Linguistics.

[KK98] Judith L. Klavans and Min-Yen Kan. The role of verbs in document access. In *Proceedings of the 36th Annual Meeting of the Association of Computational Linguistics and the 17th International Conference on Computational Linguistics (ACL/COLING-98)*, 1998.

[KK02] 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.

[Köh03] Philipp Köhn. *Noun Phrase Translation*. PhD thesis, University of Southern California, 2003.

[KPC95] Julian Kupiec, Jan O. Pedersen, and Francine Chen. A trainable document summarizer. In *Research and Development in Information Retrieval*, pages 68–73, 1995.

[KR90] Leonard Kaufman and Peter J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, New York, 1990.

[KT96] Judith L. Klavans and Evelyn Tzoukermann. Combining corpus and machine-readable dictionary data for building bilingual lexicons. *Machine Translation*, 10(3), 1996.

[Lev93] Beth Levin. *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press, 1993.

[LH03] Chin-Yew Lin and E.H. Hovy. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of 2003 Language Technology Conference (HLT-NAACL 2003)*, Edmonton, Canada, May 2003.

[Luh58] H.P. Luhn. The automatic creation of literature abstracts. *IBM Journal of research and development*, 2(2), 1958.

[Mar97] Daniel Marcu. From discourse structures to text summaries. In I. Mani and M. Maybury, editors, *Proceedings of the ACL/EACL'97 Workshop on Intelligent Scalable Text Summarization*, pages 82–88, Madrid, Spain, July 1997.

[MBE$^+$02] Kathleen R. McKeown, Regina Barzilay, David Evans, Vasileios Hatzivassiloglou, Judith L. Klavans, Ani Nenkova, Carl Sable, Barry Schiffman, and Sergey Sigelman. Tracking and summarizing news on a daily basis with columbia's newsblaster. In *Proceedings of HLT 2002 Human Language Technology Conference*, San Diego, CA, 2002.

[MBF$^+$90] George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. Introduction to wordnet: an on-line lexical database. *International Journal of Lexicography*, 4(3):235–244, 1990.

[Mel97a] 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 97–108. Association for Computational Linguistics, Somerset, New Jersey, 1997.

[Mel97b] 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 Computa-*

tional Linguistics, pages 305–312, Somerset, New Jersey, 1997. Association for Computational Linguistics.

[MMD$^+$04] 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 interlingual representation of multilingual texts. In *Language Resources and Evaluation Conference Workshop: Beyond Named Entity Recognition - Semantic Labelling for NLP Tasks*, Lisbon, Portugal, May 2004.

[MN89] Peter McCullagh and John A. Nelder. *Generalized Linear Models.* Chapman and Hall, London, second edition, 1989.

[NP04] 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.

[OD96] Douglas W. Oard and Bonnie J. Dorr. A survey of multilingual text retrieval. Technical Report CS-TR-3615, University of Maryland, 1996.

[ON03] Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

[OY04] 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.

[PHKJ01] Ari Pirkola, Turid Hedlund, Heikki Keskustalo, and Kalervo Järvelin. Dictionary-Based cross-language information retrieval: Problems, methods, and research findings. *Information Retrieval*, 4(3-4):209–230, September 2001.

[Pir98] Ari Pirkola. The effects of query structure and dictionary setups in dictionary-based cross-language information retrieval. In W. Bruce Croft, Alistair Moffat, C.J. van Rijsbergen, Ross Wilkinson, and Justin Zobel, editors, *Proceedings of* *the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 55–63, Melbourne, Australia, August 1998. ACM Press, New York.

[RJB00] 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.

[RTS$^+$03] 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.

[Sal68] G Salton. *Automatic Information Organization and Retrieval*. McGraw-Hill, New York, 1968.

[Sal71] Gerald Salton. *The SMART retrieval system - Experiments in automatic document processing*. Prentice-Hall, Englewood Cliffs, New Jersey, 1971.

[SB88] Gerard Salton and Chris Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[Sid02] Advaith Siddharthan. Resolving attachment and clause boundary ambiguities for simplifying relative clause constructs. In *Proceedings of the Student Workshop, 40th Meeting of the Association for Computational Linguistics (ACL'02)*, pages 60–65, Philadelphia, USA, 2002.

[SK98] Bonnie Glover Stalls and Kevin Knight. Translating names and technical terms in arabic text. In *Proceedings of the 1998 COLING-ACL*, Montreal, Canada, 1998.

[SMH96] Frank Z. Smadja, Kathleen McKeown, and Vasileios Hatzivassiloglou. Translating collocations for bilingual lexicons: A statistical approach. *Computational Linguistics*, 22(1):1–38, 1996.

[SNK04] 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.

[SNM02] Barry Schiffman, Ani Nenkova, and Kathleen McKeown. Experiments in multidocument summarization. In *Proceedings of the Human Language Technology Conference*, March 2002.

[SO00] Ruth Sperer and Douglas W. Oard. Structured translation for cross-language information retrieval. In *Research and Development in Information Retrieval, SIGIR*, pages 120–127, Athens, Greece, 2000.

[Som99] Harold Somers. Example-based machine translation. *Machine Translation*, 14(2):113–157, June 1999.

[Spä85] Helmuth Späth. *Cluster Dissection and Analysis: Theory, FORTRAN Programs, Examples.* Ellis Horwood, Chichester, West Sussex, England, 1985.

[TKJ97] Evelyne Tzoukermann, Judith Klavans, and Christian Jacquemin. Effective use of natural language processing techniques for automatic conflation of multi-word terms: The role of derivational morphology, part of speech tagging, and shallow parsing. In *Research and Development in Information Retrieval*, pages 148–155, 1997.

[Voo86] E. M. Voorhees. *The Effictiveness and Efficiency of Agglomerative Hierarchical Clustering in Document Retrieval.* PhD thesis, Cornell University, 1986.

[Wac98] Nina Wacholder. Simplex NPs clustered by head: A method for identifying significant topics within a document. In Federica Busa, Inderjeet Mani, and Patrick Saint-Dizier, editors, *The Computational Treatment of Nominals*, pages 70–79, 75 Paterson Street, Suite 9 New Brunswick, NJ 08901 USA, August 1998. Coling-ACL, Association of Computational Lingustics.

[WEK01] 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.

[WV98] S. Wan and M. Verspoor. Automatic english-chinese name transliteration for development of multilingual resources. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics*, pages 1352–1356, Montreal, Canada, 1998.

[Yar93] David Yarowsky. One sense per collocation. In *Proceedings of the ARPA Human Language Technology Workshop*, pages 266–271, Princeton, NJ, 1993.