The University Of Sheffield.

Niraj Aswani

[n.aswani@sheffield.ac.uk]

Designing a General Framework for Text Alignment: Case Studies with Two South Asian Languages

Submitted for the degree of Doctor of Philosophy

Supervised by

Prof. Robert Gaizauskas

July 13, 2012

Designing a General Framework for Text Alignment: Case Studies with Two South Asian Languages

Niraj Aswani

[n.aswani@sheffield.ac.uk]

Submitted for the degree of Doctor of Philosophy

Supervised by

Prof. Robert Gaizauskas

Department of Computer Science

The University of Sheffield

July 13, 2012

Abstract

Building machine translation systems for many South Asian languages (such as Hindi, Gujarati, etc.) using statistical methods is problematic. The primary reason is insufficient parallel data to learn accurate word alignment. Additionally, these languages are morphologically rich and have free word order. When it is difficult to rely purely on statistical methods due to insufficient data, research shows that better performance can be obtained by building hybrid systems that rely on language specific resources, such as morphological analysers or dictionaries, as well as statistical methods. However, it is difficult to find such language specific resources for many South Asian languages.

Since languages such as Hindi, Gujarati, Urdu, Bengali, Punjabi and Marathi are all very similar in structure and the main differences lie in the script and vocabulary used for these languages, we hypothesise that it is possible to develop resources for one of these languages and generalize the approach to allow rapid bootstrapping of similar resources for the other closely related languages – with minimal effort and similar accuracies. To verify this, we develop a few resources for the Hindi language, including a sentence alignment algorithm, a morphological analyser and a transliteration similarity component and generalize the approach to allow rapid bootstrapping of similar resources for the Gujarati language. We show that the approach works on both the Hindi and Gujarati languages and achieves results that are comparable to similar state-of-the-art (SOA) resources available for these languages.

We also hypothesise that it is possible to develop a high performance hybrid word alignment algorithm that relies on such language specific resources. To verify this, we design, implement and evaluate a novel English-Hindi hybrid word alignment system that uses the Hindi specific resources developed by us. Not only do we show our word alignment system outperforms other SOA English-Hindi word alignment systems, but also how simple it is to adapt it to the English-Gujarati language pair.

Acknowledgements

I am very grateful to my supervisor, Prof. Robert Gaizauskas, for his indispensable advice at every step helping me to make my work conceptually sound and also to draw my attention to even the smallest mistakes to improve my academic writing. With his great enthusiasm and efforts to explain things clearly and simply, he has guided me to the point where I have been able to complete the work of my thesis.

I would like to thank my advisor and my manager, Prof. Hamish Cunningham, for allowing me to take days off from work to focus on the thesis. He helped me overcome the difficulties of switching between work and PhD studies and contribute at both ends.

I will remain in debt to both my supervisor and my manager for their priceless advice at every stage that I have progressed towards becoming a professional researcher.

I am very thankful to my examiners, Prof. Josef van Genabith and Dr. Trevor Cohn, for an interesting discussion during my viva and extremely useful comments that have added a lot of value to my thesis.

I would like to thank my friends, Mr. Ravish Bhagdev, Dr. Arunangsu Chatterjee, Dr. Ajay Chakravarthy, Dr. BalaKrishna Kolluru and Mrs. Kaveri Vaswani and my brother, Mr. Amit Aswani, for assisting me in preparing a gold standard for the evaluation of my results. I would like to thank all my colleagues for sharing their valuable experience to guide me through writing the thesis.

I cannot end without thanking my parents, my wife and all my family for being so patient all these years I have worked on the PhD. Not only have they provided constant encouragement but also compromised on several ends to allow me to complete my thesis.

Publications

The work presented within this thesis is published in the following papers:

- N. Aswani and R. Gaizauskas, *A Hybrid Approach to Align Sentences and Words in English-Hindi Parallel Corpora*, In Proceedings of the ACL 2005 Workshop on Building and Using Parallel Texts: Data Driven Machine Translation and Beyond, Association for Computational Linguistics, pages 57–64, Ann Arbor, Michigan, June 2005.

- N. Aswani and R. Gaizauskas, *Aligning Words in English-Hindi Parallel Corpora*, In Proceedings of the ACL Workshop on Building and Using Parallel Texts, Association for Computational Linguistics, pages 115–118, Ann Arbor, Michigan, June 2005.

- N. Aswani and R. Gaizauskas, *Evolving a General Framework for Text Alignment: Case Studies with Two South Asian Languages*, In Proceedings of the International Conference on Machine Translation: Twenty-Five Years On, Cranfield, Bedfordshire, UK, November 2009.

- N. Aswani and R. Gaizauskas, *English-Hindi Transliteration Using Multiple Similarity Metrics*, In Proceedings of the 7th Language Resources and Evaluation Conference (LREC), European Language Resources Association, La Valletta, Malta, May 2010.

- N. Aswani and R. Gaizauskas, *Developing Morphological Analysers for South Asian Languages: Experimenting with the Hindi and Gujarati Languages*, In Proceedings of the 7th Language Resources and Evaluation Conference (LREC), European Language Resources Association, La Valletta, Malta, May 2010.

Contents

List of Figures

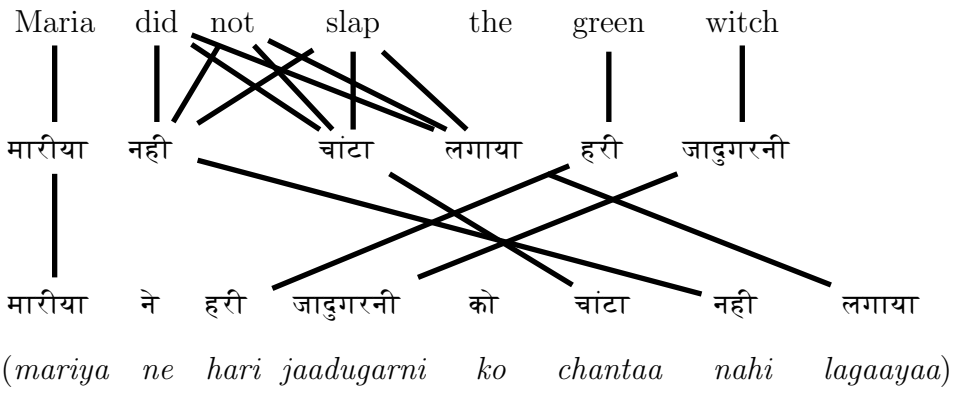
List of Tables

Chapter 1
Introduction

1.1 Motivation

Systems like Systran, Google Translate and Yahoo's Babel Fish are examples of very popular machine translation systems. They ask you to provide a piece of text and select a source (language in which the text is provided) and a target language (the language into which the provided text should be translated). What you get back is a translation of your text into the selected target language. But how does a machine translation system work?

Before a system can translate a sentence, it has to know how to translate words. For this, it has to know, for each word in the source language, its equivalent word in the target language. Let's take an example that has been adapted from the work of Koehn et al. (2007).



As illustrated above, one word in the source language does not always translate into exactly one target language word (e.g. *slap*). Also, there may be occasions when a source language word is not translated at all (e.g. *the*). Similarly, the machine has to reorder the target language words according to the syntactic structure of the foreign language and make other necessary adjustments to make the foreign language sentence syntactically correct.

These are some of the issues a machine translation system has to deal with. However, what forms the basis of a machine translation system is the mapping between words. The mapping between words is called **word alignment** and a **word alignment algorithm** tells how the words in a given sentence pair should be aligned.

Depending on the language pair and availability of data, various methods of word alignment have been proposed (see Chapter 7). However, the ones most widely discussed (in the literature) are the IBM models (Och and Ney, 2004).

IBM Models

The IBM models mainly rely on three factors: translation probability, fertility probability and distortion probability.

1. Given a source and a target word, translation probability refers to the likeliness of the source word being translated as the target word.

2. The second factor, fertility, is defined as the number of target words generated by a given source word.

3. Finally the third factor is distortion which relates the positions of translated words in the source and target sentences. If the source and target words tend to appear in the same parts of sentences, they are said to be translated roughly undistorted, while words which move far apart have high distortion.

Given pairs of sentences which are aligned with each other (i.e. are translation of each other), the IBM models try to estimate these probabilities. However, to produce acceptable results, they require a huge amount of sentence aligned data. The good thing about the IBM models is that their learning is unsupervised[1] – which means the system learns all its parameters automatically from a parallel corpus without any labelled training data (Och and Ney, 2004; Koehn et al., 2007). However, for exactly the same reason, the IBM models have certain limitations: they are time consuming and the quality produced by them is poor without enough parallel data (Moore, 2005; Ambati et al., 2010; Ananiadou, 2011).

[1]The task here is to learn mapping between words. For this task, the IBM models are not provided any reference word alignment from which they can learn. See Chapter 7 for the detailed discussion on the IBM models.

But why is having a limited amount of data a problem for statistically trained models? If a program sees a particular word or phrase a thousand times, it is more likely to learn a correct translation for such a word or phrase than if it sees it ten times, or once or never. The problem increases if a language is morphologically rich and words in the corpus (a collection of documents) are inflected significantly. In such cases, it is unlikely that word forms will be seen sufficiently often for the statistical approach to work. The problem becomes even more difficult when one of the languages in question is a free order language and contains words that are translated with high distortion.

This is particularly the case with the South Asian languages such as Hindi and Gujarati.

South Asian Languages

Although many South Asian languages (such as Hindi, Gujarati, Bengali etc.) are widely spoken, building machine translation systems for them using the statistical methods is problematic. The primary reason is that there is not enough parallel data available to learn accurate word alignment. What adds to the problem is that these languages are morphologically rich and have free order as well.

When it is difficult to rely purely on statistical methods due to a lack of data, research (Bharti et al., 2002; Lopez et al., 2002; Ayan et al., 2004; Liu et al., 2005; Venkatapathy and Joshi, 2006; Chinnappa and Singh, 2007; Riesa and Marcu, 2010; Gao et al., 2010) shows that better performance can be obtained by building systems that rely on language specific resources, such as morphological analysers or dictionaries, as well as statistical methods (see Chapter 7).

But unavailability of parallel data is not the only issue and even the language specific resources, that might be required by such hybrid systems, are difficult to find for many South Asian languages. In the past few years, the Indian NLP community has shown an ever increasing interest in the development of language resources for Indian languages. However, most of these resources are being developed in India and majority of them are only available for academic research in India. Many of these components are not public or only available

on commercial licensing terms. For the resources that are publicly available, often there are issues to be resolved such as font encodings, work still in progress and unreliable quality. We conducted a survey of available resources for the Indian languages and a list of such resources is provided in the appendix (see Section B).

Languages such as Hindi, Gujarati, Bengali, Punjabi and Marathi are all very similar in structure and the main differences lie in the script and vocabulary used for these languages. Therefore, it is likely that the problems that occur when developing resources for one of these languages will emerge for other languages in the group too.

We **hypothesise that**

- *it is possible to develop resources for one of these South Asian languages and generalize the approach to allow rapid bootstrapping of similar resources for the other closely related languages – with minimal effort and without compromising on their accuracies, and*
- *it is also possible to develop a high performance hybrid word alignment algorithm that relies on such language specific resources.*

1.2 Aims and Objectives

To verify the hypotheses, we studied the English-Hindi language pair and found that there are at least four language specific components which were absolute requirements for boosting English-Hindi word alignment results. These are a bilingual English-Hindi dictionary; a morphological analyser; a component to match named entities; and a component to capture and align multi word expressions.

First, we develop these resources for the Hindi language. Then, we generalize the approach to allow rapid bootstrapping of similar resources for the Gujarati language. Our aim is to show that the approach proposed by us works on both the Hindi and Gujarati languages and achieves results that are comparable, if not better, to the similar state-of-the-art resources

available for these languages.

We decided to do experiments on the Gujarati language for four main reasons. First, the Gujarati language is one of those South Asian languages that are similar to the Hindi language. We believe that if we can successfully adapt Hindi resources to the Gujarati language, it should be possible to adapt the Hindi resources to other South Asian languages as well. Second, it is a language which has very limited resources available. If we could develop resources for the Gujarati language, we believe it would be a very significant contribution to the research community working on the Gujarati language. Third, the Gujarati language is one of the six south Asian languages for which the EMILLE corpus (Baker et al., 2004) has parallel text. Thus, data is available for us to conduct our experiments. Finally, availability of Gujarati native speakers meant that we could easily verify the results of our algorithms and developed resources for the Gujarati language. The author himself is a native speaker of the Gujarati language which makes it easy to identify any potential problems at an early stage.

Using the four key resources as specified above, it is our aim also to design, implement and evaluate a novel English-Hindi hybrid word alignment system that relies on the resources we develop for the Hindi language and show that such a system can outperform other state-of-the-art English-Hindi word alignment systems. Finally, we aim to adapt the English-Hindi word alignment system to the English-Gujarati language pair with the least possible efforts.

It is important to mention that our aim is to tackle a resource poor language setting with limited training data, which is challenging for current statistical approaches. Whether the performance advantage of the approach developed in the thesis over statistical approaches can be maintained for larger data sets is an open (and interesting) research question. We aim to verify this by future experiments.

Below, we list several important tasks that, we think, must be completed in order to achieve the aims.

Sentence Alignment

The minimum requirement for most word alignment algorithms is pairs of sentences where sentences in a pair are translations of each other. Therefore, the first objective of our project is to prepare an English-Hindi sentence aligned dataset. We use the EMILLE corpus (Baker et al., 2004), which contains data in seven South Asian languages (including the Hindi language) organized in parallel at the document level. As the data in the EMILLE corpus is not aligned at the sentence level, we need to choose an appropriate sentence alignment algorithm or if the need arises, propose a new one suitable to the English-Hindi language pair.

Baseline for Word Alignment

Although, the literature reports that statistical methods do not perform very well on the English-Hindi language pair (for the reasons discussed earlier), it is very important to verify their performance on the English-Hindi language pair. Since the IBM Models have been the most common choice (in the literature), our second objective of the project is to use them for the English-Hindi language pair and establish a baseline for any future English-Hindi word alignment algorithms that we develop.

Gold Standard

In order to facilitate the automatic evaluation of our results, it is very important that we have a gold standard for both sentence level alignment and word level alignment. Additionally, this has to be done not only for the English-Hindi language pair but also for any other language pair we choose to experiment with in future. The data available in the EMILLE corpus is only aligned at the document level. Therefore, we require a set of tools which allow us to manually align the data both at the sentence and word level.

Language Specific Resources

Language specific resources such as a bilingual English-Hindi dictionary can be used to boost the alignment results. Therefore, it is very important that we identify such resources that can help in boosting the alignment results.

Word Alignment

Developing a word alignment algorithm for the English-Hindi language pair is our fifth objective. Having found out what resources are required, it is important to come up with a suitable arrangement for these resources to act together and produce high quality word alignments.

Bootstrapping Resources for Other Languages

To verify the hypotheses, we need to choose a South Asian language that is closely related to the Hindi language and adapt all the developed resources to the new language.

Leveraging Existing Resources

To achieve the goals of this thesis efficiently and quickly, it is very important that, instead of reinventing a wheel (at least for the very common tasks such as tokenising and sentence splitting), we reuse existing resources as much as we can.

Similarly, to maximize the chances of our resources being reused, it is important that the resources are implemented in such a way that they can be easily adapted to the other languages and easily integrated in other applications.

1.3 Contributions

In this section, we list all the actions which were taken to fulfil the objectives listed above. We also mention the contributions this project has made. All the contributions are marked with the © symbol.

Sentence Alignment

Our initial investigation into the EMILLE corpus revealed a need for removing noise from the corpus. One of the examples of corrections made to the corpus is adjusting the order of sentences in parallel documents so that the source and target sentences appear in the same order.

© Our contribution in this case is a clean English-Hindi parallel corpus containing 3440 sentences pairings.

© We have developed a length-based sentence alignment algorithm which is similar to the algorithm proposed by Gale and Church (1993). The algorithm makes it possible to align many-to-many sentence pairings and has achieved scores of 0.98 (precision) and 1.0 (recall) for the English-Hindi language pair. These scores are higher than the state-of-the-art scores for the English-Hindi dataset taken from the EMILLE corpus.

Gold Standard

Preparation of a gold standard is a very expensive and time consuming process. In our survey of existing alignment tools (see Chapter A) we found that the majority of them have very specific requirements to fulfil before they can be used for aligning text. We wanted to provide a flexible framework that can align text at any level (e.g. character, word, phrase, sentence etc.) and yet make it an acceptable experience for the annotators.

© We have developed a general framework for text alignment that allows users to manually align text at any annotation[2] level. The alignment framework makes it easy for users to define a work flow for alignment tasks, making it possible to train a model or update resources in the background. Such resources can be used further for automatically suggesting new alignments to the annotators.

© With the help of these alignment tools, we have prepared a word aligned dataset consisting of 3440 sentence pairings for the English-Hindi language pair.

© We have prepared word alignment guidelines for the English-Hindi language pair. These are based on the initial guidelines proposed by Melamed (1998b). These guidelines were distributed among native speakers who helped us in preparing the gold standard.

[2]An annotation can be described as a span over text (with start and end offsets) and with a type that describes the semantic category of the text underneath the selected span

Baseline for Word Alignment

We carried out several experiments for aligning words in the English-Hindi test dataset. Out of the 3440 sentence pairings prepared for gold standard, 2029 sentence pairings were chosen randomly for the test dataset. In addition to the test dataset, we used the English-Hindi parallel corpus prepared by Bojar et al. (2010) for the GIZA++ training (Och and Ney, 2004). We applied several heuristics (as suggested by Koehn et al. (2007)) to obtain the optimal alignment results. The best word alignment results obtained were 0.62 (precision), 0.64 (recall), 0.63 (F-Measure). These results were used as a baseline.

Language Specific Resources

We found that there are at least four language specific components which are absolute requirements for boosting English-Hindi word alignment results. These are: a bilingual English-Hindi dictionary; a morphological analyser; a component to match named entities; and a component to capture and align multi word expressions. We found that components such as a compound word de-compositor and an abbreviation matcher can help in improving the alignment results at some level.

© We have collected an English-Hindi bilingual dictionary. The dictionary has been cleaned up to remove many errors and extended to include many new entries from the EMILLE corpus (see Chapter 3).

© We have developed an FSM (finite-state-machine) based program that facilitates quick execution of suffix-replacement rules to identify a base-form and a suffix for every inflected word given to the program. We also developed an algorithm that given a monolingual corpus and a dictionary, learns suffix replacement rules to identify base-forms and suffixes. The output of this program is a rule file that can be used along with the FSM based program.

© We have developed an English morphological analyser using the the suffix-replacement rules originally written by Kevin Humphreys (Gaizauskas et al., 1995). Originally, the rules were written in a language called *flex* and we converted them into a more flexible format that is understood by our FSM based program.

© Using our algorithm for learning suffix-replacement rules, we developed a morphological analyser for the Hindi language. The Hindi morphological analyser has produced results beyond state-of-the-art on the EMILLE corpus with 0.82 (precision), 0.80 (recall) and 0.81 (F-Measure).

© In order to match cognates and mentions of named entities, we have developed bidirectional character mappings consisting of English letters and their Hindi counterparts. We also developed a similarity metric that is suitable for matching transliteration pairs when one of the languages in question is a South Asian language. We also developed a transliteration similarity algorithm to identify if the two words are transliterations of each other. The algorithm achieves an accuracy of 0.95 when used for identifying transliteration pairs.

© A Multi Word Expression (MWE) is a set of words (mostly adjacent) which altogether have a certain meaning. Such expressions are aligned (as a single unit) to a single word or similar MWE in the other language. We have developed a supervised algorithm that learns rules for identifying MWEs. For this it relies on common suffixes, prefixes and conjunctions in the training data.

Word Alignment

All our components are integrated in GATE (General Architecture for Text Engineering) (Cunningham et al., 2011). This allows them to be used as individual components and as well as a part of pipelines composed of other components from GATE.

In our case, the word alignment algorithm is a pipeline in GATE. This pipeline comprises of two types of resources. First, the language specific preprocessing components and second the word alignment specific components. Language specific preprocessing components identify things such as word and sentence boundaries, base-forms of each word, named entities and multi word expressions, etc. Here, we use the components such as the morphological analyser and local word grouping to obtain base-forms and identify multi word expressions. The word alignment components use this information and align the words.

© We have developed a word alignment pipeline for the English-Hindi language pair. The algorithm for the English-Hindi language pair produced much better results (F-Measure:0.78) in comparison to the baseline results (F-Measure: 0.63).

Bootstrapping Resources for Other Languages

In order to validate our hypothesis, it was very important that our algorithms are designed in such a way that they can be easily adapted to languages closely related to the Hindi language. Since the author is a native speaker of the Gujarati language, it was the first choice for experiments. To the best of our knowledge, very limited work has been reported for the Gujarati language and therefore we consider any contribution as a significant for the Gujarati language.

© We have prepared a corpus of 500 English-Gujarati sentence pairings aligned at the word level.

© We have developed a Gujarati morphological analyser using the same algorithm for obtaining a set of suffix replacement rules. The scores obtained for the analyser are 0.83 (precision), 0.70 (recall) and 0.76 (F-Measure).

© We have developed a transliteration mapping consisting of English letters and their Gujarati counterparts. Since, the Gujarati language is similar to the Hindi language, the mapping was derived from the English-Hindi mapping directly. Using the transliteration similarly algorithm in conjunction with this mapping produced an accuracy of 0.91.

© To develop a word alignment pipeline for the English-Gujarati language pair, we used the word alignment pipeline developed for the English-Hindi language pair and replaced the Hindi specific preprocessing components with the relevant Gujarati components. The word alignment algorithm for the English-Gujarati language pair produced 0.75 (F-Measure) on a corpus of 330 sentence pairings taken from the EMILLE corpus.

Leveraging Existing Resources

GATE is a text processing framework (Cunningham et al., 2011). It comprises of several

text processing components such as tokenisers, sentence boundary identifiers, POS taggers etc. for different languages. Not only is GATE widely accepted by scientists, researchers and students, but its stability and simplicity has also attracted several commercial organizations to use it. It has a big user community.

In order to benefit from the existing resources of the GATE system, we chose to develop an alignment framework which hosts all the components we have developed and integrate them into the GATE system.

The algorithms we have proposed are not language specific and therefore can be adapted to other similar languages by providing necessary input for the training. Also, most of the components developed as part of the PhD project are all independent and can be used for purposes other than the word alignment task.

Riding on the fact that the Hindi and Gujarati languages are similar in structure, we attempted to adapt Hindi resources to the Gujarati language. As expected, we have been able to achieve results for the Gujarati language that are not only comparable to the results of their Hindi counterparts but good enough for the Gujarati components to be used in other applications. Therefore, we believe that it is possible to adapt these resources to other South Asian languages as well – at least for the languages that are similar to Hindi and Gujarati. The Sentence Alignment algorithm can certainly be tried for the languages other than the South Asian languages. Similarly, we expect the Transliteration Similarity component to work with any language pair where a mapping among the characters of the two languages can be obtained.

These components can be useful for other areas of natural language processing. For example, morphological analysers are used by search engines to index root forms of words – both to keep index sizes low and to be able to match various inflected forms in the documents. Similarly, identification of named entities can help in populating gazetteer lists automatically which can be used in information extraction tasks.

1.4 Chapters at a Glance

Below we provide details of how rest of the report is organized with a very short summary for each chapter.

In Chapter 2, we give an overview of the Hindi language and discuss its similarities and differences with the English language. In the process, we try to identify resources that need to be developed for the English-Hindi language pair. In this case, our main focus is on the resources needed for sentence and word alignment.

In Chapter 3, we describe various external resources used in our experiments. We provide details such as how these resources were obtained, how they work, why we need them, any improvements made to these resources and from where to download the improved resources. However, the chapter does not provide any details on newly developed resources. Information about such resources is provided in subsequent chapters.

In Chapter 4, we present a very simple length-based sentence alignment algorithm. We first describe some well-known methods of sentence alignment algorithms. We then describe our alignment algorithm and give details of our experiments to evaluate it with the Hindi language.

In Chapter 5, we present a rule-based morphological analyser. We first describe a program developed in Java that facilitates execution of suffix-replacement rules to identify a base-form and a suffix for the given word. We then describe our algorithm that given a monolingual corpus and a dictionary, learns suffix replacement rules. We also describe the development of a Hindi morphological analyser. The morphological analyser is used for obtaining root forms of Hindi inflected words. These root forms are used by various components in our system. For example, the word alignment algorithm uses them to obtain their English translations in the dictionary. Similarly, there are quite a few rules in the local word grouping component that are based on the root values of words.

In Chapter 6, we present an approach to measure the transliteration similarity of English-

Hindi word pairs. The component is used in our word alignment algorithm for aligning words such as named entities and cognates. Our approach has two components. First we propose a bi-directional mapping between one or more characters in the Devanagari script and one or more characters in the Roman script (pronounced as in English). Second, we present an algorithm for computing a similarity measure which also takes into account the constraints needed to match English-Hindi transliterated words. Finally, we evaluate various similarity metrics individually and together under a multiple measure agreement scenario.

In Chapter 7, we present a hybrid word alignment algorithm for the English-Hindi language pair. The algorithm uses output of some of the components to find appropriate alignments in the bitext. Also, we look at the two key components used as part of the word alignment algorithm: a chunker and a local word grouping (LWG) system. The chapter also details our experiments with the word alignment algorithm, showing individual as well as combined performances of the different components of the algorithm.

In Chapter 8, we demonstrate the generality of our alignment algorithms and the approaches used for developing language specific resources by adapting them to the English-Gujarati language pair.

In Chapter 9, we conclude the thesis and give details of our future work.

Finally, in Chapter A (in Appendix), we introduce our framework for text alignment and discuss various features and functionalities of the same. We also give information on our setup for preparing gold-standard alignments and, at the same time, improving the English-Hindi resources in the background. We provide a list of resources related to natural language processing and machine translation for the South Asian languages – particularly, those relating to the Hindi and Gujarati languages.

Chapter 2
The Hindi Language

2.1 Outline of the Chapter

Before beginning to develop an alignment system for the English-Hindi language pair, it is important to investigate similarities and differences between the two languages. Such an investigation is important as it could reveal structural similarities and differences that could play a crucial role in deciding the methodology of the alignment system. For example, if one of the two languages has adopted words from the other, methods such as transliteration similarity could be used for identifying cognates. If the structure of two languages is known, one can use this knowledge to reduce the space of alignment search. For example, if it is known that a subject is likely to appear in the first part of a sentence, one could concentrate only on that part of the sentence to find a match.

In this chapter, we give a brief introduction to the Hindi language and discuss some important aspects of the language's grammar. We also provide guidelines for aligning words in English-Hindi parallel corpora.

2.2 Introduction to Hindi

The Hindi language is one of the many Indic languages. According to Ta (2004) about 182 million people speak Hindi as their native language making Hindi the fifth most spoken native language in the world after Mandarin, English, Spanish, and Bengali. Some estimates say that around 350 million people speak Hindi. It is the state language of Bihar, Haryana, Himachal Pradesh, Madhya Pradesh, Rajasthan, and Uttar Pradesh. It is also spoken in Surinam, Mauritius, and Fiji. It belongs to the Indo-European language group and is mostly written in the script called *Devanagari*.

2.3 The Devanagari Script

The *Devanagari* script was originally used for writing *Sanskrit*. The script has thirteen vowels and thirty six consonants[1] (see Table 2.1).

[1] `http://unicode.org/charts/PDF/U0900.pdf`

Table 2.1: Devanagari Script

Vowels	ग् (*g*)	न् (*n*)
अ (*a*)	घ् (*gh*)	प् (*p*)
आ (*aa/A*)	ङ् (*gn*)	फ् (*f*)
इ (*i*)	च् (*ch*)	ब् (*b*)
ई (*ee/I*)	छ् (*chh*)	भ् (*bh*)
उ (*u*)	ज् (*j*)	म् (*m*)
ऊ (*oo/U*)	झ् (*z*)	य् (*y*)
ऋ (*ru*)	ञ् (*gn*)	र् (*r*)
ए (*e*)	ट् (*T*)	ल् (*l*)
ऐ (*e/E*)	ठ् (*Th*)	व् (*v*)
ओ (*o*)	ड् (*D*)	श् (*sh*)
औ (*au/O*)	ढ् (*Dh*)	ष् (*Sh*)
अं (*am*)	ण् (*N*)	स् (*s*)
अः (*ah*)	त् (*t*)	ह् (*h*)
Consonants	थ् (*th*)	ळ् (*ld*)
क् (*k*)	द् (*d*)	क्ष् (*kSh*)
ख् (*kh*)	ध् (*dh*)	ज्ञ् (*gy*)

It is a syllabic script whose characters represent syllables. According to the Oxford dictionary[2], *a syllable is a unit of pronunciation having one vowel sound, with or without surrounding consonants, forming the whole or a part of a word.* For example, there are three syllables in the Hindi word आवाज (*aavaaj, sound*): आ (*aa*), वा (*vaa*) and ज (*ja*). The first syllable आ (*aa*) is a vowel. The second syllable is a combination of a consonant व् (*va*) and a vowel आ (*aa*). Finally, the third syllable is a combination of a consonant ज् and a vowel अ (*a*).

When a word starts with a vowel, it is written in a form described in the first row of Table 2.2. However, when it is merged with a consonant, the combination is written in a

[2]http://oxforddictionaries.com/definition/syllable?q=syllable

different way. Table 2.2 shows different syllables obtained for the consonants क (*k*) and ख् (*kh*) after combining them with different vowels. Here, the syllable क (*ka*) is a combination of a consonant क् (*ka*) and a vowel अ (*a*).

Table 2.2: Syllables for the Consonants क (*k*) and ख् (*kh*)

Vowels	अ	आ	इ	ई	उ	ऊ	ऋ	ए	ऐ	ओ	औ	अं	अः
Consonants													
क्	क	का	कि	की	कु	कू	कृ	के	कै	को	कौ	कं	कः
ख्	ख	खा	खि	खी	खु	खू	खृ	खे	खै	खो	खौ	खं	खः

Devanagari is a phonetic based script, where words are written largely according to phonetics (pronunciation) and therefore it is easy to read and learn. As noted the Devanagari script was originally used for Sanskrit. For sounds in Hindi that are not found in Sanskrit, letters close to those sounds are modified with a dot placed below the character. For example, the character ज (*j*) is modified to ज़ (*z*) to represent the sound of *z*.

2.4 Similarities and Differences Between English and Hindi Sentences

In India, English is one of the most popular foreign languages. Following this trend, it is becoming very popular for people to use words from both, the English and the Hindi vocabulary in the same sentence. According to Clair (2002), use of such a mixed language, also known as Hinglish, is said to have prestige, as the amount of mixing corresponds with the level of education and is an indicator of membership in the elite group.

Although the use of such a mixed code language is becoming very common, the English and the Hindi languages remain widely different in the structure. This includes differences such as the difference in word order, placements of modifiers, absence of articles and different types of genders in Hindi.

In this section, we list some of the major differences in the grammar of English and Hindi.

Word Order

The Hindi language has a different word order from the English language. In Hindi, the standard word order is Subject-Object-Verb as opposed to Subject-Verb-Object in English. In Hindi, most of the time the verbs are placed at the end of a sentence and postpositions are used instead of prepositions. Postpositions are like prepositions except that they are written after the noun. Table 2.3 explains the major structural differences between English and Hindi sentences (Williams, 1996).

Table 2.3: Structural Differences Between English and Hindi Sentences (Williams, 1996)

Declarative Sentences

Language	**Word Order**	**Sentence**
English	Subject Verb Object	*I speak English*
Hindi	Subject Object Verb	मैं (*mein, I*)अन्ग्रेजी (*angrejee, English*) बोलता हुं (*bolataa hun, speak*)
English	Subject Verb Preposition Object	*He jumped into the water*
Hindi	Subject Object Postposition Verb	वह (*vaha, he*) पानी (*paani, water*) में (*main, into*) कुदा (*kudaa, jumped*)

Imperative Sentences

Language	**Word Order**	**Sentence**
English	Verb Place Adverb	*Come here now*
Hindi	Adverb Place Verb	अभी (*abhee, now*) यहां (*yahaan, here*) आओ (*aao, come*)
English	Aux Negative Verb Adverb	*Do not eat quickly*
Hindi	Adverb Negative Verb	जलदी (*jaladee, quickly*)मत (*mata, do not*) खाओ (*khaao, eat*)

Interrogative Sentences

Language	**Word Order**	**Sentence**
English	Adverb Aux Subject Verb	*What are you doing?*
Hindi	Subject Adverb Verb Aux	तुम (*tuma, you*) क्या (*kyaa, what*) कर रहे हो (*kara rahe ho, doing*) ?

Despite the differences listed in Table 2.3, unlike English, Hindi is a relatively free order language. In other words, words in a Hindi sentence do not have to be in a particular sequence to make it a valid sentence.

Placement of Modifiers

In English, modifiers of a nominal can occur both before and after the nominal, while in Hindi, the modifiers occur before the nominal they modify. For example *(Richest) person (in the world)* will be translated as दुनिया (*duniyaa, world*) का (*kaa, in*) सबसे अमीर (*sabase ameer, richest*) ईन्सान (*insaana, person*).

Presence of Articles

There are no articles in the Hindi language. For example *In the world* in English would be translated as दुनिया (*duniyaa, world*) में (*main, in*).

Gender

English has three genders: masculine, feminine and neuter for pronouns whereas Hindi has only two, masculine and feminine. However, all nouns in the Hindi language have gender. For example किताब (*kitaaba, book*) is feminine whereas केला (*kelaa, banana*) is masculine. Also adjectives in Hindi need to agree in gender with the nouns they modify. For example, to say that a book was good, one has to use अच्छी किताब (*achchhi kitaaba, good book*) as opposed to अच्छा केला (*achchhaa kelA, good banana*) which means a good banana.

Vibhakti or KArakachinha

In Hindi, a *vibhakti* (a.k.a. *kArakachinha*) functions as a case marker and is used to describe a connection between one thing and the other. For example, to describe that an activity was carried out by a subject, the Hindi word ने (*ne, by*) is used after the subject. For example, माली ने फूल दिए (*mAlI(gardener) ne(by) phUla(flowers) diye(give), gardener gave flowers OR flowers were given by the gardener*). Here, the word माली (*mAlI, gardener*) is a subject. Since he has carried out this activity, the word ने (*ne*) appears after it. In this case, it conveys the meaning of *who carried out the activity* in the sentence.

The vibhakties को (*ko*), का (*kA*), की (*kI*), के (*ke*) are used to highlight the object on which an activity or action is performed. For example, the English passive sentence, *the ball was thrown* would be literally translated as गेंद को फेंका गया था (*genda (ball) ko phe.nka (thrown) gayA thA (was)*). Here, the use of the vibhakti को (*ko*) conveys the message that something had happened to the ball. In this case, it was thrown.

It is important to discuss the usage of vibhakties as they have no counterpart when aligned with a target English sentence in active voice. However, they could be aligned with prepositions or auxiliary verbs if the target English sentence is in passive voice. Alternatively, in case of the former, vibhakties could be grouped with the actual subject or the object they

have connection with and could be aligned as a group with the English word in the target sentence that the respective subject or the object aligns to. Consider the following example:

माली ने फूलों को पानी दिया
(*mAlI(gardener) ne(vibhakti) phUlon(flowers) ko(vibhakti) pAnI(water) diyA(give)*)

Active Voice: The gardener watered flowers.
Passive Voice: Flowers were watered by the gardener.

In case of the first English sentence, either the vibhakties ने (*ne*) and को (*ko*) both could be left unaligned, or the word ने (*ne*) can be grouped with the word माली (*mAlI*) and the word को (*ko*) with the word फूलों (*phUlon*). In case of the passive voice, the word ने (*ne*) can be directly aligned with the word *by*, however the word को (*ko*) will not be aligned.

Local Word Grouping in Hindi

Local Word Grouping (LWG) is a common term used in Hindi grammar. The function of the LWG is to form word groups on the basis of local information (i.e. information based on adjacent words).

According to Ray et al. (2003), *Local word groups in Hindi are the fixed order components of the sentence and are word groups which obey the constraints that constituent words are placed contiguously, and the group of words is fixed word order.*

In the above example, grouping of the word ने (*ne*) with the word माली (*mAlI*) is an example of a LWG. One can form groups that include head words (such as nouns or verbs), vibhakties and auxiliary verbs. Consider the following example:

माली फूलों को पानी दे रहा है
(*mAlI(gardener) phUlon(flowers) ko(vibhakti) pAnI(water) de(give) rahA hai(is doing)*)
The gardener is watering the plants

where, following are the two local word groups:

फूलों (*phoolon, flowers*) को (*ko, vibhakti*)

दे (*de, give*) रहा है (*rahA hai, is doing*)

Such a local word groupings could be very helpful in aligning one-to-many or even many-to-many alignments.

There are certain words which have more than one meaning or interpretation. One such Hindi word is पर (*para*), which, if used as postposition means *on* but if used as a noun, means *feathers*. Akshar et al. (1995) suggest that the words with more than one meaning should not be included in local word groups unless it is clear which interpretation the word carries in the sentence.

Transliterated Words

Since Devanagari is a phonetic based script where words are written largely according to pronunciation, it is possible to come up with a list of possible phonetic mappings into English for each sound in Hindi and vice versa. For example, the Hindi consonant क could be mapped to the English consonant *k* or *c* or *ch*. Furthermore, it is possible to see both English and Hindi vocabulary in the same sentence. This has led to many English words being transliterated using the Devanagari script and the opposite is also true. There are many words in Hindi which have been derived or adopted from the English. For example, words such as केंसर (*cancer*) or सायकल (*cycle*).

Morphological Variants

In English, the words *dog* and *dogs* are two different forms of the same noun *dog*. Similarly, the words *going* and *gone* are two different forms of the same verb *go*. Inflection can be useful for, for example, expressing the gender or number of nouns. Consider the above example, where the word *dog* is a singular noun, which, if inflected by the addition of the suffix *s*,

becomes a plural. In this case, the word *dog* is used for both the male and female dog and therefore no variation is needed to express gender.

In Hindi, the word कुत्ता (*kuttA*) and the word कुतिया (*kutiyA*) are used for male and female dogs respectively. Both of them have different inflections in their plural forms. For example, the कुत्ता would become कुत्ते (*kutte, male dogs*) and कुतिया would become कुतीयां (*kutIyAn, female dogs*). The common characters in all these four different inflections are कुत (*kut*) which is also called the stem of the inflected forms. For some languages, such as English and Hindi, straightforward rules can be found that describe how to form a different inflection of a word. For example, in Hindi, to convert most male singular nouns (which end with the suffix ा (*A*)) into male plural nouns, the suffix ा (*A*) is replaced with the suffix े (*e*) (i.e. कुत्ता (*kuttA, a male dog*) into कुत्ते (*kutte, male dogs*)).

In English, verbs are inflected depending on the time when the action started or is going to start and its progress status. Auxiliary verbs act as helper verbs to give further information about the main verb. Similar to the English verb *to be* and *to have*, there is a verb होना (*honA, to be*) in Hindi which acts as a helper verb. Along with the main verb, it is inflected depending on the person, number, and gender of the subject.

Study of such inflectional rules is very helpful for designing systems that can automatically convert words from one form to the other. One example of such an application is a dictionary lookup application. Most dictionaries have entries in their base-forms. Given a plural form of a word, it is unlikely that it will be located in the dictionary. If a system can automatically obtain a singular form from the plural form, it is more likely the word will be found in the dictionary. Hindi is one language that is morphologically rich and has systematic rules for converting its words from one form to the other. In other words, it is possible to come up with a small set of rules to convert inflected words into different forms and with high precision. In Table 2.4, we provide a few examples of such rules as obtained from various grammar books (Navneet, 2001) on the Hindi language. We explore these rules in detail later in this thesis and present a language independent system that, given a corpus and a

few other resources, is capable of learning such inflectional rules automatically.

Table 2.4: Examples of Rules of Inflection in Hindi

noun	**suffix**	**adjective**
तुफान (*tufAna, storm*)	ई (*I*)	तुफानी (*tufAnI, stormy*)
दया (*dayA, mercy*)	लु (*lu*)	दयालु (*dayAlu, merciful*)
रंग (*ranga, color*)	ईन (*Ina*)	रंगीन (*ra.ngIna, colorful*)
पुस्तक (*pustaka, book*)	ईय (*Iya*)	पुस्तकीय (*pustakIya, bookish*)
खर्च (*kharcha, expense*)	ईला (*IlA*)	खर्चीला (*kharchIlA, expensive*)
verb	**suffix**	**adverbs**
कमाना (*kamAnA, to earn*)	आऊ (*AU*)	कमाऊ (*kamAU, earning*)
लुभाना (*lubhAnA, to tempt*)	अवना (*avanA*)	लुभावना (*lubhAvanA, tempting*)
समझना (*samazanA, to understand*)	दार (*dAra*)	समझदार (*samazadAra, wise*)

Compound Words

A word can be either simple or compound. A simple word consists of a root or a stem together with suffixes or prefixes. A compound word is made of two or more independent words conjoined together. Akshar et al. (1995) suggests that a compound word must be broken into its constituent simple words before it is analysed. This is very important from the word alignment perspective as this could result in a single compound word being aligned to more than one word in the other language. For example, consider the compound word अत्याधुनिक (*atyAdhunika, ultra modern*) which is composed of the two independent words, अत्य (*atya, ultra*) and आधुनिक (*Adhunika, modern*). It is possible that the compound word may not be located in a dictionary, but the member words may. Here are some different examples of independent words conjoined together to make a single compound word (known as *samAsa* in Hindi):

दिन और रात (*dina aura rAta, day and night*) > दिन-रात

In this case, the word और (*aura, and*) is replaced with the - (*hyphen*) sign.

सेना का नायक (*seNa kA nAyaka, chief of the military*) > सेनानायक

In this case, the vibhakti का (*kA*) is removed.

महान कवि (*mahAna kavi, great poet*) > महाकवि

In this case, the first word महान (*mahAna, great*) is modified and conjoined with the other word.

In English, some words need to be prefixed to obtain their antonyms. For example *regular-irregular*, *proper-improper* etc. Similarly, in Hindi, there are certain prefixes which if used with appropriate words can be useful for obtaining antonyms. For example, if the word आदर(*Adara, respect*) is prefixed with the word अन्(*an*), becomes अनादर (*anAdara, disrespect*).

Thus, in order to locate such words and align them, it may be necessary to disjoin them first.

2.5 Alignment Guidelines

From the information gathered on the similarities and differences between the English and Hindi languages, we prepare a set of guidelines for the word alignment in this section.

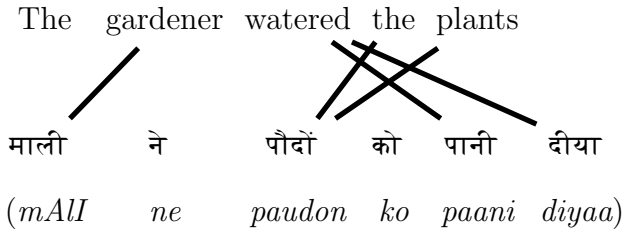
Several word alignment guidelines for different projects have been published. For example, ARCADE[3], the Blinker project (Melamed, 1998b), Kruijff-Korbayova et al. (2006) and Caseli et al. (2005). Every language pair is different and therefore the guidelines published for one language pair may not be relevant to the others. Most guidelines in the literature are based on the guidelines proposed by Melamed (1998b). These guidelines have been altered to meet the requirements of the respective language pair. Unfortunately, we could not find any existing word-alignment guidelines for the English-Hindi language pair. Therefore we take the *Annotation Style Guide of the Blinker Project* as a starting point.

In this section, we present the full alignment guidelines given to the human annotators for preparing the English-Hindi word alignment gold-standard. We explain these guidelines with examples to make the instructions clear and more illustrative.

- It is possible that a word in one language is never translated in the other language. Such words should not be aligned.

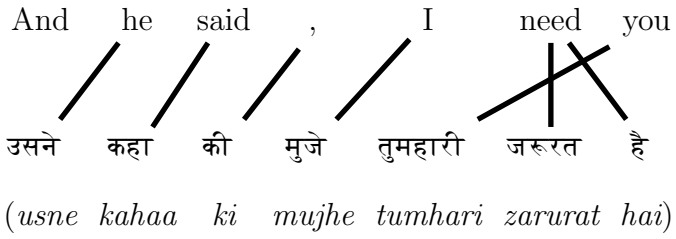
[3] `http://sites.univ-provence.fr/veronis/arcade/arcade1/2nd/word/guide/index.html`

Example 1:



In this example, the English word *The*, which is an article, and the Hindi word ने (*ne*), which is a vibhakti, remain unaligned.

Example 2:



In this example, the English word *And* is not aligned because there is no translation available for this word in the Hindi sentence.

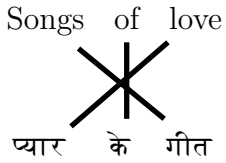
- Non-literal translations should be aligned as phrase alignments. It is possible to find examples where 1:1 translation is not possible and the meaning of a word or a phrase in one language can only be expressed with more than one word in the target language. In such cases, the words should be aligned as phrase alignments. For example,

  love songs = प्यार के गीत (*pyAra ke gIta*)

  Here, although the word *love* can be linked with प्यार (*pyAra*) and the word *songs* with गीत (*gIta*), the word के (*ke*) joins the two words and has the meaning of *of*, as if the phrase was *songs of love*.

- 1:1 word alignments should be preferred over multi-word alignments.

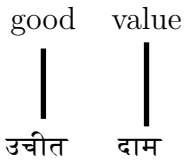
  Example 1:



(*pyaar ke geet*)

Here, one could align these phrases as phrase alignments but because it is possible to align individual words, it should be given preference. Similarly,

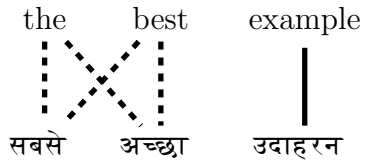
Example 2:



(*uchit daam*)

- There are no articles in Hindi. Therefore, the determiners should be aligned along with their head nouns. However, in some cases, it is possible to find separate words for the determiners.

  Example 1:

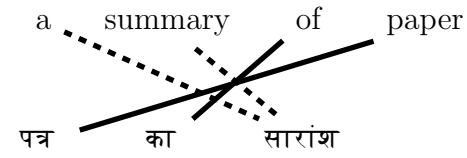


(*sabse achchha udaharan*)

Example 2:

(*ek achchha saudaa*)

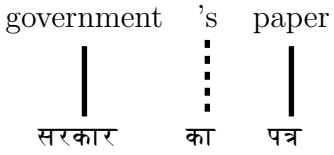
Example 3:



(*patra kaa saransh*)

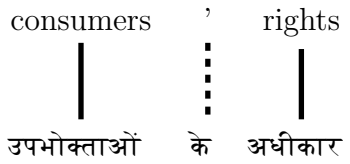
- If there are spelling errors, the words should not be aligned. If the corpus is used for constructing bilingual lexicons, it could result in invalid bilingual entries and therefore words with incorrect spellings should not be aligned.
- The English possessives **'s** and **'** should be aligned separately.

  Example 1:



(*sarkar kaa patra*)

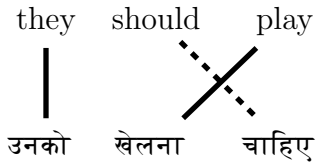
Example 2:



(*upabhoktaon ke adhikar*)

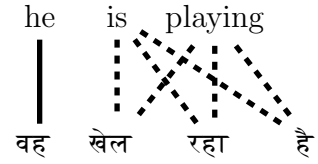
- Where possible, auxiliary verbs should be separated from main verbs.

  Example 1:



(*unko khelana chahiye*)

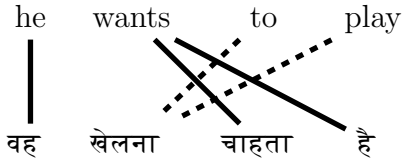
  Example 2:



(*vah khel rahaa hai*)

- Alignment of base-forms of verbs should be given special treatment.

  Example 1:

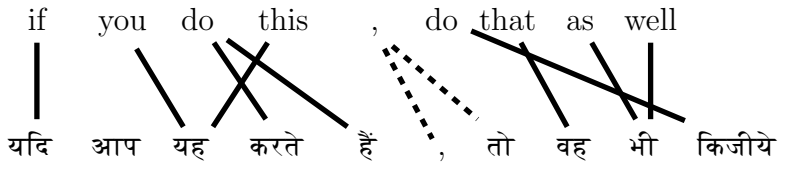


(*vah khelnaa chahataa hai*)

Note that the word *to* is aligned along with the word *play*.

- If the punctuation symbols do not match, they must not be aligned. However, it is possible that a punctuation symbol aligns with a conjunction.

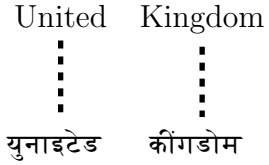
  Example 1:



(*yadi aap yah karate hain to vah bhi kijiye*)

- Words in entity names such as company names or locations should be aligned separately.

  Example 1:

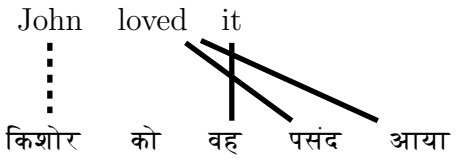


(*United Kingdom*)

- If the translation is not correct, the entire pair or the part of the sentence that is not correct should be skipped.

- Sometimes, pronouns appear in one language but not in the other language. In such cases, the entity for which the pronoun is used should be aligned with the pronoun.

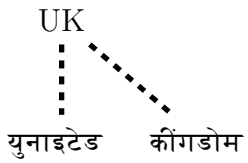
  Example 1:



(*kishor ko vah pasand aaya*)

- Sometimes abbreviations are expanded in the target text. In such cases the abbreviation should be aligned with its full form.

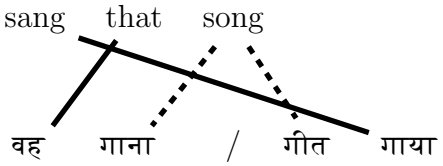
  Example 1:



(*United Kingdom*)

- If for a word in one language, there are multiple alignments in the other language, all should be aligned.

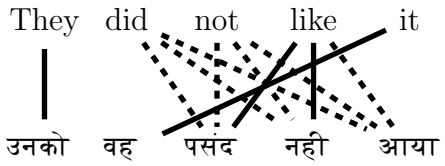
  Example 1:



(*vah gaanaa geet gaayaa*)

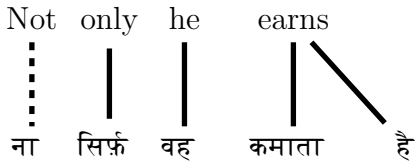
- Negation would be most probably part of a multi-word alignment. However, where possible, it should be aligned separately.

  Example 1:



(*unko vah pasand nahi aayaa*)

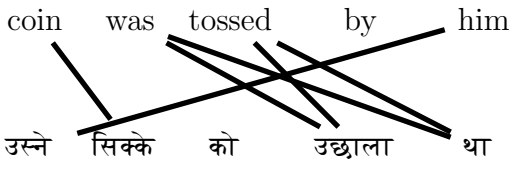
Example 2:



(*na sirf vah kamaataa hai*)

- If the translation of a sentence in active voice is in passive voice, it could be difficult to align words. In this case, one should try to align words that match. Entire sentences must not be aligned as single units.

Example 1:



(*usne sikke ko uchhaalaa thaa*)

2.6 Chapter Remarks

In this chapter we briefly looked at the Hindi language mostly from the grammatical perspective. As can be seen from the discussion above, the two languages are different as well as

similar in many ways. It can be observed that LWG and also dis-joining of compound words are two very important processes for anyone wanting to achieve one-to-many and many-to-many alignments. Having a transliteration system could also help in aligning words such as cognates and proper nouns. The study of inflections and the design on the basis of this study of a reasonable morphological analyser can also play a critical role in the alignment process.

From the information we gathered on the similarities and differences between the English and Hindi languages, we proposed a set of guidelines for the word alignment between the texts of these two languages.

From here, we take with us the knowledge of the two languages, and the issues that need most attention when designing an alignment system for the Hindi and the English languages. In the next chapter, we give information on various external resources used in our experiments. These include some of the resources discussed in this chapter. We provide details such as how these resources were obtained, how they work, why we need them, and what improvements we made to them.

Chapter 3
Resources

3.1 Outline of the Chapter

In the past few years, the Indian NLP community has shown an ever increasing interest in the development of language resources for Indian languages. These include resources such as shallow parsers, POS taggers, morphological analysers, chunkers and components for local word grouping. Machine translation among Indian languages has also been a topic of interest for many language technology groups in India[1]. There are at least three Machine Translation systems available online (see Section B) that allow users to translate texts between many of the Indian languages such as Hindi, Bengali, Marathi, Malayalam, Telugu etc.

Unfortunately, not all of these resources are public. Many of these resources are being developed in India and are only available to research groups in India. Many of these components are only available on commercial licensing terms. For the resources that are publicly available, often there are issues to be resolved such as font encodings, work still in progress and unreliable quality. We conducted a survey of tools available for South Asian languages, especially those are relevant to the Hindi and Gujarati languages. We have provided a list of such resources in the appendix (see Section B). In this chapter, we give information on various external resources used in our experiments. Many of the resources listed in the appendix have become available only very recently. This is one of the reasons why they have not been used extensively in the PhD project.

As specified in the Introduction (Chapter 1), we have chosen to integrate our alignment framework (which comprises all the components developed as part of this thesis) into GATE in order to benefit from the existing resources of GATE. Therefore, here we give information on the GATE system explaining its architecture and those components relevant to the current work. As all the experiments mentioned in subsequent chapters are based on the EMILLE corpus (Baker et al., 2004), we also give information on this corpus.

[1]Language technology groups such as Language Technologies Research Centre, IIT Hyderabad and Technology Development for Indian Languages, Ministry of Communication and Information Technology, Delhi are among the very active language technology groups in India.

3.2 GATE

GATE is comprised of an architecture, a free open source framework and graphical development environment. It already benefits from more than 15 years of exhaustive development and contributions from researchers and developers all over the globe. It is a text processing framework with a variety of language processing resources. Developing and integrating the alignment framework we have developed in GATE has two advantages.

Firstly, GATE has resources such as tokenisers, sentence splitters, part-of-taggers etc. for different languages. All our experiments are based on the three languages: English, Hindi and Gujarati. Already, GATE has various resources for the English and Hindi languages. Reusing these resources for tasks such as tokenisation and identification of sentence boundaries not only reduces the work but also makes sure that the techniques used for these tasks are widely acceptable and the results are uniform. Additionally, GATE supports a variety of document formats such as text, HTML, XML, DOC, PDF etc. When these documents are loaded into GATE, they are automatically converted into an internal representation which is the same for different types of documents. This feature of GATE allows users to concentrate more on the actual task without worrying too much about dealing with different formats.

Some of GATE's existing resources can be adapted to different requirements, while, its flexible structure allows one to easily write wrappers for third-party libraries and integrate them in the system. For example, one might want to use LingPipe's language identification functionality[2] to detect the language of a document; use the relevant plug-in to tokenise the text[3] and split it into sentences[4]; use TreeTagger's part-of-speech tagger[5] for tagging and write GATE JAPE grammars[6] to extract a variety of information such as email addresses, person names, etc. based on the information extracted earlier. In a nutshell, the purpose of the GATE system is to provide its users with a variety of language resources and allow

[2]http://GATE.ac.uk/userguide/sec:misc-creole:lingpipe:langid
[3]http://GATE.ac.uk/userguide/sec:annie:tokeniser
[4]http://GATE.ac.uk/userguide/sec:annie:splitter
[5]http://GATE.ac.uk/sale/tao/index.html#x1-50000021.3.1
[6]http://GATE.ac.uk/userguide/chap:japeimpl

them to exploit these resources in the best way.

Secondly, not only is GATE widely accepted by scientists, researchers and students but its stability and simplicity have also attracted several commercial organizations (such as IBM, HP etc.). It has a big community that uses and follows it. Integrating alignment tools, which do not exist in the current GATE system will provide a platform for thorough testing and detailed feedback from its users all over the world.

3.2.1 GATE Core Concepts

LRs, PRs, VRs, Applications and Plugins

GATE defines three types of resources namely *Language Resources* (LR), *Processing Resources* (PR) and *Visual Resources* (VR). LRs are the entities that hold linguistic data. PRs are the entities that process data. Finally, VRs are the components used for building graphical interfaces.

Resources such as corpora, documents or ontologies are examples of LRs, whereas resources such as tokenisers, sentence splitters and POS taggers are examples of PRs. Documents are resources that need to be annotated. Typically, one wants to run a set of uniform PRs across a set of documents. For this, one can use a corpus LR to group a set of documents. Similarly, to run more than one PR on a set of documents, one can use a component called pipeline (a.k.a. application). A pipeline has one or more PRs in a predefined sequence to be executed over the documents.

When a document is opened in GATE, the format is analysed and the document is converted into a single unified model that consists of text, annotation sets and features. Here, the text is the actual content as obtained from the document without any formatting information. PRs have the job of processing document contents and producing annotations. For example, tokenisers produce annotations of type *Token* where each token is a word or a punctuation symbol in the text. Similarly, sentence splitters produce annotations of type *Sentence* for each sentence in the text. These annotations can be grouped into a set. For example, one

might want to store gold-standard annotations in a set called *Key* and ask the machine to suggest new annotations in a set called *Suggestions*. A document can have one or more annotation sets each with one or more types of annotations. If a document has any sort of mark-up, this mark-up is converted into an annotation and placed under the set called *Original markups*. An annotation type can have any name, such as *Person* or *Date*, useful for describing the type of the content for which the annotation is created.

The annotation format is a modified form of the *tipster* format (Grishman, 1997) which has been made largely compatible with the Atlas format (Bird et al., 2000) and uses the now standard mechanism of 'stand-o markup'. The annotations associated with each document are a structure central to GATE, because they encode the language data read and produced by each PR. Each annotation has a start and an end offset and a set of features associated with it. These are the offsets in the underlying text document. Each feature has a *name* and a *value*, which holds descriptive or analytical information such as part-of-speech tags, syntactic analysis, co-reference information etc.

In case of the *Original markups* of the document, element names are used as annotation types and attributes on elements are converted into annotation features.

Also, every resource in GATE has a feature map attached to it. This map is useful for storing meta-data about the resource. For example, a document's source URL and mime-type etc. Figure 3.1 gives a snapshot of the GATE GUI with an article downloaded from Wikipedia. The document was processed with GATE's ANNIE application. *ANNIE* stands for *A Nearly New Information Extraction* system. It is an example of pre-configured application that comprises resources useful for performing simple tasks such as identifying tokens, sentences boundaries, annotating tokens with their POS tags, identifying persons, organizations and locations etc.

There are several different plug-ins in GATE that can be loaded and unloaded at any time. The concept behind having individual plug-ins is to group a set of PRs and make them available only when the plug-in is loaded. There are many plug-ins in GATE. There are

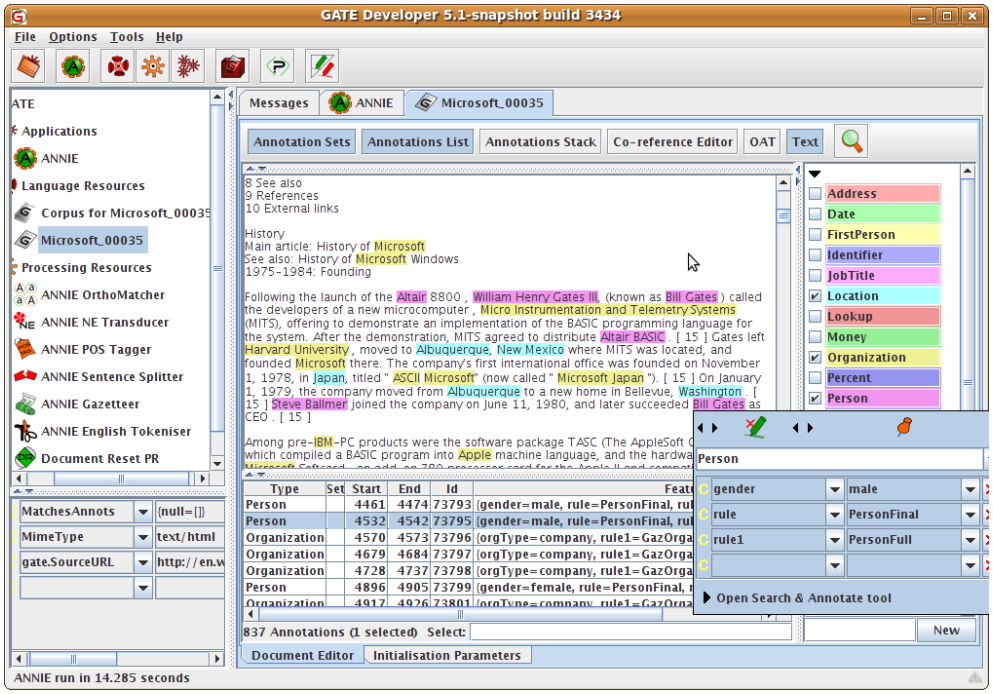


Figure 3.1: GATE GUI with a Processed Document

other versions of ANNIE or similar processing resources adapted to different languages. They are available in different plug-ins, such as the *Lang_Hindi* plug-in for the Hindi language, *Lang_Arabic* for the Arabic language, etc.

Java Annotation Patterns Engine

GATE also has an engine called JAPE[7] that stands for Java Annotation Patterns Engine. It is an engine based on regular expression pattern/action rules over annotations. JAPE is a version of *CPSL* (a Common Pattern Specification Language, developed in the TIPSTER programme by Doug Appelt and others). This engine executes one or more JAPE grammar phases where each phase consists a set of pattern/action rules. The left-hand-side (LHS) of the rule represents an annotation pattern and the right-hand-side (RHS) describes the action to be taken when pattern is found in the document. JAPE executes these rules in a sequential manner and applies the RHS action to generate new annotations over the text matched by the LHS regular expression pattern. Rule prioritisation (if activated) prevents

[7]http://GATE.ac.uk/userguide/chap:jape

multiple assignments of annotations to the same text string.

We use JAPE for recognising Hindi numbers written in words (see Chapter 7).

3.2.2 Processing Resources

In this section we describe the GATE processing resources that we used for preprocessing the English and Hindi texts.

Tokenisers

Tokenising text is the operation of splitting up a string of characters into a set of elementary units called tokens. These tokens can be numbers, punctuation or words. Tokenisation depends on the language and the writing system of the given text. In most languages words are separated by delimiters such as white space, comma, period, quotation mark, parentheses, punctuations and other special characters.

As mentioned earlier, GATE has a variety of tools for different languages. We use the GATE English tokeniser to tokenise English text. Similarly, we use GATE's Hindi tokeniser to tokenise Hindi text. These tokenisers produce annotations of type *Token* and *SpaceToken* to represent individual words and spaces respectively. In the case of the English language, for each token the tokeniser produces features including *orthography*, *kind*, *length* and *string*. Possible values for the *orthography* feature are *allCaps*, *lowerCase*, *upperInitial* and *mixedCaps*. Since, there is no concept of upper or lower case letters in the Devanagari script, this feature is irrelevant to the Hindi language. Therefore, unless the text is written in English this feature is excluded from Hindi tokens.

Tokens in both the languages can be categorised into one of the four kinds: *word*, *symbol*, *number* and *punctuation*. The length of a token is calculated as the number of characters it contains. In the case of the English language, this is the sum of the total number of consonants and the total number of vowels in a word. Calculating length of a Hindi word can be ambiguous. For example, consider the syllable क (*ka*). As described in Chapter 2, to

form such a syllable, one has to combine the consonant क् with the vowel अ. In other words, the length should be two characters long. However, this is not the case in GATE. GATE is entirely written in Java and relies on Java's underlying Unicode support for the Devanagari script. According to the Devanagari Unicode chart[8], each syllable is only one character long. To convert a syllable into the respective consonant, e.g. क into क् , one needs to add the halant character "् " at the end of the syllable. Therefore, when calculating the length of the consonant क्, Java returns its length as two characters. Finally, the *string* feature stores the string under the annotation span.

Part-of-Speech Tagger

In many language processing applications, using part-of-speech taggers – programs which can identify the grammatical parts-of-speech of words (e.g. noun, verb etc.) – is standard practice. Every language has many words that have more than one syntactic category. For example, the English word *swimming* can be annotated as both *noun* and *verb*. Which category to assign to a given word depends on its context of use in a sentence. For example, compare *Swimming is the best exercise* and *I was swimming when he arrived*. In the former case, *swimming* should be annotated as a noun, whereas in the latter case it is a verb.

The information on POS tags is useful to identify correct base-forms of the words. In the above examples, if the word *swimming* is annotated as a verb, the base-form is *swim* whereas if the word is annotated as a noun, it would be *swimming*. As we show later in Chapter 7, we use base-forms of words at several stages of our word alignment algorithm. Therefore, it is very important to have high precision when calculating base-forms. Furthermore, the morphological analyser (discussed next) used in this work requires words to be pre-annotated with their POS categories.

GATE has several POS taggers including Mark Hepple's POS tagger (Hepple, 2000), the TreeTagger, the OpenNLP POS tagger and LingPipe's POS tagger. All but Hepple's POS tagger can be used across different languages. Hepple's tagger is a modified version of Brill's

[8] `http://unicode.org/charts/PDF/U0900.pdf`

tagger. It works with English text only and produces 54 different tags in total. The tags are stored as values for the *category* feature on each token. The tagger can be trained or adapted to other languages. The Hindi tagger available as part of the *Lang_Hindi* plug-in is a perfect example of adapting Hepple's tagger to other languages.

Morphological Analyser

There are two different components in GATE for morphological analysis. Firstly, the stemmer and secondly the GATE morphological analyser. The difference between the two is that the former only returns a stem whereas the latter returns the linguistic root for a given word. Typically, a stem is the first part of the word without any inflection. For example, given three words *institutional*, *institution* and *institute*, the stem would be *institut*. Here, the string *institut* is not a word in its own right but provides a term that can be used for clustering different forms which share this stem. Often, this is useful for merging inflected terms to a common form. On the other hand, a linguistic base-form is a meaningful word in its own right. Most dictionaries tend to list words in their base-forms and therefore to be able to locate the appropriate entry for an inflected form in a dictionary, one needs to have a morphological analyser to obtain the inflected word's base-form.

We use GATE's English Morphological Analyser (MA) to analyse words and obtain linguistic roots. The morphological analyser takes POS tagged tokens as input. However, the POS tagging is optional – providing this not only increases the chances of higher precision but also speeds up the process. The MA produces lemma and affix for each token in the text. It is based on some predefined regular expression rules.

Each rule has two parts, a left-hand-side (LHS) and a right-hand-side (RHS). The LHS defines a regular expression whereas the RHS is a function name that is called when the regular expression defined on the LHS matches the word under consideration. Each rule in the rule file has a separate tag that specifies which rule to consider with what part-of-speech tag. Whether the MA PR should take part-of-speech tags into account or not can be specified as a runtime parameter. If this option is set to true, only the rules applicable

to the word's POS category are executed. The PR takes a tokenised and preferably POS tagged document along with a file with rules listed in it. One could, for instance, create one's own rule file for a choice of language and provide it to the MA PR. More information on the morphological analyser is provided later in Chapter 5.

Gazetteers

A gazetteer consists of a list of strings that one would like to match in the text. Typically, people use gazetteers to match named entities. In GATE, the implementation of gazetteer lookup is based on finite-state-machines which makes its execution very fast. A gazetteer can have a list of several string collections, where each collection has a major and minor type value specified for it. For example, one could create a list of all the countries in the world and assign *Location* as the major type and *Country* as the minor type. Similarly one could create several different collections – each with names of companies/organizations grouped together based on their field of work. One could then assign each such collection *Company* as the major type and the respective field as its minor type.

With default gazetteers, strings in the collections must match the content of the document. This might not be possible all the time. For example, suppose one is trying to locate things that are used in the game of cricket. In other words, the task is to identify if a document has mentions of *bat*, *ball*, *glove*, *stumps*, etc. If we try to match all the instances using a simple gazetteer, we need to write down all the possible inflections of these terms such as *ball* and *balls*, *stump* and *stumps*, etc. This is fine as long as there are only a limited number of entries.

We implemented a processing resource called the *Flexible Gazetteer* in GATE that provides users with the flexibility to choose their own customized input and an external gazetteer. For example, to achieve what we wanted to achieve in the cricket example, we could create a simple gazetteer with the base-forms of cricket terms only and provide it to the Flexible Gazetteer with an instruction to use the base-forms of the tokens when comparing the document content with the strings in the gazetteer. Thus, the Flexible Gazetteer performs

lookup over a document that is formed by using the values of an arbitrary feature of an arbitrary annotation type. Once annotated, the Flexible Gazetteer then transfers the lookup annotations back onto the original content.

3.3 The EMILLE Corpus

A collection of pairs of texts in two different languages, where each member of a pair is a translation of the other is called a *parallel corpus*.

EMILLE, which stands for Enabling Minority Language Engineering, was a three year research project at Lancaster University and Sheffield University. As part of the project, a 97 million word electronic corpus of South Asian languages, especially those spoken in the UK, was developed. The corpus comprises three main components: monolingual, parallel and annotated corpora. The parallel corpus consists of 200,000 words of text in English and its translation in Hindi, Bengali, Punjabi, Gujarati, Urdu, Tamil and Sinhala. The Urdu data in the corpus is annotated with part-of-speech tags. UK government advice leaflets were used as a primary source of the parallel texts. These leaflets are about issues related to personal and public health matters, social matters, security matters and housing matters in the UK. The parallel data is said to be *term-rich* as it covers a variety of topics. About 75 documents from the Departments of Health, Social Services, Education and Skills, Transport, Local Government and the Regions were collected to produce the parallel corpus. These texts were then manually converted into Unicode electronic form.

Although the quality of the EMILLE corpus is good, some modifications and corrections were required to remove noise. This noise is due to many reasons such as out-of-order translation, omitted sections and insertion of additional information (e.g. instead of one proper translation, insertion of two or more probable translations). With the help of four native speakers, we carried out the task of removing noise from a subset of the EMILLE corpus over three days. The corrected data was then used in a word alignment shared task at the ACL 2005 workshop on *Building and using parallel texts : data driven machine*

translation and beyond, organized by Joel Martin, Rada Mihalcea and Ted Pedersen (Martin et al., 2005). Below we give some details about the shared task and the work that we carried out to remove the noise from a subset of the corpus.

Shared Task:
The main focus of the word alignment shared task was on languages with scarce resources. The participating teams were provided with training and test data for three language pairs: English-Inuktitut, Romanian-English and English-Hindi. We provided the English-Hindi dataset which included more than 67,000 words of English-Hindi parallel electronic data (a subset of the EMILLE corpus). As pointed out earlier, in order to prepare a dataset for the workshop noise needed to be removed. The cleanup task involved the following[9]:

1. Adjusting the order of the sentences;
2. correcting the translation where necessary;
3. correcting spellings errors in the Hindi text.

Prior to sending the corrected corpus to the workshop, the parallel texts were aligned at the sentence level. The algorithm used for the sentence alignment is described in Chapter 4 of this report. The output was manually checked for its accuracy. The corpus, after correction, was formatted according to the conventions of the shared task at the ACL 2005 workshop. The corpus consists of two files, one for each language (English and Hindi). Corresponding lines in the English and Hindi files are sentence aligned. If a sentence in the English language aligns with more than one Hindi sentence, all the Hindi sentences that the English sentence aligns with are placed on the same line. The corpus contains 3440 parallel sentence pairings and about 122,000 words in the two files. It is available from the `http://www.cs.unt.edu/~rada/wpt05/data/English-Hindi.training.tar.gz`.

[9]The task of cleaning up the corpus was carried out in a short span of time to meet shared task deadlines and therefore, even though the subset derived from the EMILLE corpus is much better than its original version, the quality of the data is still not perfect.

We also participated in the word alignment shared task with our initial version of the English-Hindi word alignment system (Aswani and Gaizauskas, 2005b). The initial version was based on a hybrid method which performed local word grouping on the Hindi sentences and used other methods such as initial versions of the dictionary lookup, transliteration similarity and nearest aligned neighbours approaches. We used the training data to obtain a list of named entities, cognates and to collect rules for local word grouping in Hindi sentences. We were able to achieve 0.77 precision, 0.61 recall and 0.68 F-Measure on the test data. The two other participating systems (Lopez and Resnik, 2005) had achieved scores of 0.43 precision, 0.56 recall, 0.49 F-Measure and 0.44 precision, 0.56 recall, 0.49 F-Measure respectively. The former was based on IBM Model 4 with improvements made to the HMM model whereas the latter was same as the former with an integration of a distortion model based on a dependency parse built on the English side of the parallel corpus.

For our experiments with sentence and word alignment algorithms presented in the PhD report, we distributed the extracted 3440 sentences into several smaller files. In order to prepare the dataset for the word alignment experiments and to avoid repeated preprocessing, English documents were processed with PRs such as Tokeniser, Sentence Splitter, POS Tagger and the Morphological Analyser. The Hindi documents were processed with Tokeniser and Sentence Splitter only. The resulting files are available for download from the `http://www.dcs.shef.ac.uk/~niraj/resources/emille/eh/clean.zip`.

3.4 Translation Dictionaries

We use a dictionary called *Shabdanjali English-Hindi dictionary* in the word alignment experiments reported in Chapter 7. This dictionary was developed at IIT, Hyderabad (India) and is available from `http://www.shabdkosh.com/archives/content/shabdanjali_-_english_hindi_dictionary/` as an open source resource under the *General Public License*. It contains about 26,000 English entries. The work of building the dictionary was carried out by school children, teachers and others. People in about eight cities were involved in the exercise where school teachers participated to some extent in order to correct and refine the

work carried out by the school children. As mentioned on the resource's website, since the quality of these entries is not yet proved to be excellent, it is necessary to verify entries in the dictionary and correct them before their use (Sangal and Sharma, 2004). Below we list some of the problems with the dictionary and actions we took to correct them.

In the dictionary, words in the Hindi entries are not properly delimited. For example, the English verb *adopt*, which means गोद लेना (*goda lenA*) in Hindi, is written as गोदलेना (*godlenA*), that is without any space between the two words गोद (*god*) and लेना (*lenA*).

Another problem is the improper use of vowels – the way they are merged with their preceding consonants. For example, the English verb *go* which means जाना (*jAnA*) in Hindi, is written as जानआ (*jAnaA*). Here the vowel आ (*A*) should have been merged with the previous consonant न (*na*) to make ना (*nA*).

In order to fix these problems and to improve the quality of translation, it was necessary to review every entry in the dictionary and correct it where necessary. Whilst the second problem discussed above can be solved automatically, to address the first problem we needed human intervention. This work was carried out with the help of two native speakers. Since the task does not require any other knowledge apart from where to add spaces and vowels, any person with knowledge of the Hindi language can be involved in this activity.

As mentioned in Chapter 2, the Devanagari script was originally used for Sanskrit. For sounds that are not found in Sanskrit, letters close to those sounds are modified with a dot (also known as *nukta*) placed below the character. For example, the character ज (*j*) is modified to ज़ (*z*) to represent the sound of *z*. It is very common to see words in Hindi sentences mismatching with words in a dictionary just because the nukta is missing in the former. In order to solve this problem, entries in the Hindi dictionary were converted into regular expressions. Every consonant with a nukta character was converted into an entry where the nukta character was made optional. Thus the entry matched even when the word to be matched in the dictionary did not have the nukta character.

For the experiments on the English-Gujarati language pair, there was also a need for an

English-Gujarati bilingual dictionary. Gujarati is a language for which there are few resources. Hardly any resources are available free-of-charge for research purposes on the internet. The English-Gujarati dictionary available at `http://www.gujaratilexicon.com/` is one of the most comprehensive bilingual lexicons. This dictionary contains 2.5 million words and the words such as verbs and nouns are all in their root forms. The lexicon was prepared by Shri R P Chandaria, who spent two decades preparing it. The website provides an interface to access the dictionary on line, and software to install it as a desktop-application. For our experiments, we used its online version.

As noted above in our discussion of Morphological Analysis, whilst most dictionaries tend to have entries in their base-forms, words used in sentences are often inflected. Therefore, before a word can be checked in dictionary, it must be converted into its base-form. For example, the Hindi noun लडका (*laDakA, boy*) can be used with different inflections such as लडकों (*laDako.n, boys*) and लडके (*laDake, boy(s)*). GATE already has a morphological analyser for the English language. However, it does not have a lemmatiser or morphological analyser for the Hindi or the Gujarati language. The Chapter 5 gives more information on morphological analysis and our efforts to provide a general framework for developing morphological analysers for new languages.

3.5 Chapter Remarks

In this chapter, we described the resources we use for our experiments in the following chapters. Whilst most of the components needed for preparing the text for sentence and word alignment are already available in GATE, there is a need for Hindi and Gujarati morphological analysers. We explained various problems with the English-Hindi and English-Gujarati dictionaries. We also explained problems with the EMILLE corpus and provided details of the process we carried out for cleaning the corpus.

With these resources in hand, we move into the next chapter to describe our sentence alignment algorithm for the English-Hindi language pair.

Chapter 4
Sentence Alignment

4.1 Outline of the Chapter

An alignment algorithm tries to find correspondences between segments of two texts, where a segment can be a word, single sentence, a paragraph or a section. Sentence alignment is a first step towards the more ambitious task of finding correspondences among words (i.e. word alignment). Sentence alignment techniques can rely on sentence lengths, on lexical constraints and correlations, or on cognates. These techniques range from simple techniques, such as character-length or word-length techniques to more sophisticated techniques which involve linguistic data.

In this chapter, we present a sentence alignment algorithm for the English-Hindi language pair[1]. We use a simple length-based approach that is similar to the one presented by Gale and Church (1993). However, as we describe later in the chapter, the difference lies in the criteria used during dynamic programming for selecting alignment pairs.

4.2 Related Work

Previous work on sentence alignment can broadly be categorized into three main categories: length-based, lexical and hybrid approaches. Length-based approaches align texts by computing length[2] similarities whereas the lexical approaches use external resources (usually language dependent) to perform text alignment. Hybrid approaches are a mix of the two.

Different evaluation measures have been used by different researchers to report results of their alignment experiments. These include measures such as *Precision* (P), *Recall* (R) and *F-Measure* (F1), *Accuracy*, and *Alignment Error Rate* (AER). Alignment algorithms may not identify all alignment pairs specified in the reference gold standard. It is also possible that new alignment pairs have been identified which do not exist in the gold standard. Given this, a user may want to know how many of the total alignment pairs, identified by an alignment

[1]The sentence alignment algorithm described in this chapter has been published in Aswani and Gaizauskas (2009).

[2]Usually the length is counted in number of words or number of characters.

system are correct. The measure which reports this is called *Precision*.

$$P = \frac{Number\ of\ alignment\ pairs\ correctly\ identified\ by\ the\ system}{Number\ of\ alignment\ pairs\ identified\ by\ the\ system}$$

Also, a user may want to know how many of the total alignment pairs, identified by the system actually exist in the reference gold standard. The measure which reports this is called *Recall*.

$$R = \frac{Number\ of\ alignment\ pairs\ correctly\ identified\ by\ the\ system}{Number\ of\ alignment\ pairs\ in\ the\ gold\ standard}$$

A measure that combines these two measures is the harmonic mean of precision and recall and is known as *F-Measure* or balanced *F-score*.

$$F = \frac{2 \times P \times R}{P + R}$$

Often people report their results using the *accuracy* measure. It indicates what percentage of total alignment pairs have been *correctly* identified. Sometimes researchers report accuracy figures between 0 and 1 where 1 means 100% accuracy. The *AER* measure is exactly the opposite of the accuracy measure. In other words, it indicates what percentage of total alignment pairs have been *incorrectly* identified.

Partial Alignments

The advantage of considering partial alignment is that the successful linkings of multiple sentences are visible and rewarded but also that partially correct linkings are not discounted entirely (Ahrenberg et al., 2002).

In order to reward partially correct alignments, if, for example, the sentence numbers 3 and 4 in the source document are aligned with sentence numbers 5 and 6 in the target document, we create four 1:1 pairs in the gold standard (i.e. (3,5), (3,6), (4,5) and (4,6)). Given this if an algorithm finds that the third and the fourth sentences are aligned with only the fifth

sentence in the target document, we consider these two pairs (i.e. (3,5) and (4,5)) in scoring.

Using these pairs the precision is calculated by dividing the total number of pairs correctly identified by the system by the total number of pairs identified by the system (i.e. $2/2 = 1.0$) and the recall is calculated as the total number of correct pairs divided by the total number of pairs in our gold standard (i.e. $2/4 = 0.5$).

4.2.1 Length-based Approaches

Since pure length-based techniques have no concern with word identity or meaning, these approaches are considered knowledge-poor approaches. Such methods have the advantage that they can be applied on any language pair[3] without requiring any language specific resources. Lexical and hybrid approaches, which make use of linguistic data, are favoured where length-based approaches fail to establish any correlation between the lengths of the two languages under consideration.

Computationally, length-based approaches are superior as they involve less resources and thus can provide really fast implementations capable of dealing with huge amounts of data. Brown et al. (1991) and Gale and Church (1993) are amongst the most cited length-based approaches in text alignment. The program known as *align* was implemented by Gale and Church (1993). Before they perform sentence alignment, they align paragraphs in the source and the target language. Then, considering one pair of aligned paragraphs at a time, they assign a probabilistic score to each proposed pair of aligned sentences based on the ratio of lengths of the two sentences (calculated in number of characters) and the variance of this ratio.

They used a corpus to derive the parameters: c and s^2. The c parameter gives the expected number of characters in the target language per character in the source language whereas the s^2 parameter, the variance, gives an indication of the number of characters that the target

[3]Longer sentences in one language tend to be translated into longer sentences in the other language. Similarly shorter sentences in one language tend to be translated into shorter sentences in the other language. This is the assumption on which length-based approaches tend to work (Gale and Church, 1993).

language could vary per character in the source language. The value of the c parameter is estimated by counting the number of characters in source paragraphs and dividing it by the number of characters in target paragraphs. The value of the s^2 is modelled proportional to length. First, the lengths of source paragraphs is plotted on the horizontal axis. Then, the square of the length difference of source and their parallel target paragraphs (an estimate of variance) are plotted on the vertical axis. Such a plot indicates effect of a change in source length upon the variance. Finally, the parameter s^2 is determined by the slope of the robust regression line in the plot. From their experiments with the English-French and English-German languages, they report that the most European languages are similar with $c \approx 1$ and $s^2 \approx 6.8$ on average.

Using these measures and considering one pair of aligned paragraphs at a time, their program calculates the cost of aligning sentences as 1:1 (substitution), 1:0 (deletion), 0:1 (insertion), 1:2 (expansion), 2:1 (contraction), and 2:2 (merging) such that all the sentences within the aligned paragraph pair are used up and the overall distance between the source sentences and their aligned target sentences is minimised under the maximum likelihood alignment. Their distance measure is an estimate of $-logProb(match|\delta)$, where δ depends on l_1 and l_2, the lengths of the two portions of text under consideration and is computed using the following equation:

$$\delta = (l_1 \text{ - } l_2 c) \ / \ \sqrt{l_1 s^2}$$

They appeal to Bayes Theorm to estimate $Prob(match|\delta)$ as a constant times $Prob(\delta|match)Prob(match)$. They ignore the constant as it will be same for all proposed matches. They compute a value for $Prob(match)$ from their corpus (e.g. frequency for 1:1 pairs in their corpus was 1167 out of 1312 pairs giving $Prob(1:1) = 0.89$) and conditional probability $Prob(\delta|match)$ as:

$$Prob(\delta|match) = 2 \times (1 - Prob(|\delta|))$$

Here, they compute a value for $Prob(|\delta|)$ by integrating a standard normal distribution and using the *pnorm* function as described in the Abramowitz and Stegun (1964) (p. 932, equation 26.2.17).

As specified earlier, they take a pair of parallel paragraphs at a time and use a dynamic programming algorithm to compare the final scores of all possible alignments (i.e. 1:0, 0:1, 1:1, 1:2, 2:1, 2:2) within the paragraph. They choose the overall alignment that yields the least score.

They report 0.958 accuracy on the *UBS corpus*, a trilingual corpus of economic reports issued by the Union Bank of Switzerland in English, French and German, which increases to 0.979 on the *Canadian Hansard Corpus* for which the Brown et al. (1991) reports 0.994 accuracy. The main difference between the works of Brown et al. (1991) and Gale and Church (1993) is that the former used words as the unit to measure sentence length whereas the latter used characters as the unit to measure sentence length. In their experiments using the number of words instead of the number of characters as the unit for sentence length gave them a lower accuracy of 0.935. Reporting results of experiments when paragraphs are not pre-aligned, they show that the accuracy reduces by a factor of three. However, it is not necessary to pre-align paragraphs manually. Their approach can be used in two iterations – initially to align paragraphs and then to align sentences within the aligned paragraphs. The other difference is that Brown et al. (1991) use the EM algorithm to estimate values for their parameters.

4.2.2 Lexical Approaches

Length-based algorithms usually require sentences to be in the same order in source and target texts (i.e. for there to be no cross alignments). Since these approaches only rely on length they can fail to decide which target sentence should be aligned with a source sentence in the case of more than one potential match being found in the target text. Kay and Roscheisen (1993) tried a lexical method for sentence alignment. In their algorithm, assuming the first and the last sentences of the texts are aligned, they repeat a three step procedure until most of the sentences in the text are aligned. In the first step they form an envelope

of possible alignments from the Cartesian product of the lists of sentences in the source and the target language. They exclude all those alignments whose respective distances differ too greatly from their anchor points. In the second step they derive lexical correspondences between the words of partially aligned sentences. Lexical correspondences are derived on the basis of word distribution. Two words with same distribution are assumed to correspond to each other. Finally in the third step, they consider the most reliable pair of a source and target sentences to be that which contains the most possible lexical correspondences. They achieved 0.960 accuracy on Scientific American articles after four passes of the algorithm.

Warwick et al. (1989), Haruno and Yamazaki (1996) and Mayers et al. (1998) are all similar methods with minor differences in the type of resources used for performing sentence alignment. Warwick et al. (1989) make an assumption that each block of the source text corresponds to a single continuous segment of the target text. They calculate the probability of word pairings on the basis of the frequency of a source word and the number of possible translations appearing in the target segments. They suggest using a bilingual dictionary to build word-pairs. Since bilingual dictionaries contain base-forms, Mayers et al. (1998) pre-process the text to find a base-form for each word. They tried this method in an English-Japanese alignment system and got an accuracy of about 0.896 for 1:1 and 0.429 for 2:1 sentence alignments. Haruno and Yamazaki (1996) use a POS tagger for both the source and the target languages and use an online dictionary to find matching word pairs. They pointed out that though dictionaries cannot capture context dependent keywords in the corpus, they can be very useful to obtain information about words that appear only once in the corpus. Since all these methods try to find word correspondences, along with the sentence alignment, they also leave behind partial word alignment in the aligned sentences.

According to Bharti et al. (2002) there are substantial additions and deletions that can occur on either side, particularly when the languages are far apart. Their experiments show that Gale and Church (1993) did not work well on the *India-today* parallel corpus and they felt the need for lexical information. They use a medium coverage bilingual (English-Hindi) dictionary (25,000 entries) and chunkers for both the languages. Chunkers are used to break

down sentences into a number of small chunks. They mark noun phrases and verb phrases as chunks in the sentences of a given text and use the bilingual lexicons to match words from the source sentences with the target chunks. They consider the sentence in the target language with the maximum number of matched chunks to be the alignment of the source sentence. Their algorithm requires sentences to be in the same order in source and target texts and performs only 1:1 alignment in its first phase. To perform n:m alignment they first consider the number of chunks in both the source and the target sentences and align them only if the difference in number of chunks is big (e.g. source sentence has 15 chunks whereas target sentence has only 5 chunks). In this case, they consider the next target sentence appearing after the aligned target sentence and align them together with the source sentence. Using this method, they obtained 0.943 accuracy on the India-today corpus.

4.2.3 Hybrid Approaches

Wu (2000) proposes hybrid approaches as the best combination of running time, space, accuracy, and ease of implementation for aligning sentences in tightly translated bitexts. Melamed (1997a), Simard et al. (1993) and Moore (2002) are some of the examples of hybrid sentence aligners. Simard et al. (1993) combines the method proposed by Gale and Church (1993) and a method based on cognate matching. In his approach, Melamed (1997a) suggests that using only cognates, punctuation and numbers can be enough to locate parallel sentence pairings but using a bilingual dictionary increases chances of locating more translation pairs and thus yields better alignment results. Moore (2002) uses a length-based approach (Brown et al., 1991) to pre-align sentences. As a second step, his algorithm only preserves 1:1 high probability alignment pairs. Later he uses the IBM translation model 1 (Brown et al., 1993) based on the preserved alignment pairs to locate word correspondences to improve the alignment results.

Singh and Husain (2005) tested several sentence alignment algorithms (Brown et al., 1991; Gale and Church, 1993; Melamed, 1997a; Moore, 2002) under different conditions (i.e. corpus type, corpus size, differences in the sizes of source language and target language corpora and

noise). Through their experiments, they show that the performance of a sentence alignment algorithm varies significantly depending on the conditions of testing. For example, they show that Moore (2002) performed better than Gale and Church (1993) (0.922 precision and 0.915 recall compared to 0.841 precision and 0.849 recall) when the algorithms were tested on a noisy test data whereas the latter performed better than the former on clean data (0.987 precision compared to 0.951 precision with 1.0 recall on both occasions). When comparing results on a mixed dataset, Singh and Husain (2005) show that the approach suggested by Moore (2002) achieved the best precision (0.929) whereas the Gale and Church (1993) approach achieved the best recall (0.843). While Bharti et al. (2002) report poor results for using Brown et al. (1991) on the *India-Today* corpus, Singh and Husain (2005), on the contrary, show that the algorithm performs better on the *EMILLE* corpus. However, it must be noted that experiments carried out by Singh and Husain (2005) were focused only on 1:1 alignment pairs.

Ceausu et al. (2006) also present a hybrid alignment system. They describe a hybrid sentence aligner which runs through several stages. They train an SVM (Support Vector Machine) model on a gold standard corpus consisting of 400 manually aligned samples out of which half are positive and half are negative examples. For training they use features such as difference in length, difference in their relative position in the document, number of punctuation marks, number of formulas and numbers, number of dates etc. Using the trained classifier, they generate a list of possible candidates for each sentence in the source language and for each candidate pair with a score above 0.9 they estimate the table of translation equivalents using a slightly modified version of the IBM model 1 which does not consider null alignments (Moore, 2002). Having identified translation equivalents, they use them as additional features to re-train the initial SVM classifier. This time, they use the SVM classifier to obtain 1:1 pairs. The pair is considered *good* only if its score is above a set threshold. In order to find out n:m sentence pairings the 1:1 pair is expanded to include neighbouring sentences as candidate alignments. They use the EM algorithm to re-estimate the sentence-pair probabilities. Scores of the neighbouring candidate alignments are modified based on the score of the

pair identified as *good* in the current iteration. For example, they increase the probability of the closest candidate and reduce the probability for candidates cross-linked with the current pair. To locate new *good* pairs, they lower their threshold value in each iteration until there are no more *good* pairs found. They obtained higher F-Measure scores (0.989, 0.995 and 0.991 on English-Italian, English-French and English-Russian language pairs respectively) in comparison to the bilingual sentence aligner presented by Moore (2002). As can be seen, their method requires heavy processing, however the results they acquire are really good. Not only this but their algorithm is capable of aligning texts which do not follow the same linear order.

Even though hybrid methods produce better results, these approaches tend to be slow and thus despite the potential for highly accurate alignment, they may be unsuitable for very large collections of text. Given that lexical methods too can be computationally expensive, our idea was to try a simple length-based approach and use lexical methods only if the results of the former are not acceptable.

4.3 Sentence Alignment for the English-Hindi Language Pair

When examining the EMILLE corpus, we observed that in most cases, one English sentence aligns with only one Hindi sentence. However, it is also possible that one or more English sentences align with one or more Hindi sentences. Figure 4.1 shows a snippet of English-Hindi parallel texts taken from the EMILLE corpus. The alignment boundaries are marked with $<a>$ and $$ whereas the sentence boundaries are marked with $<s>$ and $</s>$. As can be seen, the English text has in total 5 sentences whereas the parallel Hindi text has 7 sentences. The first two English sentences are aligned with the first Hindi sentence. English sentences 3 and 4 are aligned with Hindi sentence 2 and 3 respectively. Finally English sentence 5 is aligned with remaining Hindi sentences 4-7.

The assumption made by Brown et al. (1991) and Gale and Church (1993) was that the lengths of two parallel sentences are positively correlated. In other words, addition of some

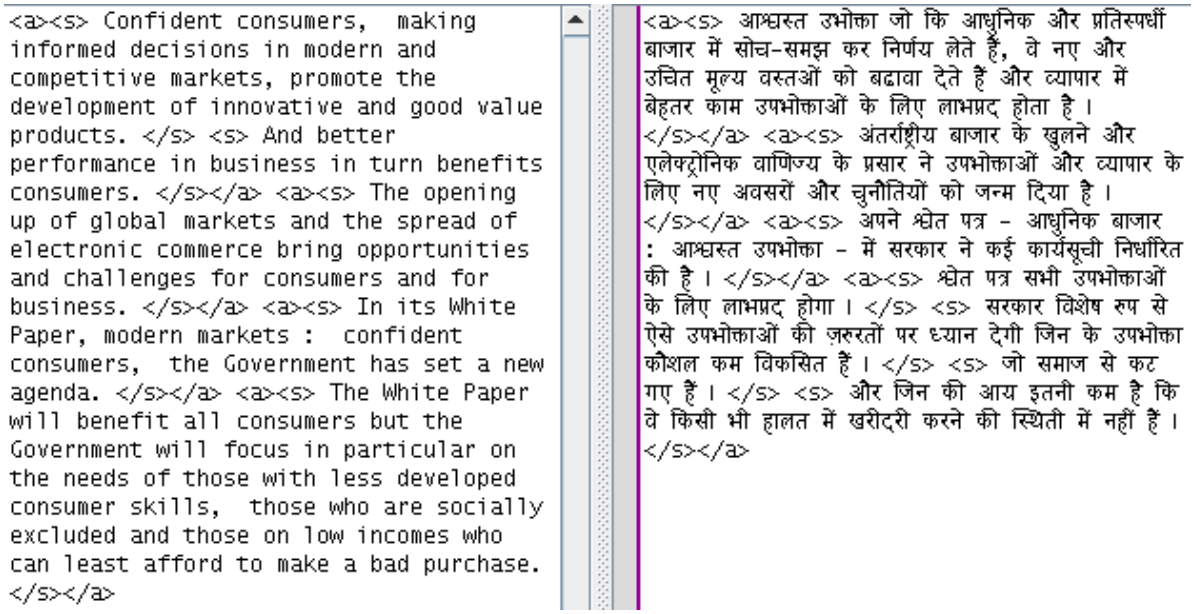


Figure 4.1: English-Hindi Parallel Text

characters in a source sentence should give rise to a certain increased number of characters in the target sentence. In their experiment with the English and the German languages, Gale and Church (1993) found that the expected number of German characters per English character is 1.1. This indicates that the lengths of the two parallel texts do not vary a lot. Statistical correlation between the lengths of the parallel sentences explains the degree at which the change in the length of a source sentence affects the length of its parallel target sentence. In other words, if the correlation is very strong, given a length of a source sentence a probable length of the target sentence can be predicted. Gale and Church (1993) found that the correlation between the lengths of English and German sentences was 0.991. Since the correlation is very high, it is possible to align sentences by finding those sentences with the nearest length. Assuming sentences and their translations occur in the same order in both texts, they also report that they observed 1:1 alignment in most cases which makes the alignment task even easier.

In order to find out how the lengths of the two languages correlate, we divided the lengths of the Hindi sentences in the EMILLE corpus with the lengths of their parallel English sentences. Figure 4.2 shows two graphs giving the distribution of ratios of lengths of characters

and words per sentence in English and Hindi.

In the first graph, the length is calculated in number of characters whereas in the second graph, the length is calculated in number of words. The *x-axis* in the first graph gives number of Hindi characters per English character and in the second graph gives number of Hindi words per English word. The *y-axis* in both the graphs indicates the number of alignment pairs for each value on the x-axis.

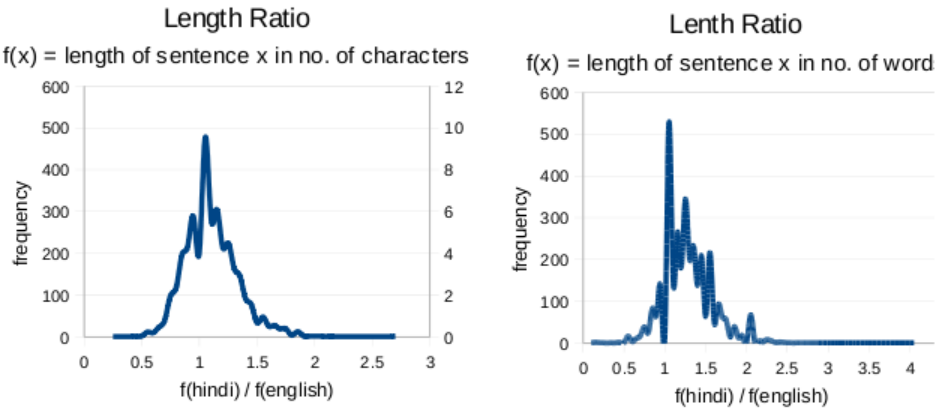


Figure 4.2: Distribution of the Ratios of Lengths in Characters and Words

The graphs were created based on 3420 sentence pairings randomly collected from the EMILLE corpus which included several 1:1 and n:m sentence pairings. The pairings were manually aligned for purposes of the experiment. In the first graph, out of the 3420 sentence pairings, 184 sentence pairings had the same length. We found that 2033 Hindi sentences were longer than their respective English sentences in comparison to 1203 pairings where the English sentences were longer than their respective Hindi sentences. Where the pair has more than one sentence in either the source or the target language, the length is the aggregated total of the lengths of the sentences. In the second graph, only 22 sentence pairings had the same number of words. We found that 2988 Hindi sentences had more words than their respective English sentences in comparison to 410 sentence pairings where the English sentences had more words than their respective Hindi sentences. Plotting their lengths in number of characters and words produces the graphs shown in Figure 4.3.

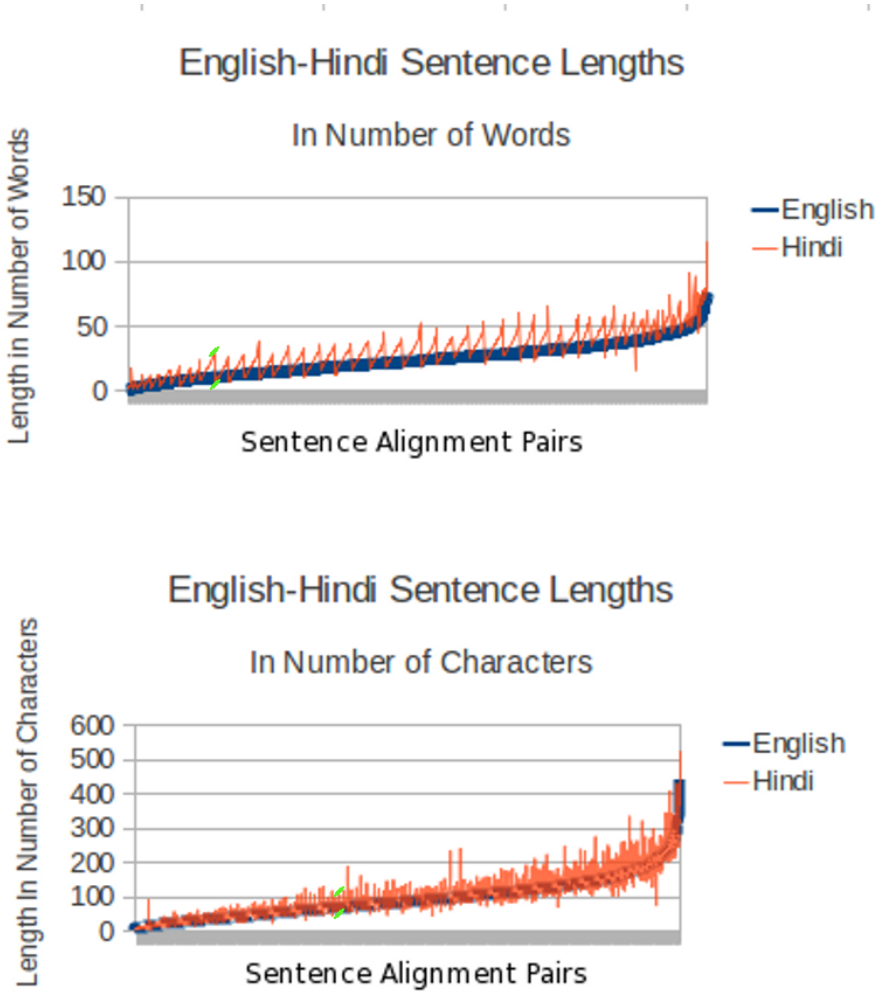


Figure 4.3: Lengths of English-Hindi Parallel Sentences

As can be seen in this figure, for instance, if the length of an English sentence is 80 characters, its corresponding parallel Hindi sentences have lengths varying between 55 and 109 characters. The same is true when the length is measured in number of words. If the length of an English sentence is 14 words, the corresponding parallel Hindi sentences have lengths varying between 7 and 29 words.

4.3.1 Experiments with Gale and Church (1993)

As described earlier, Singh and Husain (2005) report that the approach proposed by Gale and Church (1993) gave them the best result on the EMILLE corpus. As their experiments

were carried out only on 1:1 sentence pairs, we decided to use the same algorithm, i.e. Gale and Church (1993), but on a different dataset collected from the EMILLE corpus.

Dataset

Our test dataset contains 3420 sentence pairings randomly collected from the EMILLE corpus. It includes several 1:1 and n:m sentence pairings. As all the sentence pairings were collected randomly from the EMILLE corpus, we could not preserve the paragraph information.

In order to align sentences of one paragraph at a time, we drew virtual paragraph boundaries around every 10 sentences or the nearest number required to form valid sentence pairings.

Results and Discussion

We applied the algorithm proposed by Gale and Church (1993) on our dataset and obtained 0.959 (precision) and 1.0 (recall). It must be noted that the dataset we use for our experiments has been manually cleaned up. The cleanup process involved fixing out of order translations, removing duplicate translations and removing any sentences for which there was no parallel text available in the target text. Since we have fixed the ordering of all the sentences, we expect to obtain better results in comparison to the results obtained when running it over noisy data as reported by Singh and Husain (2005)

We analysed the results to check if there was any scope for improvement. The recall figure suggests that the system was able to identify all the correct pairings. However, the precision suggests that there must be a few pairing generated by the system that are wrong and do not exist in the gold standard.

When searching for such pairings, we noticed that the algorithm had generated some many-to-many sentence pairings which should have been aligned as smaller independent pairings. For example, the system had aligned two sentences in the source language with two sentences in the target language. When comparing this alignment with the alignment of the same sentences in the gold standard, we found that they should have been aligned as two separate

1:1 sentence pairings.

Because of the way we treat partially correct alignments, the 2:2 pairing was converted into four individual 1:1 pairs. When the four pairs were compared with the two 1:1 pairs in the gold standard, all the required pairs were found to have been identified by the system. Thus, resulting in scores of 0.5 (precision) and 1 (recall) for the given sentences.

As discussed earlier, the strategy employed by the Gale and Church (1993) is to choose an overall alignment that yields the least distance score. Here, we observed that the results can be improved if we give priority to forming more sentence pairings with fewer sentences in each pairing. Despite receiving such high scores for the Gale and Church's approach, we wanted to check if employing such a strategy of creating higher number of pairs would bring any improvements to the system or not.

4.3.2 Our Approach

We define our task as that of learning a function, that given the length of an English sentence(s) in number of characters/words, returns the estimated length of a parallel Hindi sentence(s) in number of characters/words. However, as observed earlier, the length differences between the two languages can deviate substantially on both sides of the mean. Therefore, what could be useful when computing the length of candidate parallel Hindi sentence(s) is the interval of possible lengths.

We obtain the average number of English characters needed per Hindi character (i.e. $\frac{f(e)}{f(h)}$, where $f(x)$ is the length of the sentence x). This is done in two steps. In the first step, considering one sentence pairing at a time, we divide the sum of lengths of the English sentence(s) by the sum of lengths of the parallel Hindi sentence(s) in that pairing. In the second step, all the values obtained in the first step are summed together and divided by the number of pairings to obtain the average number of English characters needed per Hindi character.

$$\mu = \frac{\sum_{i=1..N} \frac{f(E_i)}{f(H_i)}}{N}$$

In statistics, the standard deviation σ is a measure of the variability of a dataset. In our case, it serves as a measure of the variability in the ratios of the lengths of the English sentences and the lengths of the Hindi sentences. A low σ indicates that the lengths of the parallel sentences are very close to the expected value (μ). According to the *empirical rule*, about 68% of values drawn from a normal distribution are within one standard deviation away from the mean, about 95% of the values are within two standard deviations (i.e. 2σ) and about 99.7% lie within three standard deviations (i.e. 3σ). Therefore, we calculate the standard deviation σ and use 3σ to generate an interval with a lower boundary that is equal to $\mu - (3 * \sigma)$ and upper boundary that is equal to $\mu + (3 * \sigma)$. Given this interval and a pair of sentences and assuming that the ratio of sentence lengths between aligned sentences is normally distributed, one can tell whether the two sentences should be aligned with each other with confidence 0.997 simply by obtaining the ratio of the sentences' lengths and checking whether it lies in the interval.

However, when there are multiple possibilities, we need to decide which of the different possibilities should be considered for alignment. In our experiments, we perform alignment only for sentences within one paragraph at a time. We try to establish different pairings (i.e. n:m where n,m=[1,2,3,4]). We assume that there are no crossing alignments. If the first sentence in the source language is aligned with the first and the second sentences in the target language then the second sentence in the source language will align with at least the third sentence and may be fourth, fifth and six as well. Considering all these possible pairings we calculate the ratio R by dividing the total length of English sentences with the total length of the Hindi sentences.

$$R(E_{i..n}, H_{j..m}) = \frac{\sum_{p=i..n} f(E_p)}{\sum_{q=j..m} f(H_q)}$$

If the ratio R lies between the set interval, the pairing is said to be valid and otherwise the

pairing is discarded. If the pairing is valid, we return its distance D from the mean value μ.

$$D(E_{i..n}, H_{j..m}) = |\mu - R(E_{i..n}, H_{j..m})|$$

Given the above approach, it is possible to obtain more than one valid pairing. For example, for the first three sentences in both the source and the target languages, the system has identified the following pairings as valid pairings. Please note that the pairings have sentence numbers listed in them. (1,2;1,2) means that the sentences numbered 1 and 2 in the source language are aligned with sentences numbered 1 and 2 in the target language.

(1;1), (2;2), (3;3), (1,2;1,2), (2,3;2,3) and (1,2,3;1,2,3)

Referring to previous work, dynamic programming has been used (Brown et al., 1991; Gale and Church, 1993) to select the overall alignment that yields the least distance and thus maximizes the likelihood of correct alignment. We, too, use dynamic programming but to select the overall alignment with different criteria.

- The first priority is to choose the alignment in which no sentence is left unaligned. In other words we do not allow 0:1 and 1:0 assignments.

  Even though the algorithm proposed by Gale and Church (1993) has provisions to deal with 1:0 and 0:1 pairs, it is important to mention that their evaluation of 1:0 and 0:1 pairs suggests that all such pairs were identified incorrectly (see Table 6 on p.11 in Gale and Church (1993)). The authors, therefore, suggest to use some language specific component to deal with the problem.

  When sentences need to be aligned only based on their lengths, it can get trickier to handle 1:0 and 0:1 assignments. For example, if there was an extra sentence at the start of a source paragraph which is not a translation of any of the sentences in the target paragraph and the the length of this sentence, however, is such that it is a perfect match for the first sentence in the target text, it is not guaranteed which sentence in

the target text will be left unaligned. Also, there is a possibility that because of such an alignment, other alignments in the paragraphs may also get affected.

Although, the criteria introduced by us could lead to failure on a paragraph where there are sentences with no alignments found in the target text, we believe that if that happens, one would be able to find a paragraph that has an extra sentence which should not be there. With the provision of 0:1 and 1:0, such an error may get bypassed without noticing. Thus, not providing such a provision provides an opportunity to track down the paragraphs with undesired text in them.

- Secondly, we choose an alignment with a higher number of pairings in it. In other words, if there is a choice between the alignment with a smaller number of sentence pairings and an alignment with a larger number of sentence pairings, we choose the latter. We believe that this criterion is acceptable as all pairings which pass the interval test are valid pairs.

- Finally, if there is more than one alignment with the same number of pairings but with different overall scores, we choose the one with the least overall distance.

4.3.3 Experiments and Results

We performed 10-fold cross validation on our dataset. In this case, the data was divided into 10 folds with an equal number of sentence pairings in each of them. The sentence pairings were randomly distributed in each of the folds. 9 folds were used for estimating the parameters (μ and σ) and the remaining one for testing. Each training dataset had 3078 sentence pairings whereas the test dataset had 342 sentence pairings in it. The process was repeated 10 times, giving each fold a turn to be tested upon. The table 4.1 shows the various results collected through this experiment. Please note that the length was calculated in number of characters for this experiment.

The highest precision obtained from the 10 different accuracy figures collected during the 10-fold evaluation exercise is 1. However, since the sentence pairings are drawn randomly in

Table 4.1: 10-Fold Evaluation of the Sentence Alignment Algorithm (length measured in characters)

μ_1	σ_1	**Precision**	**Recall**
0.962	0.220	0.981	1.0
0.964	0.223	0.981	1.0
0.963	0.211	0.981	1.0
0.962	0.220	1.0	1.0
0.963	0.219	0.990	1.0
0.964	0.221	0.972	1.0
0.965	0.221	0.981	1.0
0.964	0.222	0.981	1.0
0.964	0.218	0.981	1.0
0.963	0.222	0.981	1.0
Average		**0.983**	**1.0**

these folds, it is likely to have different results every time the evaluation is performed. We performed 10-fold evaluation 10 times and obtained 0.983 as an average precision. Comparing these results with the results of Gale and Church (1993) on the same dataset, it can be seen that the changes in Gale and Church's approach improves the results.

The same experiment was repeated by calculating length in number of words. Table 4.2 shows results collected during that experiment.

The highest precision and recall obtained from the 10 different figures collected during the 10-fold evaluation exercise are 0.981 and 1.0 respectively. The averages are 0.985 precision and 0.944 recall. These results are lower than the results for length measured in number of characters. It could be due to the fact that there are more of characters then words and therefore there is less uncertainty when the length is measured in characters.

Usability of any algorithm is decided by looking at the issues such as how portable the algorithm is across different languages and how it performs in different conditions (e.g.

Table 4.2: 10-Fold Evaluation of the Sentence Alignment Algorithm (length measured in words)

μ_1	σ_1	**Precision**	**Recall**
0.853	0.335	0.990	0.942
0.853	0.333	0.980	0.932
0.853	0.332	0.990	0.942
0.852	0.316	0.990	0.942
0.853	0.335	0.980	0.932
0.853	0.333	0.990	0.942
0.852	0.324	0.980	0.932
0.852	0.314	0.980	0.932
0.850	0.281	0.981	1.0
0.854	0.336	0.990	0.942
Average		**0.985**	**0.944**

clean vs noisy data) etc. However, as pointed out by Rosen (2005) and Singh and Husain (2005), it is difficult to choose a single method with consistent performance not only across different language pairs but even across different datasets or corpora for the same language pair. There are several factors that affect the performance of an algorithm, e.g. noise in data, order of text (i.e. if the parallel text in the source and the target language are in the same order) and use of different external resources such as bilingual dictionaries for lexical methods. On the other hand, it is also important to decide if the user is interested in better precision or recall or may be in faster processing vs better results in lieu of a longer wait.

The experiment presented in this chapter is based on a pure length-based approach which is capable of dealing with many-to-many alignments and can be used for other language pairs without making any changes to it. However, it is important to mention that if there are crossing alignments (i.e. the translation does not preserve sentence ordering) or if there is any sentence in the text for which there is no aligned sentence in the other language, the method could generate an error prompting users to check their texts.

4.4 Chapter Remarks

In this chapter, we presented a sentence alignment algorithm for the English-Hindi language pair. We used a simple length-based approach that is similar to the one presented by Gale and Church (1993).

The main difference lies in criteria used during dynamic programming for selecting alignment pairs. Similar to their algorithm, we too use dynamic programming to choose an alignment with the least score however we give priority to smaller alignment pairings.

The other difference is in the types of alignment pairs that the two algorithms allow to form. For example, in Gale and Church (1993), it is possible to form pairings such as 1:0 and 0:1. In our case, it is not. We only allow aligning one or up to four sentences in a source language with one or up to four sentences in the target language. We intend to carry out further experiments, in near future, for allowing provisions for 1:0 and 0:1 alignment pairs and cross alignments.

In order to check the portability of our algorithm across other language pairs, we carried out a similar experiment on the English-Gujarati language pair. Details of the experiment are provided in Chapter 8.

Chapter 5
Hindi Morphological Analyser

5.1 Outline of the Chapter

A word can be of two types: simple and compound. A simple word consists of a root or stem together with suffixes and prefixes. When a root word is modified by suffixes and/or prefixes, the word is said to be an inflected form of the root word. Here, the appended suffix or the prefix are not all independent words and cannot occur as separate words in the text. Constituents of simple words are called morphems. A compound word, on the other hand, is composed of two or more independent words conjoined together. Each of the constituent words in a compound word is either a compound word or a simple word and may be used independently as a word.

Stemming is a process of splitting off fairly vacuous function morphems and there can be a number of morphems together in the suffix. Often the stems obtained this way are not complete root forms. For example, “institut” for the word “institution”.

Given a word, a morphological analyser retrieves information such as the root form of the word, its lexical category, gender, number, person, tense etc. For example, the root form of a word “institution” is “institute” which is different from its stem “institut”.

Often words are ambiguous carrying more than one meaning. For example, a word “swimming” can be assigned a grammatical category of noun or a verb depending on how it is used in the sentence. What is the correct root form a word also depends on the grammatical category of the word. For example, if the word is to be interpreted as a noun, “swimming” is the root form of its own but if it is to be interpreted as a verb, “swim” would be the correct root form.

The ambiguity can be resolved by looking up in the context. But when a morphological analyser is given a word without any context, it returns grammatical information for each possible meaning of the word. Opposite of an analyser is a generator. Given information such as a root word, person, gender, number etc., the generator generates an inflected form for the given root word.

For some languages dealing with its morphology is not a very serious problem. For example, in case of the Hindi language and for other similar languages such as Gujarati and Punjabi, it is possible to come up with a set of suffix replacement rules to develop a sophisticated morphological analyser. However, the problem can become very complicated when dealing with other languages, such as poly-synthetic languages, where the ratio of morpheme-to-word is very high and often the words are too long.

Indian languages are morphologically rich languages and therefore morphological analysis is seen as an essential step towards any Indian language processing task. A considerable amount of work has been put into development of stemmers and morphological analysers (Lovins, 1968; Porter, 1980; Majumder et al., 2007; Krovetz, 1993; Hull, 1996; Shrivastava et al., 2005; Ramanathan and Rao, 2003; Pandey and Siddiqui, 2008). The majority of these approaches use hand-crafted suffix-replacement rules but a few try to discover such rules from corpora. While all these approaches remove or replace suffixes, approaches such as Krovetz (1993) and Hull (1996) propose derivational stemmers which are based on prefixes as well.

In this chapter we present a rule-based morphological analyser[1] where the rules are acquired semi-automatically from corpora. We propose an approach that takes both prefixes as well suffixes into account. Given an inflected Hindi word, our system returns its base-form. It uses a dictionary, and a monolingual corpus to obtain suffix-replacement rules. Most rules in our approach are learnt automatically. However, a few of them need to be verified manually.

The morphological analyser is primarily used by our word alignment algorithm (see Chapter 7) to obtain root forms of Hindi inflected words. The root forms are used by the components such as Dictionary Lookup PR and the Local Word Grouping PR to help in aligning words.

In the following sections, we first look at some of the well-known methods used for stemming and morphological analysis. Then, we describe our morphological analyser for the Hindi

[1]The work presented in this chapter on developing a morphological analyser for the Hindi language has been published in Aswani and Gaizauskas (2010b).

language.

5.2 Related Work

Porter's stemmer is amongst the most cited stemmers in the literature (1980). His stemmer is based on predefined hand crafted suffix-replacement rules. His algorithm proceeds through a fixed succession of five stages, with a different set of rules included in each stage. The stemmer uses an explicit list of suffixes, and, with each suffix, the criterion under which it may be removed from a word to form a valid stem.

Majumder et al. (2007) present a system called YASS (*Yet Another Suffix Stripper*) that uses a corpus to learn suffix stripping rules. They define a set of string distance measures for clustering the related words. Their main intuition is to reward long matching prefixes and to penalize early mismatch. Using these metrics they decide whether the words under consideration belong to the same cluster or not. They match the longest common prefix to cluster similar words and stem them to the central word in that cluster to infer suffix-replacement rules.

Unlike other approaches, Hull (1996) proposes a derivational stemmer which not only removes suffixes but prefixes as well. For example, *superconduction* is stemmed to the word *conduct*. From their experiments with the derivational process, Hull (1996) shows that it is a bad idea to remove prefixes from words which in most cases results in over-stemming. On the other hand, Krovetz (1993) reports that a derivational stemmer performs slightly better than an inflectional stemmer.

Xu and Croft (1998) suggest that rules of stemming algorithms should be modified prior to their use on any corpus and the corpus itself should be used for this purpose. They suggest that this step is necessary as the rules learnt from one corpus might not reflect the language used in other corpora. They propose to use an aggressive rule based stemmer or a language independent (n-gram) stemmer to create equivalence classes and refine them by using corpus co-occurrence statistics. They assume that the word variants that should be conflated occur

in the same text windows. They propose to use this method for refining suffix-replacement rules.

Ramanathan and Rao (2003) propose a lightweight stemmer for the Hindi language. Their stemmer is based on some hand-crafted rules which they claimed to have derived after careful observations of the Hindi language. Based on their analysis of verbs, nouns, adjectives and adverbs they produced a list of suffixes for each of these categories. Their system conflates terms by stripping off word endings from the suffix list on a *longest match* basis.

Pandey and Siddiqui (2008) use some heuristic rules which are derived using a split-all method. From their experiments with the Hindi language, they automatically obtained a list of 51 suffixes from a set of Hindi documents from the EMILLE corpus. They strip off the longest suffixes to perform stemming. In their method, words are split to give n-gram *(n=1..l)* suffixes where l is the length of the word under consideration. Having obtained suffixes and stems for each word, they calculate suffix and stem probabilities over the corpus and multiply them to obtain split probability. They use the highest split probability to define segments and group words accordingly. For example, the words such as लडका (*ladakaa, a boy*), लडकों (*ladakon, boys*), and लडके (*ladake, boys*) with common stem लडक (*ladak*) are grouped together with suffixes ा (*aa*), ों (*on*) and े (*e*). They also apply heuristic rules that attempt to concatenate stems with whose suffixes have a common prefix. For example, आवश् (*aavash*) is a stem with three different suffixes यक (*yak*), यक्ता (*yaktaa*) and यक्तानुसार (*yaktaanusaar*). They obtain a common prefix string from these suffixes (i.e. यक (*yak*) and concatenate it with the stem to make it आवश्यक (*aavashyak, necessary*). They performed experiments on two different datasets and report 0.857 as precision and recall for the first run and 0.899 as precision and recall for the second run.

To our knowledge, the morphological analyser presented by Shrivastava et al. (2005) is the most accurate system available for the Hindi language (based on the precision score reported by the authors). They use a hand-crafted set of 86 suffix-replacement rules. They associate a grammatical category to each of their suffix-replacement rules. These suffix-replacement

rules are applied to an inflected word and the resulting candidate base-form is checked against a list of base-forms (BFList) obtained from Hindi WordNet[2]. If it is found, it is deemed to be the correct base-form. On the contrary, if a base-form is not found, the system does not suggest any base-form for such a word. They report 100% precision and 0.5481 recall.

5.3 Our Approach

In this section, we first introduces a morpher program that provides a flexible way of applying suffix-replacement rules. In order to check accuracy of the ruleset presented by Shrivastava et al. (2005), we converted their ruleset into the format that is required by this morpher program. We give details of our experiment with their ruleset. Finally, we provide details of our rule-based morphological analyser for the Hindi language.

5.3.1 Application of Suffix-Replacement Rules

Most rule based morphers take a list of suffix-replacement rules where each rule has a suffix and a replacement specified in it. If they come across a word that ends with a suffix specified in one of the rules, the suffix is replaced with a relevant replacement specified in the rule. Some programs also return part-of-speech tags for the words they process.

We have developed a morpher program that given a list of rules and a word to process, returns its base-form, suffix and its part-of-speech category. For debugging purposes, the program also returns the index of the rule that is applied to the word. If the word's grammatical category is also provided, the program only uses the rules for the provided grammatical category. If no such information is provided, the morpher program returns one result for each grammatical category that the word can be associated with. For example, if the word is *exercising*, it could be categorized as a noun or a verb depending on how it is used in the sentence. In such cases, the program returns two answers:

1. base-form:exercise, suffix:ing, pos:verb, rule:1

[2] `http://www.cfilt.iitb.ac.in/wordnet/webhwn/`

2. base-form:exercising, suffix:, pos:noun, rule:15

Figure 5.1 shows an example of a rule file and explains each element of it in detail.

```
defineVars
A ==> [-a-z0-9]
V ==> [aeiou]
EDING ==> "ed"
ESEDING ==> "es" OR "ed" OR "ing"

defineRules
#comment here
<*>"aches" ==> irreg_stem("ache","s")
<verb>"ach"{EDING} ==> stem(1,"","s")
...
...
<noun>("collie")"s" ==> stem(1,"","s")
<noun>("apologetics" OR "goings" OR "outdoors") ==> null_stem()
...
...
<*>{A}+"uses" ==> stem(2,"","s")
<verb>{A}+"used" ==> stem(2,"","ed")
<verb>{A}+"using" ==> stem(3,"","ing")
...
...
```

Figure 5.1: Example of a Suffix-Replacement Rule File

As can be seen in the example, the rule file has two sections: *defineVars* and *defineRules*. Users can define various types of variables under the section *defineVars*. An advantage of defining such variables is that they can be used in place of the actual values. There are three

types of variables:

1. Range variables.

   e.g. *A ==> [-a-z0-9]*

   This means that the variable *A* can have any character between *0* to *9* and *a* to *z* as its value. The - (*hyphen*) at the beginning of the right-hand-side indicates that A is a range variable.

2. Set variables.

   e.g. *V ==> [aeiou]*

   This means that the variable *V* can have one of the characters from *a, e, i, o* and *u* as its value.

3. String variables.

   e.g. *EDING ==> "ed" OR "es"*

   This means that the variable *EDING* can have one of the two string values *ed* or *es*. In this case, the *OR* keyword behaves likes a logical *OR* operator.

All the suffix-replacement rules are declared under the section *defineRules*. Every rule has two parts: a left-hand-side (LHS) and a Right Hand Side (RHS). The LHS part specifies a regular expression and the RHS part specifies an action that should be taken when the input word matches the LHS regular expression. The LHS and RHS are delimited by `==>`. The LHS part of a rule has the following syntax:

`<grammatical-category><regular-expression>`

For example,

```
<verb>"ach"{EDING}
<verb>{A}+"used"
```

The string *verb* in both the rules indicates that the rules should be applied only on words with grammatical category *verb*. In some cases, such as when the information on grammatical categories for individual words is not available, it could also mean that if a word matches any of these two rules, it is likely[3] to be a verb. The symbol * within the first part of the LHS is used to indicate that rule applies to all word classes. The second part of the LHS is a regular expression[4]. Variables declared earlier can be used here. The rule syntax also supports the Kleene operators + and *. The former indicates one or more occurrences of the characters specified in the variable whereas the latter indicates zero or more occurrences. Variables are enclosed with { *and* }.

The RHS part of a rule specifies a function that should be called when a word matches the LHS conditions. Users can specify one of the following predefined functions:

```
stem(n, replacement, suffix)
```

Here, *n* is number of characters to be deleted from the end of the input word, *replacement* is a replacement string for the deleted number of characters and *suffix* is a string that is returned as a suffix for this word.

```
irreg_stem(base-form, suffix)
```

Here, *base-form* is a base-form of the input word, *suffix* is a string that is returned as a suffix for the input word.

```
null_stem()
```

Here, no suffix is found for the input word.

[3]This assumption is not true for all words and is explained later in the following sections

[4]Please note that it is not a Java regular expression

Regular expressions specified on the LHS of rules are converted into a FSM (finite-state-machine) representation. The compiled FSM is only used for looking up words in the document that match the regular expressions defined on the LHS of rules. Each final state is annotated with one or more rules declared on the RHS of rules.

Given a word to process, the morpher program tries to locate a path from the initial state to a final state that matches the input string. If there's no match, it is assumed that the word is in its root form. However, if there is a match found, all the associated RHS rules are obtained and sorted in top-to-bottom order. If the user has provided a grammatical category, only the rules with the same grammatical category and with the * symbol are executed. If a grammatical category is not provided, all the rules are applied and the result of the first matching rule in each grammatical category is returned as the overall result.

The morpher program is fully customizable in such a way that one can change these rules without affecting the application logic. One can also customize it to use for a different language by providing a ruleset for that language.

5.3.2 Evaluation of Shrivastava et al. (2005)

In order to check the accuracy of the ruleset presented by Shrivastava et al. (2005), we converted their ruleset into the format that is expected by the morpher program explained in the previous section. We obtained a list of unique words from the EMILLE corpus which did not exist in the list of base-forms (BFList) obtained from the Hindi WordNet. Assuming that these are inflected words, we randomly picked 2500 words from the list for our evaluation (we call this set of words DATASET1). The morpher program was used to obtain a base-form for each word in DATASET1. We obtained 0.74 (precision), 0.68 (recall) and 0.71 (F-Measure).

Our analysis of the results revealed that the system could not find any applicable rule for 644 words. Secondly, the DATASET1 includes some words which are not exactly the proper Hindi words. For example, ऑथोरिटियाँ (*authoritiyaan, authorities*). In this case, the word is

composed from an English word *authority* and a Hindi suffix याँ (*yaan*). Finally, there were some inflected words which could not be matched with any of the rules in their ruleset. For example, a rule that converts the word जाइए (*jaaiye, go*) into जाना (*jaanaa, to go*), in which the suffix इए (*iye, a suffix to add the tone of a request*) should be replaced with ना (*naa*) does not exist in their ruleset. Here we list few more examples for which we could not find any correct root forms because there are no appropriate rules available in the ruleset.

- लीजिए (*lIjie, please take*) = लेना (*lenaa*)
- कीजिए (*kIjie, please do*) = करना (*to do*)
- लाओ (*lAO, give me*) = लेना (*to take*)
- देके (*deke, after giving*) = देना (*to give*)
- सुनकर (*sunkar, after listening*) = सुनना (*to listen*)
- लोरीयां (*lOrIyA.n, lullabies*) = लोरी (*lullaby*)

5.3.3 A Ruleset for the Hindi Language

The approach presented in this section is a novel combination of various methods described in the Related Work section above (Pandey and Siddiqui, 2008; Hull, 1996; Majumder et al., 2007; Shrivastava et al., 2005). In Hindi, most base-form verbs end with the suffix ना (*naa*). Every noun in Hindi can be categorized as a masculine or a feminine noun. While, most masculine nouns end with the vowel ा (*aa*), most feminine nouns end with the vowel ी (*ee*). Usually when using these words in sentences, these base-form suffixes are removed and the words are inflected based on varying criteria. Thus, finding a base-form for a given Hindi word involves removing inflections and attaching relevant base-form suffixes.

Discovering Base-forms

As mentioned above, the BFList is not a complete list of base-forms. Below, we describe the two methods we followed to improve the list.

First, we used a Hindi dictionary and added all the missing base-forms in the BFList.

Second, as all the entries in the BFList are in their base-forms, we used the BFList to obtain common base-form endings and used these common base-form endings to obtain candidate words from the corpus that can be base-forms. In the BFList, words are divided into four different grammatical categories: noun, verb, adjective and adverb. For the words in each category, we obtained a list of common suffixes (see Table 5.1). We used this list to identify missing words which appear in the corpus but not in the BFList. Considering one suffix at a time, we obtained words from the EMILLE corpus that end with this suffix. Each of these words was automatically checked against the BFList. If it did not exist, it was manually checked and added to the BFList. The words which were added to the BFList were excluded from the candidate list for the next suffix. This resulted in a discovery of 442 new base-forms (mostly verbs and adjectives). Although, this is a time consuming process, it certainly helps in reducing the number of candidates as compared to checking every word which does not occur in the dictionary. One can also consider to add words that end with suffixes with very high frequency. For example, the suffix ना (*naa*) appers as the suffix for 99.71% of verbs. Also, it is not a frequent suffix for words in any other category. Therefore if a word ends with suffix ना (*naa*), it will be safe to include it as a verb in a dictionary without any human verification.

Table 5.1: Most Frequent Suffixes for Hindi Base-forms

Verbs	%*	**Nouns**	%*	**Adjectives**	%*	**Adverbs**	%*
ा(*aa*)	99.81	ा(*aa*)	19.99	त (*ta*)	17.06	र (*ra*)	14.29
ना (*naa*)	99.71	ी(*ee*)	16.62	ी(*ee*)	14.89	ा(*aa*)	13.21
ाना (*aanaa*)	44.21	र (*ra*)	9.15	ा(*aa*)	10.76	:	11.52
वाना (*vaanaa*)	12.19	न (*na*)	8.06	य (*ya*)	9.20	त :(*taha*)	10.09
कना (*kanaa*)	8.03	-	-	क (*ka*)	7.87	क (*ka*)	8.13
राना (*raanaa*)	7.98	-	-	ति (*ita*)	8.74	त (*ta*)	6.43
लना (*lanaa*)	5.03	-	-	न (*na*)	6.50	-	-
-	-	-	-	रि (*ira*)	6.90	-	-

∗ percentage of number of words that have the listed suffix in that category.

The extended BFList has 2118 verbs, 48563 nouns, 16173 adjectives and 1151 adverbs.

Suffix-Replacement Rules

We identify different suffix-replacement rules for each of the four grammatical categories using the following procedure. We obtain all unique words from the EMILLE corpus which do not exist in the BFList. Assuming that these words are inflected, for each word in this list, we remove one character from the end of the string and replace it with the first suffix in Table 5.1. If the resulting word cannot be found in the BFList, we use the next suffix. This procedure is repeated until all the replacements are tried. If the word still has not been located, another character from the end of the string is removed and the entire procedure is repeated. This continues until only one character is left. We replace the removed characters with the base-form suffixes and check if the resulting word is found in the BFList. If the base-form suffix belongs to the noun category, resulting words are only checked against nouns in the BFList. If the resulting word exists in the BFList, a rule is formed using the deleted characters from the original string as suffix, and the appended characters as replacement. Also, both the original word (inflected word) and the base-form as identified by the rule are recorded for rule verification (if needed) at a later stage. Table 5.2 lists the top ten high frequency rules obtained from our experiment.

Table 5.2: High Frequency Suffix-Replacement Rules for Hindi

Suffix	Replacement	Count	Example	Category
◌े (*e*)	◌ा (*aa*)	1561	लडके (*ladake, boys*) - लडका (*ladakaa, a boy*)	noun
◌े (*e*)	◌ा (*aa*)	1561	रोने (*rone, for crying*) - रोना (*ronaa, to cry*)	verb
◌ो (*o*)	◌ी (*ee*)	492	नौकरों (*naukaron, servants*) - नौकरी (*nakaree, a job*)	noun
◌े (*e*)	◌ी(*ee*)	483	कटोरे (*katore, bowls*) - कटोरी (*katoee, a bow*)	noun
◌ों (*on*)	-	476	परिवारों (*parivaaron, families*) - परिवार (*parivaar, a family*)	noun
-	ना (*naa*)	474	फिसल (*phisal, to slip*) - फिसलना (*phisalanaa, to slip*)	verb
◌ों (*on*)	◌ा	466	नमूनों (*namoonon, samples*) - नमूना (*namoonaa, a sample*)	noun
ते (*te*)	ना (*naa*)	461	जाते (*jaate, while going*) - जाना (*jaanaa, to go*)	verb
ियों (*iyon*)	◌ी(*ee*)	432	नदियों (*nadiyon, rivers*) - नदी (*nadi, a river*)	noun
ता (*taa*)	ना (*naa*)	371	देता (*dete, while giving*) - देना (*denaa, to give*)	verb

As expected, not all rules produce correct results. For example the third rule converts a masculine plural (नौकरों (*naukaron, servants*)) into a feminine singular (नौकरी (*naukaree, a job*)). On the other hand, according to one of the rules if a word ends with ◌ाकर (*aakar*),

the suffix is replaced with ी (*ee*). Given a word कमाकर (*kamaakar, after earning*), it gives a base-form कमी (*kamee, lacking*), which, in itself is a valid base-form but not a correct base-form for the word in the question.

We found (in our dataset) that an inflected Hindi word can have multiple possible suffixes of varying lengths. In other words, there can be different base forms which if suffixed with suffixes of different lengths can result into a same inflected word. For example, the Hindi inflected word करवायेंगे (*karvAyenge, will get it done*) has two possible suffixes that can be removed. When the suffix येंगे (*yenge*) is removed and replaced with ना (*nA*), it produces a valid base form करवाना (*karvAnA, to get it done*). Also when a longer suffix वायेंगे (*vAyenge*) is removed and replaced with the word ना (*nA*), it too produces a valid base form करना (*karanA, to do*). In our dataset, we found 41.532% inflected words with only one suffix, 40.084% with two suffixes, 15.017% with three suffixes, 2.902% with four suffixes, 0.449% with five suffixes, 0.013% with six suffixes and 0.003% words with seven suffixes.

The order in which these rules are applied is also very important. For example, consider the third and the fifth rules. Given a word नौंकरों (*naukaron, servants*), if the third rule is applied first, the system returns an incorrect base-form नौंकरी (*naukaree, job*). However, if the fifth rule is applied first, the system would return a correct base-form नौंकर (*naukar, servant*). Therefore, we order the rules in the following order:

1. Given two rules, the rule with the longer suffix is executed first. This is to make sure that more specific rules are given priority over generalized rules.

2. If they have suffixes of the same length the rule with the higher frequency in our list (Table 5.1) is executed first.

3. If they have suffixes of the same length and their frequencies (Table 5.1) are same, the frequencies of the rules are looked up (Table 5.2) and the rule with higher frequency is executed first.

4. Finally, if the frequencies of replacements are the same, the one with the smallest

replacement string is executed first. This makes sure that the minimum change is applied to the word.

Since every resulting base-form is validated using the BFList, this approach restricts us from processing inflected strings whose base-forms do not exist in the BFList. For such words, we use the output of the most likely rule (i.e. the first rule that matched the inflected word). If no rule is applicable, the word itself is considered a base-form.

Finalizing the Ruleset

In order to filter the ruleset, we verified it against a set of 2500 words (DATASET2). These words were randomly obtained from the EMILLE corpus and had no word in common with DATASET1. For each word in this set we obtained its base-form[5]. We started with an empty ruleset. One rule at a time from the Table 5.2 (in order of top-to-bottom) was added to the ruleset and its performance on the DATASET2 (F-Measure) was recorded. If the addition of a rule resulted in a lower score, the rule was removed from the ruleset. If it did not bring any change[6], the user was prompted for a decision to include or exclude the rule from the ruleset. Along with the rule, a list of words (column 4 in Table 5.2) as collected earlier for this rule was shown to the user.

Although, one could use DATASET2, in the first place, to learn such rules, it is important to mention that the size of such a dataset is not very big but to manually prepare larger one could be very expensive. On the other hand, obtaining a true distribution of suffixes just based on a small dataset could lead to incorrect results. However, the rules, as shown here, are based on automatically collected high-frequency suffixes and replacements from the corpus. In order to evaluate the final ruleset, we applied it to DATASET1 and obtained 0.78 (precision), 0.61 (recall) and 0.69 (F-Measure).

Although the results are lower than that obtained with Shrivastava's ruleset, i.e. 0.74 (precision), 0.68 (recall) and 0.71 (F-Measure), it is important to mention that their ruleset was

[5]We used high-frequency rules (see Table 5.2) to obtain base-forms which were manually verified.

[6]This happens when there is no word in the dataset to which the rule can be applied.

derived manually whereas our ruleset consists of rules where the majority of them are learnt automatically without prior knowledge of the language. Even when the user is asked to verify a rule, he is shown a set of words (column 4 in Table 5.2) which are specific to the rule and were collected during the discovery phase of these rules. Such a list of words can help users to foresee the outcome of including this rule into the ruleset.

Extending the Ruleset of Shrivastava et al. (2005)

We merged our ruleset with Shrivastava's ruleset (referred to as the *extended ruleset* below) and applied it to DATASET1. Table 5.4 shows the results of our experiments.

5.3.4 Derivational Morpher

We also experimented with a derivational process whereby the most common prefixes were obtained from the BFList. Given a word from the BFList, characters from the start of the inflected word were removed one by one until the remaining string was found as another individual word in the BFList (see Table 5.3).

Table 5.3: Most Frequent Prefixes for Hindi Base-forms

Prefix	**Count**	**Example**	**Decomposition**
अ (*a*)	1616	अपरिचित (*aparichit, unknown*)	अ (*a*) + परिचित (*parichit, known*)
वि (*vi*)	307	विदेशी (*videshee, foreign*)	वि (*vi*) + देशी (*deshee, local*)
अन् (*an*)	239	अनावश्यक (*anaavashyak, unnecessary*)	अन् (*an*) + आवश्यक (*aavashyak, necessary*)
सु	197	सुपात्र (good character)	सु + पात्र (character)

For example, consider the inflected word असुवीधाऐं (*asuveedhaein, lack of facilities*). Based on the rules derived earlier, it is possible to obtain a candidate base-form असुविधा (*asuvidhaa, lack of facility*); however, if this form is not present in the BFList, we cannot verify if it is a correct linguistic root. By looking in the prefix list and removing the prefix अ (*a*), the remaining word सुविधा (*suvidhaa, facility*) is found in the BFList. In such cases, the original resulting base-form (i.e. असुविधा (*asuvidhaa, lack of facility*)) is considered a valid base-form.

Table 5.4: Results of Experiments on the Hindi Language

BFList	**Shrivastava et al. (2005)**						**Extended Ruleset**					
	Non-derivational			**Derivational**			**Non-derivational**			**Derivational**		
	P	R	F	P	R	F	P	R	F	P	R	F
Wordnet	0.736	0.683	0.708	0.735	0.683	0.708	0.743	0.717	0.730	0.743	0.717	0.730
Extended	0.817	0.768	0.792	0.817	0.768	0.792	**0.821**	**0.803**	**0.812**	0.820	0.803	0.812

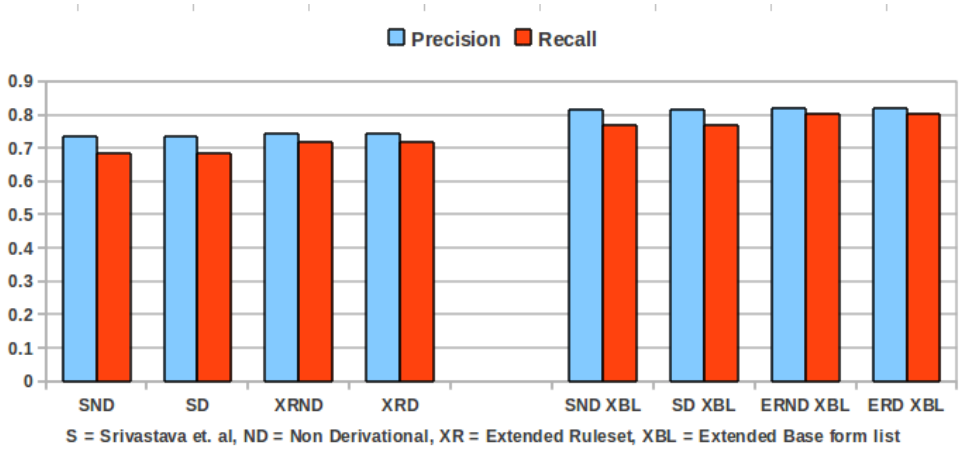


Figure 5.2: Results of Experiments on the Hindi Language

5.3.5 Results

As can be seen in Table 5.4 and Figure 5.2 we compare the results of Shrivastava's ruleset with the extended ruleset. It also shows the effect of extending the BFList by adding more words (the second row). The table also shows the results that were obtained after the addition of the derivational process. While the derivational process does not appear to make much difference this may be because very few pairs appear in the dataset to test the process.

5.4 Chapter Remarks

In this chapter, we first described a program that facilitates execution of suffix-replacement rules to identify a base-form and a suffix for a given word. Then, we presented a rule-based morphological analyser and proposed an approach that takes both prefixes as well as suffixes into account. Given a corpus and a dictionary, our method can be used to obtain a set of suffix-replacement rules for deriving an inflected word's base-form. We used this

method to develop a Hindi morphological analyser. Since the method is simple and fast, our approach can be used for rapid development of morphological analysers and yet it can obtain competitive results with analysers built relying on human authored rules.

Later, in Chapter 8, we show that our approach is portable, at least to related languages, by adapting it to the Gujarati language.

In the next chapter, we present an approach to measure the transliteration similarity of English-Hindi word pairs.

Chapter 6
English-Hindi Transliteration Using Multiple Similarity Metrics

6.1 Outline of the Chapter

When translating a sentence from a source language into a target language, Named Entities (NE) such as person names, names of places, organizations, etc. are not translated but transcribed into the writing system of the target language.

Transliteration is defined as the task of transcribing a word or text from one writing system into the another writing system such that the pronunciation of the word remains the same and a person reading the transcribed word can read it in his language. As described by Lin and Chen (2002), transliteration can be classified into two directions. Given a pair of source and target word, *forward transliteration* is the process of phonetically converting source word into target word whereas *backward transliteration* is the process of correctly finding or generating source word given target word. Forward transliteration can be achieved simply by looking up table of character mapping between the two languages. However, backward transliteration is said to be more challenging as it needs to disambiguate the noise produced in the forward transliteration and estimate the original word as close as possible (Knight and Graehl, 1998).

In this chapter, we present an approach to measure the transliteration similarity of English-Hindi word pairs[1]. Our approach has two components. First we propose a bi-directional mapping between one or more characters in the Devanagari script and one or more characters in the Roman script (pronounced as in English). Given a Hindi word, this mapping allows one or more candidate transliterated forms in the Roman script to be obtained.

To choose which of these candidates most closely matches a candidate target word requires a string similarity measure. Therefore, we review some of the well known string similarity metrics and propose an algorithm for computing a similarity measure which also takes into account the constraints needed to match English-Hindi transliterated words.

Finally, we evaluate the performance of these similarity metrics individually and in various

[1]The transliteration similarity approach presented in this chapter has been published in Aswani and Gaizauskas (2010a).

combinations to discover the best combination of similarity metrics and a threshold value that can be used to maintain the optimal balance between accuracy and coverage. To test the portability of our approach to other similar languages we adapt our system to the Gujarati language in Chapter 8.

The transliteration similarity approach presented in this chapter is used in our English-Hindi word alignment system for aligning words such as proper names and cognates (words in different languages sharing a common derivation).

6.2 Related Work

Sinha and Thakur (2005) discuss the mixed usage of the English and Hindi languages. They provide various examples of mixed usage of the two languages and present an MT system that is capable of dealing with text written in such a mixed code language. They show that although there are certain constraints that appear to be observed in the usage of the Hinglish language, people do not follow them strictly. Giving more details on the same, they explain that there are three types of constraints that are mentioned in the literature: the free morpheme constraint; the closed class constraint; and finally the principle of the dual structure. The morpheme constraint means that the words from one language cannot be inflected according to the grammar rules of the other language. The principle of the dual structure means that the internal structure of the English constituent need not conform to the constituent structure rules of the Hindi language provided the placement of the English phrase obeys the rules of the Hindi language. Finally, the closed class constraint means that the words categorized as closed class words, such as possessives, ordinals, determiners, pronouns, etc. are not used from English when the head noun used in the sentence is in Hindi.

They show by examples that out of the three constraints the morpheme constraint does not hold true and there are a large number of English words that are used in Hindi sentences according to the grammar rules of the Hindi language. For example कम्प्युटरों (*computeron,*

computers), where the कम्प्युटर (*computer*) is an English noun and ों (*on*) is used for indicating more than one computer). Similarly in बारातीस (*baaraatees, marriage guests*), बाराती (*baaraatee, a marriage guest*) is an English noun and ीस (*ees*) is the English plural marker.

The latter example shows that although transliteration similarity (TS) approaches can match the initial parts of these words, special mappings are needed to match the suffixes (such as ों (*on*) in कम्प्युटरों (*computeron, computers*) and ीस (*ees*) in बारातीस (*baaraatees, marriage guests*)). They also discuss the situation whereby their system has to deal with sentences in which the Devanagari script is used for writing English words.

TS approaches can be very helpful in identifying named entities and cognates. Kondrak et al. (2003) show that identifying cognates not only helps in improving results in word alignment but also can be very useful when machine-readable bilingual dictionaries are not available. To locate cognates in parallel texts, they experimented with three similarity metrics: Simards condition, Dices coefficient and LCSR. They create a file with all possible one-to-one translation pairs from each aligned sentence pair and calculate similarity between each pair. The pairs above a certain threshold are then considered as possible cognates and aligned with each other. They report 10% reduction in the error-rate as a result of injecting cognate pairs into their alignment system.

One approach to identify named entities (NEs) is to use precompiled lists of named entities (Cunningham et al., 2002). However, precompiled lists may not work well on unseen new documents and therefore locating named entities needs more than just using precompiled lists. Huang et al. (2003) suggest that equivalent NE pairs in bilingual texts can be found by way of surface string transliteration. Similarly Bikel et al. (1997) explain that it is possible to detect target language named entities by projecting source language named entities cross-lingually if their phonetic similarity or transliteration cost are obtained.

Another use of TS is explained by Balajapally et al. (2008). They describe a book reader tool that allows people to read books in different languages. In order to achieve this, they use sentence, phrase, word-to-word and phonetic dictionaries. In cases where they cannot find

a match in any of the sentence, phrase or word-to-word dictionaries they use the phonetic dictionary to obtain a phonetic transliteration. Their transliteration system (known as OM), given words in one language, provides their equivalent transliterations in a language that the user has requested to read the book in. The OM transliteration system gives a unified presentation for Indian languages. OM exploits the commonality of the alphabet of Indian languages and therefore the representation of a letter is the same across the many of the languages. Since the phonetic mappings are based on the sound each letter produces, if their search fails to locate a mapping for a specific character, they consider another character that sounds similar to the original character.

Pouliquen et al. (2005) highlight various approaches that have been employed by researchers to recognise NEs in the text (Kumar and Bhattacharyya, 2006). These approaches include a lookup procedure in a list of known names, analysis of local lexical contexts, use of a well known word which is part of the named entity and part of speech tags which suggest that the words might form a NE. They mention that the existing transliteration systems either use hand-crafted linguistic rules, or they use machine learning methods, or a combination of both. Similar to Kumar and Bhattacharyya (2006), they collect trigger words from various open source systems and write simple local patterns in PERL that recognise names in the text. Once they have obtained these data, they analyse the words in the left and right contexts of found NEs and collect the frequently occurring words to be used for identifying NEs in unseen data. Before they match strings in two different languages, they perform a normalization process on one of the two words. For this they use a set of approximately 30 substitution rules such as replacing accented characters with non-accented equivalents, double consonants with single consonant, *wl* (at the beginning of the word) with *vl*, *ph* with *f*, and so on. All possible strings obtained as a result of this process are then compared with the source string and if any of them has a similarity above a specified threshold, it is considered a possible match. To calculate a similarity score, they use three different similarity measures and the average of the three is used as a similarity score. These measures are based on letter n-gram similarity, where the first two measures are the cosine of bigrams and trigrams

and the third measure is the cosine of bigrams with no vowels in the text. In the following section, we give details of some of the popular string similarity metrics and how they are calculated.

6.3 String Similarity Metrics

In this section we look at some of the various methods that have been employed by researchers to compare strings. These include methods such as Dice's Coefficient, Matching Coefficient, Overlap Coefficient, Levenshtein distance (Levenshtein, 1966), Longest Common Subsequence Ratio (LCSR), Soundex distance metric, Jaro-Winkler metric (Jaro; Winkler, 1999) and n-gram metric. There are several variants of these methods or combinations of variants of these methods that are mentioned in the literature. For example, the similarity metric used in the Pouliquen et al. (2005) is an example of a combinations of three variants of the n-gram metric.

The matching coefficient is the simplest measure of all, where only the count of characters that match is taken as the similarity measure. The higher the score, the more similar the strings. However, this approach is typically used for same length strings. An immediate variant of the matching coefficient is Dice's coefficient. It allows variable length strings to be compared. Similarity is defined as twice the number of matching characters divided by the total number of characters in the two strings. Another variant of the matching coefficient is the overlap coefficient where the similarity is calculated as the number of identical characters in the two strings divided by the length of the shorter of the two strings. It is based on the assumption that if a string *s1* is a subset of the string *s2* or the converse then the similarity is a full match.

The Longest Common Subsequence Ratio (LCSR) is an another variant of Dice's coefficient where the ratio of two words is computed by dividing the length of their longest common subsequence by the length of the longer word. For example LCSR *(colour, couleur) = 5/7* as their longest common subsequence is *c-o-l-u-r*. In such approaches, where the number

of matching characters is more important, the position of the characters is not taken into consideration and therefore they can wrongly identify words such as *teacher* and *cheater*.

Gravano et al. (2001) explain an approach which is based on the n-gram similarity metric. For example while comparing the two strings *teacher* and *cheater*, a window of 2 characters can be considered and all possible bigrams can be collected for the two strings. For example, *te, ea, ac, ch, he, er* and *ch, he, ea, at, te, er*. In this case the five bigrams *te, ea, ch*, and *he* and *er* are found to be identical giving result of 2*5 / 12 = 0.83. Even though the strings are different, because they use same characters, the similarity figure is high. One can change the windows size to higher values. For example by changing the window size to 3, we get a similarity of 0.1 only. Experiments carried out by Natrajan et al. (1997) on a Hindi song database show that the window size of 3 is the optimum value for the n-gram algorithm. In their experiments, users submitted a query in Romanized Hindi script which was then matched with the Hindi database.

The basic Levenshtein edit distance algorithm was introduced by Levenshtein (1966). It is used for calculating the minimum cost of transforming one string into the other. The cost of deleting one character, inserting a new one, or cost of substituting one character for another is 1. For each character, the operation with minimum cost is considered among all other possibilities. The advantage of this method is that it also takes into account the positions of characters and returns the minimum cost that is required to change one string into the other.

Jaro-Winkler metric (Jaro; Winkler, 1999) is another measure of similarity between two strings. The metric is more suitable for short strings. The score is normalized such that 0 means no similarity and 1 means the equal strings. Given two strings *s1* and *s2*, their distance is calculated as *d(s1,s2) = 1/3 (m / |s1| + m / |s2| + (m - t)/m)*. Here, m is the number of characters that are common in two strings. To be considered as a common character, a character at position i in the string *s1* has to be within the H window of the equivalent j^{th} character in the string *s2*. Here $H = max(|s1|, |s2|)/2 - 1$. t is equal to the

number of characters matched from window H but not at the same index divided by 2.

Soundex is an algorithm that groups consonants according to their sound similarity. It is a phonetic algorithm which is used for indexing names by sound as pronounced in English. The basic idea here is to encode words that are pronounced in a similar way with the same code. Each word is given a code that consists of a letter and three numbers between 0 and 6, e.g. *Aswani* is *A215*. The first step in the algorithm is to preserve the first letter of the word and remove all the vowels and consonants (*h, w* and *y*) unless they appear at the beginning of the word. Also consecutive letters that belong to the same group are removed except the first letter. Letters *B, F, P* and *V* belong to group 1, *C, G, J, K, Q, S, X* and *Z* to group 2, *D* and *T* to group 3, *L* to group 4, *M* and *N* to group 5 and the letter *R* belongs to group 6. In a standard Soundex algorithm, only the first letter and three following numbers are used for indexing. If there are less than three, the remaining places are filled with zeros and otherwise only the first three numbers are considered. There are several variations of the Soundex algorithm. Although it is very helpful for fuzzy searching, there are certain limitations to the algorithm such as the high number of false positives due to its reliance on consonant grouping and inaccurate handling of words that start with silent letters.

6.4 Our Approach

6.4.1 Generating Transliterations

Figure 6.1 lists letter correspondences between the writing systems of the two languages where one or more Hindi characters are associated with one or more English characters. For example फ (*ph*) can be *f*, or *ph* (e.g. frame, photo) and क (*ka*) can be *c, k or ch* (e.g. cancer, kite, character). This transliteration mapping (TM) was derived manually and provides a two way lookup facility.

We also provide mappings for vowels. Vowels play a very important role in deciding how to pronounce a word. For example consider *Suman* and *Simon*. In this example, even though the words share the same consonants they are pronounced differently. Algorithms that do not

take vowels into account would incorrectly align these words. We also add special mapping for ों (*on*) and ें (*en*) = s/es). These additional pairs help in aligning words such as कम्प्युटरों (*computeron, computers*) with *computers* and साइज़ें (*saaizein, sizes*) with *sizes*.

The following illustration explains how to use the TM to obtain possible transliterations for the Hindi word केंसर (*kensar)* which means *cancer* in English. For the Hindi letter at the i^{th} position in the Hindi word HW (where $i = 1..n$ and $n = |HW|$ (i.e. the length of HW)), we define a set TS_i that contains all possible phonetic mappings for that letter.

Hindi	TT	Hindi	TT	Hindi	TT	Hindi	TT	Hindi	TT
ँ , ं	n	च	ch	म	M	०	0	जे	j
ः	h	छ	chh	य, य़	y	१	1	के	k
अ	a	ज, ज़	j, z	र, ऱ	r, rr	२	2	एल	l
आ	a, aa	झ	z	ल, ऌ, ळ, ऴ,	l	३	3	एम	m
इ	e,i	ञ	ny	व	v	४	4	एन	n
ई	ee, ii	ट	t, tt	श	sh, ss, tio	५	5	ओ	o
उ	u	ठ	th	ष	sh, ss	६	6	पी	p
ऊ	u,oo	ड, ड़	d	स	sh, ss, s, c	७	7	क्यु	q
ऋ, ॠ	ru, roo	ढ, ढ़	dh	ह	h	८	8	आर	r
ऌ	l	ण	n	ा	a, aa	९	9	एस	s
ऍ, ऐ, ए, ऐ	a, e, ae, aei	त	t	ि	i, e	ए	a	टी	t
ॐ	om, aum, oum	थ	th	ी	ee, y, i	बी	b	यु	u
ऑ, ओ, ओ	o	द	d	ु	u, ue	सी	c	वी	v
औ	ou, oau, au	ध	dh, ss, sh, tio	ू	u, oo	डी	d	डबल्यु	w
क, क़	k, c, ch	न, ऩ	n	े , ॅ , ॆ	e, ae	इ	e	एक्स	x
ख, ख़	kh	प	p	ै	ai, ei, a	एफ	f	वाय	y
ग, ग़	g	फ, फ़	f, ph	ो	o	जी	g	झे ड	z
घ	gh	ब	b	ौ	ou, au, oau	एच	h	ोंं	s, es
ङ	ng	भ	bh	ृ	ru, r	आई	i	े ं	s, es

Figure 6.1: English-Hindi Transliteration Mappings

In order to optimize the process, we remove from the TS_i all mapped characters that do not exist in the candidate target string. Below, we list mappings for the letters of the word केंसर (*kensar*). The mappings which need to be removed from the TS_i are enclosed in round brackets: क = *[c, (k), (ch)]*; ें = *[e, a, (ai)]*; ं = *[n]*; स = *[c,(s)]*; र = *[r]*. From these

mappings we define a set TS of n-tuples such that $TS = TS_1 \times TS_2 \times ... \times TS_n$ (i.e. TS is the Cartesian product of all the previously defined sets ($TS_{i=1..n}$) for each letter in the Hindi word). Each n-tuple in TS is one possible transliteration of the original Hindi word. In total there are $|TS|$ transliterated strings. In the above example the value of $|TS|$ is 2 (1 x 2 x 1 x 1 x 1) (i.e. *Cencr* and *Cancr*). Each transliterated string ($S_{j=1..|TS|} \in TS$) is compared with the English word using one of the string similarity metrics (explained in the next subsection). If the English word and any of the transliterated strings has a similarity score above a specified threshold, the strings are deemed to be transliterations.

Sometimes a word has a letter for which there is no mapping available. It is also possible that none of the mappings for a given letter appear in the English word. For example, lets compare the English word *how* with the Hindi word होमवर्क (*homework*). The Table 6.1 lists the mappings for each letter in the Hindi word.

Table 6.1: Transliteration Mappings for the Letters of Hindi Word होमवर्क (*homework*)

Hindi Letter	**Mapped English Letters**
ह (*ha*)	h
ो(*o*)	o
म (*ma*)	m
व (*va*)	v and w
र् (*r*)	r
क (*ka*)	c, k and ch

As can be seen, mappings for the Hindi letters म (*ma*), र् (*r*) and क (*ka*) do not appear in the English word *how*. Also the mapping *v* (for the letter व (*va*)) does not appear in the English word. If these mappings are excluded, there will be only one candidate transliterated string *how*. Comparing it with the English word gives 100% matching. If there are no mappings available for a given Hindi letter, we use an underscore (_) to indicate a missing mapping. Thus, the candidate word *how* will become *ho_w__* and will be compared with the English word *how*. Since the two strings are now very different, they do not match.

6.4.2 String Similarity Metrics

In the case of English-Hindi strings it was observed during our experiments that for the two strings to be transliterations the first and last characters from both the strings – the English word (E) and the transliterated string (T), must match. This ensures that the words have same phonetic starting and same ending. However some English words start or end with silent letters (e.g. *p* in *psychology* and *e* in *programme*). Therefore in such cases the first character of the transliterated string should be compared with second character of E and similarly the last character of the transliterated string should be compared with the second last character of E. Our experiments also show that unless the length of the shorter string is at least 65% of the length of the other string, they are unlikely to be phonetically similar.

The similarity algorithm (see Table 6.2) takes two strings, S and T, as input where $S_{i=1..n}$ and $T_{j=1..m}$ refer to characters at position i and position j in the two strings with lengths n and m respectively. Starting with $i = 1$ and $j = 1$, character S_i is compared with characters T_j, T_{j+1} and T_{j+2}. If S_i matches with one of T_j, T_{j+1} and T_{j+2}, the pointer i advances one position and the pointer j is set to one position after the letter that matches with S_i. If there is no match, the pointer i advances and j does not. We award every match a score of 2 and calculate similarity using the formula *TSM = matchScore/(f(s) + f(t))* where f(x) = number of letters in the string x.

6.4.3 Experiments and Results

We compare our similarity metric TSM with other string similarity metrics such as the standard Dice's coefficient (DC), LCSR, Jaro Winkler (JW), n-gram and Levenshtein distance (LD) metrics. In order to perform this comparison we manually obtained 1000 unique words pairs from the EMILLE corpus. Out of the 1000 words pairs collected, 732 pairs were correct transliterations of each other and 268 pairs were not[2].

[2]We used an ealier version of the character mapping (Aswani and Gaizauskas, 2005b) to locate named entities in parallel sentences. The collected entities were reviewed manually and some of the incorrect entities were chosen as the negative examples for this experiment.

Table 6.2: Similarity Algorithm

```
READ E, T //source and target strings
SET i=1, j=1, n=|E|, m=|T|, matches=0 //initialize variables

// if the shorter string is at least 65% of the length of the longer string
IF (|S|/|T|) >= 0.65 || (|S|/|T|) >= 0.65 THEN

  // check start and end constraints
  IF (S[1] == T[1] || S[2] == T[1] || S[1] == T[2]) &
     (S[n] == T[m] || S[n-1] == T[m] || S[n] == T[m-1]) THEN

     WHILE i <= n & j <= m  //comparing characters one by one
       FOR k = j to j+2
         IF S[i] == T[k] THEN
           INCREMENT matches, i
           SET j to k + 1
           CONTINUE WHILE
         ENDIF
       ENDFOR
       INCREMENT i //character at position i in S does not exist in T
     ENDWHILE
  ENDIF
ENDIF
COMPUTE sim = matches*2/(|S|+|T|) //computing similarity
RETURN sim
```

We obtained a set of transliterations (using the transliteration mapping) for each Hindi word in the collected sample data. For each similarity metric the task was to identify correct transliteration pairs and avoid recognizing incorrect pairs by giving them a very low similarity score. The following procedure was repeated for each similarity metric. For each Hindi word in these test pairs we used our character mapping method (see Figure 6.1) to obtain the transliteration with highest score. Then, the transliteration strings were compared with the English word in the pair and the results were clustered in six predefined groups: $>= 0.95$, $>= 0.90$, $>= 0.85$, $>= 0.80$, $>= 0.75$, and $>= 0.70$ where, the group $>= Sim$ contains pairs with similarity greater than or equal to Sim.

Here, the group $>= 0.95$ means that the pairs with similarity greater than or equal to 95%. For each group, we calculated the precision, recall and F-Measure. Precision is calculated as the ratio of the correctly identified pairs divided by the number of pairs identified by the

system. Recall is calculated as the ratio of correctly identified pairs divided by the total number of pairs in the sample data (i.e. 1000). The F-Measure is calculated to obtain the weighted harmonic mean of precision and recall. The weight is equally distributed between recall and precision and therefore the equation used was $F\text{-}Measure = 2 \times P \times R/(P + R)$.

Table 6.3: Experiments with Dice's Coefficient Metric

DC	**precision**	**recall**	**F-Measure**
>= 0.95	0.967	0.263	0.414
>= 0.90	0.932	0.454	0.611
>= 0.85	0.901	0.585	0.710
>= 0.80	0.822	0.732	**0.775**
>= 0.75	0.772	0.732	0.752
>= 0.70	0.732	0.732	0.732

Table 6.4: Experiments with Transliteration Similarity Metric

TSM	**precision**	**recall**	**F-Measure**
>= 0.95	0.994	0.170	0.290
>= 0.90	0.970	0.327	0.489
>= 0.85	0.953	0.511	0.665
>= 0.80	0.921	0.605	0.730
>= 0.75	0.809	0.732	**0.769**
>= 0.70	0.732	0.732	0.732

The best F-Measure performance each metric gave is highlighted in Tables 6.3 to 6.8. For example, in the case of Dice's coefficient, the best result is when the threshold is set to >= 0.80 similarity. Similarly, in case of the TSM, the best result is obtained when the threshold is set to >= 0.75 and so on. For the comparison purposes, we plot all the best scores we obtained by individual metrics in Figure 6.2. Here, one pair of bars (blue for precision and red for recall) is used per similarity metric. On the x axis we print name for the similarity metric followed by the threashold value for above or equals to what the similarity metric gave the best F-Measure score. The black line is used for plotting the

Table 6.5: Experiments with LCSR Metric

LCSR	precision	recall	F-Measure
>= 0.95	0.917	0.321	0.476
>= 0.90	0.917	0.341	0.497
>= 0.85	0.917	0.407	0.564
>= 0.80	0.871	0.479	0.618
>= 0.75	0.861	0.508	0.639
>= 0.70	0.846	0.550	**0.667**

Table 6.6: Experiments with Jaro-Winkler Metric

JW	precision	recall	F-Measure
>= 0.95	0.989	0.184	0.310
>= 0.90	0.953	0.325	0.485
>= 0.85	0.927	0.459	0.614
>= 0.80	0.856	0.575	0.688
>= 0.75	0.829	0.621	0.710
>= 0.70	0.798	0.680	**0.734**

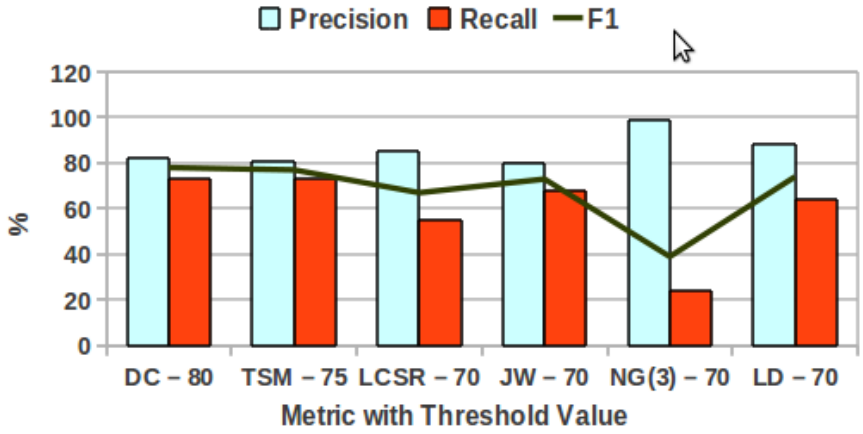


Figure 6.2: Similarity Metrics with Best F-Measure Scores

F-Measure score.

It must be noted that the DC metric does not take positions of characters into account where

Table 6.7: Experiments with N-gram(3) Metric

NG(3)	**precision**	**recall**	**F-Measure**
>= 0.95	0.994	0.153	0.265
>= 0.90	0.994	0.153	0.265
>= 0.85	0.994	0.159	0.274
>= 0.80	0.988	0.171	0.292
>= 0.75	0.990	0.203	0.337
>= 0.70	0.988	0.241	**0.387**

Table 6.8: Experiments with Levenshtein's Edit Distance Metric

LD	**precision**	**recall**	**F-Measure**
>= 0.95	0.994	0.153	0.265
>= 0.90	0.995	0.188	0.316
>= 0.85	0.972	0.317	0.478
>= 0.80	0.925	0.471	0.624
>= 0.75	0.900	0.577	0.703
>= 0.70	0.880	0.636	**0.738**

as the TSM does. Although the F-Measure figures for these two metrics are similar, this is because our dataset does not have examples such as *teacher* vs [*cheater*] which according to the DC is a correct transliteration pair (even with threshold set to 1).

The Levenshtein's distance algorithm does not return a similarity measure but a distance or a cost in number of characters that need to be either replaced, deleted or inserted into the target word. In order to compare it with other similarity metrics, we use the distance of two strings in the following equation to obtain the similarity between the source and the target string. *Sim(s,t) = 1 - (d(s,t)/max(|s|,|t|))* where, $d(s,t)$ is the distance returned by the Levenshtein distance algorithm.

Since the Soundex algorithm is a boolean function that returns only true or false based on the exact match of the two hash codes, we could not cluster the results into different groups

as defined above. The algorithm was able to identify 747 word pairs out of which 86 word pairs were incorrectly identified. Similarly out of 253 word pairs which the algorithm could not identify as transliterated pairs, 71 pairs were transliterated strings in the gold standard.

Multiple Measure Agreement Strategy

English	Hindi	GS	DC	Trans	T=80	TSM	Trans	T=75	JW	Trans	T=70	Combined
trading	ट्राडंग	T	100.00	trading	T	100.00	trading	T	100.00	trading	T	T
benchmarking	बचमाार्किंग	T	100.00	banchmirkeng	T	100.00	benchmarking	T	100.00	benchmarking	T	T
trader	**डायरक्टर**	**F**	**85.71**	**dayrectr**	**T**	**0.00**	**dayrectr**	**F**	**52.22**	**ddayyrectr**	**F**	**F**
cheater	**टाचर**	**F**	**92.31**	**teechr**	**T**	**0.00**	**teechr**	**F**	**52.38**	**tteechr**	**F**	**F**
consumers	कंज्यूमर्ज़	T	82.35	cnsyumrs	T	82.35	cnsyumrs	T	75.66	chnsyumrs	T	T
pilot	पायलट	T	80.00	poylt	T	80.00	piylt	T	73.89	piyltt	T	T
financial	**फायनाश्यल**	**T**	**80.00**	**faaynintiyl**	**T**	**73.68**	**fiynantiyl**	**F**	**58.78**	**fiynantiyl**	**F**	**F**
standards	स्टंटडड्ज़	T	88.89	stantrdds	T	77.78	stantrdds	T	84.26	stantrdds	T	T
licensing	लायसासंग	T	94.74	liycensing	T	94.74	liycensing	T	70.74	liycensing	T	T
low	**लाकल**	**F**	**57.14**	**locl**	**F**	**57.14**	**locl**	**F**	**72.22**	**locl**	**T**	**F**
crore	**क्राडट**	**F**	**54.55**	**credut**	**F**	**54.55**	**credut**	**F**	**70.00**	**credet**	**T**	**F**
trader	**डायरक्टर**	**F**	**85.71**	**dayrectr**	**T**	**0.00**	**dayrectr**	**F**	**52.22**	**ddayyrectr**	**F**	**F**
France	फ्रान्स	T	90.91	franc	T	90.91	franc	T	94.44	franc	T	T
Germany	जमना	T	83.33	grmny	T	83.33	grmny	T	80.95	ggrmony	T	T
Correct Entries					**11**			**12**			**11**	**13**

Figure 6.3: Examples of Classification Using Multiple Measure Agreement Strategy

Given the different criteria that these similarity metrics work on, it is possible that given a pair of strings one metric gives it a very high score whereas the others give it a very low score. For example, consider the pair *Advice* and अधिक (*adhika, more*). They are not related to each other. However, after obtaining various transliterations for the Hindi word अधिक and comparing each of them with the English word *Advice*, the Jaro-Winkler metric gives one of the transliterated strings a very high similarity score (82.22%). If the threshold is set to 80%, the JW metric would identify this pair as correct transliterations. On the other hand, given the same threshold, the TSM and DC metrics would reject this pair as the maximum similarity according to them is 72.72%.

Figure 6.3 shows 14 different examples of English-Hindi word pairs. The first two columns in the figure have English-Hindi word pairs, for which we would like to find out if they should be aligned or not. 3rd column indicates if the words are actually true transliterations of each other (gold standard). 4th column has the similarity score calculated using the DC method. 5th column shows the transliteration candidate for which the DC returned the highest score. 6th column indicates if the words are transliterations of each other or not according to the

DC method. This is if the threshold was set to 80 which was found to be the best threshold for the DC method during its individual assessment. 7th, 8th, and 9th columns are same as 4th, 5th and 6th columns but with values for the TSM metric. Similarly, 10th, 11th and 12th columns are for the JW metric. Finally, the last column shows the majority vote by the three methods on each of the word pairs. The last row indicates the number of pairs that are correctly identified by the individual methods by comparing their individual votes with the gold standard (3rd column). Thus, the number of correctly identified pairs by individual methods are: DC=11, TSM=12, JW=11. There are many pairs on which the three metrics have different opinion but taking the majority vote results in highest number of pairs (13) to be correctly identified.

Thus, in order to exploit a multiple measure agreement strategy where a word pair is a valid transliterated pair only if it receives majority vote from the members of the similarity metrics group, we conducted a further experiment, whereby we recorded the top combination of metrics that performed best (F-Measure) given different threshold values. We found that the combination of DC, TSM and the JW metrics works best with a threshold value set between 0.79 and 0.81.

In order to experiment with these threshold values, we used 2500 English-Hindi sentence pairings (different from the test-data), which were translations of each other. We used a part-of-speech tagger to tag the English words. Considering one sentence pair at a time, we compared each noun word from the source sentence with the every word in the target sentence. We used 0.79 as the similarity threshold and asked the three methods (DC, TSM and JW metrics) to cast their votes on each word pair. If a word pair received at least two positive votes, the pair was considered to be a correct alignment.

As a result of this exercise, the system returned 1078 unique word pairs. We checked these word pairs manually and found that the system had correctly identified 1021 word pairs giving an accuracy of 0.947. This indicates that the method works reasonably well and can be used in real applications. Since the performance of this method depends on the threshold

value, it could be set to higher value should one want to concentrate on higher precision.

Table 6.9: Multiple Measure Agreement Strategy Results

Similarity Metrics	**Threshold**	**F-Measure**
DC + TSM + JW	>= 0.75	0.85
DC + TSM + JW	>= 0.78	0.86
DC + TSM + JW	>= 0.79	**0.92**
DC + TSM + JW	>= 0.80	**0.92**
DC + TSM + JW	>= 0.81	**0.91**
DC + TSM + JW	>= 0.85	0.78
DC + TSM + LCSR	>= 0.90	0.62
DC + LCSR + JW	>= 0.95	0.4

6.5 Chapter Remarks

In this chapter, we presented an approach to measure the transliteration similarity of English-Hindi word pairs. As described in chapter, our approach has two components. First we proposed a bi-directional mapping between one or more characters in the Devanagari script and one or more characters in the Roman script (pronounced as in English). We explained with an example how this allows a given Hindi word written in Devanagari to be transliterated into the Roman script. Second, we presented an algorithm for computing a similarity measure that is a variant of Dice's coefficient measure and the LCSR measure and which also takes into account the constraints needed to match English-Hindi transliterated words. Finally, by evaluating various similarity metrics individually and together under a multiple measure agreement scenario, we showed that it is possible to achieve high accuracy in identifying English-Hindi word pairs that are transliterations.

With the help of components developed so far, we have addressed issues such as Hindi morphology and aligning cognates and named entities. Now, we move onto the next chapter where we describe our English-Hindi word alignment algorithm.

Chapter 7
Word Alignment

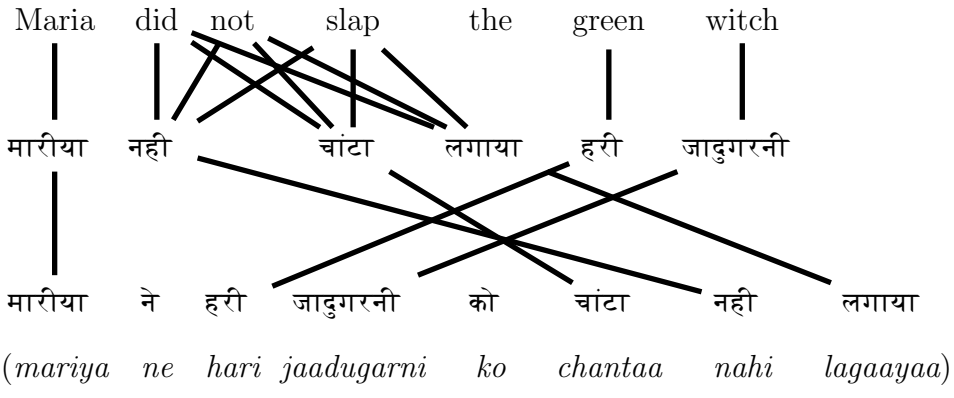
7.1 Outline of the Chapter

In this chapter, we give details of our work on the development of a word alignment algorithm for the English-Hindi language pair[1]. In the sections below, we first give an example of English-Hindi word alignment. Secondly, we look at related work and discuss various techniques that researchers have employed to solve the word alignment problem in general and in particularly for the English-Hindi language pair.

We used the GIZA++ toolkit (Och and Ney, 2004) and applied several heuristics (as suggested by Koehn et al. (2007)) to obtain baseline results. We discuss these results and finally present a hybrid word alignment algorithm for the English-Hindi language pair. We provide a detailed evaluation of our algorithm and its components.

7.2 The Problem of Alignment

Before a machine translation system can translate a sentence, it has to know how to translate words. For this, it has to know, for each word in the source language, its equivalent word in the target language. Let us return to the example introduced in the Introduction (see Chapter 1) (adapted from Koehn et al. (2007)).



[1]Two different versions of the English-Hindi word alignment algorithm presented in this chapter have been published in Aswani and Gaizauskas (2005a) and Aswani and Gaizauskas (2009).

In this example, we align the English word *witch* to the Hindi word जादुगरनी (*jadugarni*) as they both refer to a person who practices magic.

The mapping between words is called **word alignment** and a **word alignment algorithm** tells how the words in a given sentence pair should be aligned.

Given pairs of aligned sentences, the typical use of a word alignment algorithm in a machine translation system is to find out associations between the words of the source language sentence and the paired target language sentence. Later, these associations are used to obtain candidate translations for every word in a sentence given for translation. According to Huang (2009), most recently developed machine translation systems use phrase-based translations or syntax based word alignments. However, in order to obtain a phrase table (as opposed to word associations), these systems rely on the word alignments obtained in the first place – thus the quality of word alignment has significant importance in the output of machine translation.

The task of aligning text at the word level is not an easy problem to solve as there are several issues that need to be taken into consideration. As explained by Och and Ney (2003), *"Word alignment is a difficult problem as often humans cannot determine easily which target words correspond to which source words. In particular, the alignment of words within idiomatic expressions, free translations, and missing function words is problematic. Often, the word order is different, certain words are never translated or a word is translated into a phrase."*

We can witness some of these problems in our English-Hindi word alignment example discussed earlier. For example, the word order is different, there are words on both sides which are aligned to more than one word (e.g. slap) and there are words which are not aligned at all (e.g. the, ने (*ne*), को (*ko*)).

In the section below, we look at related work and discuss various techniques that researchers have employed to solve the problems associated with the word alignment task.

7.3 Related Work

Depending on the language pair and availability of data, various methods of word alignment have been proposed in the literature. In most of the literature, word alignment methods are categorized as either statistical or heuristic approaches. Statistical approaches estimate alignment probabilities from sentence aligned parallel corpora whereas heuristic approaches use associative measures derived either from corpora, or external sources such as dictionaries.

7.3.1 Statistical Approaches

Going back into the history of statistical word alignment, Brown et al. (1993) wrote one of the most important papers concerning statistical alignment of parallel texts. The five models reported in that paper, also known as IBM models, have become the basis for most of the papers written on statistical word alignment since then.

IBM Models

Brown et al. (1993) described a series of five statistical models of the translation process. For any given pair of aligned sentences each of their five models assigns a probability to each of the possible word-by-word alignments in the pair of aligned sentences. These models mainly rely on three factors: translation probability, fertility and distortion probability.

1. Given a source and a target word, translation probability refers to the likelihood of the source word being aligned to the target word.

2. The second factor, fertility, is defined as the number of target words generated by a given source word.

3. Finally the third factor is distortion which relates the positions of translated words in the source and target sentences. If the source and target words tend to appear in the same parts of sentences, they are said to be translated roughly undistorted, while words which move into far distance have high distortion.

Given an aligned pair of sentences *(e,f)* where *e* and *f* refer to a source and a target sentence respectively, in model 1, they assume initially that all alignments for each target word are equally likely. Therefore, the order of the words in *e* and *f* does not affect the translation model Pr(f|e) (i.e. probability of the sentence *e* being translated into *f*). Every time their program sees a certain word pair co-occurring in an alignment, it marks down a count for the pair. After the program has traversed the entire corpus, the program normalizes these counts and creates a new word-translation table. They use the expectation maximization (EM algorithm) (Dempster et al., 1977) to obtain the translation probabilities.

In model 2, they assume that the probability of a target word being aligned to a source word also depends on the absolute positions of the source words it aligns to. Thus, for model 2, the translation model Pr(f|e) **does** depend on the order of the words in *e* and *f*.

Model 3, then, adds the fertility parameter to the training. In other words, for each source word in *e*, the number of words in *f* it aligns to is considered. In model 4, instead of considering the absolute positions of tokens (as in model 2), relative word ordering is considered. In other words, for every target word in *f*, the model adds the relative positions of words in *e* that the target word aligns to.

According to Brown et al. (1993), models 3 and 4 are deficient, which means that they waste some of their probability on objects that are not target sentences at all. According to them, the problem with model 3 and 4 is that that they may place multiple output words at the same position. Describing model 5, they say that it is very much like model 4, except that it is not deficient in this way.

They use a method called *hill climbing* whereby a reasonable-looking alignment is selected as input for the subsequent models. For example, the output of model 1 (i.e. word-translation table) is provided to model 2 as its input. Model 2, then, takes account of absolute positions of the word pairs in the word-translation table and trains its model. Thus by initializing each model from the parameters of the model before it, they arrive at estimates of the parameters of the final model that do not depend on their initial estimates of the parameters for model

1.

GIZA and GIZA++

Based on the work of Brown et al. (1993), Al-onaizan et al. (1999) developed a toolkit called EGYPT. The toolkit was developed for researchers working with parallel corpora and those who wanted to test new methods of statistical machine translation. It includes corpus preparation software, bilingual text training software and run-time decoding software for performing actual translation. The core of the EGYPT toolkit is the system called *GIZA*, which is based on the algorithms and translation models described in Brown et al. (1993).

Och and Ney (2003) developed an extension to the *GIZA* system which is known as *GIZA++*. GIZA++ has several new features including but not limited to implementations of models 4 and 5, the HMM based alignment model (Vogel et al., 1996) and alignment models depending on word classes, etc.

The main differences among IBM models 1 and 2, and the HMM model are how the alignments are chosen. IBM model 1 assumes that the alignment of a word only depends on the length of the source sentence; IBM model 2 assumes that the alignment depends on the position of the word and the length of the source and target sentences; and the HMM model assumes that the alignment depends on the previous word's alignment and the length of the source and target sentences. Because the HMM model assumes that alignment depends on the previous word's alignment and since syntactic constituents tend to move as a unit, the HMM model can capture this locality property in a much better way than IBM models 1 and 2 (Ananiadou, 2011).

Phrase-based Translation

The IBM models proposed by Brown et al. (1993) and implemented in GIZA++ have certain limitations. They only allow at most one source word to be aligned with each foreign word. However, it is possible to align the same source word with more than one foreign word. It is a constraint on foreign words that they cannot be aligned to more than one source word. For this reason, alignment of multiple words and phrases which do not decompose easily in

word-to-word translations is not possible.

As mentioned by Och and Ney (2003), the first problem – where many-to-many translation is not possible – can be addressed if the IBM models are applied in both directions. In other words, the parallel corpus is aligned bidirectionally, e.g. source to target and target to source. This generates two word alignments that have to be merged to produce many-to-many alignments. Since the source-target and target-source alignments are not symmetric, they suggest that this approach can help in identifying phrases where word-to-word translations are not possible. However, the approach does not guarantee to recover all possible many-to-many translations. If the intersection of the two alignments is taken, it is expected to give high-precision one-to-one alignment with a high-confidence score for each alignment.

Och and Ney (2004) discuss what is the best method to extract phrase translation pairs. Their system works on the idea that lexical correspondences can be established not only at the word level but at the phrase level as well. In order to achieve better results they apply various heuristics to the alignment results produced by the IBM models.

At a minimum, they start with the high-precision one-to-one alignments using the intersection method explained above. They also obtain a set of word alignments based on the union of the bidirectional word alignment outputs. At a maximum, they retain all the phrases obtained in the union set. To illustrate this process, we use an image from Koehn et al. (2007) (see Figure 7.1). They always require that a new word alignment pair involves at least one previously unaligned word.

In order to choose alignments for their final alignment set, they explore the alignments between the intersection and union sets. The decision on which alignments to consider is made based on several criteria as listed below:

1. In which alignment set does the potential alignment point exist? It could be in the source-to-target (aka *e2f* method) or target-to-source set (aka *f2e* method).

2. Are the potential candidate words in neighbour for alignment already aligned?

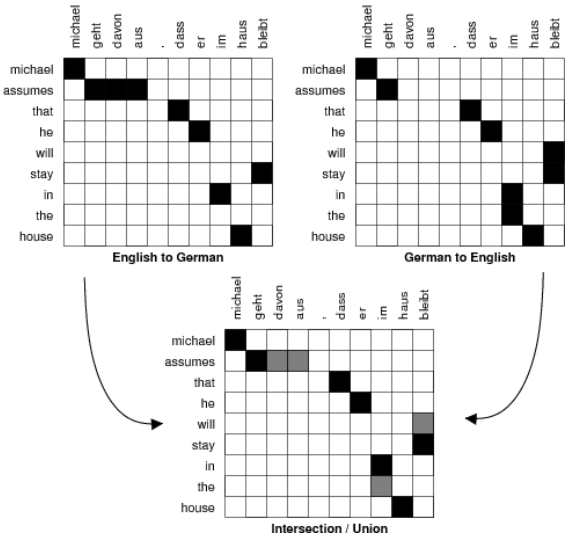


Figure 7.1: Intersection/Union of Word Alignments (Koehn et al., 2007)

3. Does the term *neighbouring* mean directly adjacent (aka *grow* method) or also diagonally adjacent (aka *grow-diag* method)?

4. Is the source or foreign word that the potential word aligns to unaligned so far? Are both unaligned? (aka *grow-diag-final* method)

5. What is the lexical translation probability for the potential candidate?

Koehn et al. (2007) use a similar method of starting with the intersection alignment set and exploring the space up to the union alignment set. They use the same set of criteria to select a word alignment pair. However, their method is different in that they first expand to only directly adjacent alignment words. They check for potential alignments starting from top right corner of the alignment matrix, checking for alignment words for the first source word, then continue with alignment words for the second source word and so on until all source words have been considered. They do this iteratively until no word alignment can be added to the final alignment set. In a final step, they consider non-adjacent alignment words, with otherwise the same conditions. The method, including the final step, is also known as the *grow-diag-and-final* method.

Deng and Zhou (2009) propose a method which they claim finds a balance between intersection and union methods of word alignment. In their case, combination is carried out as an optimization process driven by an effectiveness function. They evaluate the impact of each alignment pair with respect to the the task of phrase table creation. They consider only those phrases as candidates where boundary words are aligned. They show that considering this constraint achieves them 0.01 increase in BLEU score when compared to the grow-diag-and-final method.

As specified earlier, according to Huang (2009), most recently developed machine translation systems use phrase-based translations or syntax-based word alignments. The advantage of phrase-based models is that they allow phrase-to-phrase alignments and eliminate the need for parameters for fertility and null alignments. They also reduce the dependency on distortion. However, a disadvantage of a phrase-based model is that all possible segmentations and their alignments need to be considered – resulting in explosion of complexity.

7.3.2 Limitations of the IBM Models

Moore (2005) and Fishel (2010) give a good explanation of why the IBM models are problematic. They mention that when there is a lexical ambiguity, model 1 will select the most probable word pair in all cases and therefore it would not be able to resolve a conflict between repeated items, like punctuation marks or the same words. They mention that the same words can occur at different positions in sentences of different lengths and in the case of model 2 and 3, where absolute positions of words are taken into consideration, this can be a problem. They mention that although HMM models can capture the locality property in a much better way than IBM models 1 and 2 (as explained earlier), with the HMM model, the alignments are not independent any more.

For IBM models to produce acceptable results, they require a huge amount of data. This is because, if a program sees a particular word or phrase thousand times, it is more likely to learn a correct translation for such a word or phrase than if it sees it ten times, or once or never. The problem increases if a language is morphologically rich and words in the corpus

are inflected significantly. In such cases, it is unlikely that words will be seen sufficiently often for the statistical approach to work well. The problem becomes even more difficult when one of the languages in question is a free order language and contains the words that are translated with high distortion.

The good thing about the IBM models is that their learning is unsupervised – which means the system learns all its parameters automatically from a parallel corpus without any training data. However, for exactly the same reason, the IBM models are time consuming to train and the quality produced by them is poor without enough parallel data (Moore, 2005; Ambati et al., 2010; Ananiadou, 2011).

In the next section, we discuss discriminative approaches to word alignment where statistical methods are used in conjunction with external features to calculate scores for word alignment.

7.3.3 Discriminative Approaches

Many researchers have proposed discriminative approaches to handle word alignment. The main difference here is to use external features in calculating scores for word alignment. Such approaches often require a little annotated data to add an element of supervised learning.

Moore (2005) was amongst the initial work reported on discriminative alignment models. He explains that there are various parameters in the IBM models that should be optimized using annotated data. For example, aligning a word to null in the HMM model, smoothing parameters of the distortion probabilities and fertility probabilities. According to him, in practice these parameters are rarely optimized for the reason that they need a lot of training on a large amount of data. To overcome this problem, he proposes a method based on discriminative training of a weighted linear combination of a small number of features. For each pair of words, he obtains corresponding features and multiplies them with the features' corresponding weights to obtain an overall score for the pair. If summed together, these word pair scores give an overall alignment score for the sentence pair under consideration. In his method, he uses an association measure to obtain a translation score. Using pre-annotated

word-aligned data, he uses features such as how much word re-ordering is required, how many words are left unaligned and how often a word is linked to several words. The reported results on a subset of the Canadian Hansard bilingual corpus are slightly better than the baseline results. The reported alignment error rate for Moore (2005) is 0.075 in comparison to the baseline alignment error rate (GIZA++ with a refined IBM model 4) of 0.079.

Fraser and Marcu (2005) modify Model 4 to be a log-linear combination of 11 sub-models (5 based on standard Model 4 parameters, and 6 based on additional features) and discriminatively optimize the sub-model weights on each iteration of a Viterbi approximation to EM. Similarly Liu et al. (2005) develop a log-linear model, based on IBM Model 3. They train Model 3 using GIZA++, and then use the Model 3 score of a possible alignment as a feature value in a discriminatively trained log-linear model, along with features incorporating part-of-speech information, and whether the aligned words are given as translations in a bilingual dictionary. Blunsom and Cohn (2006) use a Conditional Random Field (CRF) which they estimate on a small supervised training set. They incorporate a large number of predictive features such as translation score based on Dice's coefficient and IBM model 1, orthographic features such as exact string similarity, with and without vowels, edit distance between the source and target words, POS tags of individual words, whether a relevant bilingual dictionary contains a possible alignment pair, distances between the alignments of adjacent words and so on. They apply this alignment model to French-English and Romanian-English language pairs and show that their model outperforms the generative Model 4 baseline. They also incorporated the output of Model 4 as one of the features for which their CRF model achieved the then highest reported results. Riesa and Marcu (2010) use features such as lexical probabilities, fertility, part-of-speech information and distance of a word from the diagonal to learn the weight that should be given to a pair. They report an increase of 6.3% accuracy above the baseline results obtained using GIZA++ IBM model 4 for the Arabic-English language pair.

Gao et al. (2010) propose a semi-supervised method whereby they use manually partially aligned data to feed into statistical models. For their experiments, they changed the GIZA++

implementation to accommodate certain constraints. For example, if one word is manually aligned to more than one word, which word to choose for alignment is decided from these links. They use manually prepared dictionaries to get initial alignments. They have made changes in the IBM models to allow many-to-many relations as long as they exist in the manually prepared alignment. With such changes made to the IBM models, they show that they were able to reduce the alignment error rate by 1.6 points to 35.61 in comparison to the baseline results obtained by using the unchanged version of IBM model 4. Ma et al. (2009) report an increase of 2.3% in the F-Measure score (in comparison to the output of IBM model 4) for their experiments on the English-Chinese language pair. They use output from IBM model 1 along with features such as POS translation probabilities and dependency parses to obtain relations between words.

In the section below, we review heuristic approaches that are used for word alignment.

7.3.4 Heuristic Approaches

Heuristic alignment methods are based on specific associative measures. Examples of such measures are the Dice's coefficient, χ^2 and mutual information.

Frequently variations of Dice's coefficient (Dice, 1945) are used (see examples below), as a similarity function to obtain a matrix of associations scores between every word at every position in a pair of aligned sentences using the following equation.

$$dice(i,j) = \frac{2 \times C(e_i,fj)}{(C(e_i) + C(f_j))}$$

Here, e_i is a word from the source language and f_j is a word from the target language. $C(e_i, f_j)$ denotes the co-occurrence count of e_i and f_j in the parallel training corpus. $C(e_i)$ and $C(f_j)$ denote the counts of e_i and f_j in the corpus respectively.

Melamed (2000) proposes an algorithm called the competitive linking algorithm which exploits this association matrix. The algorithm picks up the highest ranking (the one with

the highest association score) word position (i, j) and then removes the corresponding row and column from the association matrix. This procedure is then repeated until every word in the source and target language is aligned. The advantage of this method is the indirect alignments. That is, high frequency words are selected initially but even the low frequency words are guessed from the remaining positions. However, the result of this process is a **one-to-one** alignment with high precision. Each word is translated to at most one other word. In case of no translation for a word, it is aligned to NULL.

Fung and Church (1994) present a heuristic algorithm known as *K-vec*. Their algorithm uses several heuristic measures to estimate word correspondences. They use word occurrence vectors for alignment selection. They divide the parallel corpus into K segments and for each word construct a K-dimensional binary vector. If the word appears in segment i, they set the corresponding dimension in the vector to 1. For example, if the value of K is set to 3 (i.e. a corpus is divided into three segments), and the word *car* occurs in segments 1 and 3, the binary vector for the word *car* is set to <1,0,1>. All source and target language words are assigned a k-vector, and are later compared with each other. High correlation between segments where the source and target language words occur and do not occur raises the likelihood that the source word corresponds with the target word. Given a source language word, this method is used for short-listing a set of probable target language words. They use mutual information to choose appropriate alignments from these probable words.

The authors present a variation of this method called *DK-vec* for aligning noisy parallel corpora. This approach uses a matching algorithm called Dynamic Time Warping (Fung and Mckeown, 1994). Unlike the *K-vec* algorithm the text is not split into K segments of similar size with the occurrences of source and target language words being stored in binary vectors. In the *DK-vec* algorithm, the distances between the occurrences of the source and target language word are stored in so-called recency vectors. So each entry in the vector is the distance between a pair of occurrences of the source and target word. The algorithm, for each source and target word, computes such a recency vector. It is based on the notion that while similar words do not occur at the exact same position in each part of the parallel

corpus, distances between instances of the same word are similar across languages. In order to locate candidate alignment pairs, all pairs which occur for the first time after half of the text are filtered out. Furthermore, all pairs where one vector is less than half the length of the other are also removed. For the remaining pairs the absolute difference between their vectors is computed using Dynamic Time Warping. In the last step, they sort the pairs by their absolute difference to find closely correlated word pairs.

Ahrenberg et al. (1998, 2002) present an algorithm called LWA (Linkoping Word Aligner) which combines the K-vec approach (Fung and Church, 1994) and the greedy word-to-word algorithm of Melamed (1997b). In addition, they handle open class expressions separately from closed class expressions. In order to find lexical correspondences, the algorithm works in a number of iterations taking the following actions in each iteration. They use the K-vec approach to estimate probabilities of candidate open class expressions and then the closed class expressions individually. They use the *t-score* to rank the word pairs. They repeat the same process to align closed class expressions. Where there are sentences with only one 1:1, 1:n and n:1 pair, they form a translation pair. Once the pairs are found, they are removed from the sentences and the second iteration begins. They also integrate other modules such as a morphological analyser to obtain base-forms of the words and a phrase module that includes multi-word expressions.

Tiedemann (2003) presents a word alignment technique that is based on multiple heuristic alignment strategies. The approach uses a sentence-to-sentence matrix to represent the possible alignments and assigns a score to each available word alignment based on several clues calculated from sources such as dice-coefficient, longest common subsequence ratio (LCSR), POS tags, positional weighting, N-grams and chunks. The final score is the result from a weighted summation of these independent clues.

Berney and Perini (2005) present an algorithm to induce a translation dictionary between two similar languages using parallel corpora. The first step in their algorithm is to find equivalent phrases in the source and target languages to reduce the total number of comparisons. They

make an assumption that some words, such as names of places, names of persons etc. are identical in the source and target languages. Using such words they divide text into phrases. They also use cognates to help them match phrases. In order to match named entities and cognates they use Levenshtein's similarity score. However, to reduce the search space they make sure the words being compared have similar frequency in the corpus and also that they do not appear too far away from each other. They also make an assumption that a linear relation exists between the location of a given source language word and the target language equivalent. Thus, they assume that the translation equivalent of a word in the source language should appear within a window of a certain number of words in the target sentence. Using words matched this way and using punctuations around them as anchors points, they align phrases. After having identified phrases, they assume that a word only appears once in a phrase. Considering one phrase at a time in the source language and its respective target phrase, they use the ϕ $(phi)^2$ statistic to find translation equivalents. They make a further refinement by aligning words in this way with frequency above 2 only. Using this method, they were able to achieve 0.9 precision but with a low recall (not specified in their paper).

7.3.5 Hybrid Approaches

When it is difficult to rely purely on statistical methods due to a lack of data, research shows that it is possible to benefit from language specific resources. Depending on the type of a problem, various language specific resources can be used to handle the problem. For example, dictionaries can be used to obtain word translations; transliteration mapping can be used to identify named entities (where the only difference between the words used across the two languages is a different script). Such a system, where more than one method (or a resource) is used, is called a hybrid system.

[2]The phi coefficient is a measure of the degree of association between two binary variables. This measure is similar to the correlation coefficient in its interpretation. Please see `http://www.childrensmercy.org/stats/definitions/phi.htm` for information.

Tufiş et al. (2003) explain a word alignment system (TREQ-AL) with limited language resources. Their alignment algorithm takes a dictionary as input and is executed in four steps.

1. **left-to-right pre-alignment** Considering one word at a time, they first search for it in a dictionary and retrieve all possible translations. They link the source word with all possible matches in the target language. They also use a cognate score to choose among the possible links. When some consecutive words in a source sentence are aligned with some consecutive words in a target sentence, these are said to form an alignment chain.

2. **right-to-left adjustment of the pre-alignment** This step reduces (when possible) 1 *to* n links generated in the first step to a 1 *to* 1 link. If for one word in the source language there are several alignment possibilities, the one that belongs to an alignment chain is selected. In the case of competing alignments, the one with the highest cognate score is selected. Finally the relative positions of words in the competing link are taken into account to minimize the distance.

3. **determining alignment zones and filtering out unwanted zones** Conjunctions, prepositions and punctuation marks are used as delimiters to form alignment zones within the sentences. Using alignment links assigned in the previous steps, all alignment links that point outside zones are removed.

4. **word-alignment within alignment zones** In their final step, considering one pair of alignment zones and POS tags of the words, they try to align unaligned words. All unaligned words are then marked as `null` alignments.

According to Lopez et al. (2002) the incorporation of syntactic knowledge into the alignment model results in higher quality alignments. They claim that synchronous parsing appears to be the best model for syntactic projection. Their algorithm is a modified version of the algorithm described by Alshawi and Douglas (2000). They expect a parsed pair of

sentences as input to their system. They use a bottom-up dynamic programming procedure by initializing each word in a source sentence to be aligned with one or more words that have the same syntactic category in the target language. As in Alshawi and Douglas (2000), each possible 1:1 alignment is scored using the algorithm proposed by Gale and Church (1991). In other words, pairs are chosen based on their co-occurrence count in the corpus. To compute alignments of larger spans, their algorithm combines adjacent sub-alignments. During this step, one sub-alignment becomes a modifier phrase. Interpreting this in terms of dependency parsing, the aligned headwords of the modifier phrase become a modifiers of the aligned headwords of the other phrase. At each step, they compute the cost of the alignment. The output of their algorithm is the highest scoring alignment that covers the entire span of both the source and target sentences.

Ayan et al. (2004) present a framework for incremental testing of different alignment algorithms and their combinations. They demonstrate that a combination of statistical and linguistically-informed alignments can resolve translation divergences during the alignment process. The idea behind this system is to allow users to plug-in more than one alignment system and use the combined output to produce the final alignment. Given a sentence pair, each plugged-in aligner is asked to provide its confidence for word pairs and the word-pairs with higher confidence are chosen. The framework also allows users to specify the aligner to be taken as superior to the others on certain word pairs. They use semantic-based word classes and a set of general, linguistically-motivated rules from DUSTer (Divergence Unravelling for Statistical Translation (Dorr et al., 2002)) to induce alignment improvements over GIZA++. DUSTer is a method for systematically identifying common divergence types and transforming an English sentence structure to bear a closer resemblance to that of another language.

They obtained 0.72 (precision), 0.73 (recall) and 0.72 (F-Measure) when using GIZA++ on its own. During their error analysis of statistically induced alignments (only GIZA++) for the 99 English-Spanish sentences, they found that 80% of statistical alignment errors correspond to missing alignment links and 61% of these missed alignments are related to

verbs, functional nouns and obliques which form the main categories of words that are handled by linguistically motivated components. In another experiment, they combined the GIZA++ with the DUSTer system and took the union of the results as final output. They report 0.73 (precision), 0.73 (recall) and 0.73 (F-Measure).

7.3.6 Word Alignment Algorithms for the English-Hindi Language Pair

Bharti et al. (2002) describe a hybrid approach that is specific to the English-Hindi language pair. Although the algorithm explained by Bharti et al. (2002) is for sentence alignment, their method of chunk matching can also be used for English-Hindi word alignment. They consider noun phrases and verb phrases as chunks in sentences. While matching noun phrases they consider all words except prepositions and postpositions and for verb phrase matching they consider only head words. For matching chunks they use four methods: bilingual dictionary lookup, synonym lookups, number matching and phonetic matching.

According to Drábek and Yarowsky (2004), it requires only coarse-grained knowledge of basic word order – knowledge which can be rapidly found in even the briefest grammatical sketches – to predict the word order. They show that due to the syntactic divergences between English and Hindi, translation between texts in English and Hindi requires a high degree of reordering. The basic word order in English is subject-verb-object (SVO) whereas in Hindi it is subject-object-verb (SOV). Using a syntactic parser it is possible to identify SVO in English and then to rearrange words in English sentence to match the word order of the Hindi sentence (SOV). Since Hindi uses postpositions instead of prepositions in English, prepositions can be placed at the proper Hindi positions (Kulkarni, 2003).

In their paper, Chatterjee and Agrawal (2006) present a word alignment algorithm for the English-Hindi language pair that uses the DK-vec algorithm as presented by Fung and Church (1994). They use a corpus manually prepared using sources such as children's story books, English-Hindi translation book material and advertisements. Although they report poor results when using the original algorithm, they claim that adding new constraints such as use of root words and the i^{th} segment constraint to the recency-vector approach

significantly improves the performance. The i^{th} segment constraint relates to specifying the segment range that is relative to the current segment in which the alignment is sought. They use Levenshtein distance for comparing recency vectors. When combining the i^{th} segment constraint with the *different frequency ranges* and *multiple alignment selection (MAS)* filters, they report a significant improvement in their results. The different frequency ranges filter considers only words within the specified frequency in the current segment and the MAS constraint takes care of situations where a source word has more than one possible alignment in the target language by selecting the word pair with minimum alignment score. For example, when the frequency range was set to 2-5, they obtained 0.78 (precision), 0.50 (recall) and 0.61 (F-Measure) for the words with frequency between 2 and 5. Similarly, they obtained P=1,R=1,F=1 for the words with frequency between 200 and 300.

Venkatapathy and Joshi (2006) present an approach to align multi-word expressions (MWEs). They mainly concentrate on verbs and their dependants. Using a stochastic TAG based dependency parser (Shen, 2006) in a corpus of 400 sentence pairs they found that almost 9% of the dependants were verb dependants (193 out of 2209). They combine various local features such as Dice's coefficient calculated over morphological roots with global features such as an average distance between the words aligned to verbs, number of many-to-one alignments, POS tags of verb dependants and mutual information collected for dependants aligned to same verb in the target language. They report results for their experiments with GIZA++ and their own model. They applied GIZA++ in both the directions on a English-Hindi corpus consisting of 50000 sentence pairs. In the first experiment they did not use any morphological analyser and obtained 0.82 (precision), 0.19 (recall) and 0.31 (F-Measure). When they used a lemmatised corpus they obtained 0.81 (precision), 0.23 (recall) and 0.35 (F-Measure). In order to train their own model they used 294 sentences manually aligned at the word-level. When they used only local features they obtained 0.47 (precision), 0.38 (recall) and 0.42 (F-Measure). Finally when combining local features with global features (average distance and overlap) they obtained 0.49 (precision), 0.39 (recall) and 0.43 (F-Measure). Combining mutual information and position information obtained from GIZA++

into their models, they obtained 0.55 (precision), 0.45 (recall) and 0.50 (F-Measure).

Chinnappa and Singh (2007) use the first three IBM models along with Dice's coefficient measure to reduce the number of inappropriate alignments. They mention that for better word alignment of text in Indian languages, information about cognates is certainly needed. They use a English-Hindi bilingual dictionary along with a list of cognates for the initialization of the EM algorithm. They also experimented with using a morphological analyser. In their experiment with GIZA++, they obtained 0.37 (precision), 0.39 (recall) and 0.38 (F-Measure) whereas they obtained 0.29 (precision), 0.32 (recall) and 0.31 (F-Measure) when using the first three IBM models along with Dice's coefficient to filter out inappropriate alignments, a morphological analyser to obtain base-forms of the English and Hindi words, a list of cognates and a English-Hindi bilingual dictionary.

7.3.7 Discussion

As observed through the study of previous work, most approaches to word alignment are based on statistical methods such as presented by Och and Ney (2003). However, such statistical approaches have their own limitations (see section 7.3.2) when the languages under consideration, are different from each other.

One of the limitations of statistical approaches to word alignment is that they require a huge amount of training data. Discriminative approaches, which make use of external resources need strong signals from the underlying unsupervised learners to work well. Although, many South Asian languages (such as the Hindi, Gujarati, Bengali etc.) are widely spoken, there is not enough parallel data available to learn accurate word alignment for them. What adds to the problem is that these languages are morphologically rich and free order as well. As we have seen, many researchers have suggested use of language specific resources when there is not enough parallel data available.

There are at least three observations that can be singled out from the various approaches discussed above.

1. It seems that training word alignment models on base-forms is the one unavoidable step to improve not only the accuracy of word alignment algorithms but the coverage as well.

2. Given the high number of cognates in some language pairs, several algorithms, as discussed above, give special treatment to cognates.

3. Aligning multi-word expressions is one of the most difficult challenges researchers have come across.

Even though, we have tools such as a Hindi morphological analyser (see Chapter 5) and a transliteration similarity component (see Chapter 6), not having enough data could still cause trouble in obtaining reasonable word alignment accuracies.

Although, the literature does not favour statistical methods to perform very well on the English-Hindi language pair, it is very important to verify their performance. Since the IBM Models have been the most common choice (in the literature), we use them for the English-Hindi language pair and establish a baseline for our English-Hindi word alignment algorithm presented later in this chapter.

In the next section we give details of our word alignment experiments with the GIZA++ toolkit.

7.4 Experiments with GIZA++

Dataset

We prepared three datasets for three different experiments.

First, we used a collection of 200 English-Hindi parallel documents aligned at the sentence level. Each document in this collection contains approximately 10-15 sentence pairings (in total 2029 sentence pairings). With the help of four Hindi native speakers, we were able to prepare the word alignment gold-standard for these documents. The guidelines used for

preparing a word alignment gold-standard are provided in Chapter 2. We call this dataset WA_DATASET1.

We prepared another dataset which, in addition to all the sentence pairings from WA_DATASET1, contains 3957 sentence pairings collected from the EMILLE corpus. We call this dataset WA_DATASET2.

Finally, the third dataset is a mixture of all the sentence pairings from WA_DATASET2 and parallel sentence pairings from Bojar et al. (2010). The corpus prepared by Bojar et. al contains English-Hindi parallel sentence pairings collected from various sources. These include articles collected from `http://www.danielpipes.org`, agriculture corpus (section B), a cleaner version of the English-Hindi parallel corpus distributed during the TIDES-2003 exercise and the data distributed by us for the word alignment shared task (Martin et al., 2005)). In total the third dataset contains 59,701 unique sentence pairings. We call this dataset WA_DATASET3.

Sentences in all the three datasets were converted into the appropriate format required by the GIZA++ system. In order to reduce data sparseness, we converted all our data into lowercase and tagged every token with its base-form. We used the GATE morphological analyser (see Chapter 3) for the English language and the Hindi morphological analyser we developed (see Chapter 5) for the Hindi language. Additionally, the tokens in WA_DATASET3 were tagged with their POS tags. We used POS taggers from GATE to tag both the English and Hindi tokens.

Inter Annotator Agreement

It is important to get an idea of how much human agreement we have on the manually prepared word alignment. If the agreement between human annotators is not very well, it could be an indication of a difficult alignment problem. We follow the procedure as specified in Kruijff-Korbayova et al. (2006).

To compute agreement, we produce a cartesian product of all the words in a source language

sentence with the words in its respective target language sentence. If a word in the source sentence is aligned with a word in the target sentence, we classify the pair as *True* instance and *False* otherwise. To calculate the agreement between the alignments produced by the two annotators we use two measures.

The first measure takes into account only the True instances. For the two sets of alignments produced by the annotators, we compute $A1$ and $A2$, the number of True instances. Then we compute the intersection I – the number of alignment pairs the two annotators have agreed on. The final agreement is computed using the following equation.

$$AGR = \frac{2*I}{A1+A2}$$

The computed agreement for the True instances is about 91%.

As a second measure, we compute Kappa statistics based on two popular methods: Cohen's Kappa (Cohen, 1960) and Scott's Pi (Lombard et al., 2002). Agreements obtained for the Cohen's Kappa is 0.83 and Scott's Pi is 0.92. According to Krippendorff (1980), both the values obtained here are above the threshold of 0.8 for reliability.

Experimental Setup

We used the software toolkit called *Moses* (Koehn et al., 2007) for our experiments. It is an open source toolkit for statistical machine translation. The toolkit provides several scripts to run experiments with GIZA++ (Och and Ney, 2003) and to set various parameters for the experiments. We aligned our parallel corpus with IBM Model 4. We used the default configuration file included with the chosen version (revision 4383) of the Moses toolkit. In other words, what we used resulted in five iterations of Model 1, followed by five iterations of the HMM model, followed by five iterations of Model 4.

We trained the models in both directions, English-to-Hindi (e2h) and Hindi-to-English (h2e), and computed the union, intersection, and what Och and Ney (2004) call the *refined* combi-

nation of the two alignments[3].

Results

We applied GIZA++ on the WA_DATASET1 and obtained the results as shown in Table 7.1. Information about the various heuristics applied on the results of GIZA++ can be found in section 7.3.1.

Table 7.1: Experiment Results of GIZA++ for the English-Hindi Language Pair

Applied Heuristic	**precision**	**recall**	**F-Measure**
e2h	0.48	0.50	0.49
h2e	0.56	0.47	0.51
intersection	0.88	0.34	0.50
union	0.42	0.63	0.50
grow	0.61	0.44	0.51
grow-diag	0.56	0.50	0.53
grow-diag-final	0.54	0.55	**0.54**

As can be seen, the best results were obtained with the *grow-diag-final* heuristic: 0.54 (P), 0.55 (R) and 0.54 (F1).

It must be noted that the WA_DATASET1 contains only a limited number of sentence pairings (i.e. 2029). In order to find out if providing more data to GIZA++ can help in improving the word alignment results, we applied GIZA++ on the WA_DATASET2 which contains an extra 3957 sentence pairings. We applied the same heuristic, i.e. grow-diag-final, that had yielded the best results for WA_DATASET1. In order to compare the results of the two experiments, we only looked at the word alignment results of the same 2029 sentence pairings that appear in the WA_DATASET1. Even though, we had provided more data for the training of GIZA++ in the second experiment, we observed that the figures had gone down to 0.50 (P), 0.49 (R) and 0.50 (F1).

[3]Please see the discussion on "Phrase-based Translation" in the Section 7.3.1 for more information on the various heuristics used for obtaining the final alignment

Finally, we ran our experiments on the WA_DATASET3. Statistical approaches are expected to work better when they are provided a fair amount of data. With 19 times more data (in comparison to WA_DATASET2), we expected the GIZA++ to perform better. We experimented training our models on different factors (e.g. source form, root form, POS tags) to observe the change each factor brings in the overall alignment. We used the *grow-diag-final* heuristic for all our experiments on WA_DATASET3. Table 7.2 shows the results obtained for our experiments.

Table 7.2: Results of GIZA++ on WA_DATASET3

Used Factor(s)	**Precision**	**Recall**	**F-Measure**
surface form	0.59	0.60	0.59
root form	**0.62**	**0.64**	**0.63**
root form with dictionary	**0.62**	**0.64**	**0.63**
root forms with POS tags	0.59	0.61	0.60
surface (English), root (Hindi)	0.60	0.62	0.61

Discussion

In all the three experiments, we observed that the problem was with many-to-many alignments. Also, there are words such as numbers, names of people, etc. for which the system had difficulty in aligning them correctly. As previously mentioned in Chapter 1, if a program such as GIZA++ sees a particular word or phrase a thousand times, it is more likely to learn a correct translation for such a word or phrase than if it sees it ten times, or once or never. In case of many-to-many alignments, we believe that the system was not able to find parallel phrases with enough frequency in the first two datasets which it can use to calculate probability scores that are good enough to decide if a pair of phrases should be aligned with each other or not.

Our analysis of the results of the second experiment revealed that the system was able to align more words than it had previously aligned during the first experiment. However, the overall alignment accuracy had dropped. Some of the entries which were aligned correctly in the first experiment had been aligned incorrectly in the second experiment. This included many

NULL alignments (in the gold standard) being aligned to one or more words by GIZA++. We observed that the system had difficulties in aligning vibhakties which are often part of multi-term expressions.

With 19 times more data, we can see the real improvement in the results. With a large number of sentence aligned data, the GIZA++ is performing much better – even with the surface forms (0.59 F-Measure). However, the results get real boost when we use root forms (0.63 F-Measure). This confirms the need for a morphological analyser. Still, given the amount of additional data supplied, the results have not improved significantly. This could be partially due to the fact that the corpus compiled by Bojar et al. (2010) has data from different domains.

There are many examples in the literature when instead of relying on statistically produced translation tables, researchers use bilingual dictionaries to locate candidate translations in the other language (see the previous section). Also, there are publications such as Saha et al. (2008) and Kunchukuttan and Damani (2008), who describe methods of dealing with many-to-many alignments for the English-Hindi language pair. But even when we supply our Bilingual English-Hindi dictionary as additional sentence pairings to the GIZA++, we see no change in the results.

Using Alignments for Machine Translation

Moses currently uses GIZA++ to obtain word alignments, which, it internally uses as a basis for generating phrase tables. However, it must be noted that there are several other factors which affect the overall translation quality (Bojar et al., 2010). First and the foremost is the amount and the quality of data used for training. It is equally important that the training and the test dataset are from the same domain. Another important factor that affects the translation quality is the accuracy of the language model being used. Finally, tuning an MT model is an important step which adjusts various weights for a reference translation provided to the system.

We would like to compare results of our word alignment algorithm with that of GIZA++.

We would also like to compare the effect of word alignments produced by the two systems on the overall translation quality.

In order to train a Hindi-To-English MT model, we used the WA_DATASET3. We obtained a set of 300 consecutive sentences from a random location in the dataset. We call it MT_DATASET1. We excluded these 300 sentences from the WA_DATASET3 before using the data set of MT training. Thus, we used 59401 sentence pairings for the training. We followed the instructions available at `http://www.statmt.org/moses/?n=FactoredTraining.HomePage` for a factored training. We used root forms for GIZA++ training as specified above. We used the IRSTLM library to train a trigram language model using the English sentences from the WA_DATASET3. The reordering model was trained using the *msd-bidirectional-fe* setting. Here, *msd* refers to the three different orientations: monotone, swap and discontinuous. The *bidirectional* element tells moses to model the orientation based on both the previous and next phrases. Finally the *fe* parameter tells moses that the model should be conditioned on both the source and target languages.

We used the script *multi-blue.perl* provided with moses to compare the translation output of the MT model with its reference translation and achieved 58.68 BLEU score (Papineni et al., 2002). We trained another model using the same settings as above but using the WA_DATASET2 (5986 sentence pairings in total) and obtained only 3.94 as the BLEU score. Comparing the two BLEU scores clearly suggest that the amount of data we use has a very big influence on the overall translation quality. Figure 7.2 gives an example of a Hindi sentence with its reference English translation and the two translations as obtained from the two SMT models. As can be seen in the figure, the translation obtained from the model trained on WA_DATASET3 is close to the reference translation. But there are two problems: the ordering of words is not correct and it has failed to translate the words *residential services*. When we look into the original Hindi sentence, we can see that these words have been transliterated. Comparing this to the translation obtained from the model trained on WA_DATASET2, we can see that the former is much better.

Hindi:	रैज़िडेंशल सर्विसिफ़ आम तौर पर उन लोगों के लिए हैं , जिनकी नशीली दवाओं से सम्बन्धित गंभीर व दीर्घकालिक समस्याएँ हैं ।
Reference:	residential services are usually for serious long-standing drug problems .
moses DS3:	रैज़िडेंशल सर्विसिफ़ usually for those - standing drug problems serious long .
moses DS2:	रैज़िडेंशल सर्विसिफ़ Septicaemia to those who are of drug सम्बन्धित from serious and - समस्याएँ । .

Figure 7.2: MT Output for the Models Trained Using GIZA++

Bojar et al. (2010) have conducted several experiments using the same corpus however we cannot compare our results with theirs as the MT system produced by them was for the English-to-Hindi translation. The BLEU scores reported by them are between the range of 9.5 to 11.89. Difference in the BLEU score could be due to the fact that translating into Hindi is more complicated then English because of its rich morphology and other issues such spelling errors and use of foreign words in the reference translations. Bojar et al. (2010) have addressed this question of why it is difficult to translate into Hindi.

We refer back to the BLEU scores presented here when we evaluate our English-Hindi word alignment algorithm in Section 7.7.4.

7.5 Our Approach

In this section, we give details of our word alignment algorithm. The algorithm presented in this section is a hybrid algorithm that consists of several components. First, we show a diagram explaining the steps involved in our algorithm. Then, we provide details of each step. Finally, we provide details of our experiments and detailed analysis of every individual component involved in the algorithm.

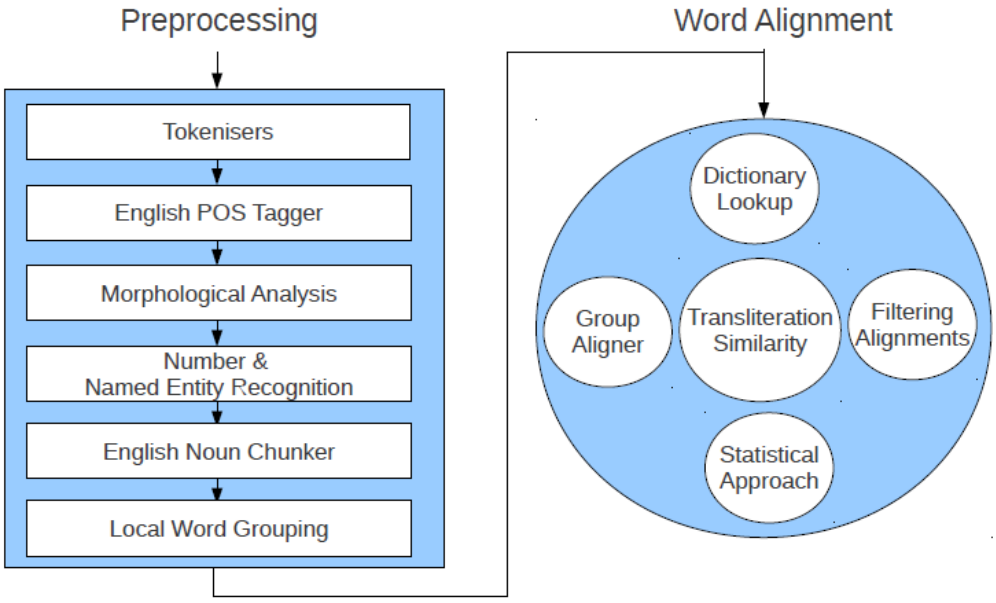


Figure 7.3: Overview of the Word Alignment Algorithm

7.5.1 Overview of the Word Alignment Algorithm

As shown in Figure 7.3, the algorithm has two modules: 1) preprocessing and 2) word alignment. Given a compound document consisting of an English text and a Hindi text, the compound document is processed by the PRs shown (see Figure 7.3). A PR, when given a compound document, may use information produced by other PRs on the same document and add new information to the document.

Assuming the data provided to the algorithm is already aligned at the sentence level, the algorithm starts with a preprocessing module. We use existing PRs in GATE to annotate specific things that are required during the word alignment stage.

- Tokenisers are used for identifying word boundaries in both the English and Hindi texts. The annotated words and symbols are considered as basic units of alignment in the word alignment stage.
- By using a POS tagger, we assign a grammatical category to each word in the English

text.

- We use the English and Hindi morphological analysers and assign a root form to every single word in both the English and Hindi texts.
- Using various PRs, we identify things such as date, numbers and named entities in the English text.
- The next step is to identify candidate word groups for multi-word alignment, i.e. groups of words in which all words are aligned to the same word or the same group of words in the other language. This PR is executed on both the English and Hindi texts.

Once the compound document has been preprocessed, it is forwarded to the word alignment stage. Here, the compound document is processed with one PR at a time. Every individual PR suggests alignments for words in the paired document and passes the document to the next PR for further processing. Please note that the order in which these PRs are executed can vary depending on the resources used for alignment. For example, one may want to execute a statistical module first if one finds that the module is producing high quality alignments. Various PRs involved in the word alignment stage of our algorithm are briefly explained below:

- Considering one sentence pairing at a time, the algorithm uses a dictionary to find potential equivalents in the target language for every word in the source language. It is possible that a source word has more than one match in the target language and thus the source word is aligned to all the matching words in the target language.
- The document is processed with the Transliteration Similarity PR which is useful for aligning named entities, cognates and phonetically transliterated words.
- It is possible that words in the English text are aligned to more than one word in the Hindi text. In order to choose correct alignments and remove any incorrect alignments, we apply a filtering algorithm.

- We use a component called the *Group Aligner* which aligns the groups that were identified by the Local Word Grouping PR during processing.
- We use a simple statistical method to align the remaining unaligned words.

Certain PRs such as the Transliteration Similarity PR and the Dictionary Lookup PR are executed more than once with different parameters. For example, the Transliteration Similarity PR is executed three times with different similarity thresholds in each iteration, i.e. 80%, 70% and 60%. Such a setting allows the algorithm to capture highly similar words prior to considering the words with lower similarity scores which may be better aligned by another module. Also, every word alignment PR we have developed has a parameter to specify if the words already aligned by other PRs should be considered for further possible alignments or not. For example, the Statistical Method PR is only used for aligning words that are not aligned by any other PR.

Preprocessing of the Data

In the following subsections, we give details of preprocessing components that are used for preparing the data for the word alignment algorithm.

Tokenisers, POS Tagger and Morphological Analysers

Tokenisers are used for identifying word boundaries and annotating them as individual tokens. They separate words from spaces and other special symbols. GATE has tokenisers for both the English and Hindi languages.

GATE has a part-of-speech tagger for the English language. We use this component to obtain the grammatical category for each of the words in the English text. Although, the grammatical information is never used by our word alignment algorithm, it is required by the GATE English morphological analyser.

We use the GATE English morphological analyser to obtain a root value for every word in

the English text. As explained in Chapter 5, we have developed a morphological analyser for the Hindi language. We use this component to obtain a root value for every word in the Hindi text.

Named Entity Recogniser

GATE has a built-in IE component set called *ANNIE* which stands for *A Nearly New Information Extraction System*. These components can be used for extracting information such as person names, organizations, and locations in the text. Usually, such entity names are not translated but written in the foreign script and pronounced in the same or similar way. The ANNIE application uses a mix of gazetteers and JAPE grammars to identify such entities. For example to identify person names, one of the gazetteers contains a list of titles such as *Mr., Mrs., Dr., Miss* etc. When such titles are followed by words in *upperInitial* or *UPPERCASE*, the sequence is annotated as a Person. Also, GATE has gazetteers with lists of well-known person names, company names and locations that help identifying mentions of them in the text. We annotate such mentions in the English text as *NamedEntity* and use them later in the word alignment algorithm.

Numbers Recogniser

In case numbers are written in words, they too need special treatment. With the help of gazetteers and JAPE grammars, we annotate numbers written in words in Hindi. We create a new annotation on each such number and assign it a feature with the string to expect in the English sentence. For example, तीनसो चार (*tinaso chaar*) is annotated as a number and a feature with value *three hundred four* is created.

As explained in Chapter 3, each entry in the gazetteer can be accompanied by a feature name and a value. When the recogniser locates any gazetteer entry in the document text, it creates an annotation and assigns the associated feature name and value to it.

Figure 7.4 shows the first of the three gazetteer lists used for annotating numbers. If in the

Hindi	Exp English	Hindi	Exp English	Hindi	Exp English
एक	one	छत्तीस	thirty six	इकत्तर	seventy one
दो	two	सैंतीस	thirty seven	बहत्तर	seventy two
तीन	three	अड़तीस	thirty eight	तिहत्तर	seventy three
चार	four	उनचालीस	thirty nine	चौहत्तर	seventy four
पांच	five	चालीस	forty	पचहत्तर	seventy five
छः	six	इकतालीस	forty one	छिहत्तर	seventy six
सात	seven	बयालीस	forty two	सतहत्तर	seventy seven
आठ	eight	तितालीस	forty three	अठहत्तर	seventy eight
नौ	nine	चौवालीस	forty four	उन्नासी	seventy nine
दस	ten	पैंतालीस	forty five	अस्सी	eighty
ग्यारह	eleven	छियालीस	forty six	इकासी	eighty one
बारह	twelve	सैंतालीस	forty seven	बयासी	eighty two
तेरह	thirteen	अड़तालीस	forty eight	तिरासी	eighty three
चौदह	fourteen	उनचास	forty nine	चौरासी	eighty four
पन्द्रह	fifteen	पचास	fifty	पचासी	eighty five
सोलह	sixteen	इकावन	fifty one	छियासी	eighty six
सतरह	seventeen	बावन	fifty two	सत्तासी	eighty seven
अठारह	eighteen	त्रेपन	fifty three	अठासी	eighty eight
उन्नीस	nineteen	चौवन	fifty four	नवासी	eighty nine
बीस	twenty	पचपन	fifty five	नब्बे	ninety
इक्कीस	twenty one	छप्पन	fifty six	इकानवे	ninety one
बाईस	twenty two	सत्तावन	fifty seven	बानवे	ninety two
तेईस	twenty three	अठावन	fifty eight	तिरानवे	ninety three
चौबीस	twenty four	उनसठ	fifty nine	चौरानवे	ninety four
पच्चीस	twenty five	साठ	sixty	पंचानवे	ninety five
छब्बीस	twenty six	इकसठ	sixty one	छियानवे	ninety six
सत्ताईस	twenty seven	बासठ	sixty two	सत्तानवे	ninety seven
अठाईस	twenty eight	तिरसठ	sixty three	अठानवे	ninety eight
उनतीस	twenty nine	चौसठ	sixty four	निन्यानवे	ninety nine
तीस	thirty	पैंसठ	sixty five	सौ	hundred
इकतीस	thirty one	छियासठ	sixty six	हजार	thousand
बत्तीस	thirty two	सरसठ	sixty seven	लाख	lakh
तैंतीस	thirty three	अड़सठ	sixty eight	करोड़	crore
चौंतीस	thirty four	उनहतर	sixty nine	अरब	arab
पैंतीस	thirty five	सत्तर	seventy		

Figure 7.4: Gazetteer List to Match Numbers

Hindi document, the recogniser can locate one of the entries mentioned in the gazetteer list, the text is annotated as *Lookup*. A feature called *majorType* with value *number* is assigned to it. The second and third gazetteer lists match associated number prefixes and suffixes. Each entry in these gazetteers also consists of a feature and a value assigned to it. Figure 7.5 lists the entries in these gazetteers. If any entry from these gazetteers is found in the Hindi document, an annotation of type *Lookup* is created with a feature called *majorType* with value *numpre* for the number prefixes and *numsuffix* for the number suffixes.

Number Prefixes		Number Suffixes			
Hindi	Expected English	Hindi	Expected English	Hindi	Expected English
पह	fir	वां	th	जी	nd
दु	seco	वीं	th	जा	nd
तिस	third	वें	th	जे	nd
चौ	four	ला	st	था	th
छठ्	six	ली	st	थे	th
चौद	fourteen	ले	st	थी	th
सत्तर	seventeen	री	nd	ठा	th
		रा	nd	ठे	th
		रे	nd	ठी	th

Figure 7.5: Gazetteer List to Match Number Prefixes and Suffixes

Once these annotations are created, we use a small JAPE grammar that combines these annotations and creates an annotation called *Number* with a feature called *english* with the value that we expect to find in the English text.

For example, consider the following Hindi string:

तीन हजार चारसो सात
(*tIna (three) hajAra (thousand) chAraso (four hundred) sAta (seven)*)

Given such a text, तीन (*tIna*) is annotated as {*Lookup majorType='number' english='three'*}, हजार (*hajAra*) as {*Lookup majorType='number' english='thousand'*}, चार (*chAra*) as {*Lookup majorType='number' english='four'*}, सो (*so*) as {*Lookup majorType='number' english='hundred'*} and सात (*saat*) as {*Lookup majorType='number' english='seven'*}. All the *Lookup* annotations except the ones on the string चारसो (*chArso*) are converted into *Number* annotations and the feature *english* is copied. For the string चारसो (*chArso*), where the token has two *Lookup* annotations without any space between them, the two annotations are concatenated and a single annotation called *Number* with the feature *english* and value *four hundred* is created.

Another example:

तीनसो चौदवां (*tInaso (three hundred) chaudavA.n (fourteenth*)

Given this text, तीन (*tIna*) is annotated as {*Lookup majorType='number' english='three'*}, सो (*so*) as {*Lookup majorType='number' english='hundred'*}, चौद (*chauda*) as {*Lookup majorType='numpre' english='fourteen'*} and वां (*van*) as {*Lookup majorType='numsuffix' english='th'*}. When concatenating them together, we get two annotations: {*Number english='three hundred'*} over the first token and {*Number english='fourteenth'*} over the last token.

English Noun Chunker

As part of the preprocessing step, we also annotate noun phrases using the *Tagger_NP_Chunking* plugin available in GATE. The plugin is a Java implementation of the algorithm for NP Chunking proposed by Ramshaw and Marcus (1995). The plugin has a set of rules where each rule can be a sequence of one or more part-of-speech tags and actual word strings. In other words, if in a document such a sequence can be located, it is annotated as a *NounChunk*. There are approximately 2000 rules. We use these annotations later in the word alignment algorithm.

Local Word Grouping

Local Word Grouping (LWG), as the name suggests, is performed based on the information collected from adjacent words. It is useful for identifying multi-word expressions (MWEs). MWEs are often aligned together as a single unit to a word or phrase in the target language.

Saha et al. (2008) describe a named entity recognition system for the Hindi language. Their work is based on orthographic features, common suffixes and prefixes, gazetteer lists and information about surrounding words. In order to learn patterns for local word grouping, they collect some seed entities and consider n words in their left and right contexts. They search the entire corpus for these patterns and drop the ones with low frequency. In order to generalize these patterns, they drop tokens in the context. Kunchukuttan and Damani (2008) suggest that MWEs can be recognised based on their linguistic characteristics. They suggest that there are four types of compound words in Hindi which involve bigrams. These

are:

1. Onomatopoeic expressions where both the words in a compound word intimate the same sound (e.g. खन खन (*khan khan*, just a type of sound, does not mean anything))
2. Complete reduplication where the individual words are meaningful (e.g. कदम कदम (*kadam kadam*, on every step))
3. Partial reduplication where one of the words is meaningful and the other one is partially copied (e.g. रंग बिरंगा (*rang biranga*, colourful))
4. Semantic reduplication where the words are from same semantic group (e.g. चाय पानी (*chAya paani*, tea and water)).

While preparing the English-Hindi word alignment development dataset (see Chapter A), we collected all multi-word alignments and counted frequencies for words that occur at the start, end and in the middle of multi-word alignments. We used only the words with frequency above 10 and formed rules. The LWG component is also useful for tagging words with their POS tags. This is achieved by using information about common suffixes used for inflecting verbs, nouns and adjectives (see table 5.2 in Chapter 5).

Rules

Word groups in Hindi and English texts are created using a rule file (separate files for English and Hindi). Every rule in the rule file is a three-tuple of form: $<PTM,AT,KWI>$ where,

1. *PTM* stands for *Pattern To Match* and must be composed of one or more of the following elements:
   - a string containing one or more words.
   - . – '.' represents any character other than a white space.
   - ( ) – round brackets are used for grouping a pattern.

- | – '|' represents the logical OR operator.
- + and * – '+' and '*' are Kleene operators. They are appended to a group declaration – '+' means one or more occurrences of the pattern specified in the group that precedes the '+' operator. Similarly, '*' means zero or more occurrences.
- $[X][X]$ – 'X' represents any single word. This pattern has a special meaning – i.e. when specified as it is, it tries to find a bigram where the first and the second words are identical (Kunchukuttan and Damani, 2008). For example, this pattern can match the Hindi bigram अलग अलग (*alag alag, different*).
- $[AnnotationType]$ – the word specified between '[' and ']' must be an annotation type that has been declared somewhere else in the rule file. It matches any string in the text that has been annotated with the annotation type specified here between the square brackets.

2. *AT* stands for annotation type. Any string that matches the *PTM* is annotated with the annotation type specified here.

3. *KWI* stands for *Key Word Index* and is a positive integer value. The word at the specified index is said to be the keyword in the group. It is possible to have more than one keyword in a group. Use of the keyword index is explained later in this section.

Below we list a couple of example:

Rule 1:
(.)+ (रहा|रहे|रही) VP 1

Rule 2:
[VP](था|थी|थे) VP 0

In the first example rule above, (.)+ indicates a word with one or more non-whitespace characters. If such a word is followed by one of the three words रहा(*rahaa*), रहे(*rahe*) and

रही(*rahi*), both the first and second words are annotated together as a single *VP*. Also a feature *kwi* with value *1* is assigned to the annotation.

In the second example rule above, we are trying to match a string where the first part of the string is already annotated as VP and is followed by one of the three words था(*thaa*), थी(*thi*) and थे(*the*). If such a pattern is found, any existing annotations over the matching text or over any part of the matching text are deleted and a new annotation covering the entire span is created. In doing so, any existing *KWI*s are adjusted such that the key word index still refers to the original word it was referring to in the deleted annotations. The *KWI* 0 indicates that there are no additional keywords found in the pattern.

7.5.2 Word Alignment Algorithm

In the following subsections we explain the components which are part of the word alignment algorithm.

Dictionary Lookup and Morphological Analyser

The use of a bilingual lexicon is the most common method for any hybrid or lexical approach. One difficulty in using a bilingual dictionary is that most dictionaries have their entries in base-form. Searching for inflected words in the dictionary can lead to a very low recall.

If the English entries in a bilingual dictionary are in base-form, the associated Hindi entries are in their base-forms too. As described earlier, we used the English morphological analyser from GATE to process English words and return their base-forms. Thus, before an English word is searched for in the dictionary, the value of its *root* feature is obtained and looked up in the dictionary. We use the Hindi morphological analyser (see Chapter 5) process Hindi words. Here, we present a few examples of Hindi inflected words, directly copied from our test data, which could not have been aligned if we had not obtained their root forms.

- पहचानने (*pahachAnane, to recognise*) can be found in the dictionary as पहचानना (*pahachAnanA*)

- समज़ (*samaza*), if searched as it is, it means understanding but if a root form of it is obtained, which is समज़ना (*samazanA*), it means to understand.
- चुनोतियों (*chunOtiyon, challenges*) cannot be found in the dictionary. However, its singular form चुनोती (*chunOti, a challenge*) can be found in the dictionary.
- जांचेंगे (*jAnchenge, will search*) – on processing this word with the Hindi morphological analyser, it returns two root forms: 1) जांच (*jA.ncha*) which means "investigation" 2) जांचना (*jA.nchanA*) which means "to investigate" in English.
- मशीनो (*mashIno, machines*). This is an example of an English word being pluralised using a Hindi suffix. Since we have a rule learnt to deal with such words, our morphological analyser analyses this word correctly and returns मशीन (*machine*) as the correct root form.

Where the alignment methods mostly rely on dictionaries, missing entries in dictionaries can result in low recall. While users were aligning words in the development dataset, we set up a task in the background to collect 1:1 alignment pairs that did not appear in the dictionary. Since each document was aligned by at least two annotators, we collected only those entries that were chosen by all annotators. We also maintained a frequency table for each entry. Finally those entries that received votes from all their annotators and had high frequency were added to the dictionary. More information about updating the English-Hindi dictionary is provided in Chapter A.

The dictionary lookup approach does not refer to the context of a particular word prior to deciding on the best match. Sometimes it is possible that a word is translated into a phrase. Similarly it is possible that the same target word appears twice in the target sentence. It is also possible that the translation of a given English word does not produce any match in the Hindi text but the translation of a synonym does. Thus, not only do we align the source English word with its all possible alignments in the target text, but we also perform lookup in a synonyms list[4] of the English word and obtain translations of all the synonyms from

[4]The synonym list for our experiments have been downloaded from the English WordNet.

the dictionary. If any of the translations appears in the target sentence, the English word is aligned with it.

The dictionary lookup approach is also responsible for aligning numbers written in words. For every *Number* annotation created earlier, we search for its expected string in the English text. If found, the Number annotation is aligned with the respective words in the English text.

As specified earlier, it is possible that a PR in our word alignment algorithm is used on more than one occasion. The Dictionary Lookup PR is executed two times. First, it is executed at the very beginning of the word alignment stage. It aligns all possible words in the source language with all possible translations found in the dictionary and in the parallel target language sentence(s). Second, the Dictionary Lookup PR is executed towards the end of the word alignment stage. In between the two executions of the Dictionary Lookup PR, it is possible that a PR such as *Filtering Alignment PR* (explained later) unaligns few words which were originally aligned during the first iteration of the Dictionary Lookup PR. The second execution of the Dictionary Lookup PR finds translations of the source words which were unaligned by the Filtering Alignment PR and have not been aligned by any other PRs.

Transliteration Similarity

The purpose of the transliteration similarity component is to align three types of words: Named Entities, cognates and the phonetically transliterated words.

Given a pair of English-Hindi words and a list of English-Hindi character mappings, the transliteration similarity component (see Chapter 6) obtains a set of candidate transliterations from the Hindi word. These candidate transliterations are then compared with the English word. The comparison is made to obtain a measure which indicates how similar the two words are. If the similarity is above a set threshold the two words are aligned with each other. The process repeats until every English word in the source language sentence is compared with every Hindi word in the target language sentence.

The Transliteration Similarity PR is executed three times with a different similarity threshold in its each iteration. The Transliteration Similarity PR gives the best outcome when the similarity threshold is set to 79% (see Chapter 6). Therefore, during the first iteration, we set its similarity threshold to 79%. Since we know that the PR is very reliable at this threshold, we allow it to consider even the words that have been already aligned by preceding PRs in the word alignment pipeline and suggest new possible alignments for these words.

The similarity threshold values are reduced to 70% and 60% during the second and third iterations respectively. Also, we only consider unaligned words during the second and third iterations. This setting allows the PR to align word pairs that are highly similar during the first iteration and suggest alignments for words with less similarity (during the second and third iterations) only if none of the preceding PRs can align them.

As explained earlier, we use the ANNIE application to identify named entities in the English text. This is where we refer to the annotations of type *NamedEntity* and try to match them with the words in the target language. Since the NEs recognised by the ANNIE application have high precision, we allow NEs to match in the target text even with a similarity score of 60% (the third iteration).

Filtering Alignments

It is possible that a word has multiple alignment possibilities in the target text and one should choose the correct one. Both the dictionary lookup and transliteration similarity components mark all possible alignments in the source and in the target language. In other words, every English word is aligned with every possible Hindi word that these methods suggest it should be aligned with. However, one has to decide which of the marked pairs are correct and which ones to delete. This component of the algorithm does exactly that.

Distortion in statistical MT refers to the likelihood of a word having a different position in the target sentence to the source language sentence. The initial assignment of alignments (as explained above) is based on the assumption that the translation of a particular word in the

source sentence can be anywhere in the target sentence. On the other hand the observation that alignments tend to respect locality has been discussed in the literature (Ananiadou, 2011). This means that neighbouring words in the source language are often aligned with neighbouring words in the target language. In order to find out if this is true for the English-Hindi language pair, we did some experiments to record jump width (distance) between the alignments of neighbouring words in the source language. The aim of these experiments was to check if the words in English that generate neighbouring Hindi words are likely to be nearby or not. To set the basis for our experiments, we ask three questions.

1. Where referring to neighbours, how do we decide which neighbours to choose from? – i.e. adjacent neighbours?, on the left or right side? or both?, how many of them?, etc.
2. Given a choice of different word pairings, and their distance from their neighbours' word alignments, how can we decide that the word pair with the least distance should be chosen?
3. What if the neighbours' alignments are wrong?

We ran our experiment on the word alignment development dataset which were manually aligned at the word level.

For every aligned word at position n in the source sentence (English), we considered its neighbouring word at position *n + 1*. For both of these words, we obtained their alignments in the target sentence (Hindi) and calculated the distance (in number of words) between these aligned words. Similar experiments were carried out for the neighbouring words at position *n-3, n-2, n-1, n+2* and *n+3*. Table 7.4 shows some very interesting results obtained from this experiment:

We consider one aligned word at a time. Thus, for a given word, the value in the first column is the distance between the current word and its neighbour (either on the left or right) in the source sentence. Then we look at the alignments of the current word and the alignments

Table 7.3: Statistics for Adjacent Words Alignment

N. Position	**Distance**	**%Frequency**	**Aggregated Frequency**
1	1	16.06	16.06
1	2	4.87	20.93
2	1	9.8	30.73
2	2	8.98	39.71
3	1	6.78	46.49
1	3	3.16	49.65
2	3	4.17	53.82
3	2	4.52	58.34
3	3	4.12	62.46

of the neighbour word. The second column shows the shortest distance between the aligned words of the current word and the aligned words of the neighbour word. The third column shows how many times (in percentages) we found the same distance between the alignments of the current word and the alignments of the neighbour word. Finally, the fourth column shows the aggregated total of frequencies (in percentages) as we move towards the bottom of the table.

For example, the table shows that on 16 occasions out of every 100 occasions, the alignment of an adjacent word was found within a distance of 1 word from the positions of the alignments of the current word. Nearly 62% of the neighbours' alignments (within the range of 3 words) appear within a distance of 3 words from the positions of the aligned words of the current word.

Based on these figures, it would be fair to conclude that even though the Hindi language is a free order language, the assumption of locality does apply to it at some extent.

Thus, when a word is aligned with more than one word in the target sentence, one could look at the alignments of its neighbours (within the range of 3 words) and choose the alignment for the current word that is nearest to its neighbours' alignments. Thus, the alignment

chosen this way would carry 62% confidence with it.

In order to find out if it was the least distance that should be used to decide which pairs to align with, we once again look at the Table 7.4. As can be seen, if the neighbouring word is at position *n + 1*, the frequency is maximum for the distance 1. The frequency decreases as the distance increases (e.g. for distance 2 it is 4.87 and for distance 3 it is 3.16). The same is true for neighbouring words at position n +2 and words at position n + 3 as well. Thus, if the alignment is chosen with the least distance, our experiment suggests that it will be the right thing to do.

Finally, the third question – what if the neighbours' alignments are wrong? The methods which are executed before this particular component in the word alignment algorithm, i.e. Dictionary Lookup, Transliteration Similarity, etc. – are all based on knowledge-rich resources. Thus most of the alignments they identify, they identify with very high accuracy. As a consequence, the probability of neighbours being incorrect is very low.

However, there is still a problem with this method. What happens when a neighbouring word has multiple matches and the distance between more than one of these matches and the current word's alignments is the same? In other words, there is one neighbour with more than one alignments each of which is same distance from current word's aligned words.

In this case, instead of referring to just one pair at a time, we consider all the pairs with more than one possible alignment. We calculate the maximum likelihood estimate by finding a combination of alignments with the least overall distance. We explain the entire process through an example.

As shown in Figure 7.6, the word *better* is aligned with बेहतर (*behatara, better*), *performance* is aligned with काम (*kAma, business, work*) and व्यापार (*vyApAra, business*), *in* is aligned with के लिए (*ke liye, for*) and में (*mein, in*), *business* is aligned with व्यापार (*vyApAra, business*) and काम (*kAma, business, work*), *in* with के लिए (*ke liye, for*) and में (*mein, in*), *turn* with के लिए (*ke liye, for*) and में (*mein, in*), *benefits* with लाभप्रद होता है (*lAbhaprada hotA hai, beneficial*) and finally the word *consumers* is aligned with उपभोक्ता (*upabhoktA, consumers*).

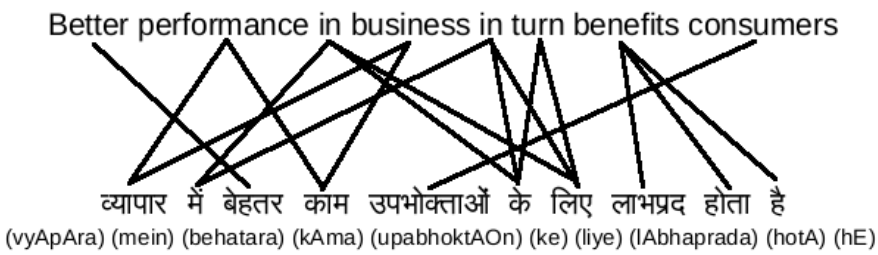


Figure 7.6: Word Alignments Before the Filtering Alignment Component is Executed

In this example, there are in total seven adjacent word pairs: *better performance*, *performance in*, *in business*, *business in*, *in turn*, *turn benefits* and *benefits consumers*.

Having identified adjacent words in the source language with one of them having multiple alignment possibilities in the target sentence, we compute distances between the alignments of the adjacent words. The distance is calculated in terms of number of words. The same step is repeated for all such pairs. Table 7.4 shows the calculated distances.

As can been seen, the pairs are categorized in different groups separated by horizontal lines in Table 7.4. Given these different groups, we generate various combinations. The idea here is to take one member from each group and generate a combination. Thus, given that there are six groups with 2, 4, 4, 4, 2, and 2 possibilities in them respectively, 512 unique combinations (i.e. 2x4x4x4x2x2) can be generated. However, we introduce a couple of constraints which reduces the number of combinations.

1. Remove combinations in which the same word is referring to different alignments. For example, look at the pairs B and C. In this case, the word *performance* has two possible alignments in Hindi काम (*kAma*) and व्यापार (*vyApAra*). Similarly look at the pairs K and O. In this case the word *in* has two possible alignments in Hindi के लिए (*ke liye*) and में (*men*).

2. Remove those sequences in which two non-consecutive source words are referring to the same alignment in the target sentence. For example, look at the pairs A and K. In this case, the English words *performance* and *business* both refer to the same Hindi

Table 7.4: Distance Between the Alignments of Adjacent Words

Label	Source Word	Aligned Words	Source Word	Aligned Words	Distance
A	performance	व्यापार	better	बेहतर	2
B	performance	काम	better	बेहतर	1
C	in	में	performance	व्यापार	1
D	in	में	performance	काम	2
E	in	के लिए	performance	व्यापार	5
F	in	के लिए	performance	काम	2
G	business	व्यापार	in	में	1
H	business	व्यापार	in	के लिए	5
I	business	काम	in	में	2
J	business	काम	in	के लिए	2
K	in	में	business	व्यापार	1
L	in	में	business	काम	2
M	in	के लिए	business	व्यापार	5
N	in	के लिए	business	काम	2
O	turn	के लिए	in	के लिए	0
P	turn	के लिए	in	में	4
Q	benefits	लाभप्रद	turn	के लिए	0
R	benefits	लाभप्रद	turn	में	6

word व्यापार (*vyApAra*) in the target sentence. Although, having such alignments is not impermissible, our analysis of the development dataset suggests that it is less likely to find such alignments.

Having removed combinations that are exceeded by the above constraints, the idea is to sum up the distances of pairs in each of the remaining combinations and locate the one with the least overall distance. We use dynamic programming to find such a combination with the least overall distance.

Initially, we generate a combination by taking one member from each group. We record the overall distance of this combination. Later, as we generate a new combination, we keep track of the overall distance after every addition of a new member to the combination. As soon as the overall distance of a new combination has exceeded the least overall distance recorded for a full combination so far, the combination in making is discarded. If it happens that the overall distance of a newly generated combination is less than the distance of the previously recorded combination, we keep the newly generated combination and discard any previously generated combinations.

Table 7.5 lists some of the low score combinations.

Table 7.5: Combinations for Filtering Alignments

Member	**Member**	**Member**	**Member**	**Member**	**Member**	**Total**
A-2	C-1	I-2	N-2	O-0	Q-0	7
B-1	D-2	G-1	M-5	O-0	Q-0	9
A-2	C-1	I-2	L-2	P-4	Q-0	11
A-2	E-5	J-2	N-2	O-0	Q-0	11
B-1	F-2	H-5	K-1	P-4	Q-0	13
B-1	F-2	H-5	M-5	O-0	Q-0	13
A-2	E-5	J-2	L-2	P-4	Q-0	15
A-2	E-5	H-5	K-1	P-4	Q-0	17
A-2	E-5	H-5	M-5	O-0	Q-0	17

As can be seen, the combination *ACINOQ* returns the lowest distance of 7 and therefore, the alignment is chosen. The Figure 7.7 illustrates the final alignment. Even though the alignment proposed by the method is largely OK, the words *performance* and *business* should have been aligned with the words काम (*kAma*) and व्यापार (*vyApAra*) respectively. Nevertheless, our evaluation show that the use of this component increases the overall precision of the system by almost 14%.

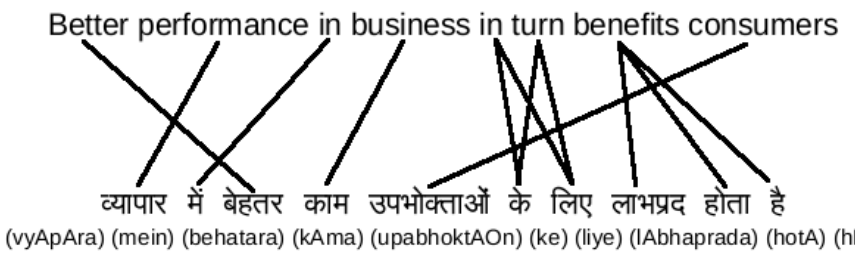


Figure 7.7: Word Alignments After the Filtering Alignment Component is Executed

Group Aligner

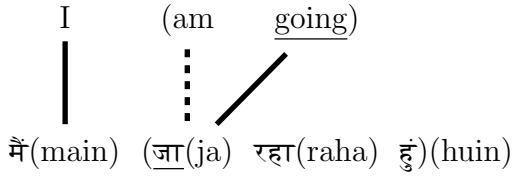
As the name suggests, the purpose of this component is to align groups. As explained earlier, the LWG component is responsible for identifying potential word groups in both the English and Hindi texts. While forming these groups, LWG assigns a feature *kwi* to every group annotation. The value of this feature refers to the word that serves as a head verb or noun for that group. Below we list different scenarios and how the Group Aligner component reacts to them. Given a group that has been identified by the LWG component, the Group Aligner component:

1. does nothing, if none of the keywords in the group have been aligned by one of the preceding components.
2. does nothing, if one of the non-keyword words in the group has been aligned.
3. aligns the entire group with the words that the keywords are aligned to.

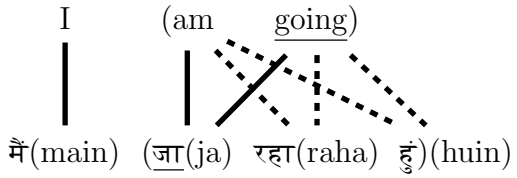
The Group Aligner component is executed in both directions (i.e. source to target and target to source). This makes it possible to align one group with the other. Below we illustrate this with an example.

Please note that the words within brackets indicate groups that are created by the LWG component. Keywords in each group are underlined. The links that are generated by the LWG component are dashed.

Results of executing Group Aligner in the source-to-target direction



Results of executing Group Aligner in the target-to-source direction



Statistical Approach

It is possible that even after executing all the above mentioned components, there are certain words which are not aligned. To align the unaligned content words, we use a simple statistical approach based on the work presented by Melamed (2000). Our method is essentially the same as the one proposed by Melamed (2000) with two differences. First, we use Dice's coefficient measure as opposed to the G^2 measure (Dunning, 1993) used by them. Secondly, we introduce a frequency constraint (see below).

We determine if an English word is a content word or not by looking up its grammatical category. If the grammatical category is one of the noun, verb, adjective and adverb, we consider the word as a content word. Since we do not have grammatical categories assigned to Hindi words, we consider all the unaligned words from the Hindi text.

Given a sentence aligned corpus with K sentence pairings, we consider one sentence pairing

$(e, h)_k$, where e is an English string and h is its Hindi translation. For this sentence pairing $(e, h)_k$, we obtain words ew_i and hw_j at positions i^{th} and j^{th} in e and h respectively and pair them. These words are the base-forms of the words found in the sentence pairing, obtained using the English and Hindi morphological analysers. Also, these words are not punctuation, special characters or any symbols.

Thus, we have a set of word pairs, W, consisting of (ew_i, hw_j). While creating W, we ignore all pairs that have occurred thrice or less in the entire corpus. We assume that such pairings (3 or less), do not reflect exact translations but are examples of rare usage or obsolete usage of the paired word.

Using Dice's coefficient measure, we calculate a co-occurrence score for every word pair in the corpus as,

$$Dice(ew_i, hw_j) = \frac{2 \cdot |(ew_i, hw_j)|}{|ew_i| + |hw_j|}$$

where,
(ew_i, hw_j) = is the frequency of the pair in W
(ew_i) = is the frequency of ew_i in W
(hw_j) = is the frequency of hw_i in W

We maintain a co-occurrence score table for the entire corpus that consists of a pair and its co-occurrence score using which we decide the word-alignment in a sentence pairing.

Given a sentence pairing $(e, h)_k$, we use the alignment methods discussed above (e.g. Dictionary lookup, transliteration similarity, etc.) to generate a set A_k containing word alignment pairings for the sentence pairing k.

Now, we obtain pairs $(ew_i, hw_j)_k$ from W which do not exist in A_k (i.e. all the unaligned word pairs from the sentence pair k). From this we select one or more pairs that have the

highest value in the co-occurrence table.

Such pairs are aligned and added to A_k. Before the next iteration, all pairs involving either of the words from the last aligned word is removed and the greedy search for the next pair with maximum score is continued. We iterate until all the words in the k^{th} sentence pairing are aligned or there is not a single pair left in the co-occurrence table for the current sentence pairing k that is not aligned.

7.6 Example – Putting it Altogether

In this section, we show an example sentence pairing and execute the various components of the word alignment algorithm one after the other to align the words. At every step, we give details of the component being used and provide an explanation for the outcome.

Step 1: Morphological Analysis

For both the English and Hindi sentences, their morphological analysers return their base-forms. For all the subsequent steps, we will be using their root values.

On the first of January I will buy three computers for my home from PC world

On the first of January I will buy three computer for my home from PC world

पहली जनवरी को मैं घर के लिये पीसी वर्ल्ड से तीन कम्प्युटर खरीद लुंगा

(*pahalI janavarI ko main ghara ke liye pIsI varlDa se tIna kampyuTar kharIda lungA*)

पहली जनवरी को मैं घर के लिये पीसी वर्ल्ड से तीन कम्प्युटर खरीदना लेना

Step 2: Recognizing Named Entities

This step involves identifying named entities in the text and we use GATE's ANNIE application for the same. In the following example, only the string *PC world* is recognised as a company name and surrounded by round brackets.

On the first of January I will buy three computer for my home from (PC world)

पहली जनवरी को मैं घर के लिये पीसी वर्ल्ड से तीन कम्प्युटर खरीदना लेना

Step 3: Recognizing Numbers

This step involves identifying numbers in Hindi. In this case, it recognises the string पहली as a number which is assigned a feature *english* with the value *first*. This value indicates that the expected English word for this number in the English sentence is *first*. If such a word is found in the English sentence, the Hindi word पहली should be aligned to it. Similarly, the string तीन is also recognised as a number and assigned a feature *english* with the value *three*. To make it easier to read, we will replace the Hindi numbers with their respective English strings.

On the first of January I will buy three computer for my home from (PC world)

पहली जनवरी को मैं घर के लिये पीसी वर्ल्ड से तीन कम्प्युटर खरीदना लेना

First जनवरी को मैं घर के लिये पीसी वर्ल्ड से three कम्प्युटर खरीदना लेना

Step 4: LWG

As part of this step, we create groups of tokens that may be aligned as a single unit. The groups found in the following example are surrounded by square brackets and keywords are underlined.

The LWG rules that formed these groups are listed below:

{*(.)+ of (.)+ Group 1,3*} responsible for grouping the words *First of January*
{*will (.)+ VP 2*} responsible for grouping the words *will buy*
{*(.)+* लुंगा *VP 1*} responsible for grouping the words खरीद लुंगा

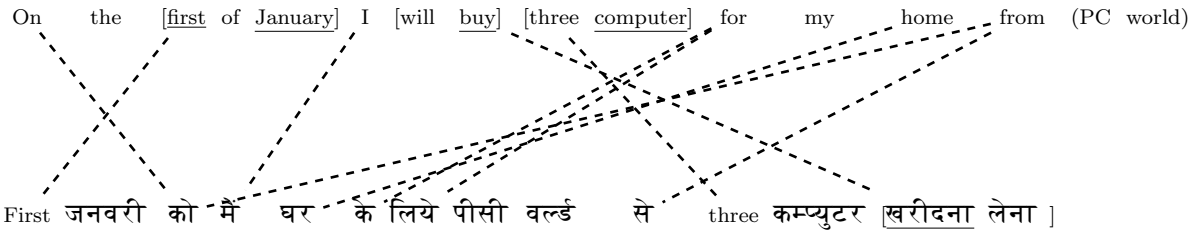
The plugin *Tagger_NP_Chunking* annotates the words *three computers* as a *NounChunk* annotation.

On the [first of January] I [will buy] [three computer] for my home from (PC world)

First जनवरी को मैं घर के लिये पीसी वर्ल्ड से three कम्प्युटर [खरीदना लेना]

Step 5: Dictionary Lookup

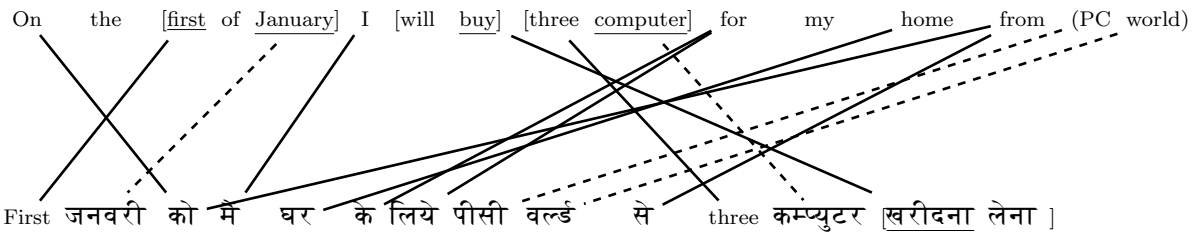
In this step, we use the English-Hindi bilingual dictionary to align words. The *Number* annotations created in the step 3 are also aligned in this step.



As can be seen, the word *from* is aligned to two Hindi words को (*ko*) and से (*se*).

Step 6: Transliteration similarity

The Transliteration Similarity component is responsible for aligning named entities, cognates and words that are phonetically transliterated. As discussed in Chapter 6, words are said to be transliterations of each other only if at least two of the three methods *TSM*, *DC* and *Jaro-Winkler* give them similarity above 79%. In other words if the similarity obtained is less than 79%, the word pair under consideration is not aligned.



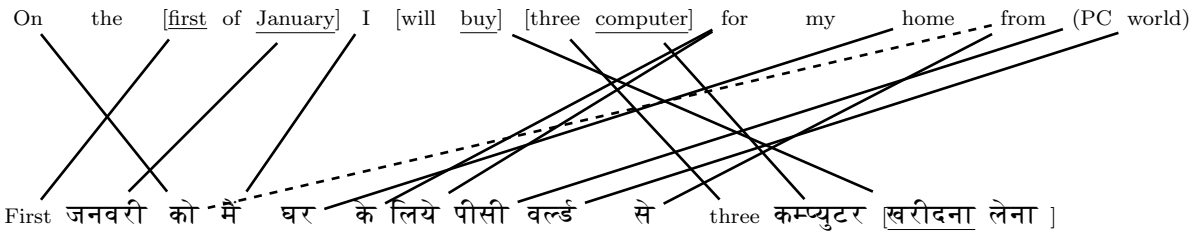
Step 7: Filtering Alignments

In this step, we look at the words for which there are more than one alignment in the target sentence and try to preserve the one that is most suitable according to method discussed in

Section 7.5.2. In the following example, there is only one English word *from* for which there are two possible alignments.

from = को
from = से

For each Hindi word in the above cases (call it *H1*), we first find out the aligned English word's neighbour whose aligned word is closest to the Hindi word. Let's call this aligned word *H2*. Then we calculate the distance between *H1* and *H2* for each case. In the case of the Hindi word को (*ko*), the neighbour whose aligned word is closest to it is *home* and the aligned word is घर (*ghar*). The distance between the words को (*ko*) and घर (*ghar*) is 2 words. On the other hand, the neighbour whose aligned word is closest to the Hindi word से (*se*) is *world* which is aligned to वर्ल्ड (*world*) and the distance between the words से (*se*) and वर्ल्ड (*world*) is 1. Therefore we choose the latter alignment. Thus it is able to remove the incorrect alignment successfully. In the following figure, we show the link that needs removing as a dashed line.



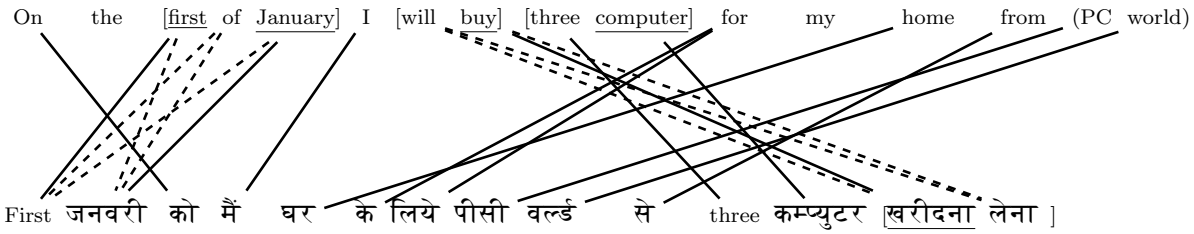
Step 8: Group Aligner

There are three groups in the English sentence and one in the Hindi sentence. The first group *first of January* has two keywords, *first* and *January*, which are aligned to the Hindi words, पहली (*pahali*) and जनवरी (*janvari*) respectively. These Hindi words are adjacent words. Therefore, the words of the group *first of January* should be aligned altogether with the words पहली जनवरी(*pahali janvari*).

The second group in the English sentence is *will buy*. The keyword *buy* is aligned to the Hindi word खरीद (*kharid*). Here, the Hindi word खरीद (*kharid*) is a keyword in the Hindi

group खरीद लुंगा(*kharid lunga*). Thus, the groups can be aligned to each other.

The group *three computers* has all its words aligned and therefore we do not need to take any action for this group.



Step 9: Statistical Approach
In this step we align the remaining unaligned content words. The two words, *my* and *the*, are not aligned in the English sentence. As none of them is a content word, the statistical method does not align them.

7.7 Evaluation

In this section, we give details of our experiments with the English-Hindi word alignment algorithm described in the previous sections. First, we give details of the evaluation measures used for evaluating the word alignment results. Secondly, we provide details of the test data used in our experiments. Finally, we provide detailed evaluation of our algorithm and its components.

7.7.1 Evaluation Measures

Evaluation of a word alignment system can be performed with two different traditional measures: precision and recall. We calculate the precision, recall and F-Measure. Precision (P) is calculated as the ratio of the correctly identified pairs divided by the number of pairs identified by the system. Recall (R) is calculated as the ratio of correctly identified pairs divided by the total number of pairs in the sample data. The F-Measure (F) is calculated to obtain the weighted harmonic mean of precision and recall. The weight is equally distributed

between precision and recall and therefore the equation used is:

$$F = \frac{2 \times P \times R}{(P+R)}$$

Some researchers use a measure called AER which stands for Alignment Error Rate. As the name suggests it is useful for measuring the errors that an alignment system makes. It is calculated using the following equation:

$$AER = 1 - F;$$

Null and Partial Alignments

Some systems do not consider *null* alignments while others do. According to Tufiş et al. (2003), null alignments play a vital role in the evaluation. Their experiments with a Romanian-English gold standard show that about 13.35% of the total number of alignments were null.

Our results show that almost 23% of the total number of word alignment pairs were null alignments. In general where null alignments are not taken into consideration, they are removed from both the test results and the gold standard. In our case, we do consider the null alignments.

It is also important to consider partial alignments. As described in Ahrenberg et al. (2002), the advantage of considering partial alignment is that the successful linkings of multi-word units are visible and rewarded but also partially correct linkings are not discounted entirely. The approach has been used by many (Melamed, 1998b; Och and Ney, 2000; Mihalcea and Pedersen, 2003; Martin et al., 2005) and is a well-established method of reporting word alignment results. A consequence of converting a multi-word unit into many 1:1 alignments is that the number of alignment pairs increases, which certainly has an impact on the evaluation measures precision and recall[5].

Where a source word is aligned to more than one word in the target language, we create

[5] `http://stp.ling.uu.se/~joerg/phd/html/node6.html`

multiple 1:1 alignments. For example, suppose words *e1* and *e2* in English are aligned to words *h1* and *h2* in Hindi. Instead of representing the pair as *e1,e2 = h1,h2*, four 1:1 pairs are formed: *e1 = h1*, *e2 = h1*, *e1 = h2* and *e2 = h2*. The proposed arrangement respects the partial alignment. From this, we calculate precision and recall measures.

7.7.2 Test Data

With the help of four Hindi native speakers, we were able to prepare a word alignment gold-standard for 300 documents. Each document in this collection contains approximately 10-15 sentence pairings. The guidelines used for preparing the word alignment gold-standard are provided in the Chapter 2.

Out of the 300 documents, 100 documents were used as a development dataset to update various English-Hindi resources (see Chapter A) and the remaining 200 documents were used as test data (same as the WA_DATASET1 as explained in the Section 7.4) to evaluate the word alignment algorithm.

The test data contains 2029 sentence pairings with 30,672 English words and 34,121 Hindi words in total.

7.7.3 Component Evaluation

The word alignment algorithm described in this report is a hybrid system that consists of several different components. These components are responsible for aligning different types of words. For example, the Transliteration Similarity (TS) method aligns words that are phonetic transliterations of each other. Similarly, the Group Aligner component aligns groups resulting in many-to-many alignment.

Ordering of PRs in the Word Alignment System

One could question the order in which these components are executed. In other words, how did we decide in which order the components should be executed. Before providing details on this, it is important to mention that all our PRs have a parameter which decides whether the respective PR should attempt to align only the unaligned words or all the words without

considering their alignment status.

In our word alignment system, the first two PRs are the Dictionary Lookup PR and the Transliteration Similarity PR. The Dictionary Lookup PR uses both the English-Hindi dictionary and the custom dictionary. The Transliteration Similarity PR is executed with the similarity threshold set to 79%. Both of these PRs are allowed to align every word from the source language to every possible match they can find in the target language. The Group Aligner PR and the Filtering Alignments PR require that at least some words are already aligned before they can bring any improvement to the overall word alignment. The remaining PRs only consider the words which are not aligned already.

As the Dictionary Lookup PR is very much language specific and the Transliteration Similarity PR is set to use a very high similarity threshold, the alignments produced by these two PRs should be of very high quality. Therefore they should be executed at the beginning of the pipeline. Because they align words without considering their alignment status (aligned or unaligned), the order in which the two of these are executed does not make any difference.

To decide the order in which all the other PRs should be executed, we ran an experiment on the word alignment development dataset. The PRs were programmatically rearranged in different orders and the results were collected for every permutation. The order in which the PRs produced the best results on the development dataset was chosen as the order of the PRs in the final word alignment pipeline.

Overall Results of the Word Alignment System

The overall results are calculated in terms of precision and recall. We obtained 0.90 (precision), 0.68 (recall), 0.78 (F-Measure) and 0.22 (AER). When we compare our results with the results obtained through GIZA++ experiments on the same data set, i.e. 0.63 (precision), 0.64 (recall), 0.63 (F-Measure) and 0.37 (AER), we can see that our algorithm has produced much better results.

Contribution of Each Component in the Word Alignment

Let M be a set of components used in the word alignment algorithm, n be the total number word pairs aligned by the algorithm, and W_m be the number of words aligned by component

m. The contribution of each component C_m is calculated using the following equation:

$$C_m = \frac{W_m}{n}$$

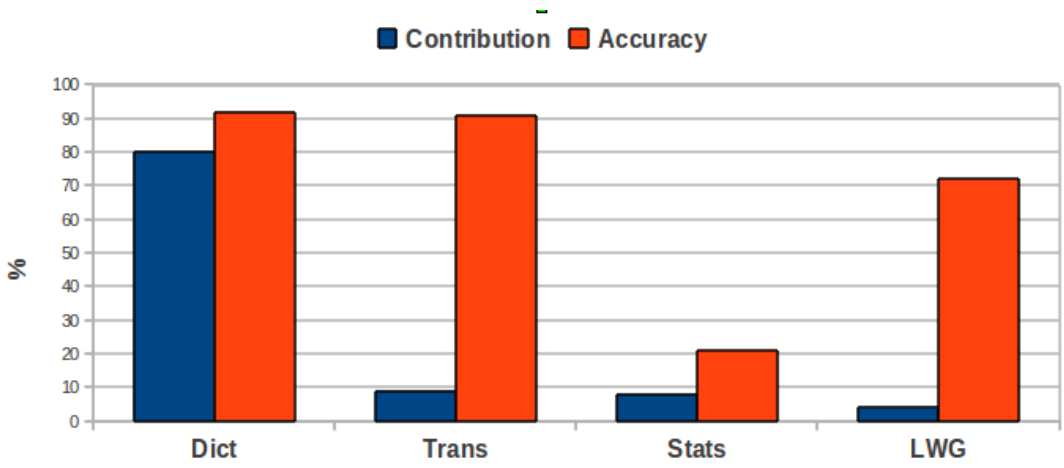


Figure 7.8: Contribution of Each Component in the Word Alignment

Figure 7.8 shows contribution of each component in the word alignment pipeline. The blue bar indicates the contribution of an individual component and the red bar its accuracy.

1. Dict – 80% of the total words were aligned with 92% of accuracy using the Dictionary Lookup PR. Out of these 80%, 1.74% of the words were aligned with the help of a synonyms list obtained from the English WordNet. In other words, when an English word could not be located in the English-Hindi dictionary, we used the synonyms list to lookup its synonyms. These English synonyms were then searched in the English-Hindi dictionary to find a possible match in the target Hindi sentence. The Dictionary Lookup PR also uses a custom dictionary – a dictionary that was built from the development dataset specifically prepared for updating resources of the word alignment algorithm. Almost 13% of the words were aligned using the custom dictionary.

2. Trans – The Transliteration Similarity (TS) method is useful for aligning words such as cognates, proper names and named entities. The method is used three times in the algorithm with a different threshold value in each iteration. When the TS method is used for the first time, the threshold value is set to 79%. In this mode, the TS method

is asked to compare strings of even the already aligned words, provided that the source and the target words being compared are not already aligned with each other. The second time the TS method is called, the threshold is set to 70%. Third time, it is set to 60%. In both the second and third iterations, the TS method is asked to consider only the unaligned words. As can be seen, the TS method is responsible for aligning almost 9% of the total words which it has aligned with 91% accuracy.

3. LWG – The Group Aligner component is responsible for aligning groups that are created as part of the LWG step. This component was responsible for aligning 4% of the total words which it has aligned with 72% accuracy.

4. Stats – Finally, the statistical PR was responsible for aligning almost 8% of the total aligned words. However, the accuracy of this component is only measured at 21%.

Having decided and evaluated the final word alignment pipeline, we ran two more experiments. In the first experiment, one component at a time was excluded from the word alignment pipeline (see Table 7.7.3). In the second experiment, we formed various combinations of components and ran each combination of these components over the test data. The former experiment gives us an idea of results when a certain component is not available whereas the latter gives us an indication of what to expect when only certain components are available. For each of these experiments, we report contributions made by individual components (marked as "**C**") and their individual performance in terms of precision (or accuracy – marked as "**A**").

Figure 7.9 shows the results of word alignment algorithm when a specific component was missing in the pipeline. Here, **D** is for the Dictionary Lookup PR, **L** is for the Grouping Alignment PR, **T** is for the Transliteration Similarity PR, **F** is for the Filtering Alignment PR and **S** is for the Statistical Alignment PR. Similarly, Figure 7.10 shows the contribution of each component when used on its own and with other components in the word alignment pipeline. In both the figures, the blue bar gives precision, red bar gives recall and the black line is for the F-Measure.

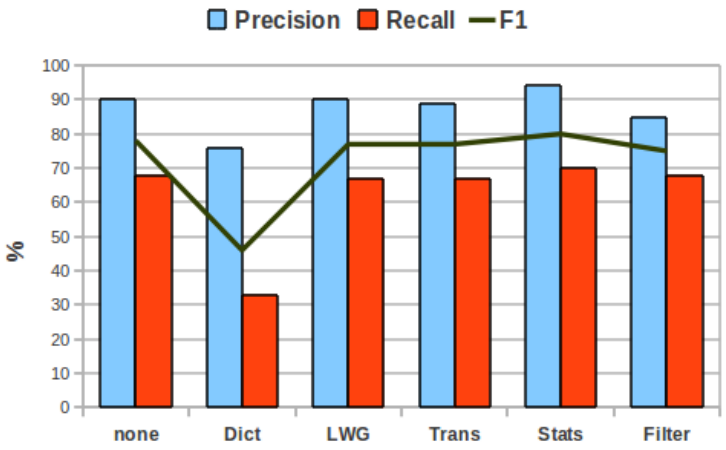


Figure 7.9: Scores Without Specific Components

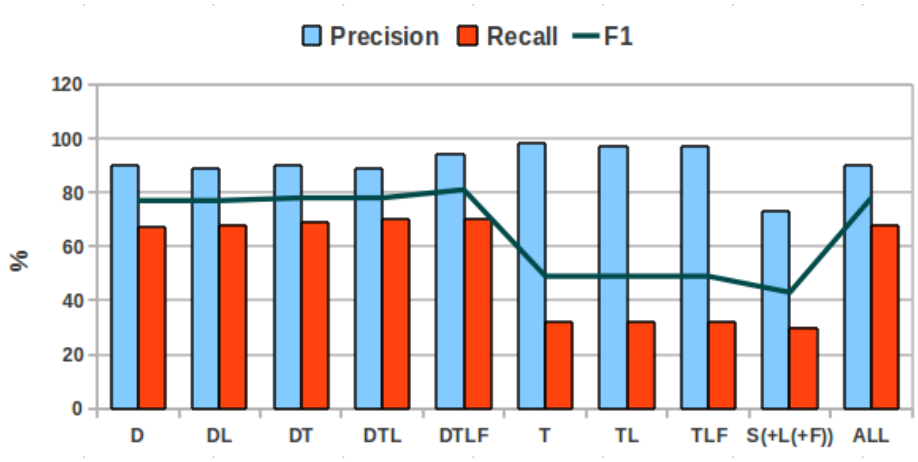


Figure 7.10: Results of Various Components Used Together

It is clear from these tables (Tables 7.7.3 and 7.7.3) that the Dictionary Lookup PR is the most important components of all. The Filtering Alignment PR tries to unalign words that are incorrectly aligned by the preceding PRs in the word alignment pipeline. We can clearly see the difference the Filtering Alignment PR brings in terms of accuracy when used in conjunction with the Dictionary Lookup PR and/or the Transliteration Similarity PR.

It is the combination of Dictionary Lookup, Transliteration Similarity, LWG and the Filtering PR that produces the best F-Measure of 0.81. The F-Measure score comes down to 0.78 when the statistical PR is added to the pipeline. It suggests that it is introducing more noise to the results rather then helping the system to find new correct alignments. However, it

Table 7.6: Word Alignment Results Without Specific Components

Missing Component	**P, R, F1**	**Individual Contributions**
none	P:0.90 R:0.68 F1:0.78	Dict (C:80%, A:92%) Trans (C:9%, A:91%) Stats (C:8%, A:21%) LWG (C:4%, A:72%)
Dict.	P:0.76 R:0.33 F1:0.46	Trans (C:23%, A:85%) Stats (C:76%, A:30%) LWG (C:1%, A:9%)
LWG	P:0.90 R:0.67 F1:0.77	Dict (C:82%, A:92%) Trans (C:9%, A:91%) Stats (C:9%, A:21%)
Trans.	P:0.89 R:0.67 F1:0.77	Dict (C:82%, A:92%) SS (C:6%, A:91%) Stats (C:9%, A:22%) LWG (C:4%, A:72%)
Stat. DICE	P:0.94 R:0.70 F1:0.80	Dict (C:87%, A:92%) Trans (C:9%, A:91%) LWG (C: 4%, A:72%)
Filter	P:0.85 R:0.68 F1:0.75	Dict (C: 81%, A:84%) Trans (C:9%, A:84%) Stats (C:7%, A:21%) LWG (C:3%, A:70%)

is important to mention that the Statistical PR used for our experiments has been trained only on a small set of dataset and it would be interesting to replace the Statistical PR with something like GIZA++ trained on a huge amount of data (such as WA_DATASET3).

7.7.4 System Evaluation

We also did a thorough analysis of the word alignment results and found many reasons why the system did not perform any better than the current results. Below, we list our observations and provide examples where possible.

- The English-Hindi dictionary has approximately 26K entries and more entries were added to the custom dictionary from the word alignment development data. We ob-

Table 7.7: Word Alignment Results With Specific Components

Components	**P, R, F1**	**Individual Contributions**
Dict	P:0.90 R:0.67 F1: 0.77	
Dict + LWG	P:0.89 R:0.68 F1:0.77	SS (C: 6%, A: 82%) Dict (C: 90%, A: 84%) LWG (C: 4%, A: 69%)
Dict + Trans.	P:0.90 R:0.69 F1:0.78	Dict (C: 90%, A: 85%) Trans (C: 10%, A: 84%)
Dict + Trans + LWG	P:0.89 R:0.70 F1:0.78	Dict (C: 87%, A: 84%) Trans (C: 9%, A: 84%) LWG (C: 4%, A: 70%)
Dict + Trans + LWG + Filter	P:0.94 R:0.70 F1:0.81	Dict (C: 87%, A: 92%) Trans (C: 9%, A: 91%) LWG (C: 4%, A: 72%)
Trans	P:0.98 R:0.32 F1:0.49	
Trans + LWG	P:0.97 R:0.32 F1:0.49	Trans (C: 96%, A: 82%) LWG (C: 4%, A: 83%)
Trans + LWG + Filter	P:0.97 R:0.32 F1:0.49	Trans (C: 96%, A: 84%) LWG (C: 4%, A: 9%)
Stat.	P:0.73 R:0.30 F1:0.43	
Stat. + LWG	P:0.73 R:0.30 F1:0.43	
Stat. + LWG + Filter	P:0.74 R:0.30 F1:0.43	
All	P:0.90 R:0.68 F1:0.78	Dict (C:80%, A: 92%) Trans (C: 9%, A: 91%) Stats (C: 8%, A: 21%) LWG (C: 4%, A: 72%)

served that there are quite a few content words, i.e. verbs, nouns, adjectives and adverbs, that are missing from the dictionary.

For example, if we count the total number of English words we have in our English-

Hindi dictionary, along with their synonyms in our synonyms list, the total number of unique words we have is 53,569. When this number is compared to the total number of unique content words available in the English WordNet[6], i.e. 155,287, we can see that our dictionary contains less than 35% of the total entries found in the WordNet. From this, it is very clear that our dictionary needs to be improved and this will certainly improve the overall results.

- We do not take POS tags of individual words into account when looking up for their translations in the dictionary. Because of this, certain words become ambiguous. For example, the word *May* in the following examples:

  – Miss **May** is very naughty.

  – **May** I come in?

  – **May** is usually warm enough to sunbathe.

  If the sentences are like those shown above, the alignment algorithm does not have any problem aligning words appropriately. However, the problem occurs when the same word *May* is used more than once in the same sentence in different contexts. For example:

  Miss **May may** come sometime in mid **May**.

  This is a difficult example where the current alignment system makes mistakes. However, the problem can be solved by running a POS tagger for both languages and considering POS tags of individual words. For example, if we know that *May* in *Miss May* is a proper noun, we can reduce the number of possible candidate words for alignment to only the proper nouns found in the parallel Hindi sentence. Similarly, if we know that the second mention of the word *may* is an auxiliary verb, it would be treated differently. Even though the use of POS tagger does not guarantee only one possibility of alignment per word, it certainly reduces the number of possibilities a word can be aligned to.

[6] `http://wordnet.princeton.edu/wordnet/man/wnstats.7WN.html`

- We observed that in certain cases our word alignment system had problems in aligning pronouns in the text. Consider the following example:

  John sings because he loves singing.
  जोह्न गाता है क्योंकि जोह्न को गाना पसंद है
  (*John gaataa hai kyonki John ko gaana pasand hai*)

  As can be seen in the above example, the pronoun *he* is used for coreferring to *John* in the English sentence. However, the name जोह्न (*John*) is used twice in the Hindi sentence. In such cases, the system could not align the pronoun *he* with the second mention of जोह्न (*John*) in the text.

- There are many instances in the EMILLE corpus where for a word in the English text, multiple Hindi translations, as alternatives of one other, are provided in the parallel Hindi text. The same is true with the abbreviations. While in certain cases, the abbreviation is written as it is in English in the Hindi parallel text, in some cases, a transliterated version of the same abbreviation is also provided as an alternative. For example:

  I live in the UK.
  मैं UK/युके में रहता हुं
  mE.n UK main rahataa huin

  In some cases, even the full form of an abbreviation is provided in the Hindi text. For example:

  I live in the UK.
  मैं युनाइटेड कींगडोम में रहता हुं
  mE.n united kingdom main rahataa huin

We had asked our aligners to align such a word with all the possible translations. However, this is not how our word alignment system works. Even though the system may initially associate such an English word with all its alternative translations in the parallel Hindi text, the Filtering Alignment PR may reduce the associations of such a word to only one alignment (see section 7.5.2 for more information on the constraints applied by the Filtering Alignment PR).

- We observed that there are many Hindi compound words in our test data that could not be aligned by our word alignment system. An example of this is the word अत्याधुनिक (*atyAdhunika, ultra modern*) which is composed of two independent words, अत्य (*atya, ultra*) and आधुनिक (*Adhunika, modern*).

  The Dictionary Lookup PR considers only one English word from the English text at a time. It looks it up in the English-Hindi dictionary, collects all its possible Hindi translations and matches them in the Hindi text. In this case, it tries to find out Hindi translations for the English words *ultra* and *modern* separately. Both of these words exist in our English-Hindi dictionary and have more than one Hindi translation. The relevant translations are अत्य (*atya*) and आधुनिक (*Adhunika*) respectively. However, when the Dictionary Lookup PR tries to match them in the Hindi text, it cannot find the words अत्य (*atya*) and आधुनिक (*Adhunika*).

  The problem can be solved in two steps:

  1. by introducing a PR that decomposes such compound words into individual words
  2. by asking the Dictionary Lookup PR to use the decomposed words for dictionary matching

  We have developed such a PR that decomposes Hindi compound words into individual words. However, more work needs to be done before it can be successfully integrated in the word alignment system.

- Dealing with misspelled words is one of the other challenges we found in our word alignment system. We had asked our aligners to ignore misspelled words. Following this guideline, they had not aligned any misspelled word in the gold standard. However, if the misspelled word is a named entity or a cognate, our word alignment system tries

to align it using the Transliteration Similarity PR (see Chapter 6). In such cases, even though the alignment is correct, it is not rewarded because it does not exist in the gold standard. Here are some examples of misspelled words:

  - वस्त (*vaston*) – should have been spelled as वस्तु (*vastuon*) which means *things*' in English.
  - आधुनिकीकरन (*aadhunikikaran*) – should have been spelled as आधुनिकरन (*aadhunikaran*) which mean *modernisation* in English.

- Local word grouping helps in aligning phrases such as noun phrases and verb phrases. Some of LWG rules make it possible to align even idioms in the text. One example of such an idiom is *hand in hand* in English and साठ साठ (*saath saath*) in Hindi. However, there are many idioms which cannot be aligned yet. For example, उल्ता पुल्ता (*UlTa PulTa*) which means *upside down* or *mess* in English. Our dictionary has only single term entries and therefore having a proper dictionary of idioms could help solving the problem.

- Improper translation is also one of the causes of words not getting aligned properly. For example, *second hand* in English is translated as नयी और पुरानी (*nayI Aur purAnI*) which means *new and old* in English. Such a translation in the text results in poorer translation quality.

- In most cases, numbers in the EMILLE corpus are translated fine. However, in case of help line numbers, we found that these numbers were diffrent. The same is true for many addresses including building addresses, email address and website addresses.

  In such cases, where the details were different, our annotators did not align them with each other. However, if the phone number, for example, is written across multiple tokens, and there are common tokens found in the two numbers, they get automatically aligned by our word alignment algorithm. Here is one such example from our test data. The number specified in the English text was *071 - 638 - 3700* and the number specified in the Hindi text was it 071 - 405 3923. Here, the algorithm had aligned *071* and -.

- Often incorrect tokenisation is a cause of words not being aligned correctly.

  - 8,000 in the Hindi text is tokenised as three tokens: "8", ",", and "000". However, such a number is correctly recognised as a single token in the English text.
  - Often it is the case in Hindi that postpositions are conjoined with the words appearing before them. One such example is व (*vargOmEn*) which means *inside classrooms*. If the tokeniser was able to separate the postposition from the previous word व , it would have been possible to align both the words व and . In the above case, because the post position was conjoined with the word व , which is a plural form of the word वर्ग , our morphological analyser was not able to find a proper root form of the word.

- One of the components that did not perform well, in terms of precision and recall, is the Transliteration Similarity PR with the similarity threshold set to 60%. Our analysis of the results suggest that it could only align 2% of the total words aligned by the system and has achieved precision just over 60%.

  Similarly, the the Statistical Method PR has very low coverage. This could be due to fact that we use this method at the end of the pipeline, only for the remaining unaligned content words, and also with a minimum frequency constraint that requires a word pair to occur at lease three times in the parallel sentence pairings in the test dataset. Therefore, the PR may not have many words to align. Also, the amount of data is not sufficient for the PR to collect enough frequencies for all the unaligned words.

  The choice whether these two PRs should be included in the word alignment system or not depends on factors such as:

  - whether the user has enough data to collect enough frequencies for content words.
  - whether the user is looking for better coverage or better precision. In the former case, the user may want to include both the PRs and may change the similarity threshold and minimum frequency parameters. In the latter case, it would be better to avoid using both the PRs.

Using Alignment Results for Machine Translation

In Section 7.4, we presented results of an SMT system trained on the alignments produced

by GIZA++. In order to check if better word alignment results produce better MT output, we conducted another experiment with moses where we fed the moses system the word alignments produced by our word alignment algorithm for the datasets WA_DATASET2 and WA_DATASET3. We call these models *aaswan DS2* and *aaswan DS3* respectively. We applied these models on the same test data (MT_DATASET1), which we had used for testing the MT models *moses DS2* and *moses DS3*.

Hindi:	रैज़िडेंशल सर्विसिफ़ आम तौर पर उन लोगों के लिए हैं , जिनकी नशीली दवाओं से सम्बन्धित गंभीर व दीर्घकालिक समस्याएँ हैं ।
Reference:	residential services are usually for serious long-standing drug problems .
moses DS3:	रैज़िडेंशल सर्विसिफ़ usually for those - standing drug problems serious long .
moses DS2:	रैज़िडेंशल सर्विसिफ़ Septicaemia to those who are of drug सम्बन्धित from serious and - समस्याएँ । .

Figure 7.11: Comparion of Output from various MT Models

For the MT model *aaswan DS2* (trained on WA_DATASET2) we obtained a BLEU score of 5.33 and for the model *aaswan DS3* (trained on WA_DATASET3) we obtained a BLEU score of 59.6. These scores are only slightly better than those obtained for the models trained on alignments from GIZA++. In Figure 7.11, we compare the outputs of these MT models. As can be seen, the translation produced by the *aaswan DS2* model is certainly better than the translation produced by the *moses DS2* model. However, the translations produced by these models do not make any sense. When we compare the translation produced by *aaswan DS3* with the translation produced by *moses DS3*, we can see that the *aaswan DS3* model has been able to solve one of the two issues we had with the translation produced by the *moses DS3*. It has been able to translate the words *residential services* correctly. This could be an effect of having the Transliteration Similarity component in our word alignment algorithm. The problem associated with reordering is not resolved yet however the output of the *aaswan DS3* is much better than the output of the *aaswan DS2*.

Overall, we cannot see a much difference in the output of the translation systems even though we had seen a significant difference in the alignment results. As specified earlier, there are several other factors which affect the translation quality. However, one point that is worth

mentioning here is that the dataset WA_DATASET3 is composed of documents from different domains. It is possible that our dictionary does not contain many words associated with these domains. Therefore, considering dictionaries that belong to specific domains may help improve the word alignment and eventually the output of machine translation at some level.

Statistical Significance Test for the SMT models

Koehn (2004) has described a bootstrap re-sampling method to compute statistical significance of test results of an MT system. This method, in particular, is suitable for small test sets – e.g. 300 sentences. The author of the paper suggests that estimating the confidence interval from a large number of test sets with n test sentences drawn from a set of n test sentences with replacement is as good as estimating the confidence interval for test sets size n from a large number of test sets with n test sentences drawn from an infinite set of test sentences. We follow this method to compute statistical significance for results of our models.

In our experiment, we assembled an English-Hindi test set of 300 sentence pairings. We used both of our models , *moses DS3* and *aaswan DS3*, to translate one Hindi sentence at a time and to produce its output in English. Thus we could compare how well the English output of the *aaswan DS3* is in comparison to the English output of the *moses DS3*. From this set, we repeatedly (1000 times) generated blocks of 300 sentences each where the sentences were randomly picked up from the initial set of 300 sentences.

Considering one block of translation at a time, we compared if the BLEU score of *aaswan DS3* for that particular one block was better than the BLEU score of *moses DS3*. We found that the output of *aaswan DS3* was better on 624 occasions (out of total 1000 occasions). Using the bootstrap sampling method suggested by the author, we sorted the BLEU scores of both the models, ignored the top 25 and the bottom 25 scores and generated an interval of BLEU scores for both the models. Intervals of BLEU scores obtained for the *aaswan DS3* and the *moses DS3* models are [56.02, 62.88] and [55.59, 62.16] respectively.

According to Koehn (2004), if the significance level is between 60% and 69.9% (62.4% in our case), one could predict with 72% confidence that the system is likely to produce a BLEU score that is between the interval obtained above (i.e. between 56.02 and 62.88).

7.8 Chapter Remarks

In this chapter, we have presented a hybrid word alignment algorithm for the English-Hindi language pair. Also, we looked at the two key components used as part of the word alignment algorithm: a component to group local words and a component to filter alignments. We learnt from the literature that when there is not enough data, statistical methods do not perform well. The same can be confirmed from our own experiments with GIZA++. By introducing external resources such as a bilingual dictionary, and transliteration similarity method, we are able to raise the F-Measure score from 0.63 to 0.78. As far as we are aware, no other English-Hindi word alignment system has yet achieved F-Measure so high. We did a thorough analysis of our word alignment results and discovered the areas where the system can be improved by making further improvements.

In order to find out an effect of better alignments on the overall translation produced by an SMT system, we carried out an experiment whereby we trained four SMT models: two based on limited amount of data and two based on larger training sets. We experimented with using alignments produced by GIZA++ and compared the output with the output of an SMT system trained on the alignments produced by our word alignment algorithm.

Even though the alignments produced by our algorithm produced better results, the difference in the output of the MT system was not substantial. It was observed in the MT output that GIZA++ had problems aligning terms such as named entities, whereas the MT system trained on the alignments produced by our system was able to translate them correctly. However, a little improvement in the overall BLEU score is an indication that the quality of an MT system relies heavily on factors other than the quality of word alignment only. The other factors being the amount of data supplied for training, domains of the training and testing datasets etc.

Having developed a number of components for the English-Hindi language pair, we, now, shift our focus onto the English-Gujarati language pair. According to our hypothesis, the components developed for the Hindi language should be adaptable to the Gujarati language as well. We take with us all the components into the next chapter where we describe their adaptation to English-Gujarati and our experiments with the English-Gujarati language pair.

Chapter 8
Experiments with the Gujarati Language

8.1 Outline of the Chapter

In this chapter, we demonstrate the generality of our alignment algorithms and the approaches used for developing Hindi resources by adapting them to the English-Gujarati language pair.

The Gujarati Language

The Gujarati language is native to the Indian state of Gujarat. Some estimates suggest that there are about 46.1 million people speaking the Gujarati language all over the world. Similar to the Hindi language, it also belongs to the Indo-Aryan family of languages.

We decided to do experiments on the Gujarati language for four main reasons.

- The Gujarati language is one of those South Asian languages that are similar to the Hindi language. Both the English and Hindi languages are similar in many ways.

    1. Both the languages are similar in structure. For example, to translate a Hindi sentence into the Gujarati language, the minimum thing one have do is to replace the Hindi words with their respective Gujarati words.
    2. Both the languages are free-order language.
    3. Even though, their vocabulary is different, they share quite a few words between them. There are many words which may not be identical but have a very little difference in them. For a Hindi speaker to be able to understand and speak the Gujarati language, he would need to learn only the vocabulary.
    4. The one major difference, however, between the two languages is that the Gujarati language has an additional gender, neuter, which is not the case with the Hindi language which has only two genders: masculine and feminine.
    5. Both the languages have different writing scripts but their alphabets are very similar where the consonants and vowels are pronounced in the same way.

    Considering these similarities, we believe that if we can successfully adapt Hindi resources to the Gujarati language, it should be possible to adapt the Hindi resources to other South Asian languages as well.

- It is a language which has very limited resources available. We have provided a list of some of the Gujarati resources in the appendix. However either these resources are not publicly available or are very small in size or incomplete to be fully utilised as an NLP resource. Despite being one of the languages with a huge number of speakers, the Gujarati language has just a few resources available online. If we could develop resources for the Gujarati language, we believe it would be a significant contribution to the research community working on the Gujarati language.

- The Gujarati language is one of the six south Asian languages for which the EMILLE corpus has parallel text. Thus, data is available for us to conduct our experiments.

- Availability of Gujarati native speakers meant that we could easily verify the results of our algorithms and developed resources for the Gujarati language. The author himself is a native speaker of the Gujarati language which makes it easy to identify any potential problems at an early stage.

In the following sections, we first look at the sentence alignment algorithm. The algorithm is same as the one proposed by us in the Chapter 4 but trained on a subset of the Gujarati data. Then, using the same approach as suggested by us in Chapter 5, we develop a morphological analyser for the Gujarati language. Here too, we use the same algorithm for learning suffix-replacement rules but use the Gujarati resources instead. We also experiment with the transliteration similarity method. Finally, we use all these new components together to produce an English-Gujarati word alignment system.

Not all the components we used for the English-Hindi experiments need adaptation to the Gujarati language. For example, one can use the mean and the standard deviation values as obtained for the English-Hindi language pair. This is possible because both the Hindi and Gujarati languages are very similar in structure. Similarly, the Filtering alignment PR (from the Word Alignment chapter) can be used directly for the Gujarati language without making any modifications to the PR.

Since both the languages are very close to each other, the process of adapting resources of one language to the other is very easy. It took us only 3 person days to adapt all the English-

Hindi resources to the English-Gujarati language pair. The duration reported here does not include the time required to prepare a word alignment gold standard for the English-Gujarati language pair. To prepare the gold standard, we needed help from 2 native speakers of the Gujarati language for 4 more days. Thus, it took us only 11 person days in total to prepare the gold standard and adapt all the resources to the English-Gujarati language pair.

8.2 Sentence Alignment

We used the EMILLE corpus for our experiments. When examining the English-Gujarati parallel corpus, we observed that in most cases one English sentence aligns with only one Gujarati sentence. However, it is also possible that one or more English sentences align with one or more Gujarati sentences.

In order to find out how the lengths of the two languages correlate we divided the lengths of the Gujarati sentences with the lengths of their parallel English sentences. As in Figure 4.2 for Hindi, Figure 8.1 shows two graphs. In the first graph, the length is calculated in characters whereas in the second graph, the length is calculated in words. The *x-axis* in both graphs gives the observed number of Gujarati characters per English character and the *y-axis* gives the number of sentences for each observed value.

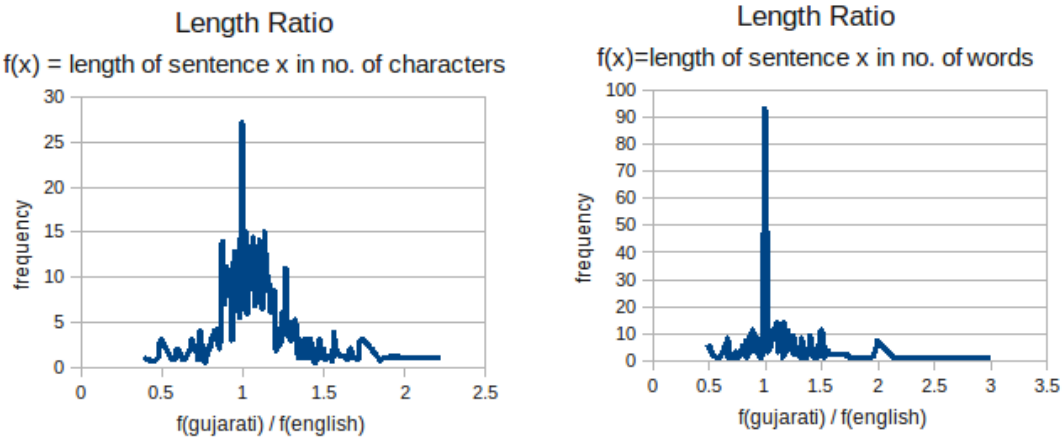


Figure 8.1: Distribution of the Ratios of Lengths in Characters and Words

The graph was created based on 500 sentence pairings randomly collected from the corpus which included several one-to-one and many-to-many sentence pairings. The pairings were

manually aligned for the purpose of the experiment. In the case of the first graph, out of the 500 parallel sentence pairings, 22 sentence pairings had the same length. We found that 305 Gujarati sentences were longer than their respective English sentences in comparison to 168 sentence pairings where the English sentences were longer than their respective Gujarati sentences. In case the pair has more than one sentence in either the source or the target language, the length is the aggregated total of the lengths of the sentences. In the case of the second graph, 93 sentence pairings had the same number of words. We found that 282 Gujarati sentences had more words than their respective English sentences in comparison to 125 sentence pairs where the English sentences had more words than their respective Gujarati sentences. Plotting their lengths in number of characters and words produces charts as shown in Figure 8.2.

As can be seen in this figure, for instance, if the length of English sentences is 80 characters, their parallel Gujarati sentences have lengths varying between 57 and 117 characters. The same sort of variation is observed when the length is measured in number of words. If the length of English sentences is 12 words, their parallel Gujarati sentences have lengths varying between 7 and 17 words. These graphs are similar to the ones obtained for the English-Hindi language pair (see Figure 8.2). Thus, the approach used for the English-Hindi language pair should also work for the English-Gujarati language pair.

Using all 500 sentence pairings, we performed 10-fold evaluation on our dataset. In this case, the data was divided into 10 folds with an equal number of sentence pairings in each of them. The sentence pairing were randomly distributed in each of the folds. 9 folds were used for training and the remaining one for testing. The process was repeated for 10 times, giving each fold a turn to be tested upon.

Our algorithm takes aligned paragraphs as input and aligns sentences within one pair of paragraphs at a time. As all the 500 sentence pairings were collected randomly from the EMILLE corpus, we could not preserve the paragraph information. In order to align sentences of one paragraph at a time, we formed paragraphs with 5 sentence pairings in each of them.

Table 8.1 shows the results of this experiment.

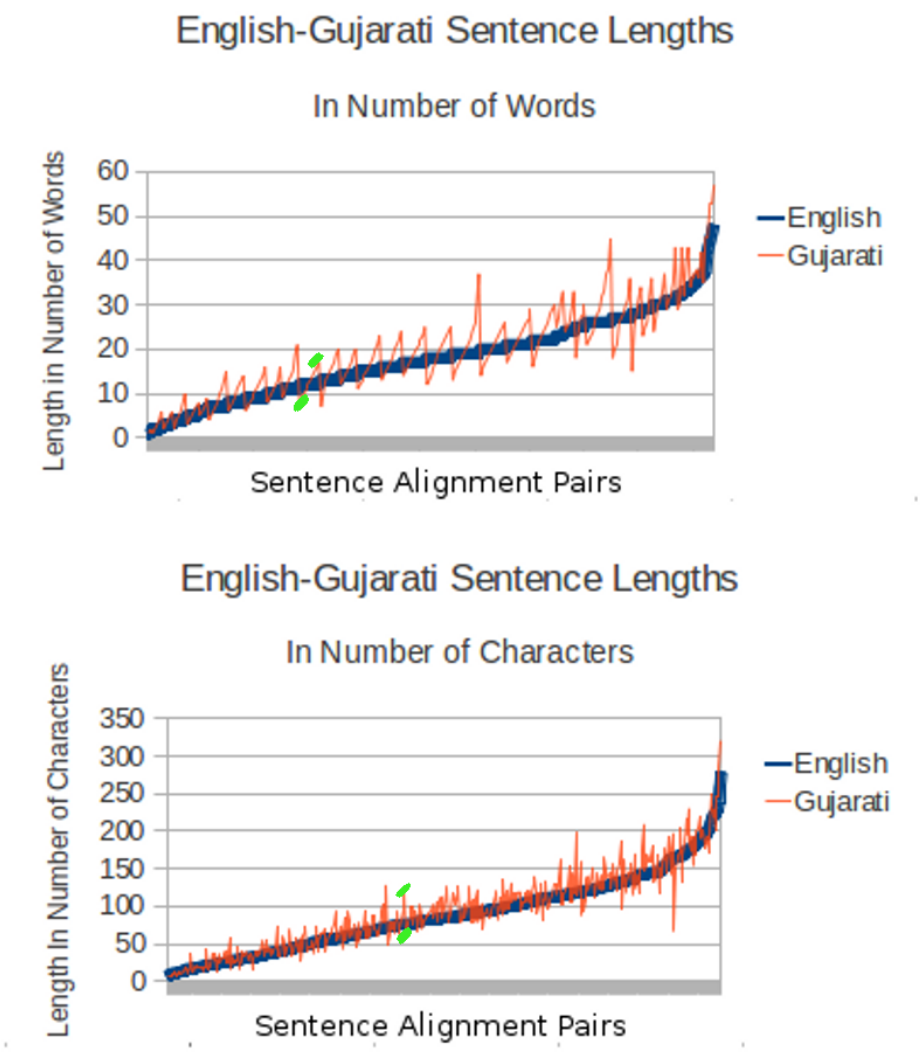


Figure 8.2: Lengths of English-Gujarati Parallel Sentences

The highest precision obtained during the 10-fold evaluation exercise is 0.950. Since the sentence pairings are drawn randomly in these folds, it is likely to have different results every time the evaluation is performed. We performed 10-fold evaluation ten times and obtained 0.938 precision as an average calculated over the ten 10-fold evaluations.

The same experiment was repeated by calculating length in number of words. Table 8.2 shows the results collected from the experiment. The highest precision obtained from the 10 different precision figures collected during the 10-fold evaluation exercise is 0.910. We performed 10-fold evaluation ten times and obtained 0.901 precision as an average calculated

Table 8.1: 10-Fold Evaluation of the Sentence Alignment Algorithm (length measured in characters)

μ_1	σ_1	**Precision**	**Recall**
0.844	0.012	0.931	1.0
0.843	0.012	0.938	1.0
0.844	0.011	0.933	1.0
0.844	0.012	**0.950**	1.0
0.844	0.012	0.938	1.0
0.844	0.011	0.948	1.0
0.843	0.012	0.940	1.0
0.845	0.012	0.929	1.0
0.844	0.012	0.923	1.0
0.844	0.012	0.946	1.0
Average		**0.938**	**1.0**

over the ten 10-fold evaluations.

The two Tables 8.1 and 8.2 clearly show (as expected) that considering length in number of characters gives better results.

In the case of the sentence alignment algorithm, where there is no element of any language dependency, we were able to use the algorithm without needing any modifications for the Gujarati language. We believe the real challenge lies ahead when we try to adapt other resources. In the next section, we use the same algorithm that we used for learning suffix-replacement rules for the Hindi morphological analyser and try to obtain a similar rule set for the Gujarati language.

8.3 Gujarati Morphological Analyser

In this section, we use the same algorithm as described in Chapter 5 to learn suffix-replacement rules for the Gujarati language. We do not use anything else other than the algorithm from the Chapter 5 to learn suffix-replacement rules for the Gujarati language. The

Table 8.2: 10-Fold Evaluation of the Sentence Alignment Algorithm (length measured in words)

μ_1	σ_1	**Precision**	**Recall**
0.804	0.014	0.900	1.0
0.805	0.014	0.906	1.0
0.805	0.014	**0.910**	1.0
0.805	0.014	0.902	1.0
0.804	0.014	0.904	1.0
0.805	0.014	0.908	1.0
0.804	0.014	0.886	1.0
0.805	0.014	0.898	1.0
0.805	0.014	0.900	1.0
0.805	0.014	0.898	1.0
Average		**0.901**	**1.0**

learning algorithm requires two resources: 1) a monolingual Gujarati corpus and 2) a Gujarati dictionary. We use the Gujarati dictionary available at `http://www.gujaratilexicon.com`.

Since most dictionaries have entries in base-forms, we added all single word entries from the dictionary into a base-form list (GBFList). From the dictionary, we were able to add in total 11758 entries which included 3408 adjectives, 425 adverbs, 7096 nouns and 829 verbs. However, since these entries are not verified, the GBFList is likely to have some spurious entries in it.

All the steps we followed for the Hindi language are repeated for the Gujarati language. The first step is to find out the most common suffixes for base-forms. Later, these common suffixes, along with the monolingual Gujarati corpus can be used to learn suffix-replacement rules. Once the rules are obtained, we need to finalize the ruleset. Please see Chapter 5 for more information on these individual steps.

8.3.1 Base-form Suffixes

Table 8.3 lists some of the frequent suffixes as obtained from the GBFList for each of the four word class categories.

Table 8.3: Most Frequent Suffixes for Gujarati Base-forms

Verbs	%	**Nouns**	%	**Adjectives**	%	**Adverbs**	%
વું (*vun*)	92.57	ી(*ee*)	12.27	ં (*n*)	22.36	ં (*n*)	22.82
વવું (*vavun*)	17.91	ા(*aa*)	11.27	ુ ં (*un*)	22.95	ુ ં (*un*)	17.41
ાવવું (*aavavun*)	12.95	ં (*n*)	8.86	ક (*ka*)	14.22	ા(*aa*)	12.00
ાવું (*aavun*)	9.90	ર (*ra*)	8.62	ી(*ee*)	9.50	ે(*e*)	11.52
ડવું (*davun*)	9.68	ઓ (*o*)	8.05	િત (*ita*)	7.48	ર (*ra*)	7.64
રવું (*ravun*)	9.23	ુ ં (*un*)	7.99	િક (*ika*)	7.46	ક (*ka*)	7.29
ળવું (*lavun*)	6.41	તા (*taa*)	6.04	ર (*ra*)	7.20	ત (*ta*)	7.29
કવું (*kavun*)	5.29	ંુ (*nun*)	0.14	ુ (*lu*)	6.91	થી (*thee*)	7.05
-	-	ાલ (*aal*)	0.14	વાળુ (*vaalu*)	6.09	ાં (*aan*)	6.17
-	-	વાદ (*vaad*)	0.13	ાળુ (*alu*)	6.09	-	-

In order to add more words to the GBFList, we considered one suffix from Table 8.3 and tried to match words in the corpus that end with that suffix. Assuming that the matched word is a base-form, if it did not exist in the GBFList, we manually checked it and added the base-form to the GBFList. We stopped examining suffixes down the list when there were more negative examples found than positive ones. As a result of this exercise we added 576 more words (mainly verbs and adjectives).

8.3.2 Suffix-Replacement Rules

The next task was to learn suffix replacement rules. In order to do that, we obtained a set of unique and inflected words (84,489) from the corpus that did not exist in the GBFList. Similar to the Hindi language, we repeated all the steps to obtain a set of suffix-replacement rules. Table 8.4 lists some high-frequency rules as obtained from our experiment.

Table 8.4: High Frequency Suffix-Replacement Rules for Gujarati

Suffix	Replacement	Count	Example	Category
ી(*ee*)	વું (*vun*)	392	પીગળી (*peegalee, melting*) - પીગળવું (*pigalavu, to melt*)	verb
ના (*naa*)		358	સલાહકારના (*salaahakaaranaa, advisor's*) - સલાહકાર (*salaahakaar, advisor*)	noun
ની (*nee*)		347	સલાહકારની (*salaahakaaranee, advisor's*) - સલાહકાર (*salaahakaar, advisor*)	noun
ને (*ne*)		318	સલાહકારને (*salaahakaarane, to advisor*) - સલાહકાર (*salaahakaar, advisor*)	noun
ા(*aa*)	ુ઼ (*un*)	278	બતાવવા (*bataavavaa, for showing*) - બતાવવું (*bataavavun, to show*)	verb
ા(*aa*)	ુ઼ (*un*)	275	જોડાયેલા (*jodaayelaa, connected(p)*) - જોડાયેલું (*jodaayelu, connected(s)*)	adj
માં (*maan*)		273	વિચારમાં (*vichaarmaan, in thought*) - વિચાર (*vichaar, thought*)	noun
ે (*e*)	વું (*vun*)	269	જાળવે (*jaalave, preserve*) - જાળવવું (*jaalavavun, preserve*)	verb
તા (*taa*)	વું (*vun*)	258	બતાવતા (*bataavataa, showing*) - બતાવવું (*bataavavun, to show*)	verb
નો (*no*)		258	કટોકટીનો (*katokateeno, of urgency*) - કટોકટી (*katokatee, urgent*)	noun

8.3.3 Finalizing the Ruleset

Having obtained such a list of rules, the next task was to finalize a ruleset that yields maximum performance. The rule validation dataset (GUJDATASET1) we used for our experiments contains 2000 words randomly obtained from the EMILLE corpus. In order to make sure the words we collect are inflected, only the words that do not exist in the GBFList were collected and included in the GUJDATASET1. In order to find a base-form for each word in the GUJDATASET1, we used the rules with high frequency above 75 from the previous step. The resulting base-forms were manually checked and corrected if necessary.

We also prepared a separate test dataset (GUJDATASET2) for testing the final ruleset which also contains 2000 words randomly obtained from the EMILLE corpus. In order to make sure the words we collect are inflected, only the words that do not exist in the GBFList were collected and included in the GUJDATASET2.

Similar to our experiment with the Hindi language, we started with an empty ruleset. One

rule at a time was added to the ruleset and its performance over the GUJDATASET1, in terms of F-Measure, was recorded. If the addition of a rule resulted in a lower F-Measure, the rule was removed from the ruleset. If the addition of a rule did not make any difference in the value of the F-Measure, the user was shown a list of words which were collected during the phase of rule discovery (fourth column in Table 8.4). Based on the word pairs shown to the user, the user was asked to either include or exclude the rule from the ruleset. This procedure was repeated until all the rules with frequency above 20 were added to the ruleset. Finally, we obtained a ruleset that yielded best F-Measure figure.

We used this ruleset to experiment with GUJDATASET2 and obtained 0.834 (precision), 0.699 (recall) and 0.760 (F-Measure). When we compare these results with the results of the Hindi morphological analyser, i.e. 0.821 (precision), 0.803 (recall) and 0.812 (F-Measure), the results of the Gujarati morphological analyser are very encouraging, especially given the very limited amount of efforts we had to put in to learn these rules. However, the results are not as good as the results of the Hindi morphological analyser.

The morphological analyser needs to check if the base-form returned by the underlying suffix-replacement rules exists in the GBFList. If it exists, the morphological analyser returns the base-form as a final result. Therefore, it was the GBFList which was investigated first to find out a cause for low recall. As expected, our analysis of the GBFList revealed that there are many incorrect entries in the list (i.e. most of them to do with spelling errors), which, if corrected, could lead to much improved results of the system. Below, we provide a few examples:

- વાચનનો (*vAchanano, of reading*). The root form વાચન (*vAchan, a reading*) is correctly produced by the Gujarati morpher but not added as a root form because it does not exist in the list of base forms.
- ફાલો (*fAlo, contribution*) is a root form itself. However, because it does not exist in the list of base forms, it is processed with the morpher. Morpher converts it into another root form ફલ (*fal, result*) which exist in the list of base forms. In this case, even though the root form ફલ is a valid Gujarati word, its meaning is very different from the original word.

- શીકશાકો (*shIkashako, teachers*) is correctly converted into a singular form શીકશાક , however the correct spelling of the word as found in the dictionary is શિક્ષક (*shIkshaka, a teacher*). Similarly, વસ્તૂઓં (*vastUon, things*) is correctly converted into its root form વસ્તૂ (*vastU, vastu*), however the correct speeling is વસ્તુ .

- જે (*je, the one which*) is converted into જવું (*javu.n, to go*). This happens because there is no appropriate rule to handle such a word.

- હેડટીચરો (*headteachero, headteachers*). Here, the English word Headteacher is transribed in the Gujarati language and then inflected using a suffix from the Gujarati language. Even though, the analyser is capable of finding a corret root form here, it does not assign it to the word because the transcribed word does not exist in the dictionary.

Given that the entire process took only a couple of days (with most time spent on preparing the two datasets), the results are very encouraging. One could also examine rules that did not make it to the final ruleset and use the respective examples (fourth column in Table 8.4) collected from the corpus to decide if they should be included in the ruleset.

In the next section, we give details of our experiments with developing a transliteration similarity component for the English-Gujarati language pair.

8.4 Transliteration Similarity

Balajapally et al. (2008) have suggested that since the writing systems of the Indian languages are similar, it should be possible to exploit the commonality among them.

The Gujarati and Devanagari scripts are very similar. They have similar consonants and vowels, which are pronounced in the same way. Since the transliteration work presented here is solely based on letter correspondences, using the mapping produced for the English-Hindi language pair, we tried to produce a mapping between the letters of the Gujarati script and the English alphabet. With the help of a native Gujarati speaker, we replaced the Hindi letters in our mapping table with their Gujarati counterparts. Figure 8.3 lists letter correspondences between the writing systems of the Gujarati and English languages.

0	૦	a	એ, અ, ા, આ, ઍ, ે, ૈ	c	સ, સી, ક	f	ફ, એફ
1	૧	aa	ા, આ	cc	સ, ક	ff	ફ
2	૨	ae	એ, ઍ, ે	ch	ક, ચ	g	ગ, જ, જી
3	૩	aum, om, oum	ૐ	chh	છ	gg	ગ, જ
4	૪	b	બ, બી	ai	ૈ	gh	ઘ
5	૫	o	ઓ, ઓ, આ, ો, અ, ા	dd	દ, ડ,	h	હ, એચ
6	૬	oau	ૌ, ઔ	dh	ઢ, ધ	hh	હ
7	૭	n	ં, ન, એન, ણ, ઁ	ei	ૈ	ii	ઈ
8	૮	q	ક્યુ, ક્યૂ, ક	k	ક, કે	kh	ખ
9	૯	ru	ુ, ઋ, ૂ	ll	લ	m	મ, એમ, ં
tt	ત, ટ	t	ત, ટ, ટી	mn	ણ, ન	un	ન
kk	ક	w, ww	વ	on	ન	oo	ૂ, ઊ, ુ
mm	મ	d	જ, ડ, દ, ડ, ડી	ph	ફ	pp	પ
l	લ, એલ	e, ee	ે, એ, ૈ, િ, ઍ, ઈ, ઈ, ી	th	થ, ઠ	roo	ઋ
ou	ઔ, ી	ea	ૈ, ે, એ, ઍ	i	ા	sh	ષ, સ, શ, ધ
qu	ક	r	ર, આર, ુ, ઋ, ૃ	vv	વ	ti	શ
rr	ર, ઋ	s	સ, એસ, જ, ોં, ાઁ	y	ય	ue	યુ, યૂ, ુ, ૂ
z, zz	જ, ઝ	u	ઉ, િ, ઊ, ી, ુ, યુ, યૂ, ૂ	er	યર	p	પ, પી
tio	શ, ધ	es	ે, ોં, ાઁ	aei	ઍ, એ	bb	બ
v	વ, વી	i	ઈ, િ, ઈ, ી, આય, આઈ	j	જ, જે	bh	ભ
yy	ય	ss	શ, ષ, ધ, સ, જ	au	ઔ, ી	jj	જ

Figure 8.3: English-Gujarati Transliteration Mappings

In order to experiment with the English-Gujarati mapping, we used 500 sentence pairings, randomly taken from the EMILLE corpus. The sentences in each pairing were translations of each other. Considering one sentence pairing at a time, we compared each word from the source sentence(s) with every word in the target sentence(s). Similar to our earlier experiment with the English-Hindi parallel data, we only used nouns from the English data. We used 79% as the similarity threshold and asked the three methods (DC, TSM and JW metrics) to cast their votes. If a word pair received at least two positive votes, the words in that pair were marked as transliterations of each other.

The system returned 450 word pairs, out of which 172 pairs were unique. We checked these word pairs manually and found that the system had correctly identified 156 word pairs giving an accuracy of 0.910. When we compare this figure with the similar experiment carried out on the Hindi dataset (see Chapter 6), the accuracy we had obtained was 0.947.

When we tried to investigate into the errors made by the Transliteration Similarity component, we found that there are mainly two types of errors:

1. A missing character mapping
2. We noticed that on many occasions, vibhakties are conjoined with their preceding word.

   For example, એક્ટમાં (*actmAn*), ફોનીલ (*phonenI*).

   The first example could not be picked up by the PR because of the two reasons. First, it does not fulfil the requirement of endings of the both words being same. Because it has a postposition માં at the end, it is rejected by our TSM similarity metric. Second, it is still possible that it receives enough votes from the other two methods and gets aligned; However, in this case, the word itself is very short (3 letters for the actual word and 2 letters for the postposition). We have a predefined condition that the pairs should be aligned with the Transliteration Similarity PR only if the difference of length is not above 35%. Because of this reason, even the other two methods fail to align such words. The same is true for the second example as well. Here, it gets picked up by the TSM metric as the same endings are same but it is rejected by all the three participating similarity metrics because of the difference in the lengths of the two strings.

There are many other South Asian languages with similar characteristics to the Hindi and Gujarati languages. Given that we had to spend just a couple of hours to adapt the English-Hindi mapping into the English-Gujarati mapping (and spend another few hours to manually prepare a dataset for the experiment), and given that the scores are very encouraging, we are confident that the approach would work on other similar languages as well.

Having a Gujarati Morphological Analyser and a Transliteration similarity component, we move onto the next section where we try to integrate all of these components into an English-Gujarati word alignment pipeline.

8.5 Word Alignment Setup for the English-Gujarati Language Pair

In this section, we describe our experiments with adapting the English-Hindi word alignment algorithm to the English-Gujarati language pair.

For the English-Hindi word alignment algorithm, we have a GATE pipeline consisting of a number of GATE processing resources. Below we list the GATE components used in this pipeline:

1. English and Hindi Tokenisers and Sentence Splitters
2. English POS Tagger
3. English and Hindi Morphological Analysers
4. English Noun Chunker to form English Local Word Groups
5. English Named Entity Recognisers
6. Hindi LWG Component
7. Hindi Number Matcher (identifies words written in Hindi)
8. Dictionary Lookup Component
9. Transliteration Similarity Component
10. Group Aligner
11. Alignment Filter
12. Statistical Aligner

For the experiments with English-Gujarati language pair, we only need to replace the Hindi components with the relevant Gujarati components.

Most GATE processing resources can be applied to different languages by providing a different set of parameters for each different language. For example, gazetteers in GATE take a file definition consisting of a list of files containing strings to match in the text. The gazetteer program converts these entries into finite-state-machines (FSMs) for a faster matching process. Algorithms for converting entries into FSMs and matching over the text are language independent.

Gujarati Tokeniser and Splitter

Since the English and Gujarati languages have similar word and sentence delimiters, we use the English Unicode Tokeniser and English Sentence Splitter to identify token and sentence boundaries in the Gujarati texts.

Gujarati Morphological Analyser

As explained earlier, we now have a Gujarati Morphological analyser. We use this to find base-forms for Gujarati words.

Gujarati Number Matcher

In order to prepare a number matcher for the Gujarati language, we used the Hindi gazetteer lists (which were used to locate Hindi numbers) and replaced the Hindi numbers and prefixes with their Gujarati counterparts.

Gujarati LWG Component

In order to learn LWG rules for the Hindi language, we had used a set of English-Hindi files aligned at the word level. In case of the Gujarati language, we used the Hindi LWG rule file and adapted it to the Gujarati language simply by replacing the Hindi words with their Gujarati counterparts.

Other Alignment Components

We use the same method of filtering alignments and the same statistical method in the final pipeline component to align any remaining unaligned words in the English-Gujarati language pair. We are using an English-Gujarati dictionary (from the http://gujaratilexicon.com) and the Transliteration similarity component discussed above in section 8.4 with the English-

Gujarati character mappings.

8.5.1 Experiments with the English-Gujarati Word Alignment

Dataset

For our experiments with the English-Gujarati word alignment algorithm, we used a corpus of 500 English-Gujarati sentence pairings. The sentences in this corpus were aligned using our sentence alignment algorithm. We prepared a set of 30 compound documents, each containing approximately 15-20 sentence pairings. We asked two native speakers of the Gujarati language to use our alignment tools and prepare a gold standard for word alignment.

We used one third of the total 30 documents as the development dataset and the remaining two third for testing the English-Gujarati word alignment algorithm. The development dataset was used for updating the Gujarati-specific resources such as the LWG ruleset for the Gujarati language and character mappings for the transliteration similarity PR. We followed the same procedure we had followed to update the Hindi-specific resources (see Section A.3.2).

Results

We carried out two experiments. In the first experiment, we used the original resources, without any modifications using the development dataset, that were adapted from the Hindi resources. In this case, the word alignment algorithm produced 0.858 (precision), 0.609 (recall), 0.712 (F-Measure) and 0.305 (AER). These results are an indication of what one can expect from the word alignment pipeline when it has been adapted from a similar language such as Hindi. Our analysis of the results shows that:

- the system failed to align many nouns, verbs, vibhakties and postpositions. Unlike, Hindi, where the vibhakties (case markers in English) are used as separate words, in Gujarati they are concatenated with the preceding nouns and verbs on many occasions. For example, in the Gujarati word સમાજના (*samAjanA, of society*) the vibhakti ના (*nA ,of*) is concatenated with the Gujarati word સમાજ (*samAja, society*).

  The word ગીતમાં (*gItamaan, in the song*), is another example of illustrating the same

problem. Here, the postposition માં (*maan, in*) is concatenated with the Gujarati word ગીત (*gIta, song*).

- The Gujarati Morphological Analyser assigns a correct base-word for many Gujarati words where vibhakties and postpositions are concatenated with the preceding words. However, this means vibhakties and postpositions that are concatenated with the previous words are considered as suffixes and removed from the actual words. Therefore, such words are never seen by the Dictionary Lookup PR and never aligned.

  If the morphological analyser cannot separate vibhakties and postpositions from the preceding words because there is no appropriate rule available in the morphological analyser, the Dictionary Lookup PR has difficulty in aligning such words.

  For example, when using the Dictionary Lookup approach for locating translations of the English words *society* and *song*, we could find સમાજ (*samAja*) and ગીત (*gIta*) respectively as root forms of the Gujarati words સમાજના (*samAjanA, of society*) and ગીતમાં (*gItamaan*) but the vibhakties ના (*nA*) and the postposition માં (*maan*) are never aligned.

  If the morphological analyser can not find a proper root form for a word, it is very unlikely that it will be aligned by the Dictionary Lookup PR (see the examples of errors made by the Gujarati morphological analyser in Section on 8.3).

- We have found many cases of incorrect spellings. This include errors such as:

  - incorrect use of vowels (e.g. વસ્તૂ instead of વસ્તુ (*vastu, a thing*), શીકશક instead of શીક્ષક (*shikshak, a teacher*)),
  - missing or additional hyphen in compound words (e.g. માતા - પીતા instead of મતાપીતા (*mAtApItA, parents*)

- The Transliteration Similarity PR works very well on the English-Gujarati language pair. Most of the word pairs aligned by the Transliteration Similarity PR are correct. But because of the errors as specified in the Section 8.4, there are still quiet a few words which do not get aligned properly.

- We have a separate PR that matches numbers and tries to align them. However, we found examples of numbers (e.g. four digits years) followed by vibhakties. Such words are never processed by the morphological analyser and cannot be aligned. For example, an English sentence has a phrase "in 2002" which is translated as "2002માં .
- The LWG component had the same problem. It expects postpositions and vibhakties as separate words whereas in the case of Gujarati language, often these words are concatenated with the preceding words.
- We found many other errors that are similar to the errors found for the English-Hindi language pair. For example, alignment of idioms, improper translations, availability of multiple Gujarati translations in the Gujarati text, alignment of pronouns and compound words etc.

Although, the problems associated with conjoined vibhakties and postpositions can be addressed by introducing a simple preprocessing component to separate vibhakties and postpositions from their preceding words, it is clear from the analysis that there is a need for some subtle language specific tuning after the adaptation.

Prior to running the second experiment, we used the development dataset to update the resources. The custom dictionary contains 276 word pairings that had minimum frequency of two in the development dataset. We were also able to learn seven additional character mappings which did not exist in the English-Gujarati character mappings for the Transliteration Similarity PR. We could not learn any new rule for the LWG ruleset as the amount of data we used as the development dataset was not sufficient to learn new rules. Using the updated resources, the word alignment algorithm produced 0.869 (precision), 0.631 (recall), 0.731 (F-Measure) and 0.269 (AER).

The improvement in the results was mainly due to the additional entries added into the dictionary. When we compare results of the two experiments, it shows how using even a small amount of data to update the language-specific resources can improve the word alignment results.

It is also important to mention that in case of the English-Hindi language pair, the reference word alignment was prepared by two annotators and also reviewed by a third person. But in case of the English-Gujarati language pair, only one native speaker has been involved. Therefore, it would be interesting to see the changes in results after it has been reviewed by another person as well.

8.6 Chapter Remarks

In this chapter we presented our experiments with the Gujarati language and showed that the approaches used for the Hindi language do work for the closely related Gujarati language. We explained our experiments with the sentence alignment algorithm, morphological analyser, the transliteration similarity approach and the English-Gujarati word alignment algorithm. We analysed the results and found certain problems that affect the accuracy of these components.

As expected, the problems we found are very similar to the problems we had encountered when aligning words in the English-Hindi parallel text. Since, we use the same components (algorithms) for both the language pairs, when we fix errors, it is very likely that the errors will be resolved for both the language pairs.

Our experiments with these resources show that even with minimal effort put into the adaptation work, i.e. 11 person days in total, it is possible to achieve fairly good results. However, it is clear from the analysis that there is a need for some subtle language specific tuning after the adaptation.

Similar to the Gujarati language, languages such as Punjabi, Bihari, and Bengali are too very similar to the Hindi language. Therefore, we strongly believe that it should be possible to develop similar resources for such similar languages in a similar time frame.

In the next chapter, we conclude all the work that has been carried out as part of this thesis. We also discuss our immediate future plans relating to the components developed in this project and new components for other languages as well.

Chapter 9
Conclusion and Future Work

9.1 Outline of the Chapter

In this final chapter of the PhD thesis, we conclude our work. We revisit the aims and objectives of the thesis and compare them with the tasks accomplished.

In the future work section, we provide details of some of the tasks that can be carried out based on the work presented in this thesis and set out our plan for immediate future activities relating to the further improvement and development of the alignment framework and its components.

9.2 Conclusion

9.2.1 Validation of the Hypotheses

At the beginning of our work, we hypothesised that

- *it was possible to develop resources for one of the South Asian languages and generalize the approach to allow rapid bootstrapping of similar resources for the other closely related languages – with minimal effort and without compromising on their accuracies, and*
- *it was also possible to develop a high performance hybrid word alignment algorithm based on such language specific resources.*

To verify the hypotheses, we developed many resources specific to the English-Hindi language pair. The decision about which specific components should be developed was taken based on the following two studies carried out by us:

1. a systematic study of the Hindi language and structural differences between the English and Hindi languages (see Chapter 2), and
2. literature review of many word alignment approaches such as statistical, heuristic and hybrid for the English-Hindi and other language pairs (see Section 7.3).

We found that there are at least four language specific components which were absolute requirements for boosting English-Hindi word alignment results. These are a bilingual English-Hindi dictionary; a morphological analyser; a component to match named entities; and a component to capture and align multi word expressions. Our findings were confirmed by our analysis of the word alignment results of GIZA++ on the the English-Hindi dataset (see Section 7.4).

Therefore, we developed these resources for the Hindi language (see Chapters 5 and 6) and designed and implemented an English-Hindi hybrid word alignment system that relies on these resources (see Chapter 7). Our evaluation of the English-Hindi word alignment algorithm meets our expectations and shows that we are able to achieve better than state-of-the-art word alignment results for the English-Hindi language pair (see Section 7.7).

The thesis tackles a resource poor language setting with limited training data, which is challenging for current statistical approaches. Whether this performance gain compared to statistical methods also manifests itself for larger data sets would need to be verified by future experiments.

We generalized the approach we used for developing the English-Hindi resources and adapted these resources to the English-Gujarati language pair (see Chapter 8). The total time required to adapt the English-Hindi resources, i.e. Sentence Alignment PR, Morphological Analyser, Transliteration Similarity PR, LWG Ruleset and other components of the English-Hindi word alignment pipeline, was only 11 person days. This duration also includes the time required to prepare a small English-Gujarati word alignment gold standard (consisting of 500 sentence pairings) using the alignment framework described in this thesis.

The results show that the approach to develop language-specific resources specified in the thesis works on both the Hindi and Gujarati languages when these languages are paired with the English language and therefore the approach should also work with other closely related languages in the group too, when they are paired with the English language. We also believe that the development of resources for such similar languages can be done in similar time frame.

There are at least some of the components (such as sentence alignment algorithm and the filtering alignment component) which we think can be applied to languages other than the South Asian languages. Experimenting with other languages and developing similar resources for them is our immediate objective in the near future.

We believe that the results of individual components of both the Hindi and Gujarati languages and the word alignment systems of the English-Hindi and English-Gujarati language pairs validate both the hypotheses we had set out at the beginning of the project.

In the next section, we summarise the resources that were developed for the English-Hindi language pair and adapted to the English-Gujarati language pair.

9.2.2 Resources and Contributions

Sentence Alignment

in Chapter 4, we presented a sentence alignment algorithm for the English-Hindi language pair. Our goal was to develop an alignment system that could work across different South Asian languages when they are paired with the English language.

Our approach is similar to the one presented by Gale and Church (1993). The main difference lies in criteria used during dynamic programming for selecting alignment pairs. Similar to their algorithm, we too use a dynamic programming to choose sentences that should be aligned together in the source and target languages. However, we give priority to smaller alignment pairs. The other difference is in the types of alignment pairs that the two algorithms allow to form. For example, in Gale and Church (1993), it is possible to form pairings such as 1:0 and 0:1. In our case, it is not. Even though their algorithm identifies 1:0 and 0:1 pairs, it is important to mention that their evaluation of 1:0 and 0:1 pairs suggests that all such pairs were identified incorrectly (see Table 6 on p.11 in Gale and Church (1993)). In our algorithm, we only allow aligning one or up to four sentences in a source language with one or up to four sentences in the target language.

In our evaluation of the algorithm on a subset of English-Hindi parallel text taken from the EMILLE corpus, we show that our algorithm produces better results than Gale and Church

(1993). Furthermore, our experiments with the English-Gujarati language pair confirms that our algorithm is adaptable to other languages that are similar to the Hindi language with no additional effort.

Word Alignment

As observed in the literature review of existing word alignment methods (see Section 7.3), statistical approaches currently dominate research on word alignment. It was also observed that researchers have urged the need to go beyond statistical approaches (and employ other approaches such as discriminative models, heuristic models and hybrid approaches) when there is not enough data available to train a statistical model.

As described above, this was also confirmed by our experiments with GIZA++ on the English-Hindi language pair (see Section 7.4). It was observed that these methods struggled to go beyond an F-Measure score of 0.63 on the word alignment task. Our analysis of the results revealed that the data used for training was too sparse.

Based on our study of the Hindi language (see Chapter 2), we concluded that components that exploit morphological analysis, transliteration similarity and LWG can contribute significantly towards achieving successful alignment. Having developed these components and integrated them in our word alignment algorithm, we applied the algorithm on the English-Hindi dataset. We show that the use of such resources, in a hybrid setup, produces much better results (F-Measure = 0.78) (see Section 7.7).

Furthermore, we adapted the English-Hindi word alignment algorithm to the English-Gujarati language pair (see Section 8.5). We show that the pipeline of components in our word alignment algorithm can be reused with only minimal changes with individual components on language pairs those are similar to the English-Hindi. Replacing Hindi specific components with their Gujarati counterparts produces 0.731 F-Measure.

In order to find out an effect of better alignments on the overall translation produced by an SMT system, we carried out an experiment whereby we trained four SMT models: two based on limited amount of data and two based on larger training sets. We experimented with using alignments produced by GIZA++ and compared the output with the output of

an SMT system trained on the alignments produced by our word alignment algorithm.

Even though the alignments produced by our algorithm produced better results, the difference in the output of the MT system was not substantial. It was observed in the MT output that GIZA++ had problems aligning terms such as named entities, whereas the MT system trained on the alignments produced by our system was able to translate them correctly. However, a little improvement in the overall BLEU score is an indication that the quality of an MT system relies heavily on factors other than the quality of word alignment only. The other factors being the amount of data supplied for training, domains of the training and testing datasets etc.

Morphological Analyser

A considerable amount of work has been put into development of stemmers and morphological analysers. We reviewed several methods in our literature study on morphological analysers in Chapter 5. The majority of these approaches use hand-crafted suffix-replacement rules but a few try to discover such rules from corpora. While most of the approaches remove or replace suffixes, there are examples of derivational stemmers which are based on prefixes as well.

We proposed an approach that takes both prefixes as well as suffixes into account. Given a corpus and a dictionary, our method can be used to obtain a set of suffix-replacement rules for deriving an inflected word's base-form. We developed an approach for the Hindi language but showed that the approach is portable, at least to related languages, by adapting it to the Gujarati language (see Section 8.3). Given that the entire process of developing such a ruleset is simple and fast, our approach can be used for rapid development of morphological analysers and yet it can obtain competitive results with analysers built relying on human authored rules.

Transliteration Similarity

In Chapter 6, we presented an approach to measure the transliteration similarity of English-Hindi word pairs. Our approach has two components.

First we proposed a bi-directional mapping between one or more characters in the Devana-

gari script and one or more characters in the Roman script (pronounced as in English). This allows a given Hindi word written in Devanagari to be transliterated into the Roman script and vice-versa. Although, the initial mapping, which was obtained manually, was not exhaustive, it gave us a good start. As explained in Chapter A, the list was updated from the annotators' feedback during the preparation of English-Hindi development dataset using the alignment editor.

Second, we presented an algorithm for computing a similarity measure that is a variant of Dices coefficient measure and the LCSR measure and which also takes into account the constraints needed to match English-Hindi transliterated words. Finally, by evaluating various similarity metrics individually and together under a multiple measure agreement scenario, we showed that it is possible to obtain 0.947 accuracy in identifying English-Hindi word pairs that are transliterations. In order to assess the portability of our approach to other similar languages we adapted our system to the Gujarati language with minimal effort and showed that it obtained 0.910 accuracy (see Section 8.4).

Alignment Framework in GATE

A gold standard is an essential requirement for automatic evaluation of text alignment algorithms and approaches such as semi-automatic or incremental learning can be used to speed up the process of creating one.

We described a general framework for text alignment (see Chapter A in the Appendix) that supports manual creation of a gold-standard while in the background updating the language resources used to suggest an initial alignment. We reviewed several existing alignment tools and developed an alignment editor that allows users to setup a work flow for alignment tasks. The alignment editor consists of several features which can be used for facilitating incremental learning.

We gave details on the procedure of how the gold standard was prepared and how the existing resources were used for suggesting annotations to the users. We also provided details on how setting up a proper work flow has helped us to update our resources. For example, as reported in section 7.7, we saw a 13% increase in the number of entries matched by the updated dictionary.

In order to benefit from the existing resources of GATE, we chose to integrate our alignment framework in GATE and host all the components we develop as part of this thesis in GATE.

Our word alignment algorithm is a GATE pipeline which in addition to the components we have developed contains various existing processing resources from GATE. These existing components are used to preprocess documents and generate information that is used by other components of the word alignment algorithm. For example, the English noun phrase chunker helps the LWG component to identify multi word expressions. Similarly, the named entities suggested by the English named entity recognition system in GATE are used by the transliteration similarity component to align names of organizations, persons and locations. We believe the pipeline is a very good illustration of how components of GATE can be easily integrated and used for generating necessary information that is useful for other tasks such as word alignment.

To summarise, the decision to integrate our tools in GATE has proven to be very useful, at least for the development of English-Hindi and English-Gujarati resources. Furthermore, by integrating our components in GATE, we have made it easy for other users to use our components.

Contributions

Below, we list the contributions this project has made.

- **Parallel Corpora:** We have prepared a clean English-Hindi parallel corpus consisting of 3440 sentence pairings aligned at the word level. We have also prepared an English-Gujarati parallel corpus consisting of 500 sentence pairings aligned at the word level.
- **Word Alignment Guidelines:** We have prepared word alignment guidelines for the English-Hindi language pair.
- **Sentence Alignment:** We have developed a length-based sentence alignment algorithm that is suitable for at least the South-Asian languages when they are paired with the English language.

- **A General Framework for Text Alignment:** We have developed a general framework for text alignment that allows users to manually align text at any annotation level. The alignment framework makes it easy for users to define a work flow for alignment tasks, making it possible to train a model or update resources in the background. Such resources can be used further for automatically suggesting new alignments to the annotators.

- **Bilingual Dictionaries:** We have collected an English-Hindi bilingual dictionary. The dictionary has been cleaned up to remove many errors and extended to include many new entries from the EMILLE corpus.

- **Morphological Analysers:** We have developed an FSM (finite-state-machine) based program that facilitates quick execution of suffix-replacement rules to identify a base-form and a suffix for every inflected word given to the program. We also developed an algorithm that given a monolingual corpus and a dictionary, learns suffix replacement rules to identify base-forms and suffixes. The output of this program is a rule file that can be used along with the FSM based program. We have developed morphological analysers for the English, Hindi and Gujarati languages.

- **Transliteration Similarity:** We have developed character mappings for English-Hindi and English-Gujarati languages. We also developed a similarity metric that is suitable for matching transliteration pairs when one of the languages in question is a South Asian language. We also developed a transliteration similarity algorithm to identify if the two words are transliteration of each other.

- **Local Word Grouping:** We have developed a supervised algorithm that learns rules for identifying MWEs. For this it relies on common suffixes, prefixes and conjunctions in the training data which can be obtained when preparing a gold-standard using the alignment framework presented in this thesis.

- **Word Alignment:** We have developed a generic word alignment pipeline in GATE that if used with relevant language specific resources can be used for aligning words in parallel texts. Based on this pipeline, we have developed word alignment systems for the English-Hindi and English-Gujarati language pairs.

Progress Beyond the State-Of-The-Art

Of all the contributions mentioned above, the sentence alignment algorithm for the South Asian languages, the Hindi morphological analyser and the English-Hindi word alignment system have achieved accuracies beyond the state-of-the-art results when used on the EMILLE corpus.

To the best of our knowledge, very limited work has been reported for the Gujarati language and therefore we consider any contribution as a significant for the Gujarati language. Importantly, the results obtained for the Gujarati-specific resources are similar to their Hindi-specific counterparts.

The general framework for text alignment presented in this thesis supports manual alignment of text at different levels (e.g. word, phrase, sentence, paragraph, section, etc.). No other alignment tool that we reviewed provides such flexibility. The framework has the distinctive feature that it improves the linguistic resources it uses to perform alignment in the background while human annotators are carrying out alignment. Integration of the alignment tools in GATE has made it possible to exploit many existing text processing and information extraction components for the purpose of aligning bilingual text.

9.3 Future Work

In this section, we give details of our immediate future activities related to expanding the current work. We also list possible applications that could be developed using the resources we have presented in this thesis.

Improvements to Existing Resources

Although, the resources developed as part of this thesis have produced very encouraging results, we believe, there is a scope for improvement. For example, see section 7.7.4 for examples of improvements that can be made to the English-Hindi word alignment system.

Even though the South Asian languages are very similar to each other in structure, it is possible that there are at least a few changes, specific to a particular language, that should be made to the resources of that language. For example, in case of the English-Hindi bilingual

dictionary, we had to convert Hindi entries into regular expressions where every consonant with a nukta character was converted into an entry where the nukta character was made optional. Thus, when using the Dictionary Lookup approach, the Hindi entry matched even when the Hindi word to be matched did not have the nukta character. We believe that this kind of language-specific improvements can only be made if the language-specific resources are analysed individually and thoroughly. Therefore, we would also like to study the resources of English-Gujarati language pair in detail and make necessary changes to improve the resources.

Similarly, different languages can be similar, but there is always a possibility that some differences between them may prove to be significant. We have been able to prove that (without studying the Gujarati language in detail), the approaches developed for English-Hindi word alignment can also be used for the English-Gujarati language pair. However, understanding the differences between the Hindi and Gujarati languages can guide us in making our algorithms more generic for South Asian languages. We plan to take results of the existing resources and analyse them in detail to propose better algorithms with better scores.

Experiments with Other South-Asian Languages

Other South-Asian languages such as Bengali, Punjabi and Marathi are similar to the Hindi language. Having successfully developed resources for the Gujarati language, we aim to do similar experiments with other languages as well and produce more resources.

Using Word Alignment Results in Other Applications

- **POS Tagging by Projection:** Yarowsky et al. (2001) introduced a method for developing a POS tagger by projecting tags across bilingual parallel word aligned corpora. They applied existing text analysis tools for English to bilingual text corpora and projected their output onto the second language via statistically derived word alignments. They show that simple direct annotation projection is quite noisy. However, if the quality of alignment is good, the method of annotation projection produces reliable results.

They used Brill's POS tagger (Brill, 1995) to tag a corpus of English text. Their bootstrapping algorithm is explained in two basic steps. In a first step they project the basic noun, verb, adjective, adverbs, etc. to the aligned words. This is done for one-to-one alignment mapping. In the second step they map tags across the one-to-many alignments. They show that projection across one-to-many alignments is difficult as only some of the words over the target language accept the tag derived from the source word. To solve this they first project the POS tag over the phrase and then learn the local structure by probabilistic mapping. They obtained 96% accuracy when applied this method to the English-French pairs.

We aim to experiment the same method where POS tags from the English words will be projected over the Hindi aligned words. We believe, in our case, we will be in a slightly better position when projecting tags across many-to-many alignments. In our case, the LWG component has information about keywords in many-to-many alignments. The keywords are like head words in noun and verb phrases. This information can be useful when projecting tags across many-to-many alignments. However, there are many structural differences between the English and Hindi languages and therefore it is expected that the tagsets for both languages will differ from each other and thus mapping between the tagsets will be required. Which bootstrapping method (or a learning method) to use for training a model is a question that can be answered at a later stage by carrying out various experiments with different algorithms.

Finally, it will be interesting to see if the same approach can be used for other similar language pairs such as the English-Gujarati, English-Bengali, English-Punjabi and English-Marathi.

- **Named Entity Tagging by Projection:** ANNIE in GATE (as described in Chapter 3) is a rule based IE application. Having identified tokens, sentence boundaries and a part-of-speech tag for each token, it identifies individual components such as job titles, person titles, company name endings, locations etc. with the help of gazetteers. It uses regular expressions over these annotations to identify named entities such as person names, organization names, etc. With the help of alignment tools, there are at least two possible ways the extracted information in the source language can be used to do

IE in the target language:

1. Similar to the approach of POS tagging by projection, annotations identifying named entities in the source text can be directly projected onto the aligned words in the target language (Riloff et al., 2002). Then, the target annotated corpus can be used for bootstrapping a NE tagger for the target language.

2. One could use the alignments to change the annotation patterns for the target language. For example, a rule to identify university names in English looks like the following:

   {Location} "University"

   Here, the {*Location*} refers to any text annotated as *Location*. Such a pattern matches strings such as *Sheffield University* and *Manchester University*.

   From the alignment information, we can find out how *University* is translated in Hindi – महाविध्यालय (*mahA vidhyalaya*). Also, using alignments based on transliteration mappings, one could find out how locations are transliterated in Hindi and thus gazetteers can be updated to identify locations in the Hindi text. Assuming word order is the same for such patterns in both the languages, the above rule can be changed to the following and used for identifying university names in Hindi.

   {Location} "महाविध्यालया "

Chapter A
A General Framework for Text Alignment

A.1 Outline of the Chapter

Text alignment can be achieved at the document, section, paragraph, sentence or word level. Given two parallel corpora, where the first corpus contains documents in a source language and the other in a target language, the first task is to find parallel documents and align them at the document level. Then alignments can be performed at lower levels such as the section, paragraph, sentence and word level.

We wanted to provide a flexible framework that allows users to align text at any level (e.g. character, word, phrase, sentence, etc.) and yet make aligning text an easy and acceptable experience for the annotators.

In this chapter, we introduce our framework for text alignment and discuss various features and functionalities of the same. We first look at some of well-known alignment tools and discuss the various features they provide.

The main purpose of creating a new framework was to benefit from the concept of incremental or active learning and thus provide a tool that not only allows its users to prepare a gold-standard but also to build in the background a system that gradually shifts from being a completely human process into a semi-automatic process and finally into a fully-automatic process. The framework presented in this chapter, allows a user to set-up a learning task in the background while providing the facility to prepare a gold-standard.

A.2 Related Work

There are two categories of alignment tools – viewers and editors. Viewers only facilitate viewing alignments, whereas editors also allow editing or carrying out new alignments. In order to understand the various features provided by various existing alignment tools, we reviewed several of them. The tools we reviewed include *Blinker – a bitext aligner tool* presented by Melamed (1998a), *UMIACS Word Alignment Editor*[1], HandAlign[2], *Cairo* by Smith and Jahr (2000), web-based alignment interfaces known as ISA, a sentence alignment tool

[1]http://www.umiacs.umd.edu/∼nmadnani/alignment/forclip.htm
[2]http://www.cs.utah.edu/∼hal/HandAlign/

and ICA, a word alignment tool by Tiedemann (2006), Stockholm TreeAligner (Samuelsson and Volk, 2007), *Yawat* (Germann, 2008) and *I*Link* (Ahrenberg et al., 2002, 2003).

All these systems have various unique features as well as many common features. Most of them share similar visualization methods – use of links, colour-coding, horizontal-view vs vertical-view, support for longer sentences and mouse driven approaches to name a few. Some of these tools have a web-based interface which allows a user to align data virtually from anywhere.

However, most of these tools have restrictions on the type of documents they can work with. They expect these files to be formatted in specific ways. Even in interactive mode, the type of work they can automate is limited (e.g. the meta-data they can utilize for automatic word-alignment needs to be previously encoded in the files in a specific format). All of these tools are specific to word-alignment tasks except for the *ISA tool* which is specific to sentence alignment.

The nearest tool to the one presented here is that developed by Ahrenberg et al. (2002, 2003). They present a system and a method to improve the performance of word alignment by putting a human in the alignment loop. Their interface is useful for creating new alignments, reviewing existing ones and modifying existing ones. The tool can be used for configuring resources to be used by the system (e.g. for storing new links in a different resource than the original bilingual dictionary). Even though this tool is very powerful, one of the major constraints of this tool is making changes to its configuration. The tool only works with XML documents and therefore anyone wishing to process documents other than in XML format is forced to convert them to XML. Another major constraint of the tool is that it only works for word alignment. For the tool to work, one has to provide text that is already aligned at the sentence level.

A.2.1 Our Approach

Having reviewed various alignment tools, we introduce a new set of alignment tools that in addition to providing the functionalities provided by the tools reviewed, provides features which help in overcoming the difficulties one faces when using the existing tools. The tools

presented in this section have been developed in GATE and can be used for tasks such as alignment at character, word, phrase, sentence, section or even at document level. In addition, since the tools need to be interactive to support incremental learning, they not only support automatic alignment but also a way for humans to provide feedback to the system and thus allow the machine to improve. There are several issues such as what is the best way to visualise alignments and how to store alignments but before we deal with them, it is very important to look at the existing facilities provided by GATE and also study the limitations of the same.

Limitations of GATE

Below, we briefly review the core concepts of GATE (see Chapter 3 for more information) and discuss various limitations of GATE.

Core Elements

GATE comprises four main types of resources: Language Resources (LRs), Processing Resources (PRs), Visual Resources (VRs) and Controllers.

- Resources such as documents, corpora, ontology etc. are examples of LRs.
- PRs are tools that are executed over LRs to generate appropriate information (e.g. Tokeniser identifies tokens, Sentence Splitter marks sentence boundaries).
- VRs provide visual interfaces to LRs and PRs.
- Controllers, often called pipelines, consist of one or more PRs in the order a user wants them to execute on documents.

For example, a typical controller would have a tokeniser, followed by a sentence splitter followed by a part-of-speech tagger. In this case, given a document, the tokeniser tokenises the document content and produces annotations of type tokens which are then used by the sentence splitter to recognise sentence boundaries. The part-of-speech tagger uses tokens and sentences and produces part-of-speech tags for each token.

In GATE, an annotation is some data associated with a span of text represented by the span's start and end offsets in the text. Annotations can have features associated with them and are used for storing linguistic information.

Some pipelines take a document as input which is then passed between the PRs, whereas others may take a corpus as input, in which case all the documents, one by one, are passed between the PRs in the pipeline.

Thus, the standard processing arrangement within the GATE framework deals with one document at a time. All the existing PRs in GATE process one document at time.

Limitations

The basic requirement for any sort of alignment is a source and a target text. If the source and target texts appear in two different documents, the underlying components (PRs) which perform alignment need to be able to refer to them at the same time. Thus, where access to more than one parallel document is required, the need arises for PRs to accept more than one document as parameters (e.g. a Sentence Alignment PR would take a set of parallel documents and align them at the sentence level). At the same time, other types of resources such as Tokenisers and Sentence Splitters will have to process one document at a time.

One requirement in a text alignment framework is for parallel text to be grouped together. Other requirements include facilitating users with easy-to-use alignment and annotating tools (e.g. tools to help preparing a gold standard for alignment evaluation and tools that allow users to annotate co-reference across the multiple documents). The standard behaviour of the GATE PRs contradicts the above mentioned requirements. As explained above, all PRs in GATE accept one document at a time. Even a corpus pipeline which accepts a corpus as an input, considers only one document at a time. The following subsection deals with this issue and suggests a solution.

Compound Documents

We have added new functionality to GATE that allows an arbitrary number of documents (which may be translations of each other) to be grouped as a single document called a

Compound Document. Such a document can be passed around to be processed with different PRs. Within a compound document the focus may be set to one of its member documents. Thus, if the focus is set on one of the members, when a PR such a tokeniser tries to process the compound document, all the changes happen to only the member document which is in focus.

PRs such as the sentence aligner that are mostly language specific, can be explicitly instructed about which members to use for alignment. Thus, given a compound document, which appears as a single document from the system's perspective, it can be passed around to both the existing resources in GATE that can process only one document at a time and the new alignment resources which need more than one document for processing.

Storing Alignment Information

Information about the alignments between the two member texts of a compound document is stored in a special object called *Alignment*. It is stored as a document feature in the compound document itself. Since the document features are stored in a map, every object stored as a document feature needs to have a unique name that identifies that feature. There is no limit on how many features one can store, provided they all have different names. This allows for storing alignment information, carried out at different levels, in separate alignment instances. For example, if a user is carrying out alignment at the word level, he/she can store it in an alignment object with the name *word-alignment*. Similarly, sentence alignment information can be stored in an alignment object with name *sentence-alignment*. If multiple users are annotating the same document, alignments produced by different users can be stored with different names (e.g. *word-alignment-user1*, *word-alignment-user2* etc.).

Alignment objects can be used for:

- aligning and unaligning two annotations;
- checking if the two annotations are aligned with each other;
- obtaining all the aligned annotations in a document;
- obtaining all the annotations that are aligned to a particular annotation.

Alignment Editor

We have implemented an alignment editor that has several features discussed in the related work. We preserve standard ways of aligning text but at the same time provide advanced features that can be used for facilitating incremental learning. The alignment editor can be used for performing alignment at any annotation level.

Given a compound document containing a source and a target document, the alignment editor starts in the alignment viewing mode.

Alignment Viewing Mode

In this mode the texts of the two documents are shown side-by-side in parallel windows. The purpose of the alignment viewer is to highlight annotations that are already aligned. Figure A.1 shows the alignment viewer. In this case the selected documents are English and Hindi, titled as *en* and *hi* respectively.

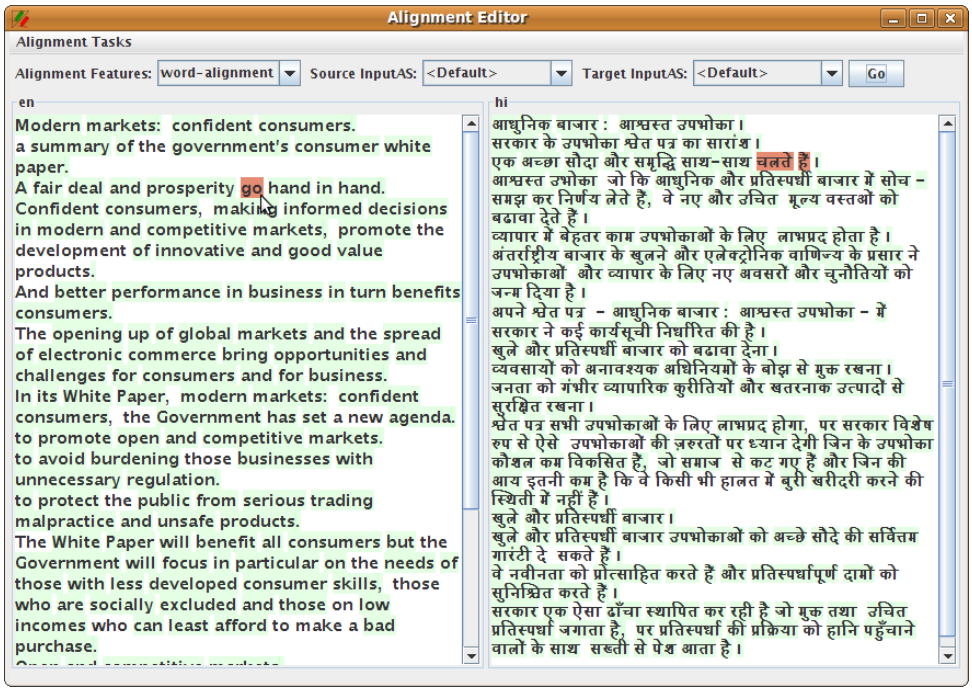


Figure A.1: Alignment Viewer

To see alignments, the user needs to select the alignment object (stored as a document

feature) that he/she wants to see alignments from. Along with this, the user also needs to select annotation sets – one for the source document and one for the target document. Given these parameters, the alignment viewer highlights the annotations that belong to the selected annotation sets and have been aligned in the selected alignment object.

When the mouse is placed on one of the aligned annotations, the selected annotation and the annotations that are aligned to the selected annotation are highlighted in red. In this case (see Figure A.1) the word *go* is aligned with the words चलते हें (*chalate hEin*).

Alignment Editing Mode

Before the alignment process can be started, the tool needs to know a few parameters of the alignment task.

- *Unit Of Alignment:* This is the annotation type that users want to perform alignment at.
- *Data Source:* Generally, if performing a word alignment task, people consider a pair of aligned sentences one at a time and align words within sentences. If the sentences are annotated, for example as *Sentence*, the *Sentence* annotation type is called *Parent of Unit of Alignment*. The *Data Source* contains information about the aligned parents of the unit of alignment. In this case, it would refer to the alignment object that contains alignment information about the annotations of type *Sentence*. The editor iterates through the aligned sentences and forms pairs of parent of unit of alignments to be shown to the user one by one. If the user does not provide any data source, a single pair is formed containing the entire documents.
- *Alignment Feature Name:* This is the name given to the alignment object where the information about new alignments is stored. The name is used as an identifier for the alignment object when stored as a document feature.

The editor comes with two different editing modes for performing alignment which the user can switch between: the horizontal view (see Figure A.2 – suitable for character, word and

phrase level alignments) and the vertical view (see Figure A.3 - suitable for annotations which have longer texts, e.g. sentences, paragraphs, sections).

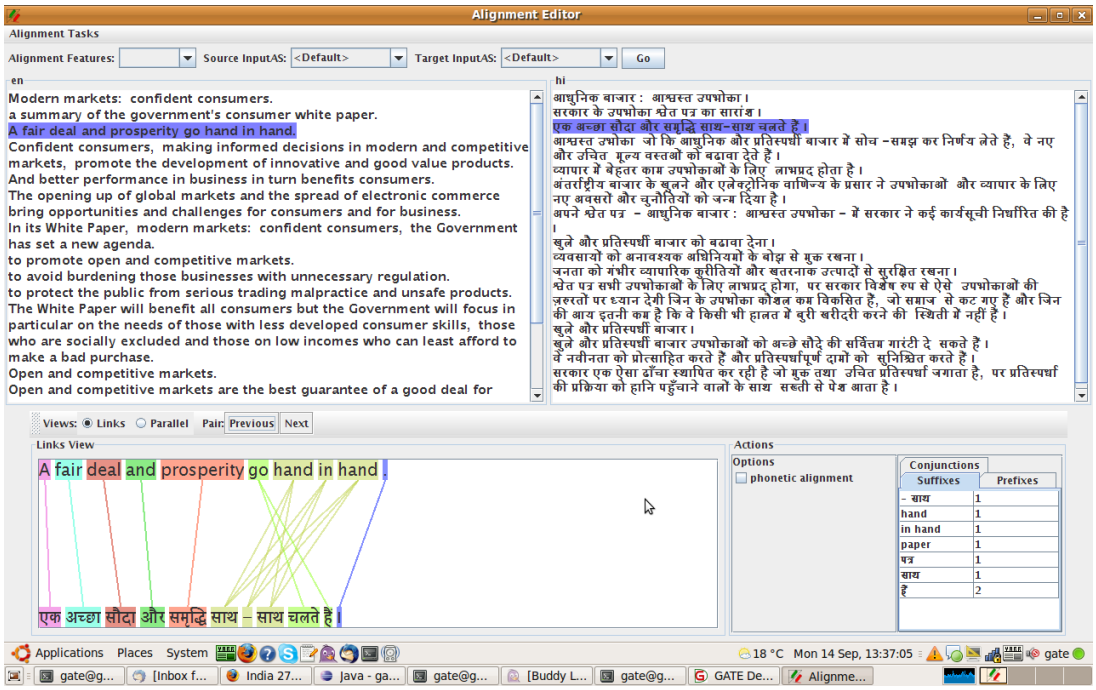


Figure A.2: Horizontal View in the Alignment Editor

Alignment Procedure

Let us assume that the user wants to align words in sentences.

To align one or more words in the source language with one or more words in the target language, the user needs to select individual words (in both texts) by clicking on them individually. Clicking on words highlights them with an identical colour. Right clicking on any of the selected words brings up an option menu with the two default options: *Reset Selection* and *Align*. Different colours are used for highlighting different pairs of alignments. This helps distinguishing one set of aligned words from other sets of aligned pairs. Also a link between the aligned words in the two texts is drawn to show the alignment. To unalign, user needs to right click on the aligned words and click on the *Remove Alignment* option. Only the word on which user right-clicks is taken out of the alignment and the rest of the words in the pair remain unaffected.

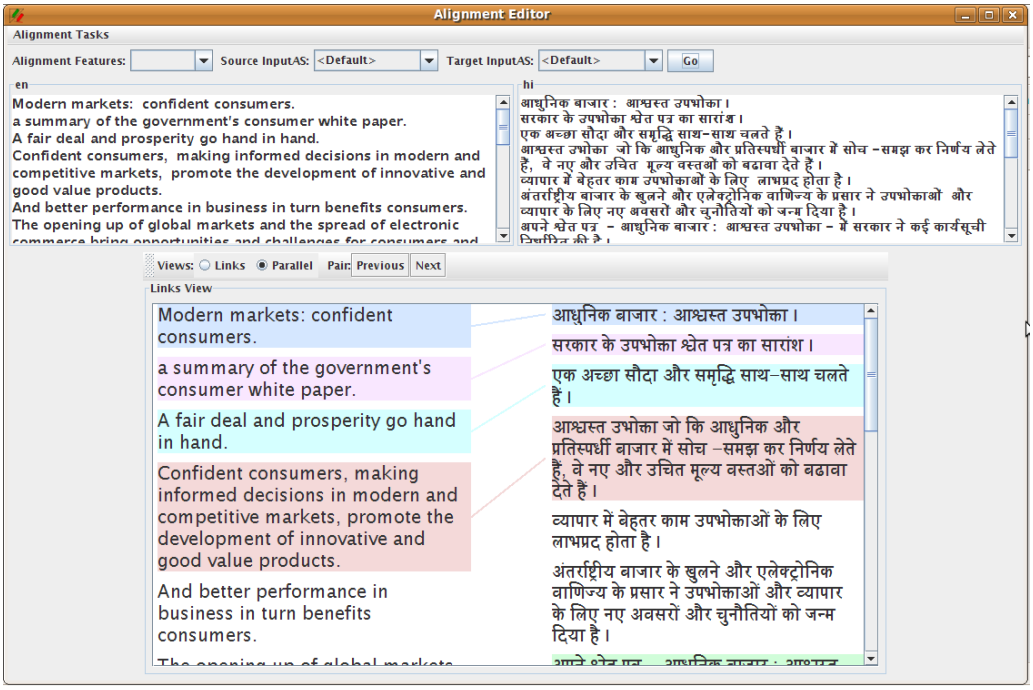


Figure A.3: Vertical View in the Alignment Editor

Advanced Features

We provide advanced features that can be used for setting up an alignment work flow and facilitating incremental learning.

In addition to the parameters required for setting up an annotation task, the user can specify three different types of actions that they want to take at different stages of the alignment process. These include:

- actions that should be taken before the user starts aligning words (PreprocessingAction);
- actions that should be taken when the user aligns annotations (AlignmentAction),
- actions that should be taken when the user has completed aligning all the units (e.g. words) in the given pair (e.g. a pair of sentence) (PostprocessingAction).

For example, to help users in the alignment process by suggesting word alignments, one

may want to wrap a pre-trained statistical word alignment model as a *PreprocessingAction*. Similarly, actions of the type *AlignmentAction* can be used for submitting examples to the model in order for the model to update itself. When all the words in a sentence pair are aligned, one may want to sign off the pair and take actions such as comparing all the alignments in that sentence pair with the alignments carried out by some other user for the same pair – this is when the registered PostprocessingActions are executed.

Additionally, the editor allows for configuring different actions for annotations in different alignment states: aligned, selected for alignment but not yet aligned and unaligned annotations not selected for alignment. Similarly, the editor also allows specific actions to be associated with any of the default alignment actions. For example, one can specify different actions that should be taken when a pair of annotations is aligned and unaligned. Such actions can be registered as *OnDemand* actions. For such actions, the editor provides a checkbox in the editor. If before aligning a pair, the user has selected the check box, the respective action is called after the pair is aligned.

A configuration file is an XML file with the required information to setup a work flow. Below we give an example of such a configuration file where the system is asked to cache all the pairs that are seen at least three times, with their root values (base-forms). The same action is used for automatically aligning word pairs whose root values appear in the cache and have not been previously aligned.

```
<?xml version="1.0"?>
<actions>
  <action>
    <jar> actions.jar </jar>
    <class> GATE.alignment.actions.DictionaryCache </class>
    <param name="filename">file://home/username/dictionary.txt</param>
    <param name="featureToUse">root</param>
    <param name="minFreq">3</param>
    <executionOrder>1</executionOrder>
  </action>
```

```
</actions>
```

The editor uses the same mechanism (as illustrated above) to provide an inbuilt caching mechanism. The idea here is to eliminate human effort required to re-annotating pairs previously seen.

Exporting Alignment Results

The editor provides different exporters to export the alignment results.

GATE XML Format

The framework allows the whole compound document to be stored in a single XML file which can be easily loaded back into GATE. Having a single XML file makes it possible to port the entire file from one destination to another easily.

XML File with Alignment Results

It is also possible to store the alignment information in a separate XML file. For example, once the annotators have aligned documents at the word level, the alignment information about both the unit and parent of unit annotations can be exported to an XML file. Figure A.4 shows an XML file with word alignment information in it.

When aligning words in sentences, it is possible to have one or more source sentences aligned with one or more target sentences in a pair. This is achieved by having *Source* and *Target* elements within the *Pair* element. Each of the *Source* and *Target* element contains one or more *Sentence* elements. Each word or token within these sentences is marked with *Token* element. Every *Token* element has a unique id assigned to it which is used when aligning words. It is possible to have one-to-one or one-to-many and many-to-one alignments. The *Alignment* element is used for mentioning every alignment pair with *source* and *target* attributes that refer to one of the source token ids and one of the target document ids respectively. For example, according to the first alignment entry, the source token *markets* with id 3 is aligned with the target token बाजार (*bAzAr*) with id 3. The exporter does not export any entry for the unaligned words.

Figure A.4: Word Alignment XML File

ACL Style

We also added an ACL style exporter that follows the guidelines as published at the `http://www.cse.unt.edu/~rada/wpt05/WordAlignment.Guidelines.txt`. The result file includes one line for each word-to-word alignment identified by the system. The lines in the results file follows the following format:

```
sentence_no  position_L1  position_L2  [S|P]  [confidence]
```

Here,

- *sentence_no* represents the id of the sentence within the test file. Where words are being aligned in sentences, the first pair of sentences is given the id 1, the second pair the id 2, the third pair the id 3 and so on.

- *position_L1* represents the position of the token that is aligned from the text in language L1; the first token in each sentence is token 1. The exporter also exports unaligned words with their position_L2 set to 0 which means the token at position L1 is aligned to null.
- *position_L2* represents the position of the token that is aligned from the text in language L2; again, the first token is token 1. Please note that the exporter also exports unaligned words with their position_L1 set to 0 which means the token at position L2 is aligned to null.
- *S|P* can be either S or P, representing a *Sure* or *Probable* alignment (see ACL guidelines for more information on this). The exporter allows users to specify the name of the feature on the annotation that it should use to look for this value.
- *confidence* is a real number, in the range 0 and 1 (1 meaning highly confident, 0 meaning not confident); this field is optional, and by default the PR assumes a confidence number of 1. As with the S|P field, the exporter allows users to specify the name of the feature on the annotation that contains the value for confidence.

A.3 Integrating Word Alignment in the Framework

We introduced our alignment tools in the previous section. In this section, we explain how different components of our word alignment algorithms are integrated in the alignment framework. We also explain how these components were used in preparing the gold-standard and give details on how different external resources were updated over the course of manual word alignment process.

The initial version of our word alignment algorithm (Aswani and Gaizauskas, 2005a) produced low recall (0.77 (precision) and 0.68 (recall)). Analysis of this system revealed that one of the reasons for low recall was insufficient data in our external resources. For example, there were several words in the English-Hindi parallel corpora, which could not be found in the dictionary. Similarly, although the transliteration mappings collected by hand were able to produce high precision, the absence of many possible mappings resulted in many possible candidates being left out. The analysis concluded that these resources needed updating.

There was already a need for a proper gold-standard for the purpose of automatic evaluation. However, work on the development of a word alignment gold-standard using the EMILLE corpus was started after the development of the initial version of the word alignment algorithm. As a result, instead of developing the gold-standard as a full manual process, the entire process emerged as a semi-automatic method in which the existing algorithm together with a human, produced a word aligned corpus in a manner that combined the strengths of each.

A.3.1 Alignment Task

Dataset

Four native speakers of the Hindi language were consulted to prepare a word-aligned gold-standard for the English-Hindi parallel subset of the EMILLE corpus. The subset contains 3440 sentence pairings in total. These pairings were distributed among 300 documents (GATE Compound documents) such that each of these document has between 10-15 sentence pairings. There are approximately 49,500 English words and 63,000 Hindi words in the English-Hindi parallel subset.

Out of the 300 documents, 100 documents were used as the development dataset to update various English-Hindi resources and the remaining 200 documents were used as the test dataset to evaluate the English-Hindi word alignment algorithm.

Preprocessing

All 300 documents were preprocessed. Sentences in the English documents were pre-processed with the English tokeniser, sentence splitter, part-of-speech tagger and the morphological analyser from the GATE system. Similarly, sentences in the Hindi documents were pre-processed with the Hindi tokeniser, sentence splitter and the morphological analyser as presented in Chapter 5.

We used an earlier version of the word alignment algorithm (Aswani and Gaizauskas, 2005a) to pre-process a sentence pairing and suggest initial alignments to the annotators.

How It Was Done

The task of preparing a word alignment gold standard for all the 300 documents was carried out in two steps. First, the annotators were asked to annotate the development dataset. It was at this time when the underlying resources were updated in the background. More information on updating the resources is provided in the next section. Second, the annotators were asked to annotate the remaining 200 documents. It was at this time when the updated resources (based on the development dataset) were used for suggesting initial alignments.

Each document was annotated by two annotators using the alignment tool described in the section A.2.1. The alignments produced by annotators were compared and only the alignments with both users' agreement were accepted. Alignment pairs on which the users could not agree were shown to a curator who was responsible for correcting any mistakes made by the annotators.

The guidelines used for preparing the word alignment gold-standard is provided in the Chapter 2.

Using Alignment Cache

As specified earlier, the alignment editor has built-in support for caching alignment entries. However, it only caches entries those are manually annotated. When the user aligns a pair of words which does not exist in the alignment cache, the editor adds the pair to the alignment cache. Similarly, if the user unaligns something that is in the cache, the entry is deleted. The alignment editor caches word pairings in their base-forms. Caching base-forms has an advantage over caching the actual surface-forms. For example, look at the following two sentence pairs:

Summary of the government's consumer white paper
सरकार के उपभोक्ता श्वेत पत्र का सारांश
(*sarakAra ke upabhoktA shveta patra kA sArAnsha*)

Modern markets : confident consumers
आधुनिक बाजार : आश्वस्त उपभोक्ता

(*Adhunika bAjAra : Ashvasta upabhoktA*)

If in the first sentence pair, a user aligns the word *consumer* with the Hindi word उपभोक्ता , the alignment editor caches their base-forms. In this case, both the English and Hindi words are in their base-forms.

Now, when the second pair is displayed, the alignment editor is given a chance to look into its cache. In this case, the alignment editor obtains the base-form of the word *consumers*, and looks into its cache for a match. The base-form of the word *consumers* is *consumer* and therefore the word *consumers* is aligned to the word उपभोक्ता .

A.3.2 Updating External Resources

In this section, we explain how different components of our word alignment algorithms are integrated in the alignment framework.

Updating the English-Hindi Bilingual Dictionary

Although the English-Hindi dictionary used for the alignment originally contained 26,000 entries, the evaluation of our first version concluded that there was a need to extend the dictionary. In order to collect missing entries in the English-Hindi dictionary, we implemented an action called *DictionaryUpdater*.

The action was registered as a *PostprocessingAction*. In other words, when users had completed aligning words in a sentence pair, all one-to-one alignments were recorded and only the entries that did not exist in the dictionary and had agreement from majority of annotators (in this case two annotators) were added to the dictionary. As a result of this, we were able to update dictionary with 2748 new words collected from development dataset.

A.3.3 Transliteration Mappings Collector

For our experiments with the English-Hindi system, we manually derived a mapping between the letters of the two scripts (Devanagari and Roman). Although this list was not exhaustive,

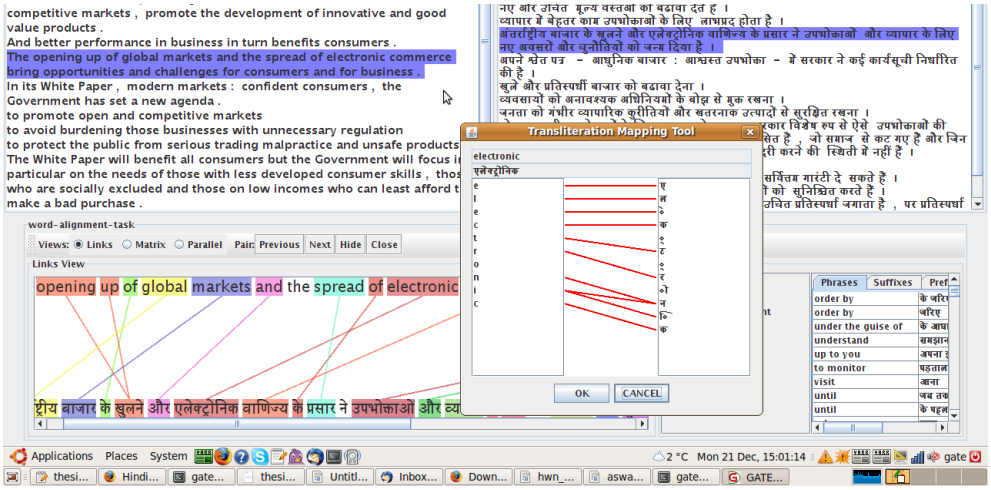


Figure A.5: Transliteration Mappings Tool

it gave us a good start. The mapping for each letter in the Devanagari script contains a sequence of one or more characters in the Roman script such that they both have identical or similar sound. Using the framework for alignment, we asked users to submit pairs which they thought were transliterations of each other.

To do this, we implemented an action called *TransliterationMappingsCollector* and registered it as an *OnDemand* action. For an implementation of the *OnDemand* action, the alignment editor provides a checkbox. If before aligning a pair, the user has checked the check box, the associated action is called after the pair is aligned.

In the case of the *TransliterationMappingsCollector*, a window is shown to the user (see Figure A.5), which allows him to align words at the character level. The tool, based on the previous character mappings, aligns characters in the pair and the user is asked to align the unaligned characters. All the new character mappings, selected by the user, are then recorded.

Thus, feedback from users was used to gradually update the transliteration mapping. Along with the newly suggested character mappings, the tool keeps track of the words to which these mapping belong. Only the character mappings with frequency above 2 were added to the original mapping. As a result of this exercise, we were able to learn 78 new mappings.

A.3.4 LWG Members Collector

Different Hindi phrases such as खुशी (*khushI, happiness*) के (*ke, of*) गीत (*gIta, songs*) and आलु (*Alu, potatoes*) के (*ke, of*) तीखे (*tikhe, spicy*) पकोरे (*pakore, bhaji as in onion bhaji*) are aligned as many-to-many alignments on different occasions. The LWG Members Collector, leaving the first and last words aside, collects information in the form of uni-grams, bi-grams and so on up to n-grams, as specified by the parameters, and maintains a frequency table.

Here, the idea is to collect base-forms of words that often occur in the middle of LWGs (i.e. which serve as conjunctions). Later, this information is used for the LWG algorithm. For example, if the LWG algorithm finds out that there are many manually aligned examples where the Hindi word का (*kA*) or its different inflections, i.e. के (*ke*), की (*kI*), को (*ko*), etc.) appear in middle of phrasal expressions, it reasons out that the word का is a phrase conjunction. Therefore, if it comes across similar examples where the two words are separated by a word whose base-form is का(*kA*), it automatically forms a group of those words.

The action *LWGMembersCollector* also performs the task of building up a frequency table for words that commonly precede or follow phrasal conjunctions in many-to-many alignments.

For example, different Hindi phrases such as गा (*gA, sing*) रहा (*rahA, ing*) था (*thA, was*) and खा (*khA, eat*) रही (*rahI, ing*) है (*hai, is*) are aligned as many-to-many alignments on different occasions. The LWGMembersCollector collects base-forms of words from the beginning and end of many-to-many alignments and maintains a frequency table for them. For example, a frequency table for the examples illustrated here looks like following:

Preceding words

- uni-grams
  - गाना (*gAnA, to sing*) with frequency 1
  - खाना (*khAnA, to eat*) with frequency 1

- bi-grams
  - गाना (*gAnA, to sing*) रहना (*rahanA*, base-form of the word रहा (*rahA, ing*)) with frequency 1
  - खाना (*khAna, to eat*) रहना (*rahanA*, base-form of the word रही (*rahI, ing*)) with frequency 1

Following words

- uni-grams
  - होना (*honA*, base-form of the words था (*thA, was*) and है (*hE, is*)) with frequency 2
- bi-grams
  - रहना (*rahanA*, base-form of the words रहा (*rahA*) and रही (*rahI*)) होना (*honA*, base-form of the words था (*thA, was*) and है (*hE, is*)) with frequency 2

Later, this information is used by the LWG algorithm. For example, from the frequency table above, the LWG algorithm reasons that consecutive Hindi words whose base-form values match with the pattern रहना (*rahanA*) होना (*honA*) should be grouped with the word that precedes them (e.g. चल (*chala, to walk*) रहा (*rahA, ing*) हुं (*hun, am*)).

Table A.1 shows the top 20 common words that occur at the start of phrases, in the middle of phrases and at the end of phrases. The high frequency words (above 20 frequency) are used by the LWG component (see Chapter 7).

Table A.1: English: Common Preceding, Middle and Following Words

Preceding Words				**Words in Middle**				**Following Words**			
Eng	**freq**	**Hin**	**freq**	**Eng**	**freq**	**Hin**	**freq**	**Eng**	**freq**	**Hin**	**freq**
to	1338	के	660	to	275	के	354	to	1172	के	686
a	334	हो	176	be	181	-	210	a	264	सकते	178
will	318	नहीं	145	of	107	की	199	will	189	की	176
have	172	आप	128	a	68	करने	150	be	183	-	162
may	158	सकते	128	the	55	कर	115	of	105	आप	97
in	138	काम	88	not	38	को	83	are	93	सकता	96
has	136	पता	81	been	37	बारे	79	in	91	करने	95
as	120	कर	80	it	35	का	77	may	78	बारे	91
are	102	न	79	able	30	करते	73	have	76	से	88
for	96	मदद	77	be able	30	सकता	61	do	64	कर	86
do	84	संभव	77	out	30	हो	61	to be	63	करते	85
need	78	सुनिश्चित	76	n't	28	से	57	for	59	के बारे	79
be	64	माता	65	have	26	सकते	50	the	59	न	78
Do	64	अदा	59	well	25	है	47	make	57	नहीं	71
can	62	थीक	56	able to	22	किया	41	has	49	का	64
is	62	कोई	55	make	21	न	40	not	48	सकती	64
make	58	से	54	will	21	करने के	39	by	43	को	52
should	54	र	54	in	20	सकती	36	can	41	पता	48
by	48	लागू	53	out of	20	रुप	35	will be	39	करने के	47
does	48	किसी	50	or	19	जाने	31	been	37	है	47

A.4 Chapter Remarks

In this chapter, we described a general framework for text alignment that supports manual alignment of text at different levels (e.g. word, phrase, sentence, paragraph, section etc.). The framework has the distinctive feature that it improves the linguistic resources it uses to perform alignment in the background while human annotators are carrying out alignment. We also explained how the integration of the alignment framework in GATE allows us to take advantage of its existing resources such as tokenisers, POS taggers, etc. We also explained the advanced features of the alignment editor that can be used for facilitating incremental learning.

We explained the usage of our word alignment algorithm in suggesting initial alignments to our annotators and the procedure of collecting entries for a bilingual dictionary and other external resources. We also explained how different components were integrated in the system – providing good examples for anyone wishing to integrate their own components.

Chapter B
List of South-Asian Language Resources

B.1 Natural Language Processing Tools

We have conducted a survey of available resources for South Asian languages. Our focus is on the Hindi and Gujarati resources that are suitable for the text alignment purpose. For each of these resources, we provide details such as what it is, languages supported by the resource, its publisher, where to find it on the internet and its licensing terms.

1. **Title:** Vibhakti Splitter
   Description: Splits the vibhakti which is in the head of the chunk and creates new words inside the chunk.
   Languages supported: Hindi
   Publisher: Technology Development for Indian Languages
   URL(s): `http://tdil-dc.in/index.php?option=com_up-download&task=view-download-tool&view=download&toolid=308&set=1&id=61&lang=en`
   License: Available for Academic Research in India

2. **Title:** Morphological Analysers
   Description: Produces root form and other features such as gender, number and tense.
   Languages supported: Telegu, Hindi, Marathi, Kannada and Punjabi
   Publisher: Language Technologies Research Centre, Indian Institute of Information Technology
   URL(s): `http://ltrc.iiit.ac.in/showfile.php?filename=onlineServices/morph/index.htm`
   License: GPL

3. **Title:** POS Tagger
   Description: Tags words with their respective grammatical categories. For this tagger to work, input words need to be provided in the Susha font.
   Languages supported: Hindi

Publisher: Technology Development for Indian Languages

URL(s): http://tdil-dc.in/index.php?option=com_up-download&task=view-download-tool&view=download&toolid=558&set=1&id=61&lang=en

License: Available for Academic Research in India

4. **Title:** Chunker

   Description: The program groups constituents of the language.

   Languages supported: Hindi

   Publisher: Technology Development for Indian Languages

   URL(s):http://tdil-dc.in/index.php?option=com_up-download&task=view-download-tool&view=download&toolid=559&set=1&id=61&lang=en

   License: Available for Academic Research in India

5. **Title:** Tools for Chunking and Pruning

   Description: The program identifies chunks in sentences using the POS information.

   Languages supported: Indian languages

   Publisher: Technology Development for Indian Languages

   URL(s):http://tdil-dc.in/index.php?option=com_catalogue&task=viewTools&id=62&lang=en

   License: Available for Academic Research in India

6. **Title:** Word Generator

   Description: Given a root form and features such as number, gender and person, the program generates a suitable inflected word.

   Languages supported: Hindi

   Publisher: Technology Development for Indian Languages

   URL(s):http://tdil-dc.in/index.php?option=com_up-download&task=view-download-tool&view=download&toolid=560&set=1&id=61&lang=en

   License: Available for Academic Research in India

7. **Title:** A tool for capturing structural differences between source and target languages in MT
   Languages supported: Indian languages
   Publisher: Technology Development for Indian Languages
   URL(s):`http://tdil-dc.in/index.php?option=com_up-download&task=view-download-tool&view=download&toolid=304&set=1&id=62&lang=en`
   License: Available for Academic Research in India

8. **Title:** Sanchay Toolkit
   Description: An open source toolkit for developing NLP components. The toolkit has been tagged as Work In Progress. Only a few graphical utilities such as text editors, font converts, syntactic annotation tool, parse tree viewer etc. have been released so far.
   Languages supported: South Asian Languages
   Publisher: Technology Development for Indian Languages (http://anilkumarsingh.me/)
   URL(s):`http://www.sanchay.co.in`
   License: Freeware

9. **Title:** Head Computation for Natural Language **Description:** A tool for calculating heads of chunks.
   Languages supported: South Asian Languages
   Publisher: Technology Development for Indian Languages
   URL(s):`http://tdil-dc.in/index.php?option=com_up-download&task=view-download-tool&view=download&toolid=564&set=1&id=62&lang=en`
   License: Available for Academic Research in India

10. **Title:** Intra and Inter Chunk agreements
    Languages supported: Hindi
    Publisher: Technology Development for Indian Languages

URL(s):http://tdil-dc.in/index.php?option=com_up-download&task=view-download-tool&view=download&toolid=562&set=1&id=68&lang=en
License: Available for Academic Research in India

11. **Title:** Shallow Parsers
Description: Shallow parsers for producing features such as root forms, POS tags, person, gender and number information.
Languages supported: Bengali, Hindi, Marathi, Punjabi, Tamil, Telugu, Urdu
Publisher: Technology Development for Indian Languages
URL(s):http://tdil-dc.in/index.php?option=com_catalogue&task=viewTools&id=60&lang=en
License: Available for Academic Research in India

12. **Title:** POS Tagger for the Indian Languages using Hybrid Approach. **Description:** The work is still in progress.
Languages supported: Assamese, Bengali, Bodo, Gujarati, Hindi, Malayalam, Manipuri, Nepali, Oriya, Punjabi, Tamil and Urdu
Publisher: Linguistic Data Consortium for Indian Languages (LDC-IL)
URL(s):http://www.ldcil.org/workInProgress.aspx
License: No information available

13. **Title:** Various Text Processing Tools
Description: Automatic Transliterator, Frequency Analyser, Manual POS annotation tool, ISCII to Unicode converter, Pronunciation Dictionary etc.
Languages supported: not specified but for many of the Indian languages
Publisher: Linguistic Data Consortium for Indian Languages (LDC-IL)
URL(s):http://www.ldcil.org/toolsCorpora.aspx
License: No information available

14. **Title:** Shallow Parsers
    Description: Gives the analysis of a sentence in terms of morphological analysis, POS tagging, chunking etc.
    Languages supported: Hindi, Punjabi, Urdu, Bengali, Tamil, Telugu, Kannada, Malyalam, Marathi
    Publisher: Language Technologies Research Centre, Indian Institute of Information Technology
    URL(s):`http://ltrc.iiit.ac.in/showfile.php?filename=downloads/shallow_parser.php`
    License: GPL

15. **Title:** Spell Checker for Named Entities (An Open Office Plugin)
    Languages supported: Nepali, Bodo, Manipuri, Assamese
    Publisher: Technology Development for Indian Languages
    URL(s):`http://tdil-dc.in/index.php?option=com_up-download&task=view-download-tool&view=download&toolid=556&set=1&id=61&lang=en`
    License: Freeware

16. **Title:** Local Word Grouper
    Description: Annotates groups of words as verb or noun phrases.
    Languages supported: Indian languages (no specific mention on the website)
    Publisher: Technology Development for Indian Languages
    URL(s):`http://tdil-dc.in/index.php?option=com_up-download&task=view-download-tool&view=download&toolid=304&set=1&id=62&lang=en`
    License: Available for Academic Research in India

17. **Title:** Local Word Grouper
    Description: Annotates groups of words as verb or noun phrases.
    Languages supported: Hindi
    Publisher: Language Technologies Research Centre, Indian Institute of Information

Technology
URL(s):http://ltrc.iiit.ac.in/downloads/tools/hindiLWG.tgz
License: GPL

18. **Title:** Font Converters
Description: Converts text written in Font specific encoding to ISCII (standard character coding scheme) or to Unicode.
Publisher: Language Technologies Research Centre, Indian Institute of Information Technology
URL(s):http://ltrc.iiit.ac.in/showfile.php?filename=downloads/FC-1.0/fc.html
License: GPL

B.2 Machine Translation Systems

1. **Title:** Anusaaraka Systems
Description: Machine Translation among Indian Languages. Authors have warned for incorrect grammatically structure in the output.
Languages supported: Kannada-Hindi, Marathi-Hindi, Punjabi-Hindi and Telugu-Hindi
Publisher: Language Technologies Research Centre, Indian Institute of Information Technology
URL(s):http://ltrc.iiit.ac.in/showfile.php?filename=downloads/anu/index.htm
License: GPL

2. **Title:** VAANEE A Statistical Machine Translation System
Description: Dependency Based Statistical Machine Translation system from English to Hindi. It is based on the Moses tool kit.

Languages supported: English-to-Hindi

Publisher: Language Technologies Research Centre, Indian Institute of Information Technology

URL(s):`http://ltrc.iiit.ac.in/downloads/tools/Vaanee.tgz`

License: GPL

3. **Title:** Sampark

   Description: A Machine Translation system for Indian languages. It is an experimental setup.

   Languages supported: (English/Hindi/Punjabi/Telugu/Urdu) to (Bengali/Hindi/Malayalam/Marathi/Oriya/Punjabi/Tamil/Urdu)

   Publisher: Indian Language Technology Proliferation and Deployment Centre (ILTP-DC)

   URL(s):`http://sampark.iiit.ac.in/sampark/web/index.php/content`

   License: Online experimental system

B.3 Dictionaries

1. **Title:** Bilingual Electronic Dictionaries (ISCII format)

   Description: Input is accepted in ISCII, Shusha, Roman Readable, Itrans and RomanWX format.

   Languages supported: English-Hindi, Marathi-Hindi, Kannada-Hindi, Telugu-Hindi, Punjabi-Hindi, Calita Bengali-Hindi, Dishi Bengali-Hindi, English-Telugu dictionaries, English-Hindi (UTF-8 format)

   Publisher: Language Technologies Research Centre, Indian Institute of Information Technology

   URL(s):`http://ltrc.iiit.ac.in/onlineServices/Dictionaries/Dict_Frame.html`

   License: GPL, Online

2. **Title:** A Bilingual Dictionary (windows executable)
   Languages supported: English-Hindi
   Publisher: Technology Development for Indian Languages
   URL(s):`http://ildc.in/Hindi/hdownloadhindi.html`
   License: Shareware

3. **Title:** A Bilingual Dictionary
   Description: In the domains of health and tourism. Generic version is also available. Contents of the dictionaries are provided in the ISCII format.
   Languages supported: Manipuri-English
   Publisher: Technology Development for Indian Languages
   URL(s):`http://tdil-dc.in/index.php?option=com_up-download&task=view-download-tool&view=download&toolid=786&set=1&id=69&lang=en`
   License: Available for Academic Research in India

4. **Title:** Bharatiya Bhasha Kosh
   Description: A collection of 5000 words and their corresponding translation in 14 South Asian languages. Words from day-to-day life, words pertaining to Indian culture and tradition.
   Languages supported: South Asian Languages
   Publisher: Technology Development for Indian Languages
   URL(s):`http://tdil-dc.in/index.php?option=com_up-download&task=view-download-tool&view=download&toolid=282&set=1&id=63&lang=en`
   License: Freeware

5. **Title:** Trilingual Dictionary
   Description: ISCII based, tagged with POS tags and decorated with examples for each word.
   Languages supported: Hindi, English, Malayalam
   Publisher: European Language Resources Association

URL(s):http://universal.elra.info/product_info.php?products_id=530
License: Research

6. **Title:** Ectaco Dictionaries
   Languages supported: Many languages including Hindi as the only Indian language.
   Publisher: European Language Resources Association
   URL(s):http://universal.elra.info/product_info.php?products_id=509
   License: Research

7. **Title:** Bilingual Tourism Lexicon
   Languages supported: English-Marathi, English-Hindi, English-Tamil, English-Bengali, English-Oriya and English-Urdu
   Publisher: Technology Development for Indian Languages
   URL(s):http://tdil-dc.in/index.php?option=com_up-download&task=view-download-tool&view=download&toolid=278&set=1&id=71&lang=en
   License: Available for Academic Research in India

8. **Title:** Gujarati Lexicon
   Description: Dictionary, opposites, thesaurus, idioms, proverbs, phrases.
   Languages supported: English-Gujarati, Gujarati-English, Gujarati-Hindi, Hindi-Gujarati
   Publisher: http://gujaratilexicon.com/
   URL(s):http://gujaratilexicon.com/
   License: Online

9. **Title:** Hindi-Japanese Lexicon
   Languages supported: Hindi, Japanese
   Publisher: European Language Resources Association
   URL(s):http://universal.elra.info/product_info.php?products_id=525

License: Research

10. **Title:** Multilingual Lexicon
    Languages supported: Many languages including Hindi and Urdu
    Publisher: European Language Resources Association
    URL(s):http://universal.elra.info/product_info.php?products_id=1844
    License: Research

B.4 Gazetteer Lists

1. **Title:** List of Female Nouns
   Languages supported: Hindi
   Publisher: Technology Development for Indian Languages
   URL(s):http://tdil-dc.in/index.php?option=com_up-download&task=view-download-tool&view=download&toolid=565&set=1&id=68&lang=en
   License: Available for Academic Research in India

2. **Title:** Geographical Gazetteer Lists
   Languages supported: English, Hindi, Chinese, Arabic
   Publisher: European Language Resources Association
   URL(s):http://universal.elra.info/product_info.php?products_id=390
   License: Research

3. **Title:** Various Gazetteer Lists
   Description: Foods and spices, baby names, Indian middle names, Indian words in English etc.
   Languages supported: Hindi
   Publisher: various sources on the web
   URL(s):http://www.cs.colostate.edu/~malaiya/hindilinks.html

License: Research

B.5 WordNets

1. **Title:** Hindi Wordnet
   Description: Consists of 63,800 unique words, synset (28,687 synonyms), gloss (concept description), position in ontology, hypernymiess, hyponymies, meronymies, troponymies, antonymies, entailments.
   Languages supported: Hindi
   Publisher: Technology Development for Indian Languages
   URL(s):`http://www.cfilt.iitb.ac.in/wordnet/webhwn/`
   License: Freeware

2. **Title:** Gujarati Wordnet
   Description: 500 synsets
   Languages supported: Gujarati
   URL(s): `http://universal.elra.info/product_info.php?cPath=42_45&products_id=527`
   License: Research

3. **Title:** IndoWordNet
   Description: A lexical semantic network (work is still in progress)
   Languages supported: Assamese , Bengali, Bodo, Gujarati, Hindi, Kashmiri, Manipuri, Nepali, Oriya, Sanskrit, Urdu, Hindi, Punjabi, Marathi, Bengali, Telugu, Tamil, Kannada, Malayalam
   Publisher: European Language Resources Association
   URL(s):`http://universal.elra.info/product_info.php?products_id=2296`
   License: Research

B.6 Corpora

1. **Title:** Parallel Corpora
   Languages supported: English-Hindi
   Publisher: Institute of Formal and Applied Linguistics, Charles University in Prague
   URL(s):`https://ufal-point.mff.cuni.cz/xmlui/handle/11858/00-097C-0000-0001-BD17-1?show=full`
   License: Creative Commons

2. **Title:** Agriculture Domain Parallel Corpus
   Description: 64 parallel documents and various other monolingual corpora
   Languages supported: English-Hindi-Marathi
   Publisher: Center for Indian Language Technology
   URL(s):`http://www.cfilt.iitb.ac.in/Downloads.html`
   License: Freeware

3. **Title:** Uppsala Hindi Corpus
   Description: A part of parallel treebank with approximately 100K words (sources: Bible Text, EMILLE corpus, UN Declaration of Human Rights and a Hindi novel)
   Languages supported: Hindi, English, Swedish
   Publisher: Department of Linguistics and Philology, Uppsala University
   URL(s):`http://www2.lingfil.uu.se/personal/anjusaxena/hemsida.html`
   License: Research

4. **Title:** Parallel Corpus
   Languages supported: Manipuri-Hindi
   Publisher: Technology Development for Indian Languages
   URL(s):`http://ildc.in/Hindi/hdownloadhindi.html`
   License: Available for Academic Research in India

5. **Title:** GyanNidhi Parallel Corpus
   Description: Aligned at the sentence level (sources: National Book Trust India, Sahitya Akademi, Navjivan Publication House, Publication Division, SABDA Pondicherry, Pustak Mahal). 50,000 pages per language.
   Languages supported: English, Gujarati, Hindi, Punjabi, Marathi, Bengali, Telugu, Tamil, Kannada, Malayalam, Assamese
   Publisher: European Language Resources Association
   URL(s):`http://universal.elra.info/product_info.php?products_id=1736`
   License: Research

6. **Title:** The EMILLE/CIIL Corpus
   Description: It consists of three components: monolingual (14), parallel (6) and annotated (1) corpora.
   Languages supported: Monolingual (Assamese, Bengali, Gujarati, Hindi, Kannada, Kashmiri, Malayalam, Marathi, Oriya, Punjabi, Sinhala, Tamil, Telegu and Urdu in total 92m words), Parallel (English Hindi, Bengali, Punjabi, Gujarati, Urdu 200,000 words of data for each language pair), Annotated (Urdu with POS tags)
   Publisher: European Language Resources Association
   URL(s):`http://catalog.elra.info/product_info.php?products_id=714`
   License: Research

7. **Title:** Comparable Text corpora with English
   Languages supported: Bengali (146k – size reported in number of words), Dogri (267k), Hindi (3.6m), Kannada (873k), Kodava (182k), Maithili (225k), Nepali (301k)
   Publisher: Linguistic Data Consortium for Indian Languages (LDC-IL)
   URL(s):`http://www.ldcil.org/resourcesParallelTextCorp.aspx`
   License: No information available

8. **Title:** ERDC Noida corpus
   Languages supported: Hindi

Publisher: European Language Resources Association
URL(s):http://universal.elra.info/product_info.php?products_id=310
License: Research

9. **Title:** Tagged Hindi corpus (no more information available on tags)
   Languages supported: Hindi
   Publisher: European Language Resources Association
   URL(s):http://universal.elra.info/product_info.php?products_id=314
   License: Research

10. **Title:** POS Tagged Corpora for the Indian Languages.
    Description: Approximately 80K+ words tagged in each language.
    Languages supported: Assamese, Bengali, Bodo, Gujarati, Hindi, Malayalam, Manipuri, Nepali, Oriya, Punjabi, Tamil and Urdu
    Publisher: Linguistic Data Consortium for Indian Languages (LDC-IL)
    URL(s):http://www.ldcil.org/workInProgress.aspx
    License: No information available

11. **Title:** Monolingual Corpora
    Languages supported: Assamese (6.7m – size reported in number of words), Bengali (7.2m), Bodo (1.2m), Dogri (0.2m), English (2.3m), Gujarati (4.3m), Hindi (31m), Kannada (3.5m), Kashmiri (0.9m), Kodava (0.18m), Konkani (2.18m), Maithili (2.68m), Malayalam (5.21m), Manipuri (3.51m), Nepali (6.24m), Oriya (1.04m), Punjabi (3.95m), Sanskrit (0.51m), Tamil (9.29m), Urdu (5.18m), Yarava (13k) and Telugu (1.9m)
    Publisher: Linguistic Data Consortium for Indian Languages (LDC-IL)
    URL(s):http://www.ldcil.org/resourcesTextCorp.aspx
    License: No information available

12. **Title:** Corpus of Classical Hindi Literature.
    Description: Writings of Meera, Surdasa, Tulsidas, Premchand, Rahim etc. Provided in ISCII and Shusha fonts.
    Languages supported: Hindi
    Publisher: Language Technologies Research Centre, Indian Institute of Information Technology
    URL(s):`http://ltrc.iiit.ac.in/showfile.php?filename=downloads/Classical_Hindi_Literature/index.html`
    License: GPL

13. **Title:** Monolingual Corpora (Health, Tourism)
    Languages supported: Urdu, Punjabi, Assamese, Bodo, Manipuri, Nepali, Bangla, English, Gujarati, Konakani, Malyalam, Marathi, Tamil, Telugu
    Publisher: Technology Development for Indian Languages
    URL(s):`http://ildc.in/Hindi/hdownloadhindi.html`
    License: Available for Academic Research in India

14. **Title:** Websites with English-Hindi Parallel Texts
    Languages supported: Hindi, English
    URL(s): `http://www.24dunia.com/english-news/business-news.html` (English), `http://www.24dunia.com/hindi-news/business-news.html` (Hindi), `http://daily.bhaskar.com/business/` (English), `http://business.bhaskar.com/` (Hindi), `http://rajyasabha.nic.in/rsnew/rsweb.asp` (English), `http://rajyasabhahindi.nic.in/rshindi/hindipage.asp` (Hindi), `http://rajbhasha.nic.in/dolst_eng.htm` (English), `http://rajbhasha.nic.in/dolst_hin.htm` (Hindi)
    License: Online

15. **Title:** Online Newspapers
    Languages supported: Gujarati

URL(s): http://www.gujarattimesusa.com/, http://www.divyabhaskar.co.in/, http://www.kutchmitradaily.com/, http://www.bombaysamachar.com/, http://www.gujaratsamachar.com/, http://www.sandesh.com/, http://www.akilanews.com/, http://www.sambhaav.com/, http://www.gujaratinews.co.uk/, http://www.janmabhoominewspapers.com/, http://sharooatdaily.wordpress.com/

License: Online

Bibliography

M. Abramowitz and I. Stegun. Handbook of Mathematical Functions. US Government Printing Office, New Delhi, India, 1964.

L. Ahrenberg, M. Andersson, and M. Merkel. A Simple Hybrid Aligner for Generating Lexical Correspondences in Parallel Texts. In Proceedings of the 17th International Conference on Computational Linguistics, pages 29–35, Morristown, NJ, USA, 1998. Association for Computational Linguistics.

L. Ahrenberg, M. Andersson, and M. Merkel. A System for Incremental and Interactive Word Linking. In Third International Conference on Language Resources and Evaluation, pages 485–490, 2002.

L. Ahrenberg, M. Merkel, and M. Petterstedt. Interactive Word Alignment for Language Engineering. In EACL '03: Proceedings of the 10th Conference on European Chapter of the Association for Computational Linguistics, pages 49–52, Morristown, NJ, USA, 2003. Association for Computational Linguistics. ISBN 1-111-56789-0. doi: http://dx.doi.org/10.3115/1067737.1067746.

B. Akshar, V. Chaitanya, and R. Sangal. Natural Language Processing: A Paninian Perspective. Prentice Hall of India, New Delhi, India, 1995.

Y. Al-onaizan, J. Curin, M. Jahr, K. Knight, J. Lafferty, I. D. Melamed, F. J. Och, D. Purdy, N. A. Smith, and D. Yarowsky. Statistical Machine Translation. Technical report, Final Report, JHU Summer Workshop, 1999.

H. Alshawi and S. Douglas. Learning Dependency Transduction Models from Unannotated Examples. Philosophical Transactions of the Royal Society, 358:1357–1372, 2000.

V. Ambati, S. Vogel, and J. Carbonell. Active Semi-Supervised Learning for Improving Word Alignment. In Proceedings of the NAACL HLT 2010 Workshop on Active Learning for Natural Language Processing, pages 10–17, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL http://portal.acm.org/citation.cfm?id=1860625.1860627.

S. Ananiadou. On Word Alignment Models for Statistical Machine Translation. PhD thesis, University of Rochester, New York, 2011.

N. Aswani and R. Gaizauskas. A Hybrid Approach to Align Sentences and Words in English-Hindi Parallel Corpora. In Proceedings of the ACL 2005 Workshop on Building and Using Parallel Texts: Data Driven Machine Translation and Beyond, pages 57–64, Ann Arbor, Michigan, June 2005a. Association for Computational Linguistics.

N. Aswani and R. Gaizauskas. Aligning words in English-Hindi parallel corpora. In Proceedings of the ACL Workshop on Building and Using Parallel Texts, pages 115–118, Stroudsburg, PA, USA, 2005b. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1654449.1654472.

N. Aswani and R. Gaizauskas. Evolving a General Framework for Text Alignment: Case Studies with Two South Asian Languages. In Proceedings of the International Conference on Machine Translation: Twenty-Five Years On, Cranfield, Bedfordshire, UK, November 2009.

N. Aswani and R. Gaizauskas. English-Hindi Transliteration Using Multiple Similarity Metrics. In Proceedings of the 7th Language Resources and Evaluation Conference (LREC), La Valletta, Malta, May 2010a. ELRA.

N. Aswani and R. Gaizauskas. Developing Morphological Analysers for South Asian Languages: Experimenting with the Hindi and Gujarati Languages. In Proceedings of the 7th Language Resources and Evaluation Conference (LREC), La Valletta, Malta, May 2010b. ELRA.

N. Ayan, B. Borr, and N. Habash. Multi-Align: Combining Linguistic and Statistical Tech-

niques To Improve Alignments for Adaptable MT. In Proceedings of the 6th Conference of the Association for Machine Translation in the Americas, pages 17–26, September 2004.

P. Baker, K. Bontcheva, H. Cunningham, R. Gaizauskas, O. Hamza, A. Hardie, B. D. Jayaram, M. Leisher, A. M. McEnery, D. Maynard, V. Tablan, C. Ursu, and Z. Xiao. Corpus Linguistics and South Asian Languages: Corpus Creation and Tool Development. Literary and Linguistic Computing, 19(4):509–524, 2004.

P. Balajapally, P. Pydimarri, M. Ganapathiraju, N. Balakrishnan, and R. Reddy. Multilingual Book Reader: Transliteration, Word-to-Word Translation and Full-text Translation. In Proceedings of the 13th Biennial Conference and Exhibition, Crown Tower, Melbourne, Australia, February 2008.

J. Berney and J. Perini. Word Alignment of Parallel Texts. In Proceedings of the Class of 2005 Senior Conference on Natural Language Processing, Swarthmore, Pennsylvania, USA, April 2005.

A. Bharti, V. Sriram, A. Vamshi Krishna, R. Sangal, and S. Bendre. An Algorithm for Aligning Sentences in Bilingual Corpora Using Lexical Information. In Proceedings of the International Conference on Natural Language Processing, Mumbai, India, 2002.

D. Bikel, S. Miller, R. Schwarz, and R. Weischedel. Nymble: A High-Performance Learning Name Finder. In Proceedings of the Conference on Applied Natural Language Processing-97, pages 194–201, Washington DC, 1997.

S. Bird, D. Day, J. Garofolo, J. Henderson, C. Laprun, and M. Liberman. ATLAS: A Flexible and Extensible Architecture for Linguistic Annotation. In Proceedings of the 2nd International Conference on Language Resources and Evaluation, Athens, 2000.

P. Blunsom and T. Cohn. Discriminative Word Alignment with Conditional Random Fields. In Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics, ACL-44, pages 65–72, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. doi: 10.3115/1220175.1220184. URL http://dx.doi.org/10.3115/1220175.1220184.

O. Bojar, P. Stranak, and D. Zeman. Data Issues in English-to-Hindi Machine Translation. In Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10), Valletta, Malta, May 2010. European Language Resources Association (ELRA). ISBN 2-9517408-6-7.

E. Brill. Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging. In Computational Linguistics, volume 21, pages 543–565. MIT Press, Cambridge, MA, USA, December 1995.

P. Brown, J. C. Lai, and R. Mercer. Aligning Sentences in Parallel Corpora. In Proceedings of the 29th Conference on Association of Computational Linguistics (ACL), Berkeley, California, 1991.

P. F. Brown, V. J. D. Pietra, S. A. D. Pietra, and R. L. Mercer. The Mathematics of Statistical Machine Translation: Parameter Estimation. Computational Linguistics, 19 (2):263–311, 1993. ISSN 0891-2017.

H. M. Caseli, M. A. G. Scalco, and M. G. V. Nunes. Annotation Style Guide for Lexical Alignment. Série de Relatòrios do, ICMC, 256 (NILC-TR-05-09), April 2005.

A. Ceausu, D. Stefanescu, and D. Tufis. Acquis Communautaire Sentence Alignment Using Support Vector Machines. In Proceedings of the 5th International Conference on Linguistic Resources and Evaluation (LREC-2006), pages 2134–2137, Genoa, Italy, 2006.

N. Chatterjee and S. Agrawal. Word Alignment in English-Hindi Parallel Corpus Using Recency-Vector Approach: Some Studies. In ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics, pages 649–656, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

G. Chinnappa and A. Singh. A Java Implementation of an Extended Word Alignment Algorithm Based on the IBM Models. In Proceedings of the 3rd Indian International Conference on Artificial Intelligence, pages 1897–1913, Pune, India, December 2007.

St. R. N. Clair. Managing Multilingualism in India: Political and Linguistic Manifestations. volume 26, pages 336–339. John Benjamins Publishing Company, 2002.

J. Cohen. A Coefficient of Agreement for Nominal Scales. Educational and Psychological Measurement, 20(1):37–46, 1960. URL http://epm.sagepub.com/cgi/doi/10.1177/001316446002000104.

H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics, July 2002.

H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, N. Aswani, I. Roberts, G. Gorrell, A. Funk, A. Roberts, D. Damljanovic, T. Heitz, M. A. Greenwood, H. Saggion, J. Petrak, Y. Li, and W. Peters. Text Processing with GATE (Version 6). 2011. ISBN 978-0956599315. URL http://tinyurl.com/gatebook.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. Journal of the Royal Statistical Society, 39(1):1–38, 1977.

Y. Deng and B. Zhou. Optimizing Word Alignment Combination for Phrase Table Training. In Proceedings of the ACL-IJCNLP 2009 Conference Short Papers, pages 229–232, Suntec, Singapore, August 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P/P09/P09-2058.

L. R. Dice. Measures of the Amount of Ecologic Association Between Species. Ecology, 26 (3):297–302, July 1945.

B. Dorr, L. Pearl, R. Hwa, and N. Habash. DUSTer: A Method for Unraveling Cross-Language Divergences for Statistical Word-level Alignment. In Proceedings of the 5th Conference of the Association for Machine Translation in the Americas, pages 31–43, October 2002.

E. F. Drábek and D. Yarowsky. Improving Bitext Word Alignments via Syntax-Based Reordering of English. In Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions, page 14, Morristown, NJ, USA, 2004. Association for Computational Linguistics.

T. Dunning. Accurate Methods for the Statistics of Surprise and Coincidence. Computational Linguistics, 19:61–74, March 1993. ISSN 0891-2017. URL http://portal.acm.org/citation.cfm?id=972450.972454.

M. Fishel. Simpler Is Better: Re-evaluation of Default Word Alignment Models in Statistical MT. In Proceedings of the 24th Pacific Asia Conference on Language, Information and Computation (PACLIC 24), Sendai, Japan, November 2010.

A. Fraser and D. Marcu. ISI's Participation in the Romanian-English Alignment Task. In Proceedings of the ACL Workshop on Building and Using Parallel Texts, ParaText '05, pages 91–94, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. URL http://portal.acm.org/citation.cfm?id=1654449.1654466.

P. Fung and K. W. Church. K-vec: A New Approach for Aligning Parallel Texts. In Proceedings of the 15th Conference on Computational Linguistics, pages 1096–1102, Morristown, NJ, USA, 1994. Association for Computational Linguistics.

P. Fung and K. Mckeown. Aligning Noisy Parallel Corpora Across Language Groups: Word Pair Feature Matching by Dynamic Time Warping. In Proceedings of the 1st Conference of the Association for Machine Translation in the Americas, pages 81–88, Columbia, Maryland, October 1994.

R. Gaizauskas, K Humphreys, H. Cunningham, and Y. Wilks. University of Sheffield: Description of the LaSIE System As Used for MUC-6. In Proceedings of the 6th Conference on Message Understanding, MUC6 '95, pages 207–220, Stroudsburg, PA, USA, 1995. Association for Computational Linguistics. ISBN 1-55860-402-2. URL http://dx.doi.org/10.3115/1072399.1072418.

W. Gale and K. Church. A Program for Aligning Sentences in Bilingual Corpora. Computational Linguistics, 19(1):75–102, 1993. ISSN 0891-2017.

W. A. Gale and K. W. Church. Identifying Word Correspondence in Parallel Texts. In Proceedings of the Workshop on Speech and Natural Language, pages 152–157, Morristown, NJ, USA, 1991. Association for Computational Linguistics.

Q. Gao, N. Bach, and S. Vogel. A Semi-supervised Word Alignment Algorithm with Partial Manual Alignments. In Proceedings of the Joint 5th Workshop on Statistical Machine Translation and Metrics, WMT '10, pages 1–10, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. ISBN 978-1-932432-71-8. URL http://portal.acm.org/citation.cfm?id=1868850.1868851.

U. Germann. Yawat: Yet Another Word Alignment Tool. In HLT '08: Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies, pages 20–23, Morristown, NJ, USA, 2008. Association for Computational Linguistics.

L. Gravano, P. G. Ipeirotis, H. V. Jagadish, N. Koudas, S. Muthukrishnan, L. Pietarinen, and D. Srivastava. Using q-grams in a DBMS for Approximate String Processing. IEEE Data Engineering Bulletin, 24:28–34, 2001.

R. Grishman. TIPSTER Architecture, Design Document Version 2.3 Technical Report, DARPA. 1997. URL http://www.itl.nist.gov/div894/-894.02/relatedprojects/tipster/.

M. Haruno and T. Yamazaki. High-Performance Bilingual Text Alignment Using Statistical and Dictionary Information. In Proceedings of the 34th Annual Meeting on Association of Computation Linguistics (ACL), pages 131–138, Santa Cruz, California, 1996.

M. Hepple. Independence and Commitment: Assumptions for Rapid Training and Execution of Rule-Based POS Taggers. In Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-2000), Hong Kong, October 2000.

F. Huang. Confidence Measure for Word Alignment. In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, pages 932–940, Suntec, Singapore, August 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P/P09/P09-1105.

F. Huang, S. Vogel, and A. Waibel. Extracting Named Entity Translingual Equivalence with

Limited Resources. ACM Transactions on Asian Language Information Processing, 2(2): 124–129, June 2003.

D. A. Hull. Stemming Algorithms: A Case Study for Detailed Evaluation. Journal of the American Society for Information Science, 47(1):70–84, 1996. ISSN 0002-8231.

M. A. Jaro. Advances in Record-linkage Methodology as Applied to Matching the 1985 Census of Tampa. In Journal of the American Statistical Association, volume 84, pages 414–420. Florida.

M. Kay and M. Roscheisen. Text Translation Alignment. Computational Linguistics, 19(1): 121–142, 1993.

K. Knight and J. Graehl. Machine Transliteration. Computational Linguistics, 24(4): 599–612, 1998. ISSN 0891-2017. URL http://dl.acm.org/citation.cfm?id=972764.972767.

P. Koehn. Statistical Significance Tests for Machine Translation Evaluation. In Dekang Lin and Dekai Wu, editors, Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, pages 388–395, Barcelona, Spain, July 2004. Association for Computational Linguistics.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. Moses: Open Source Toolkit for Statistical Machine Translation. In Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions, ACL '07, pages 177–180, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. URL http://portal.acm.org/citation.cfm?id=1557769.1557821.

G. Kondrak, D. Marcu, and K. Knight. Cognates Can Improve Statistical Machine Translation Models. In Human Language Technology (NAACL), 2003.

K. Krippendorff. Content Analysis: An Introduction to its Methodology. Sage Publications, Beverly Hills, 1980. URL http://books.google.co.uk/books/about/Content_Analysis.html?id=q657o3M3C8cC&redir_esc=y.

R. Krovetz. Viewing Morphology as an Inference Process. In SIGIR '93: Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 191–202, New York, USA, 1993. ACM. ISBN 0-89791-605-0.

I. Kruijff-Korbayova, K. Chvatalova, and O. Postolache. Annotation Guidelines for Czech-English Word Alignment. pages 1256–1261, 2006. URL http://www.mt-archive.info/LREC-2006-Kruijff.pdf.

A. Kulkarni. Design and Architecture of Ä̈nusaaraka– An Approach to Machine Translation. Satyam Technical Review, 3, October 2003.

N. Kumar and P. Bhattacharyya. Named Entity Recognition in Hindi Using MEMM. Technical report, Indian Institute of Technology, Bombay, India, 2006.

A. Kunchukuttan and O. P. Damani. A System for Compound Nouns Multiword Expression Extraction for Hindi. In Proceedings of 6th International Conference on Natural Language Processing (ICON 2008), Pune, India, December 2008.

V. I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. In Proceedings of the Soviet Physics Doklady 10, pages 707–710. 1966.

W. Lin and H. Chen. Backward Machine Transliteration by Learning Phonetic Similarity. In Proceedings of the 6th Conference on Natural Language Learning, volume 20 of COLING-02, pages 1–7, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1118853.1118870. URL http://dx.doi.org/10.3115/1118853.1118870.

Y. Liu, Q. Liu, and S. Lin. Log-linear Models for Word Alignment. In Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05), pages 459–466, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P/P05/P05-1057.

M. Lombard, J. Snyder-Duch, and C. C. Bracken. Content analysis in mass communication: Assessment and reporting of intercoder reliability. Human Communication Research, 28: 587–604, 2002.

A. Lopez and P. Resnik. Improved HMM Alignment Models for Languages with Scarce Resources. In Proceedings of the ACL Workshop on Building and Using Parallel Texts, pages 83–86, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1654449.1654464.

A. Lopez, M. Nossal, R. Hwa, and P. Resnik. Word-Level Alignment for Multilingual Resource Acquisition. In Proceeding of the Workshop on Linguistic Knowledge Acquisition and Representation: Bootstrapping Annotated Language Data, Las Palmas, Canary Islands, Spain, 2002.

J. Lovins. Development of a Stemming Algorithm. Mechanical Transactions - Computational Linguistics, 11:22–31, 1968.

Y. Ma, P. Lambert, and A. Way. Tuning Syntactically Enhanced Word Alignment for Statistical Machine Translation. In Proceedings of the 13th Annual Meeting of the European Association for Machine Translation (EAMT 2009), pages 250–257, Barcelona, Spain, 2009.

P. Majumder, M. Mitra, S. K. Parui, G. Kole, P. Mitra, and K. Datta. YASS: Yet Another Suffix Stripper. ACM Transanctions and Information Systems, 25(4):18, 2007. ISSN 1046-8188.

J. Martin, R. Mihalcea, and T. Pedersen. Word Alignment for Languages with Scarce Resources. In Proceedings of the ACL 2005 Workshop on Building and Using Parallel Texts: Data Driven Machine Translation and Beyond, pages 65–74, Ann Arbor, Michigan, June 2005.

A. Mayers, R. Grishman, and M. Kosaka. A Multilingual Procedure for Dictionary-Based Sentence Alignment. In Proceedings of the 3rd Conference of the Association for Machine Translation in the Americas on Machine Translation and the Information Soup, volume 1529, pages 187–198, 1998.

I. Melamed. A Portable Algorithm for Mapping Bitext Correspondence. Annual Meeting - Association for Computational Linguistics, 35:305–312, 1997a. ISSN 0736-587X.

I. D. Melamed. A Word-to-Word Model of Translation Equivalence. In Proceedings of the 35th Conference on Association for Computational Linguistics (ACL)/EACL 8, pages 490–497, 1997b.

I. D. Melamed. Manual Annotation of Translational Equivalence: The Blinker Project. Technical Report 98-07, University of Pennsylvania, Philadelphia, Pennsylvania, 1998a.

I. D. Melamed. Annotation Style Guide for the Blinker Project, version 1.0.4. Technical Report 98-06, 1998b.

I. D. Melamed. Models of Translational Equivalence Among Words. Computational Linguistics, 26:221–249, June 2000. ISSN 0891–2017.

R. Mihalcea and T. Pedersen. An Evaluation Exercise for Word Alignment. In R. Mihalcea and T. Pedersen, editors, HLT-NAACL 2003 Workshop: Building and Using Parallel Texts: Data Driven Machine Translation and Beyond, pages 1–10, Edmonton, Alberta, Canada, May 2003. Association for Computational Linguistics.

R. Moore. Fast and Accurate Sentence Alignment of Bilingual Corpora. Lecture Notes in Computer Science, pages 135–144, 2002. ISSN 0302-9743.

R. C. Moore. A Discriminative Framework for Bilingual Word Alignment. In Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing, HLT '05, pages 81–88, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. URL `http://dx.doi.org/10.3115/1220575.1220586`.

A. Natrajan, A. L. Powell, and J. C. French. Using n-grams to Process Hindi Queries with Transliteration Variations. Technical Report Technical Report CS-97-17, Department of Computer Science, University of Virginia, July 1997.

Navneet. Bal Anand, Hindi Grammar Books for Standard 5 to Standard 10. Navneet Press, India, 2001.

F. J. Och and H. Ney. Improved Statistical Alignment Models. In Proceedings of the 38th Conference on Association for Computational Linguistics (ACL), ACL '00, pages 440–447, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

F. J. Och and H. Ney. A Systematic Comparison of Various Statistical Alignment Models. Computational Linguistics, 29(1):19–51, 2003.

F. J. Och and H. Ney. The Alignment Template Approach to Statistical Machine Translation. Computational Linguistics, 30(4):417–449, 2004. ISSN 0891-2017.

A. Pandey and T. J. Siddiqui. An Unsupervised Hindi Stemmer with Heuristic Improvements. In Proceedings of the 2nd Workshop on Analytics for Noisy Unstructured Text Data, pages 99–105, New York, USA, 2008. ACM. ISBN 978-1-60558-196-5.

K. Papineni, S. Roukos, T. Ward, and W. J. Zhu. BLEU: a method for automatic evaluation of machine translation. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, ACL '02, pages 311–318, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL http://dx.doi.org/10.3115/1073083.1073135.

M. F. Porter. An Algorithm for Suffix Stripping. Program, 14(3):130–0137, 1980.

B. Pouliquen, R. Steinberger, C. Ignat, I. Temnikova, A. Widiger, W. Zaghouant, and J. Zizka. Multilingual Person Name Recognition and Transliteration. In Proceedings of the CORELA - COgnition, REpresentation, LAnguage, volume 3/3, pages 115–123, Poitiers, France, 2005.

A. Ramanathan and D. Rao. A Lightweight Stemmer for Hindi. ACM Transactions on Asian Language Information Processing (TALIP), 2(2):130–142, 2003.

L. Ramshaw and M. Marcus. Text Chunking Using Transformation-Based Learning. In Proceedings of the 3rd ACL Workshop on Very Large Corpora, 1995.

P. Ray, V. Harish, S. Sarkar, and A. Basu. Part of Speech Tagging and Local Word Grouping Techniques for Natural Language Parsing in Hindi. In Proceedings of the 1st International Conference on Natural Language Processing (ICON 2003), Mysore, India, 2003.

J. Riesa and D. Marcu. Hierarchical Search for Word Alignment. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL '10, pages

157–166, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL http://portal.acm.org/citation.cfm?id=1858681.1858698.

E. Riloff, C. Schafer, and D. Yarowsky. Inducing Information Extraction Systems for New Languages via Cross-Language Projection. In Proceedings of the 19th International Conference on Computational Linguistics, pages 1–7, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

A. Rosen. In Search of the Best Method for Sentence Alignment in Parallel Texts. In Proceedings of the 3rd International Seminar on Computer Treatment of Slavic and East European Languages, Veda, Bratislava, 2005.

S. Saha, S. Sarkar, and P. Mitra. A Hybrid Feature Set Based Maximum Entropy Hindi Named Entity Recognition. In IJCNLP-08: Proceedings of the 3rd International Joint Conference on Natural Language Processing, pages 343–349, Hyderabad, India, Junary 2008.

Y. Samuelsson and M. Volk. Alignment Tools for Parallel Treebanks. In Proceedings of GLDV Frhjahrstagung 2007, Tbingen, 2007.

R. Sangal and D. Sharma. Creating Language Resources for NLP In Indian Languages. In Proceedings of the Sharing Capability in Localisation and Human Language Technologies, Kathamandu, Nepal, January 2004.

L. Shen. Statistical LTAG Parsing. PhD thesis, Philadelphia, PA, USA, 2006. AAI3225543.

M. Shrivastava, N. Agrawal, B. Mohapatra, S. Singh, and P. Bhattacharya. Morphology Based Natural Language Processing tools for Indian Languages. In Proceedings of the 4th Annual Inter Research Institute Student Seminar in Computer Science, IIT, Kanpur, India, April 2005.

M. Simard, G. Foster, and P. Isabelle. Using Cognates to Align Sentences in Bilingual Corpora. In Proceedings of the 1993 Conference of the Centre for Advances Studies on Collaborative Research: Distributed Computing, volume 2, pages 1071–1082, Toronto, Ontario, Canada, 1993.

A. Singh and S. Husain. Comparison, Selection and Use of Sentence Alignment Algorithms for New Language Pairs. In Proceedings of the ACL 2005 Workshop on Building and Using Parallel Texts: Data Driven Machine Translation and Beyond, pages 99–106, Ann Arbor, Michigan, June 2005.

R. M. K. Sinha and A. Thakur. Machine Translation of Bi-lingual Hindi-English (Hinglish) Text. In Proceedings of the 10th Machine Translation Summit (MT Summit X), Phuket, Thailand, September 2005.

N. A. Smith and M. E. Jahr. Cairo: An Alignment Visualization Tool. In Proceedings of the 2nd International Conference on Linguistic Resources and Evaluation, 2000.

A. Ta. A Door into Hindi. Technical report, NC State University, 2004. URL `http://www.ncsu.edu/project/hindi_lessons/lessons.html`.

J. Tiedemann. Combining Clues for Word Alignment. In Proceedings of the 10th Conference on European Chapter of the Association for Computational Linguistics, pages 339–346, Morristown, NJ, USA, 2003. Association for Computational Linguistics. ISBN 1-333-56789-0.

J. Tiedemann. ISA and ICA – Two Web Interfaces for Interactive Alignment of Bitexts. In Proceedings of the 5th International Conference on Linguistic Resources and Evaluation (LREC-2006), Genoa, Italy, 2006.

D. Tufiş, A. Barbu, and R. Ion. TREQ-AL: A Word Alignment System with Limited Language Resources. In Proceedings of the HLT-NAACL 2003 Workshop on Building and Using Parallel Texts: Data Driven Machine Translation and Beyond, volume 3 of HLT-NAACL-Parallel '03, pages 36–39, 2003.

S. Venkatapathy and A. K. Joshi. Using Information about Multi-word Expressions for the Word-Alignment Task. In MWE '06: Proceedings of the Workshop on Multiword Expressions, pages 20–27, Morristown, NJ, USA, 2006. Association for Computational Linguistics. ISBN 1-932432-84-1.

S. Vogel, H. Ney, and C. Tillmann. HMM-Based Word Alignment in Statistical Translation. In Proceedings of the 16th International Conference on Computational Linguistics, pages 836–841, Copenhagen, August 1996.

S. Warwick, R. Catizone, and R. Graham. Deriving Translation Data from Bilingual Texts. In Proceedings of the 1st International Lexical Acquisition Workshop, Detroit, 1989.

P. Williams. A Short Introduction to Hindi. March 1996. URL `http://www.it-c.dk/people/pfw/hindi/index.html`.

W. E. Winkler. The State of Record Linkage and Current Research Problems. In Statistics of Income Division. Internal Revenue Service Publication R99/04, 1999.

D. Wu. Alignment. In Handbook of Natural Language Processing, pages 415–458. CRC Press, New York: Marcel Dekker, July 2000.

J. Xu and W. B. Croft. Corpus-Based Stemming Using Cooccurrence of Word Variants. ACM Transactions on Information Systems, 16(1):61–81, 1998. ISSN 1046-8188.

D. Yarowsky, G. Ngai, and R. Wicentowski. Inducing Multilingual Text Analysis Tools via Robust Projection Across Aligned Corpora. In Proceedings of the 1st International Conference on Human Language Technology Research, San Diego, California, 2001.