ABSTRACT

Title of dissertation: RESOURCE GENERATION FROM STRUCTURED DOCUMENTS FOR LOW-DENSITY LANGUAGES

Burcu Karagol-Ayan
Doctor of Philosophy, 2007

Dissertation directed by: Professor Amy Weinberg
Department of Computer Science

The availability and use of electronic resources for both manual and automated language related processing has increased tremendously in recent years. Nevertheless, many resources still exist only in printed form, restricting their availability and use. This especially holds true in low density languages or languages with limited electronic resources. For these documents, automated conversion into electronic resources is highly desirable.

This thesis focuses on the semi-automated conversion of printed structured documents (dictionaries in particular) to usable electronic representations. In the first part we present an entry tagging system that recognizes, parses, and tags the entries of a printed dictionary to reproduce the representation. The system uses the consistent layout and structure of the dictionaries, and the features that impose this structure, to capture and recover lexicographic information. We accomplish this by adapting two methods: rule-based and HMM-based. The system is designed to produce results quickly with minimal human assistance and reasonable accuracy.

The use of an adaptive transformation-based learning as a post-processor at two points in the system yields significant improvements, even with an extremely small amount of user provided training data.

The second part of this thesis presents Morphology Induction from Noisy Data (MIND), a natural language morphology discovery framework that operates on information from limited, noisy data obtained from the conversion process. To use the resulting resources effectively, however, users must be able to search for them using the root form of morphologically deformed variant found in the text. Stemming and data driven methods are not suitable when data are sparse. The approach is based on the novel application of string searching algorithms. The evaluations show that MIND can segment words into roots and affixes from the noisy, limited data contained in a dictionary, and it can extract prefixes, suffixes, circumfixes, and infixes. MIND can also identify morphophonemic changes, i.e., phonemic variations between allomorphs of a morpheme, specifically point-of-affixation stem changes. This, in turn, allows non-native speakers to perform multilingual tasks for applications where response must be rapid, and they have limited knowledge. In addition, this analysis can feed other natural language processing tools requiring lexicons.

RESOURCE GENERATION FROM STRUCTURED DOCUMENTS FOR LOW-DENSITY LANGUAGES

by

Burcu Karagol-Ayan

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2007

DEDICATION

This thesis is dedicated to my family.

ACKNOWLEDGEMENTS

First for all, I would like to thank my co-advisor David S. Doermann. He always believed in the work I did, and guided me with his advice to be a better researcher. I am grateful to him for being available whenever I needed and supporting me financially. I sincerely thank my advisor, Amy Weinberg, for her ideas and support. Her insightful comments and positive attitude helped me when I was down.

I also thank Bonnie J. Dorr for her help and advice at the first stages of my work and for being a critical reader. I thank Eileen Abels for accepting to serve in my committee and sparing her invaluable time during her move to read my dissertation. I also thank William Hanna for sparing his time during his recovery and Norbert Hornstein for being my Dean's Representative on such a short notice.

Many people worked on BRIDGE project, and Huanfeng Ma and Tandeep Sidhu deserve special thanks. I should also express my gratitude to Mathius Creutz, Murat Saraclar and Ebru Arisoy for providing me Turkish data and allowing me to use them in my experiments.

I would like to acknowledge the support my friends gave and thank Okan Kolak, Mustafa Murat Tikir, Kevser Ozer, Aydin Balkan, Rezarta Islamaj, Fatma Betul Atalay, Evren Sirin and several others. I would have felt very lonely without them in Maryland. I am especially grateful to Fusun Yaman-Sirin for listening to all my problems, giving me support and encouragement when I needed it most, and being my best friend.

I also thank Carol and John Mattimore for loving and taking care of my son, and giving me the piece of mind to do my work.

I can never thank my family enough. My parents Recep Karagol and Melek Nur Karagol made sacrifices for my education, always believed in me, and provided me love, understanding, and encouragement. I thank my brother Ahmet Karagol, my sister Ahsen Karagol, my grandmother Nuran Cinali, and my great-grandmother Fatma Zehra Cibir—who couldn't see that I finished— for their love and support. I especially thank my sister Tuba Karagol for her cheer and positive attitude. Her phone calls and visits helped me through hard times. My whole family made me feel their pride in me and I am grateful to them.

I thank my beloved husband, Necip Fazil Ayan, for accepting me despite my flows, always believing in me and loving me. He helped me to be a better researcher and person without knowing by sharing his thoughts and problems, and by being a role model in many ways.

Finally, I thank my son, Alperen Ayan—the apple of my eye, for being such a wonderful boy and bringing great joy to my life. His unconditional love makes me happy.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

Chapter 1

Introduction

In recent years, the digitization of data, such as newspapers, books, telephone books, and dictionaries, has increased tremendously. The popularity of the Internet has contributed to the availability of electronic data. Most electronic data, however, appears in only a few languages. Many languages still have very limited online resources, so these *low-density* languages are represented in scarce quantities on the web. Of the world's more than 7,000 known living languages specified in Ethnologue (Gordon, 2005), 10 languages dominate the content of Internet with 95.4% of Internet pages. The other languages comprise only 4.6% of the total web content[1]. Thousands of languages are not represented on the web at all.

We use "density" to denote the amount of computational resources available, rather than the number of speakers of any given language. By resources, we refer not only web content but also other types of electronic resources, such as dictionaries. These resources may be used by human users directly, but often they serve as the input to resources required for automatic natural language processing (NLP) applications. Resources such as lexicons establish word level correspondences for

[1]http://www.glreach.com/globstats/refs.php3

multilingual applications, linguistically annotated data (like part of speech (POS)), or morphologically annotated data used to build automatic word stemmers, analyzers, sentence analyzers. Methods exist to extract this information from parallel corpora for languages with ample electronic resources (Brill, 1995; Goldsmith, 2001). Such resources are often necessary for many NLP applications, and the absence of these resources limits the usability and access of these languages. Applications include human or machine translation, automatic speech recognition, and automatic multilingual or monolingual (for foreign language) information retrieval. Electronic resources may also be required for human-oriented tasks. For example, human translators may use electronic dictionaries and may require morphological analyzers to access the dictionaries (see below). We can divide the world's languages into three classes with regard to their densities. High-density languages have abundant resources (such as English, Mandarin Chinese, Arabic, French, German, and Spanish). A few more languages (mostly European languages) represent the medium-density languages, which have existent and growing resources. Together, high-density and medium-density languages number only 20 or 30. All other languages are low-density languages, which have scarce resources.

Maxwell and Hughes (2006) asked an important question, which describes the motivation of this thesis:

> "Given that the vast majority languages documented in the Ethnologue fall into the class of low-density languages, what should we do? Equally important, what can we realistically do?"

Two approaches can rectify the problem of the lack of resources for low-density languages:

1. Increasing available resources

2. Reducing data requirements of the data-rich methods

The first approach, increasing available resources, involves creating more data from which the required resources can be built. Different avenues have been explored for this purpose. The most obvious way, manual construction, is expensive and time-consuming, and it needs expert and native speaker knowledge. Especially for languages without a strong political, economic, or social status, supporting such an approach is rarely feasible (Maxwell and Hughes, 2006). Computer elicitation for creating certain types of resources may reduce the time, or at least force consistency, for the structure of the data, but it also requires a long time. Regardless of the density of the language, the cost of producing resources, especially linguistically annotated corpora of a substantial size, remains significant. A few automatic approaches can create more resources, such as collecting language specific web content (Ghani et al., 2001; Scannell, 2003, 2006) or building an approximate parallel corpus from web data (Resnik and Smith, 2003). Some low-density languages have some kind of computational resources, but they do not usually exist in the required format. Resnik (1999), for instance, proposed using the Bible as parallel corpus. This expands the data set somewhat, but a Bible derived corpus is still considered small for most automatic resource development methods. These automatic methods assume the existence of some data on the web for the language in question.

Unfortunately, many languages still do not have web data as an option.

The second approach reduces the large data requirements of some methods (Somers, 1997, 1998; Hughes et al., 2006). For example, developing machine translation systems that require less parallel text presents an interesting topic. Another approach involves the use of information in smaller resources or developing new methods to extract more information from smaller resources. This has produced marginal results to date.

1.1 Proposed Solution

In our approach, we examine improving the use of information in smaller resources by developing new methods to harvest data from specific resources that contain small data sets with annotations. This thesis focuses on resource generation from documents with consistent structure. The implicit information in the structure of such documents may provide more information than a corpus that consists of raw text. Moreover, we show that it is possible for non-expert users to generate resources using semi-automatic or automatic methods.

In particular, bilingual dictionaries are studied in this thesis. Reasons for choosing dictionaries as the focus of this study include:

- Printed bilingual dictionaries exist in many languages that map the low-density language to a high-density language, such as English. Thus, this is a resource available for many languages.

- Linguistically, dictionaries are useful and may provide many kinds of information, such as translation, part-of-speech, gender, how the word used in an example sentence, related inflected forms, etc. Such diverse information can be applied in different situations.

- Electronic versions of dictionaries are used both by NLP applications and humans. For example, they often provide additional evidence to better train the model parameters in statistical machine translation (Nießen and Ney, 2004). Human translators benefit from electronic bilingual dictionaries as well, given electronic dictionaries exhibit several advantages over printed dictionaries, such as faster search and retrieval, a variety of search options, exhaustiveness of electronic medium, etc.

For many low-density languages that lack computational resources, input data appears mostly in printed form, such as dictionaries. However, two challenges face us in using these printed resources: they must be converted to electronic form, and they must be parsed so users or systems can easily search the relevant information. In this thesis we will demonstrate that we can perform both these tasks (semi-automatically). In addition, we will show that information in bilingual dictionaries, when mined correctly, can assist in developing higher level NLP resources. We use information in a printed dictionary to develop a word (morphological) segmenter.

The different kinds of information dictionaries provide may be used in NLP applications. For instance, many dictionaries include POS for each entry. Applications such as machine translation or information retrieval use this information to guide the decoder or search process respectively. When no POS taggers exist for a language, the POS tagged entry words extracted from a dictionary may be used as a seed to built the POS tagger. Seeding with gold standard tags should improve the quality of the tagger, which may then be built from a smaller additional dataset. Traditional methods for building POS taggers (Church, 1988; DeRose,1988; Brill, 1995) require large data, and thus not applicable for languages with very limited or no electronic data.

Manual conversion of printed dictionaries is both costly and labor-intensive. Another option, using optical character recognition (OCR), is fast and cost-effective. However, two problems limit the use of OCR. First, not all languages have OCR to support their script and alphabet. Naturally, these languages present the target of this study: low-density languages. It is, however, possible to train an OCR for a specific language, even tune it for a specific document (Ma and Doermann, 2004b). The other limitation of OCR appears when OCR produces a text version of printed data, because it does not provide annotated data, which is what we seek. The raw text of a dictionary is not enough to extract the implicit information given in dictionary entries. A need occurs for "transition" from OCR output to usable data to make a dictionary searchable. Performing this transition manually is time-consuming and costly. In the first part of this thesis we propose methods to annotate OCR output to transform it into a usable computational resource.

We propose a generic entry tagging approach to annotate the lexicographic information in dictionary entries. The approach relies on the consistent layout and structure of dictionary pages and entries. The typography, and other formatting clues, dictionary publishers and human readers use to distinguish different types of information in each entry also can provide annotation for the entries. These clues include the keywords that indicate POS, gender, etc.; different font styles; and various separators and symbols that impose structure.

Two entry tagging methods are presented. The rule-based method depends on human-defined configuration of the entries, and the stochastic method uses Hidden Markov Models (HMMs) (Rabiner and Juang, 1986, 1989) and requires training data from the user. Both methods are generic, fast, and rely on minimal non-expert human input. The methods achieved quite good performance when evaluating three dictionaries with different characteristics, but the rule-based method proved more reliable for phrase annotation.

In order to make the entry tagging results usable in various applications, a generation module is also presented. It generates different output formats from the same tagged dictionary entries.

However, in two cases the rule-based method performed badly: it does not consider the sequential ordering of the tags, and it can produce incorrect splitting and/or merging of phrases. To overcome these problems, this thesis presents another approach that adapts transformation-based learning (TBL) (Brill, 1995) as a post-processor. TBL automatically generates a list of ordered transformation rules, tailored to the initial annotation of the rule-based method, to improve the baseline results. Moreover, TBL can improve the font style accuracy of OCR output. This post-processing method requires only a miniscule amount of user generated training data for extracting transformation rules. The experiments using two different dictionaries demonstrated that this method is adaptable and shows significant improvements on the initial font style, which affects the tagging performance and tagging results.

The entry tagging, generation, and adaptive TBL post-processing were implemented as part of the Bilingual Resource Inference and Dictionary Generation Environment (BRIDGE) tool developed at the University of Maryland. The tool has a user-friendly GUI that provides a seamless integration of the system's stages and the ability to correct errors and prepare ground truth in the processing at will. BRIDGE has undergone user and usability tests, and it has converted dictionaries in house, at the Center for Advanced Study of Language (CASL), at MITRE, and at the National Geospatial-Intelligence Agency (NGA).

Turning from data mining to higher level resource creation, we show that dictionaries can develop into adequate morphology learners. The annotated (tagged) dictionary entries may be used as a tool for human translators. However, if the language has a rich morphology, the dictionary user will unable to find the root word when searching for a morphological variant form. So, a method becomes needed to discover morphemes of the language and provide the user with the root form. Data intensive methods do not apply to low-density languages. To address this problem and show the capacity of structured resources to serve as seeds for the development of higher level analyzers, this thesis proposes a natural language morphology induction framework that operates on dictionary information. The *Morphology Induction from Noisy Data (MIND)* framework uses the headwords and examples of usage found in many dictionary entries to discover the prefixes, suffixes, circumfixes, and infixes of a language. To extract the affixes from the headword variant found in most examples of usage, three string algorithms apply, string edit distance, longest common substring, and k-difference matching algorithm. MIND can also find morphophonemic rules, in particular the ones that appear at morpheme boundaries. The experiments on two morphologically rich languages revealed that MIND can discover affixes, and the automatically induced affixes can help segment words and find root forms. In a morpheme segmentation application, MIND output achieved better results on inflectional morphology than other methods that use much more data.

1.2 Contributions

This thesis makes the following contributions:

- Demonstration of the possibility to reduce the need for large amounts of data to produce annotated, searchable resources with machine learning methods, when these methods couple with structured documents.

- Introduction and implementation of an adaptable entry tagging module that annotates dictionary entries rapidly, with minimal non-expert human input.

- Adaptation of Hidden Markov Models (HMMs) to the dictionary entry problem, using minimal training data. Training of HMMs to search for content and information type (headword, etc.) in the dictionary.

- Introduction of a flexible generation component capable of producing different formats, targeting different applications from the same annotated dictionary entries.

- Adaptation of transformation-based learning (TBL) to a new field in NLP, using limited training data.
- Introduction and implementation of a new framework for robust morpheme discovery. The framework finds morphemes and induces morphophonemic rules using limited, noisy dictionary data, and and it needs no knowledge of the language. This presents a novel use of dictionary data are used for morpheme acquisition.
- Application of string algorithms (string edit distance, longest common substring, and k-difference string matching) in morpheme and morphophonemic rule discovery.

1.3 Thesis Layout & Brief Overview of Chapters

This dissertation proceeds as follows:

- Chapter 2 briefly surveys earlier work in the fields of resource generation for low-density languages, dictionaries in general, and electronic dictionary acquisition processes.
- Chapter 3 presents a dictionary entry tagging system to convert printed dictionaries into usable electronic representations. Two methods, a rule-based and an HMM-based stochastic method are discussed. The experiments are explained, with discussions of results.

- Chapter 4 extends the methods of Chapter 3 and presents an adaptive transformation-based error-driven learning method used to post-process at two points in the system, improving the initial results of the entry tagging system. The system works with a limited training data, and experiments on different data sets demonstrate the robustness of the method and confirm the improvements of this method.

- Chapter 5 begins with background in morphology and NLP. This chapter then presents a new method, *Morphology Induction from Noisy Data (MIND)*, a natural language morphology discovery framework to discover morphemes from the limited, noisy data obtained from a dictionary with string searching algorithms.

- Chapter 6 provides discussion and conclusions, states limitations of the methods, and outlines a number of areas for future work.

Chapter 2

Related Work

This thesis aims to help the resource generation process for languages with scarce electronic resources. Resources consist of input needed for high level NLP applications, such as machine translation (MT) and speech recognition. The resources required range from large parallel texts needed for statistical MT systems to bilingual lexicons (and the electronic rendering of dictionaries needed to induce these lexicons), and beyond. For example, a rule-based MT system may require a morphological analyzer or a syntactic rule generation component. Resources may also be required for non-automatic tasks. For example, we have found that human translations benefit from an electronic dictionary and morphological analyzer that can map words in a text to related dictionary entries (Abels et al., 2005).

This chapter first discusses the work done on providing the resources for low-density languages, then describes dictionaries, specifically bilingual and electronic ones. Methods used for electronic dictionary acquisition and systems specifically designed to help translation process are also briefly described.

2.1 Resources for Foreign Language Applications

Natural Language Processing (NLP) applications may require the following enabling resources to operate:

- A lexicon that provides the correspondence between words in the source and target languages.
- Part-of-speech (POS) tags: Word and sentence analyzers require POS tags as the input because affix choice and sentence structure depends on the POS of a particular lexical item. (For example *-ing* applies only to verbs in English.)
- Morphological analysis: Segmenting words into roots and affixes is necessary for identifying the root.
- Collocational restrictions: For example in the sentence *Promise Mary John would come* the knowledge that the verb *promise* is a transitive verb that takes one object allows us to parse the sentence correctly). In some models these are given to the system by rules whereas; in some models, the systems automatically learn these rules from the data.

The problem with low-density languages arises with the acquisition of these resources, which requires either time-consuming and expensive human effort or large amounts of electronic data. When large amounts of data is unavailable, we need new approaches to work with low-density languages. Techniques like complex reasoning, which humans employ to figure out how to translate a language they do not know using a small bilingual corpus, remains too complicated to be applicable to machine learning methods (Al-Onaizan et al., 2000).

In this chapter we begin by discussing the acquisition of resources using large data sets such as parallel corpora. This section introduces some machine learning methods used to develop these resources. We will show in the following chapters how these methods can adapt to much smaller data sets given structured data. We then discuss studies on resource acquisition for low-density languages in Section 2.1.2.

In Section 2.2, we discuss the information relevant to NLP applications found in structured documents, in particular dictionaries. We will describe the properties of dictionaries and electronic dictionaries, as well as methods proposed for dictionary acquisition. This review demonstrates that methods obtained from large data sets can be employed on much smaller data, assuming we can semi-automatically employ the implicit structure of the data found in these structured resources.

2.1.1 Exploiting Large Data Sets for NLP Applications: Parallel Corpora

Statistical MT systems (such as proposed by Brown et al. (1993)) require significant amounts of parallel bilingual texts, i.e., texts accompanied by their translations in one or more languages. These systems use parallel corpora for finding the word alignments (i.e., detection of corresponding words between two sentences that are translations of each other) and for decoding. Other NLP applications utilize parallel texts, such as cross-language information retrieval (Fluhr, 1995; Oard and Dorr, 1996), extracting transfer rules or examples (Kaji et al., 1992; Meyers et al., 2000; Watanabe et al., 2000), and word-sense disambiguation (Brown et al., 1991; Gale et al., 1992; Gale and Church, 1993; Chang et al., 1996; Diab and Resnik, 2002). All of these applications need large amounts of corpus for training. In fact, several researchers have shown that more training data yield better results (Och and Ney, 2003; Koehn et al., 2003). This requirement for large data renders these approaches impractical for languages with scarce data.

Development of speech processing systems presents another NLP application that requires significant amount of resources, which limits the speech processing only to economically viable languages. Such systems require resources including pronunciation dictionaries. The methods described in the next chapters may also create pronunciation dictionaries either from general dictionaries or from special printed pronunciation dictionaries.

These NLP applications may require various resources enabling them, as described in Section 2.1, and some resources can be developed from corpora. For instance, word-alignment systems produce lexicons as a by-product. There are POS tagging algorithms (Church, 1988; Cutting et al., 1992; Merialdo, 1994; Brill, 1995), and unsupervised morphological segmentation methods (described in Section 5.3) that also depend on large corpora.

2.1.2 The Lack of Data Problem for Low-Density Languages

The methods described above depend on large data sets, which limits their application only to languages that have such data. For low-density languages, we need to discover ways to address this problem and to make low-density languages more accessible both by NLP applications and humans.

Two general approaches may overcome the lack of large amount of data for low-density languages:

1. Increasing available resources.

2. Reducing data requirements of the data-rich methods.

In the next two sections, we will discuss these approaches.

2.1.2.1 Increasing Available Resources

For a few languages, many parallel data sets are readily available in different formats, such as parliamentary proceedings (e.g., Canadian Parliament proceedings), news archives, and the Bible. Some web sources even exist, such as Wikipedia[1], for use as a parallel text source. Unfortunately, apart from the Bible, not many languages can boast such resources.

Given a lack data sets, another approach involves human induced methods. First, in some efforts (usually funded by governments) participants receive ... for creating (mostly annotated) data. One example is the EMILLE (Enabling Minority

[1]http://www.wikipedia.org/

Language Engineering)[2] project in United Kingdom. Such methods involve long term projects requiring large funds, multiple experts, and native speakers.

A number of catalogs of information about languages and their resources are also available. The Linguistic Data Consortium (LDC)[3], for instance, represents an open consortium of universities, companies, and government research laboratories. It creates, collects, and distributes speech and text databases, lexicons, and other resources for research and development purposes. One of its projects, LCTL (Less Commonly Taught Languages), targets a number of "less commonly taught languages". Its goal is to create and share resources to support additional basic research and initial technology development in these languages.

The Open Language Archives Community (OLAC)[4], another effort, provides a platform for, "creating a worldwide virtual library of language resources", on a meta-level. It allows lookup of data by type, language, etc., for all data repositories that OLAC claims as its own.

Computer-directed elicitation by a native speaker presents another method for human-induced data creation. The BOAS (Nirenburg, 1998; Nirenburg and Raskin, 1998) and the AVENUE (Probst et al., 2002; Lavie et al., 2003; Monson et al., 2006) projects offer examples of such an approach. The AVENUE project, for instance, has a specially designed elicitation tool that allows users to align words (Probst et al., 2001; Lavie et al., 2004). The Montage (Markup for ONTological Annotation and

[2]http://www.emille.lancs.ac.uk/index.php and

http://bowland-files.lancs.ac.uk/corplang/emille/

[3]http://www.ldc.upenn.edu/

[4]http://www.language-archives.org/

Grammar Engineering) project also allows manual annotation of different resources, and it especially targets documentation of grammars of underdescribed languages by field linguists (Bender et al., 2004).

Although human-induced methods can achieve success at developing a set of resources, their non-automatic nature makes them labor intensive and time consuming, therefore impractical. Also, human-generated resources usually prove not robust. Moreover, such methods depend either on bilingual speakers or expert linguists, which limits their applicability when experts are not readily available for long periods of time.

Automatic methods also exist for data creation. Several researchers have experimented with the automatic creation of corpora using web crawling (Ghani et al., 2001; Baldwin et al., 2006) or harvesting the web for bilingual texts (Resnik and Smith, 2003). The aim is collecting especially linguistically interesting resources such as translation sets and multilingual documents, on the web. However, the collected resources are often appear as just collection of texts. General web search often cannot address this issue well because it operates as a low precision activity and web search for low density languages involves even lower precision. This approach produced quite large corpora for some low-density languages (Scannell, 2003, 2006). Such a method, however, can be used only for languages with some web content. However, many low-density languages do not meet this standard, thus collecting data from the web does not apply for all languages.

Another collaborative effort, the Translingual Information Detection, Extraction and Summarization (TIDES) exercise[5], included data collection and building an MT for a new language in one month (Oard, 2003). It quickly produced a number of resources and a large body of translated text for two languages. This presents a similar approach to resource creation of numerous open source projects, with the work being done by the public. It differed in that the results were not made publicly available. The methods described in Chapter 3 were used in the TIDES exercise to produce bilingual translation lexicons.

2.1.2.2 Reducing data requirements

The second solution to the problem of data shortage in low-density languages appears when reducing the large data requirements of some methods. This can also be achieved by improving the way the information in smaller resources is used or by developing new methods to extract more information from smaller resources.

A few studies have focused on developing machine translation systems that require less parallel text (Somers, 1997, 1998). Using a controlled but small data set instead of large corpora, presents another method proposed for languages with limited resources (Nirenburg and Raskin, 1998; Sherematyeva and Nirenburg, 2000; Probst, 2002). Although these studies decrease the amount of data needed, they still require elicited data.

In another study addressing the problem of translating with scarce resources, Han (2001) proposed an approach based on genetic algorithms (Holland, 1975; Goldberg, 1989), in which translation becomes an optimization problem. However, prob-

[5]http://language.cnri.reston.va.us/TeamTIDES.html

lems arise, such as an overly simplified representation of lexical entries and running time, that need to be addressed for this approach to become practical. Moreover, it is unclear whether the fitness function and genetic algorithm operators give the best results.

Another approach investigated to address the problem of large data requirement is using semi-supervised methods. In statistical MT systems, for instance, two methods are used. In co-training, existing machine translations present a resource to bootstrap more training data and to create data automatically for new language pairs (Callison-Burch and Osborne, 2003). Selective sampling, on the other hand, attempts to reduce the number of bilingual parallel sentence pairs through the use of active and semi-supervised learning (Ronald and Barnard, 2006). Despite these interesting studies, such methods require initial annotated data. For co-training, for instance, existing translations may be used, and if such data are unavailable, co-training cannot be used.

Projection of existing resources for creating new ones has also been tried for different text analysis tools, such as POS taggers, noun-phrase bracketers, named-entity taggers, and morphological analyzers (Yarowsky et al., 2001; Yarowsky and Ngai, 2001). Although this method eliminates the need for hand-annotated training data, it relies on corpora that limit its applicability to many languages.

An increased interest exists in low-density languages. Given the methods proposed for well-studied languages require large amounts of data, a need has arisen either for new methods that work with limited data or for efficiently creating resources for languages with scarce resources. When developing language resources, a trade-off occurs between the volume of resources that can be developed and the quality of the resulting resource with the same effort. Davel and Barnard (2006) argued that, with so many resource scarce languages, usable resources in multiple languages may be more valuable than highly accurate resources in a single language. They claimed that, "expensive techniques that are appropriate for the development of highly accurate resources in the developed world may not be as appropriate within a developing-world context." This is our approach in this thesis. We look to generate resources rapidly that are not perfect but usable.

Dictionaries provide many useful information in a compact and structured format. Section 2.1 lists some resources required by some NLP applications, including lexicons found in dictionaries. Many dictionaries contain POS tags for words and sometimes list morphological variants. Automatic or semi-automatic parsing of the dictionary entries is necessary to extract the relevant information. The core of the work presented in Chapters 3 and 4 involves the dictionary entry parsing to obtain the entry information and to make dictionaries searchable by human users. We discuss dictionaries and information acquisition from dictionary in the rest of this chapter.

2.2 Dictionaries

In this section, we describe characteristics of dictionaries. A dictionary, a reference text, lists the words of a language systematically and describes the vocabulary of a language or a coherent subset of a language. It delineates the meanings of words, illustrates how they are used in context, usually indicates pronunciation, and may provide part-of-speech or other information.

The word or expression being defined (or the lexical unit) in a dictionary is called the *headword* for the entry. Each entry contains a number of pieces of information. Typically in a hierarchical structure today's dictionaries often combine all part-of-speech (POS) forms of a word within a single dictionary entry to conserve space. The headwords in a dictionary indicate the preferred spelling and the usual printed form of the lexical units (whether capitalized; whether considered foreign (and italicized) or naturalized (Landau, 2001)).

Entries contain *items* and *structural indicators*; distinguished by their respective purposes (Wiegand, 1989). An item in a dictionary entry enables the user to retrieve *lexicographic information (content)* regarding the headword. Items are traditionally regarded as the different information categories in an entry. An example of an item is the headword itself. The purpose of a structural indicator in a dictionary entry, on the other hand, assists the user in identifying and distinguishing the different items and in finding them as quickly as possible. These can be divided into two subtypes:

1. typographic (formed by using different graphical aids, e.g., italic, bold, etc.)

2. non-typographic (signs like asterisk, parentheses or punctuation marks) structural indicators.

Figure 2.1 shows some entries from a bilingual dictionary. An example of a structural indicator in this dictionary would be the use of bold font to distinguish

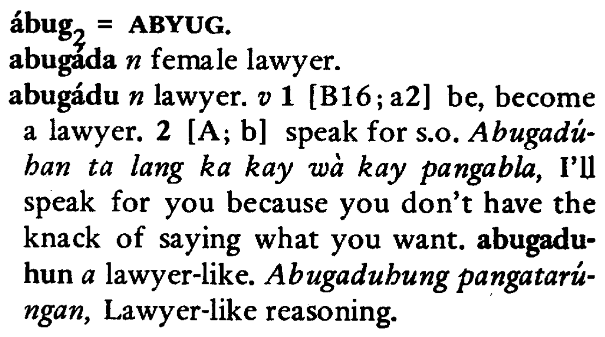

ábug$_2$ = ABYUG.
abugáda *n* female lawyer.
abugádu *n* lawyer. *v* **1** [B16; a2] be, become a lawyer. **2** [A; b] speak for s.o. *Abugadúhan ta lang ka kay wà kay pangabla,* I'll speak for you because you don't have the knack of saying what you want. **abugaduhun** *a* lawyer-like. *Abugaduhung pangatarúngan,* Lawyer-like reasoning.

Figure 2.1: Entry Samples from a Bilingual Cebuano-English Dictionary

headwords and derived words from other types of information. Other structural indicators used in this dictionary involve normal font for translations and italic font for part-of-speech (POS) and example of usage. Another feature distinguishes POS using special abbreviations (i.e., keywords) such as *a*, *n*, *v*, etc. An equal sign (=) before a word indicates a synonym, while a period (.) follows translations. A numbering system (**1, 2**) identifies translations with different POS. The methods proposed in this thesis make use of these structural indicators to find, identify, and interpret lexicographic information given in an entry.

The range and type of information within an entry will vary according to the dictionary and kind of headword. The typical information found in entries of a monolingual dictionary can be identified as follows (Jackson, 2002):

- **Spelling:** The headword indicates the normal spelling, but variations will follow.
- **Pronunciation:** All variations are given.
- **Inflections:** Given if they are formed irregularly or some spelling adjustment

occurs.

- **Part-of-speech (POS):** Conventional abbreviations indicate this information.

- **Senses:** If a word has more than one meaning, each sense is usually numbered; If a sense, or group of senses, belong to a different POS or subclass, this is indicated before the sense(s) concerned.

- **Definition:** Each sense is defined, which is an explanation of its meaning.

- **Examples:** Given the elucidation of a sense benefits from an illustrative phrase or sentence.

- **Usage:** If a word, or a sense of the word, is restricted in its contexts of use, an appropriate label indicates this restriction.

- **Run-ons:** These include undefined derivatives, related words, idioms, phrasal verbs, usually in bold type.

- **Etymology:** The history of the word (usually a root form).

Some dictionaries include additional information, such as pictorial illustrations.

Usage covers many different kinds of information about the headword. The most common kinds of usage information include: currency or temporality; regional or geographic variation; technical or specialized terminology; taboo, sexual, and scatological usage; insult; slang; style, functional variety, or register; and status or cultural level (Landau, 2001).

Lexicographers use different approaches to express compounds and derivations in a dictionary (de Caluwe and van Santen, 2003). One approach includes the derived words as subentries, usually in an abbreviated form. For example, the derived word *helpfulness* may be given in an abbreviated form as ∼*ness* under the headword *helpful*. Another common practice, in order to save space, lists the derivative words at the end of the entries of their root forms without definitions. These are called *run-on entries*. The publishers presume that if one understands the entry word and the productive affix –usually defined in its alphabetical place as a separate entry with the POS to which they can be added– one will have no difficulty understanding the derived word. This method helps by suggesting possible combinations. A related question asks to what extent dictionaries should outline possible new combinations. While some dictionaries include many possibilities, most dictionaries contain only the most commonly derived forms.

Some dictionaries also give information concerning idiom. In idiom words are not defined in a literal sense, as in "kick the bucket". Most dictionaries prefer to list idioms under the first word, though exceptions are common (Landau, 2001). In some print dictionaries, some verbal idioms may be given as run on entries. In dictionaries where each sense has its own entry, the idiom falls within the entry giving the sense word under which the idiom appears.

This section gives general information about dictionaries. Next, we will describe the bilingual dictionaries relevant to the applications with which we are concerned in this thesis.

2.2.1 Bilingual Dictionaries

Although the techniques described in this thesis may apply to any dictionary, the main focus will look at bilingual dictionaries, being those dictionaries that provide translations of vocabulary in two languages. However, a monolingual dictionary and a bilingual one c differ not only in their number of languages but in their essential purpose. A bilingual dictionary is essentially a translation-related problem-solving tool for users with different needs (Hannay, 2003). A bilingual dictionary is also described as a list of words or expressions, usually in alphabetical order, in one language (the *source language*), for which, ideally, exact translational equivalents are given in another language (the *target language*) (Landau, 2001). The purpose aims to help someone who understands only one of the languages. More, the presumption holds one of the languages is the user's native language. Bilingual dictionaries may be unidirectional (monodirectional) or bidirectional. A monodirectional dictionary contains translations from one language into another only. A bidirectional dictionary really consists of two dictionaries, and it contains translations from one language into another and vice versa.

Theoretically bilingual dictionaries can include all the information in monolingual ones except the definition, which is replaced by the translation equivalence. However, in practice, this does not hold true most of the time. Swanepoel (2003) specifies the following as the kind of information that translation dictionaries should (ideally) provide:

- A translation equivalent for every word in the source language
- Full coverage of the vocabulary of the source language
- Grammatical, syntactic, and semantic information
- Information on language variation
- Proper names
- Special vocabulary items
- Guidance on spelling
- Guidance on pronunciation

Generally, though, bilingual dictionaries do not include all the information, or do so in an inconsistent way.

In theory, monolingual and bilingual dictionaries differ only with the one additional piece of data added to an entry, i.e., a translation equivalent. However, the differences from a content and structure point of view are quite important. No one-to-one correspondence exists between words of different languages, nor is there a simple correspondence between word senses from one language to another. Major discrepancies occur between the word senses for monolingual purposes and distinctions necessary for translation purposes. In some cases a translator would find two or more translations for a given sense, and in other cases two senses may correspond to the same translation (Armstrong-Warwick, 1995). Because of this lack of one-to-one correspondence between any two languages, common practice uses a monolingual dictionary for understanding another language instead of a bilingual dictionary.

For centuries, dictionaries could be found only in printed form. With the rise of the digital age, though, dictionaries may also be transformed into electronic format. The next section discusses the differences between printed and electronic dictionaries.

2.2.2 Electronic Dictionaries

The last few decades have seen a great change in dictionary format: dictionaries have appeared in electronic format. The electronic medium redefines what a dictionary should contain and opens new possibilities in how dictionaries can be used and also in how dictionary material can be organized and presented. Electronic dictionaries offer many advantages compared to printed dictionaries:

1. Speed: Electronic dictionaries can be consulted faster, and the information can be retrieved in an expedited manner.

2. Search: Printed dictionaries offer only one way of searching of information, usually alphabetically. In electronic dictionaries, on the other hand, one can follow various roots to find the information they contain. For instance, it is possible to find the opposite meaning through the antonym or a particular synonym by consulting a list of synonyms (van Sterkenburg, 2003). Moreover, multiword entities, such as compounds or idioms can be searched as lexical entities in electronic dictionaries. The user does not have to know the headword

with which it is stored as long as the information is indexed.

3. Chaining or hyperlinking: This search mechanism calls up another dictionary entry and takes a dictionary user directly to this entry of the dictionary. Hyperlinks allows users to jump to another entry of the dictionary immediately. The most obvious usage of hyperlinks comes with cross-references, which refer users from one entry to another and are quite user-unfriendly in traditional printed dictionaries. In electronic dictionaries, however, the required information can be either given on the spot, or the cross reference will become a hyperlink, so the user can simply click to the desired entry.

4. Sophisticated information exploitation; some examples include: A lookup using wildcards (i.e., searching only part of a word), searching with operators ’and’, ’or’, and ’not’, using filters to restrict some kind of information (Jackson, 2002). For lexical research purposes, the electronic medium appears more flexible and sophisticated tool than its counterpart.

5. Incorporation of “fuzzy matching” or even stemmer algorithms into matching algorithm in electronic dictionaries: The fuzzy matching allows the user to find headwords which approximately match the user’s query. Stemmers can take a morphological form of a word and return its headword form.

6. Support of a variety of word organizations besides alphabetical listing (Byrd, 1995): An electronic dictionary can support reverse dictionaries based-on spellings, rhyming dictionaries based on pronunciation, thesauri based on

meaning, even etymological dictionaries where "cognate words"[6] are neighbors.

7. Exhaustiveness of electronic medium: The traditional dictionary user must interpret all kinds of symbols and abbreviations (Oppentocht and Schutz, 2003). Symbols and abbreviations can be made explicit in electronic dictionaries because lack of space is no longer an issue. Using the tilde, for instance, to represent headword in collocations and phrases is no longer necessary. The use of new lines, color, and other typographic features could further visually distinguish between the various kinds of information. Moreover, the dictionary parts could link to a background corpus, which allows the user to check the meanings, usage, frequency, etc. formulated by the lexicographers.

8. The ease of maintenance of electronic dictionaries: The maintenance can be performed online. Electronic dictionaries can be edited on a daily basis which makes an electronic dictionary dynamic, unlike its printed version. Keeping the dictionary more current is also possible, which has particular importance in technical domains.

Electronic dictionaries offer many advantages not possible with printed dictionaries. They may also have a different kind of "user" than printed ones: computer programs, so, two types of computerized dictionaries exist: dictionaries for people and those for computer programs. The dictionary needs of humans vary from those

[6]Cognates words are pairs of tokens of different languages which, usually due to a common etymology, share obvious phonological or orthographic properties, as well as semantic properties.

of computers, with humans having a complex reasoning mechanism. Computers, on the other hand, are endowed with almost unlimited information stores and, as a result, they have perfect recall (Slocum and Morgan, 1995). Thus, human users and computer programs emphasize different aspects of lexical information and require different types of access to the data (Zampolli, 1994).

Dictionaries for human users usually present an interface at a computer terminal or workstation with which the user can request information associated with words. Dictionaries for programs, on the other hand, provide information that may only indirectly be made available at the computer system's interface. The computerized dictionaries for people provide access (usually) to enhanced information found in normal printed dictionaries (Byrd, 1995). To make a dictionary searchable, however, the information in the entries must be segmented into its parts, i.e., to parse the dictionary.

Dictionaries for programs provide lexical information for a wide variety of computer applications that support natural language user interfaces. Electronic dictionaries present a source of lexical knowledge for computer programs, especially in NLP, for purposes of, machine translation, cross-language information retrieval, web crawling, and automatic summarizing. A computer program, rather than a human user, requests and uses the information. Byrd (1995) points out that, for a given language, these applications require much of the same information. Most of the dictionaries contain information, such as words, pronunciations, POS, and syntactic, and (general) semantic information, which are properties of the language not of particular applications. Nevertheless, each application does have unique additional information requirements. Furthermore, different systems can have vastly different formats for representing their lexical requirements.

Although the main difference between electronic and printed dictionaries are the medium in which they are presented, they have different capabilities and features. In the next section, the acquisition of electronic dictionaries is discussed.

2.2.3 Electronic Dictionary Acquisition

In this section, we discuss methods to turn paper dictionaries into useful resources for automated or non-automated applications such as translation or speech recognition. Electronic dictionaries can have different formats, and many different approaches are used to acquire electronic dictionaries.

Electronic dictionary generally refers to a dictionary in electronic form. An *online dictionary* will refer to dictionaries for humans that can be consulted through the network. Computers use the *Machine Readable Dictionary* (MRD) and *lexicon*. An MRD is the text of a dictionary readable by a computer, and a lexicon contains information required by an NLP application and in a format usable by computer applications. In most instances, the MRD is simply a computer tape that the dictionary publisher provides to a printer for typesetting. It contains the text of the dictionary with special commands to determine page formatting and font changes which signal different parts of the entry. With considerable cleaning and re-coding, these rough typesetting tapes can be converted into lexical databases suitable for electronic access. MRDs can be simple word-lists or elaborate lexicons containing grammar codes, detailed sense distinctions, and example sentences.

Two approaches happen for lexicon building: the demo approach and the book approach (Amsler, 1982; Miller, 1985a). The demo approach involves hand-coding a small but rich lexicon designed for a system with a specific purpose in mind. Every entry is constructed individually, with foreknowledge of its intended use, and of the knowledge it should contain.

The book approach attempts to develop methods for transforming the knowledge within dictionaries or encyclopedias into a format usable for NLP tasks, usually with the aim of covering as large a portion of the language as possible. Dictionary and encyclopedia entries become problematic because although they are constructed in a principled manner by professional lexicographers and encyclopedists, they are designed for human use (Wilks et al., 1988, 1990).

The demo approach had been the dominant paradigm in NLP (and AI in general) during 70s and 80s. However, interest in the book approach has expanded since 80s. Today, research is interested mainly in the book approach.

Within the book approach, three major classes of techniques have been developed for lexicon acquisition: manual construction (usually machine-aided), extraction from MRDs, and extraction from corpora. Below, each of these techniques and other alternative approaches are described.

2.2.3.1 Manual Construction of Electronic Dictionaries

Manual construction crafts a dictionary by hand. It is slow, expensive, and cumbersome. However, manual construction has two advantages (Fellbaum, 1998). First, it allows the creation of entries with the content that will prove useful in certain applications. These contents may be richer than the information extracted from standard dictionaries. Second, the manipulation of the contents for information extraction can be minimal given control of the format.

Perhaps the most popular and widely used lexical resource for NLP systems that has been manually constructed is the WordNet (Miller, 1985b). WordNet, based on psycholinguistic principles, has been developed solely for use on computers and has the structure of a 'vocabulary matrix' (Kegl, 1995). It presents an on-line representation of a major part of the English lexicon that aims to be psychologically realistic. Several studies and applications have used WordNet.

Another handcrafted resource is the CYC Project (Lenat et al., 1986; Lenat and Feigenbaum, 1987). The project aims to hand-code one million entries from an encyclopedia, taking an estimated two person/centuries of work, which some believe is a mistaken approach and waste of precious human resources (Wilks et al., 1996).

Next we will discuss how MRDs can be used for electronic dictionary acquisition.

2.2.3.2 Acquisition of Electronic Dictionaries from MRDs

The main motivation for acquiring lexicons from MRDs involves the use of already existing and constructed dictionaries. As Boguraev and Briscoe (1989) stated, existing dictionaries typically involve tens of lexicographer/years to develop. Furthermore, lexicographers have experience in dictionary creation. MRDs also provide a labor-saving resource to augment a manually constructed core lexicon (Copestake et al., 1995). However, an obstacle occurs in using the electronic versions of printed dictionaries (derived from MRDs): published dictionaries have the human reader foremost in mind, therefore they are not formatted for computer use. They can be, however, an invaluable translation aid tool.

One obstacle with machine-readable text comes when the initial text is full of inconsistencies and is not robust. Most dictionaries are compiled by hand using slips of paper with indications of what information needs to be coded (Kegl, 1995). The individual lexicographer responsible for the entry makes many of the necessary choices. Since different lexicographers generally handle different letters of the alphabet (Murray, 1977), entries within the same verb or noun class are seldom handled in the same way. Such differences prove problematic when attempting the automatic extraction of entries or parts of entries with shared lexical features.

The works described below assume the availability of tapes of printed dictionaries. Several publishers already produce such tapes. Converting printed dictionaries to tape involves either manual construction or scanning a printed dictionary, converting the raw data to a structured format. Manual construction can be done in two ways. In the first method all formatted structure is prepared manually, usually by typing. This allows for the conversion and manipulation of the raw data, containing typographic and lexicographic codes, which requires automatic or semi-automatic techniques (Berment, 2002). When the files are unavailable, the second method involves scanning and Optical Character Recognition (OCR). Our work concerns this "dictionary recycling". However, to our knowledge, not much work has been done on this subject. In two studies with the same goal, Palowitch and Stewart (1995) implemented an iterative process to identify structural features automatically in printed dictionaries, and Mao and Kanungo (2001) relied on stochastic language models based on manually created context-free grammars (CFG) and dictionary-specific stochastic production rules for automatic extraction of a Chinese-English bilingual dictionary. Although these studies have similar approaches to ours, such as using OCR, some important differences occur. First, they were small attempts applied only to one dictionary. It is not clear how adaptable they are and how complex dictionary entries they can convert. Our method, on the other hand, is generic and has been introduced to multiple dictionaries with different structures and complexities. Second, the two studies described above need either a PERL program or a CFG written for each dictionary. Such input can only be provided by people like computer scientists. Our system does not need an expert (in any field) to operate.

One important issue concerning the conversion of printed dictionaries into electronic medium involves copyright. Publishers are reluctant to relinquish electronic rights to the material they own without fully understanding the financial opportunities available and the impact that electronic publication will have on their traditional book businesses. Possible solutions to the copyright problem have been proposed (Byrd, 1995). At one extreme, the publishers can own (or license) the technology for the electronic publication of their data. At the other extreme, the system providers can own the data. An interesting middle ground allows any publisher to format the data accordingly, place it in electronic media, and sell it through its own marketing channels (like Borland International's Turbo Lightning system).

The two complementary processes in the general conversion of MRDs into lexical databases include:

1. Recovery of the dictionary structure from the typographical markings which exist on dictionary distribution tapes and embody the publishers' notational conventions, and

2. Making explicit all of the codified and elided information packed into individual entries (Neff and Boguraev, 1989).

Once the dictionary sources are edited and parsed, i.e., the useful information is extracted, the next step involves structuring the contents into a database (Byrd et al., 1987; Calzolari, 1984; Ahlswede et al., 1986). At this point, the dictionary material can be exploited in various ways. We adopt this approach in our work as well. The difference comes from our input not being MRDs, but OCRed dictionary pages. This difference occurs because low-density languages do not have readily available MRDs, thus we use OCR to digitize the text.

Many computational lexicography projects typically concentrate on the conversion of a single dictionary into a usable form. A few such projects include:

- *The Webster's Seventh New Collegiate Dictionary* (Gove, 1969) by Ahlswede et al. (1986)
- *The Collins English Dictionary* (Hanks, 1987) by Fox et al. (1988)
- *The Longman Dictionary of Contemporary English* (Procter, 1978) by van der Steen (1982); Nakamura and Makoto (1988); Boguraev and Briscoe (1989)

These projects use specialized programs based on the properties of a particular dictionary. Although the analysis of the *Oxford English Dictionary* (Murray et al., 1928) by (Kazman, 1986) has a more modular architecture consisting of a parser and a grammar, the output is not a structurally rich and explicit lexical database.

Focusing on the conversion of a single dictionary requires the implementation of new tools for each new dictionary, which has lead to the proposal of systems capable of handling multiple MRDs. We borrowed ideas from two studies the Lexical Systems project (Byrd et al., 1987) and the dictionary entry parsing tool LEXParse (Hauser and Storrer, 1994). Our methods, described in Chapter 3, are also generic, but we need user information about the structure of the dictionaries being processed. We also offer methods for handling structural indicators, fonts, etc. during entry parsing, and present various display methods to the user. Both these methods rely on grammars generated by experts to parse entries. Our methods, on the other hand, require information in a much simpler form, thus can be used by anybody.

In the ACQUILEX project (Briscoe, 1991), MRDs were seen as tools to speed the production of the lexicon, or to extend a core lexicon, not to automate lexicon construction entirely. It appears much closer to manual construction than construction from MRDs.

The work by Boguraev and Briscoe (1987) is interesting because it allows the user to intervene and correct errors during the actual processing of lexical entries from the *Longman Dictionary of Contemporary English*.

The electronic dictionary acquisition has the advantage of exploiting already existing resources. It is definitely faster than pure manual construction, but the end product is usually of a lower quality than its hand-generated counterpart. Plus, humans are still in the loop. The challenge is to find ways handle multiple resources with different structures/formats. Moreover, the end product is not completely error free. Another issue arises from the reliance on the existing MRDs. This does not present a problem for languages like English, where many studies have been done for several years. However, for many languages, MRDs are not presently available. For such languages, we need either efficient and reliable methods for creating MRDs or other resources for electronic dictionary construction. The next section discusses the use of corpora for electronic dictionary acquisition.

2.2.3.3 Corpus-based Approaches to Electronic Dictionary Acquisition

Corpora, i.e., collections of writings or recorded remarks, have been an object of study of some decades. Their application to NLP has recently increased interest in their use and construction. Corpora appear as optimal resources for lexicon and other resource creation in NLP applications. One reason claim that neither the contents nor form of any existing published dictionary meet all the requirements of a NLP system for a lexicon (Boguraev and Briscoe, 1987). For instance, published dictionaries do not reflect the recent changes in the vocabulary of the language because of their static nature. A large parallel or comparable corpus, however, provides the bilingual lexicographer with better and more naturally occurring equivalents from which to choose, and fills the gaps in the lexical coverage of the dictionary (Al-Ajmi, 2002). For another benefit, the corpus analysis allows the derivation of information that in some respects surpasses that available from MRD sources, such as information about the frequency and usage of words. All these advantages lead to an increased interest in the field of corpus-based approaches for the acquisition of electronic dictionaries, in particular for NLP systems.

One interesting and unique usage of corpus appears in the *Collins Birmingham University International Language Database* (COBUILD) project, which resulted in the publication of the *Collins COBUILD English Language Dictionary* (Sinclair, 1987). It is important because, "For the first time, a dictionary has been compiled by the thorough examination of a representative group of English texts, spoken and written, running to many millions of words." (Sinclair et al., 1987). The COBUILD dictionary has many features that makes it different from other English dictionaries: it is a wholly new dictionary, it reflects present-day usage of English, and it shows a radical departure from existing lexicographic conventions in that definitions appear as complete natural language sentences.

The COBUILD dictionary presents a unique application of how to use a corpus for lexicon acquisition. Lexicon acquisition from corpora usually results from a by-product of some other application such as word alignment using parallel text, a pair of texts that are mutual translations. Nearly every system that attempts to align words also generates bilingual lexicons based on these alignments. Many different methods have been used for this purpose (Brown et al., 1988, 1990; Gale and Church, 1991; Brown et al., 1993; Fung and Church, 1994; Kay and Röscheisen, 1993; Wu and Xia, 1994; Melamed, 1997b; Resnik and Melamed, 1997; Resnik, 1999; Melamed, 2000; Utsuro et al., 2002). Sentence or word alignments were employed to acquire of technical lexicons as well (Dagan et al., 1993; Dagan and Church, 1994; Gale and Church, 1991; Smatja et al., 1996). A few studies focused on aligning Indo-European and non-Indo-European language pairs (Utsuro et al., 1992; Wu and Xia, 1994; Fung and McKeown, 1994; Fung, 1995). A more difficult problem involves building a bilingual lexicon of phrasal expressions, such as collocations and idiomatic expressions (Kupiec, 1993; der Eijk, 1993; Dagan and Church, 1994; Daille et al., 1994; Kumano and Hirakawa, 1994; Haruno et al., 1996; Smatja et al., 1996; Melamed, 1997a; Gaussier, 1998). This offers an interesting research area because standard dictionaries lack phrasal translations. Another type of resource is comparable corpus, a collection of texts from pairs or multiples of languages, which can be contrasted because of their common features, such as topic, domain, authors, or time period. During recent years, the growing availability of comparable corpora, through the Internet or via distribution agencies providing newspapers articles in different languages, has led to an increased interest in using comparable corpora, especially in the areas of bilingual terminology acquisition (Tanaka and Iwasaki, 1996; Fung and Yee, 1998; Dejean et al., 2002; Koehn and Knight, 2002; Sadat et al., 2003a,b) and lexical resource enrichment (Peters and Picchi, 1995; Rapp, 1999; Fung, 2000; Dejean et al., 2002).

The major advantage of a corpus-derived dictionary in compared to a general purpose dictionary comes from tuning it to the way corpus translates its sentences, therefore it becomes more useful for translation (Brown, 1997). However, the limited number of parallel text and limited context these parallel texts exhibit presents an important problem of corpus-based research. The dependency on the existence of large electronic text resources limits the applications of corpus-based methods in low-density languages. For many languages, there is not even any comparable corpus available. Other approaches are needed to deal with the resource-limited languages.

2.2.3.4 Acquisition of Electronic Dictionaries for Low-Density Languages

The work described thus far assumes the availability of electronic resources, namely MRDs and text. However, this does not hold true for all languages, and creating such resources is time consuming and expensive. This section describes some alternative methods to obtain resources for creating electronic dictionaries.

One method employs a “generalized linguistic contribution”, which is a collaborative work, probably on the web, of a large distributed team. This method requires people who can provide data for the planned resource, a format to which everybody agrees , and tools each contributor can use (Berment, 2002; Shimohata et al., 2001; Ampornaramveth, 1998; Ampornaramveth and Aizawa, 2001).

Another method capitalizes on the fact that people often consult an intermediate third language to look up words in bilingual dictionaries, especially when working with uncommon languages in a specific domain. This idea was exploited to extract supplemental information for nouns in an existing Japanese-French bilingual dictionary using English as the bridging language (Tanaka and Umemura, 1994).

Another interesting direction uses the similarities between languages. Machine translation projects between Spanish and two languages closely related to Spanish, Catalan (Canals-Marote et al., 2001) and Galician (Diz, 2001) were based on this approach. Similarly, the MAJO system (Mahsut et al., 2001) used the syntactic similarities in Japanese and Uighur. In another study, Paul (2001) reused the existing knowledge resources of high-quality translation engines for translation into different, but related, languages with similar grammatical characteristics.

These alternative approaches do not apply to all languages. Some are time-consuming and rely on the collaborative effort, some can only help for supplemented additional information, and some need a well-studied similar language. When we need to acquire more data quickly, and the language differs greatly from the resource-rich languages, these will not prove helpful. This thesis addresses this gap by automatically providing an electronic resource from a printed dictionary.

The electronic dictionaries discussed so far were generally meant for NLP systems. Another type of electronic dictionary that targets mainly human users, specifically translators. The next section discusses these translation aid systems.

2.2.4 Translation Aid Systems

Technological advances help humans during the translation process as well. One end of the spectrum holds the online versions of dictionaries and pop-up dictionaries (such as Babylon[7], WordPoint[8], iFinger[9], and Technocraft's RoboWord[10]). On the other end of the spectrum, automatic machine translation systems appear. In between are several systems with different capabilities. Some of these include:

- Glossing programs in which a user can click a word on the computer screen and receive a, usually, rough translation (Nerbonne et al., 1997; Poznanski et al., 1998).

- More complex environments and help systems that can survey multiple dictionaries simultaneously, including users' glossaries, and find words and expressions in corpora, thus dynamically providing the translators with examples from their earlier translations or other translators' works (Prószéky, 1998; Prószéky and Kis, 2002).

- Systems that support collaborative translation work on the web in a user-friendly, efficient way (Shimohata et al., 2001).

- Systems that provide more useful and intelligent answers to translators' queries, recognizing their goals and anticipating their needs and allows user interaction (Agirre et al., 1995).

[7]http://www.babylon.com/

[8]http://wordpoint.grapho.net/

[9]http://www.ifinger.com

[10]http://www.technocraft.co.jp/english/

The Bilingual Resource Interface and Dictionary Generation Environment (BRIDGE) system developed by our group and described in Section 3.4 provides another attempt to target translators. BRIDGE provides the means to convert a printed dictionary into electronic form and to correct errors. It also offers different views of the generated resource and allows search of the resulting dictionaries. BRIDGE functions to transform printed resources into electronic ones. These resources are also searchable, thus users can easily find the information (such as POS or translation) they require. For instance, a translator can more effectively use the dictionary directly for tasks required. The work described in this thesis is implemented as an important component of BRIDGE.

Chapter 3

Dictionary Entry Tagging

The availability and use of digital data increase each year. Most of the data in electronic format, however, appears in a few languages. Many low-density languages still have only limited online resources available. For these and many other languages, data are available mostly in printed form and must be converted to electronic form. Optical character recognition (OCR) often offers the only feasible method to perform this conversion, owing to its speed and cost-effectiveness. OCR, however, gives only the content of printed source, and cannot perform satisfactorily when annotation of the data is needed. In order to extract the implicit information, we have to extend the traditional OCR approach; there is a need for transition from OCR output to usable electronic data.

As discussed in the previous chapter, an important electronic resource is the bilingual dictionary. The availability of dictionaries and thesauri is crucial both for translators and many Natural Language Processing (NLP) systems. For instance, machine translation (MT) and cross-language information retrieval (CLIR) systems use bilingual lexicons containing large sets of source-language/target-language correspondences. Electronic or online dictionaries can also serve as translation aid tools. The electronic medium offers functionalities that printed documents cannot duplicate, such as search. Printed documents allow only one access method, (by headword), which is static and pre-determined. Electronic documents, on the other hand, are dynamic, and users can define their own search methods with appropriate software. The possibility ever exists to search multiple electronic documents simultaneously. Moreover, the dynamic feature of electronic resources makes it possible to update, correct, and add new information easily and efficiently. The electronic medium also appeals with the speed of search and information retrieval. Therefore, digitizing printed documents offers new alternatives to the users that printed documents cannot provide.

Online dictionaries can either be constructed manually or using automatic methods. Manually constructing online dictionaries requires too much human effort. Several researchers have noted that, even for monolingual entries, the average time needed to construct a single entry can amount as much as 30 minutes (Copestake et al., 1995; Neff and McCord, 1990; Walker and Amsler, 1986). The construction of bilingual entries becomes more complicated because it requires native-speaker knowledge in both languages (Boas, 2002; Calzolari and Lenci, 2002; Neff et al., 1993). Costly manual construction results in resources that usually have limited coverage or application area. Thus, the automation of the bilingual lexical acquisition process is crucial for multilingual processing of any kind.

In Chapter 2, we have described several automated or semi-automated methods for generating electronic dictionaries. These methods rely on other online resources, which constrains the application of the described approaches to languages most frequently used in MT and CLIR tasks (such as English, French, Spanish, and Chinese). The same approaches are difficult to apply to language pairs involving low-density languages (such as Cebuano, Swalili, etc.) where not enough electronic resources exist. Yet, for these low-density languages, *printed* bilingual dictionaries often effectively map the low-density language to a high-density language, such as English. This thesis has the use of printed dictionaries in electronic dictionary generation as its motivation.

Printed dictionaries can be converted into electronic format in several ways. The most trivial idea is typing the information in dictionaries and annotating the data simultaneously. This may take at least six months for an abridged dictionary (personal communication). Thus this approach is impractical and costly. Another approach is scanning the dictionary pages, and using OCR to get the text content. However, as explained above, OCR is only a first step in extracting the implicit information in the dictionary entries. This implicit information tells a user of system what type of information a particular piece of content represents and so is necessary to convert a printed dictionary into a searchable electronic resource. For example, as just discussed, the font style may need to be explicitly marked if it indicates part of speech so that an automatic system can use this information effectively. The third approach is using OCR to get the content, and then performing the annotation of the data manually. This process also requires that a user go through all the information in the dictionary, and is thus time consuming and expensive. The last approach is automating the annotation process. However, there is still a need for human involvement since data in dictionaries is not enough for unsupervised methods to achieve the required accuracy levels. This thesis uses the last approach and users are only required to provide information about the dictionary data and/or training data. This approach greatly reduces the amount of human labor required, compared to the other methods (It requires on the order of one day of human effort.).

This chapter describes two new methods for tagging entries in bilingual dictionaries and generating an online lexicon from printed bilingual dictionaries, especially for low-density languages. 'Tagging' means assigning the lexicographic information (or information type) each entry word, thus annotating their data. A search algorithm must target different tagged types of information, making a dictionary searchable. Ultimately, the objective is to discover important information components provided in bilingual dictionaries, e.g., POS, pronunciation, and usage examples. In addition to a (desirable) lack of reliance on pre-existing online resources, the speed of this lexical-acquisition approach distinguishes this work from previous work. We aim to generate an online bilingual lexicon very quickly (usually, in one day). The methods described in this chapter are explained, then used on bilingual dictionaries. They have a general enough application to employ on a wide variety of language pairs and structured text, such as telephone books or tables. Entry tagging methods are rule-based or stochastic. Both methods utilize the repeating structure of the dictionaries to identify and label the different information types. Human assistance is required for both techniques, but is minimal.

The entry tagging is one of the components in the Bilingual Resource Inference and Dictionary Generation Environment (BRIDGE) system, an implemented prototype for electronic dictionary acquisition.

This chapter continues as follows. First, the entry tagging and the two proposed methods are described in Section 3.1. Then, the evaluation of tagging method and the results are discussed in Section 3.2. The generation module is described in Section 3.3. Finally, the overall BRIDGE system is presented in Section 3.4.

3.1 Entry Tagging

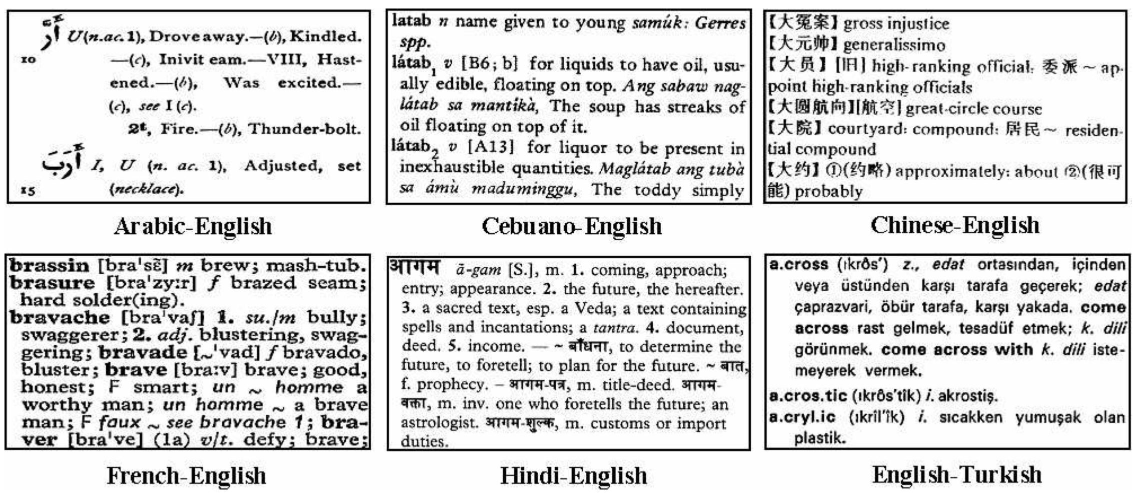
In this section, entry tagging is explained in the context of printed bilingual dictionaries, however, the same principals hold for other structured documents as well.

We focus on converting printed dictionaries into electronic format, especially for low-density languages, so the information in the dictionaries is usable in NLP applications or for human translation. This approach makes use of already existing and constructed printed dictionaries, reducing the time and labor necessary to produce online dictionaries. However, simply conducting OCR on text to digitalize will not suffice to make documents such as dictionaries useful. OCR produces only text and does not annotate data. To employ the information in the dictionary entries fully, we need to further process and annotate the data, making explicit each piece of information. Furthermore, we would like to accomplish this task rapidly with minimal human input. The data annotation in dictionary entries occurs by entry tagging, which involves recognizing, parsing, and tagging printed dictionaries to reproduce the representation electronically.

Dictionaries provide lexicographic information about the vocabulary of a language in a systematic way[1]. Although dictionaries can have different formats and

[1]Human related errors/inconsistencies always appear in printed dictionaries because several

provide a variety of information, they contain a consistent layout of entries and a consistent structure within entries. Dictionary publishers typically use a combination of structural indicators to impose this structure, including functional properties (changes in font, font style, font-size, etc.) that imply the role of the content implicit, keywords that provide an explicit interpretation of the lexicographic information, and various separators that impose an overall structure on the entry (as described in Section 2.2). Entry tagging aims to use the consistent layout and structure of dictionaries, along with such clues, to capture and recover the lexicographic information in the entries.


أَرَّ *U*(*n.ac.* **1**), Drove away.—(*b*), Kindled. —(*c*), Invit eam.—VIII, Hastened.—(*b*), Was excited.—(*c*), *see* I (*c*).
2, Fire.—(*b*), Thunder-bolt.
أَرَبَ *I*, *U* (*n. ac.* **1**), Adjusted, set (*necklace*).

Arabic-English

latab *n* name given to young *samúk: Gerres spp.*
látab$_1$ *v* [B6; b] for liquids to have oil, usually edible, floating on top. *Ang sabaw naglátab sa mantikà,* The soup has streaks of oil floating on top of it.
látab$_2$ *v* [A13] for liquor to be present in inexhaustible quantities. *Maglátab ang tubà sa ámù maduminggu,* The toddy simply

Cebuano-English

【大冤案】gross injustice
【大元帅】generalissimo
【大员】[旧] high-ranking official: 委派～appoint high-ranking officials
【大圆航向】[航空] great-circle course
【大院】courtyard: compound: 居民～ residential compound
【大约】①(约略) approximately: about ②(很可能) probably

Chinese-English

brassin [bra'sɛ̃] *m* brew; mash-tub.
brasure [bra'zy:r] *f* brazed seam; hard solder(ing).
bravache [bra'vaʃ] **1.** *su./m* bully; swaggerer; **2.** *adj.* blustering, swaggering; **bravade** [~'vad] *f* bravado, bluster; **brave** [bra:v] brave; good, honest; F smart; *un ~ homme* a worthy man; *un homme ~* a brave man; F *faux ~ see bravache 1*; **braver** [bra've] (1a) *v/t.* defy; brave;

French-English

आगम *ā-gam* [S.], m. **1.** coming, approach; entry; appearance. **2.** the future, the hereafter. **3.** a sacred text, esp. a Veda; a text containing spells and incantations; a *tantra.* **4.** document, deed. **5.** income. — ~ **बाँधना**, to determine the future, to foretell; to plan for the future. ~ **बात**, f. prophecy. – **आगम-पत्र**, m. title-deed. **आगम-वक्ता**, m. inv. one who foretells the future; an astrologist. **आगम-शुल्क**, m. customs or import duties.

Hindi-English

a.cross (ıkrôs') *z., edat* ortasından, içinden veya üstünden karşı tarafa geçerek; *edat* çaprazvari, öbür tarafa, karşı yakada. **come across** rast gelmek, tesadüf etmek; *k. dili* görünmek. **come across with** *k. dili* istemeyerek vermek.
a.cros.tic (ıkrôs'tîk) *i.* akrostiş.
a.cryl.ic (ıkrîl'îk) *i.* sıcakken yumuşak olan plastik.

English-Turkish

Figure 3.1: Examples of Bilingual Dictionaries

Entry tagging involves the automatic conversion of a document image analysis output of a dictionary into a format which explicitly represents the lexicographic information in dictionary entries. The entry tagging system described in this chapter can be adapted to different bilingual dictionary formats, as well as different

lexicographers compile one dictionary (Murray, 1977).

languages. Figure 3.1 illustrates how dictionary formats vary from simple term and phrase translation pairs to full descriptions that contain several different lexicographic information. For example, the Chinese-English dictionary contains the Chinese headword, English translation, and, sometimes, a phrasal use of the Chinese word and its English translation. The entries in the French-English dictionary, however, contain a French headword, its pronunciation, English translation, and POS or gender information. The entries may also contain derived forms with corresponding pronunciations and English translations, examples of usage and their translations, number, domain and etymology information, and cross references.

Entry tagging relies on minimal human input and the output of a document image analysis to identify different information types (POS, pronunciation, example of usage) for each bilingual dictionary entry. The pre-existing document image analyzer segments scanned images of dictionary pages into individual entries and extracts the functional properties of various elements of dictionary entries.

The user's first task requires the determination of the kind of information given in the specific dictionary and the kind of structural indicators used to distinguish these different types of lexicographic information. To reduce human involvement, we prepared a pre-defined list of widely used information types (given in Appendix A). This list was obtained through manual examination of several printed dictionaries. More information types appear than mentioned in Section 2.2 in the hope that more detailed information can be recovered from the dictionary entries. For instance, instead of having one information type as *inflections* or *run-ons*, we preferred to have several information types such as *Tense*, *Number*, and *Compound/Derived*, which are types of morphological inflections or derivations.

Entry tagging processes the OCRed dictionary pages, the information provided by a document image analyzer, and the human input about the dictionary[2]. Then, each piece of lexicographic information is determined for each entry using this input. The entry tagging requires an OCR that supports languages in the dictionaries. Moreover, the image analyzer provides the following information as input to the entry tagging:

- division of each page into dictionary entries,
- association of each entry with an entry type,
- identification of each entry, line and token,
- font style for each token, and
- OCR result for each token.

The entry tagging module was implemented using both non-tokenized and tokenized input. In the non-tokenized version, a *token* is defined as a set of glyphs (i.e., a visual representation of a set of characters) in the OCRed output, separated by white space. In the tokenized version, however, each punctuation counts as a token as well. Given input in this format, the entry tagging system associates labels with each information type provided by a token or group of tokens in the entry. The group of consecutive tokens with the same information type forms a *phrase*. In the system, each token is assigned a *phrase flag* stating whether it appears as the first

[2]Two methods rely on different human input, as described in Section 3.1 and Section 3.1.2

token in a phrase. If it is not the first token, then it is part of the continuation of a phrase. This information becomes useful during the generation process.

The entry tagging system requires input from a human operator familiar with, but not necessarily expert in, the language of interest. In some cases, not having knowledge of the language is necessary. Using the information given in the preface of some dictionaries and common dictionary information, the user may provide the information needed by the system.

We have implemented two different methods for entry tagging:

1. A rule-based method

2. A stochastic Hidden Markov Model (HMM) method

We faced a challenge in handling noisy input provided by the OCR and document image analyzer. The rule-based method accommodates noise by allowing for a relaxed matching of OCRed output to information types. The HMM method is *inherently* noise-tolerant due to the statistical nature of the training procedure underlying the models.

The architecture of the rule-based entry tagging appears in Figure 3.2. The input to the rule-based method includes the list of information types, separators, keywords provided by the user, and page segmentation from document image analysis. Each entry is divided into meaningful segments using this information, and then each segment is assigned an information type. In the end, some clean-up (usually provided by the user) is performed. The clean-up may include removing punctuation marks, replacing abbreviations with their full forms, etc. Then, the cleaned

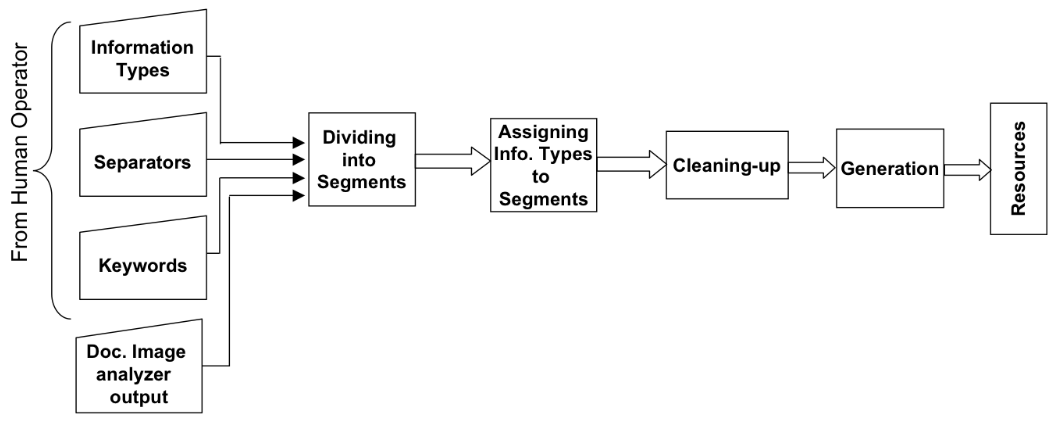


Figure 3.2: Rule-Based Entry Tagging Design

data generates different resources.

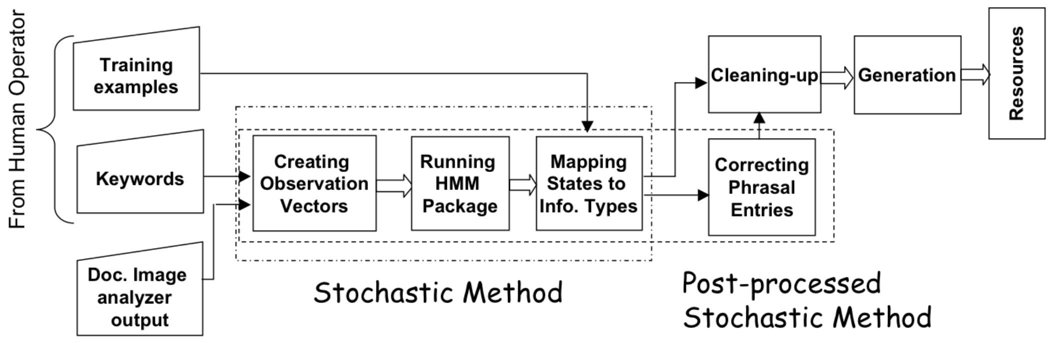

From Human Operator
Training examples
Keywords
Doc. Image analyzer output
Creating Observation Vectors
Running HMM Package
Mapping States to Info. Types
Cleaning-up
Generation
Resources
Correcting Phrasal Entries
Stochastic Method
Post-processed Stochastic Method

Figure 3.3: Stochastic Entry Tagging Design

The architecture of the stochastic entry tagging is shown in Figure 3.3. The input to the stochastic method contains some small amount of training data provided by the user and page segmentation. The first step produces the observation vectors for each token in the all entries, then trains HMM using these observation vectors. The training data maps the states assigned by HMM to the actual information types given in the training data. The phrasal entries are further processed to improve their accuracy in the post-processing phase. The same clean-up and generation in the rule-based method works for the stochastic method as well.

The following sections describe each of these two methods in detail.

3.1.1 Rule-Based Entry Tagging

The rule-based tagging approach uses the functional properties of tokens and their relationships to each other to assign labels to each information type in a dictionary entry. Rule-based tagging utilizes three different types of clues—font style, keywords, and separators—to tag the entries in a systematic way. The key is to discover the regularities in the occurrences of these clues and to employ them to assign labels to the different information types associated with each token.

3.1.1.1 The Input

The system relies on human input to identify the different information types in the dictionary and to describe their properties. A list of pre-defined information types exists, and users can define new information types if necessary. During the *configuration* of the dictionary, the user must provide the system with information types given in the dictionary, the specifications for these information types (such as font, separator, etc.), and whether they are represented with any keywords.

We defined five operands to describe different kinds of separators and their functions. For example, <cat> refers to the information types defined for the dictionary, and <sym> is a symbol used as a separator for this specific information type. The <sym> argument of an operand does not necessarily need to be a character value, it may contain multiple characters that are treated as a whole.

- `<cat>` *InPlaceOf* `<sym>` : Used as a shortcut for an information type, primarily for headwords, for space-saving reasons

  Headword *InPlaceOf* ∼

- `<cat>` *StartsWith* `<sym>` : Information type begins with this separator

  Pronunciation *StartsWith* [

- `<cat>` *EndsWith* `<sym>` : Information type ends with this separator

  Translation *EndsWith* ;

- `<cat>` *PreviousEndsWith* `<sym>` : Previous information type ends with this separator

  Translation *PreviousEndsWith* ,

- `<cat>` *Contains* `<sym>` : Information type contains this separator

  Compound/Derived *Contains* •

We recommend the usage of the operand *Contains* to be restricted to define some identifying separator used in the middle of a information type because using the other operands produces more specific information. Some pre-defined values specify some class of separators:

- *number* represents numbers used to divide different POS of a word, such "1", "2", etc.

- *numberletter* is similar, representing number and letter combinations, such as "1a", "1b", etc.

- *letter* represents letters, such as “a”, “b”, etc.

These can be used just like the other separators, as in:

Translation *PreviousEndsWith number*

For any information type, no limit exists to the number of separators the system operator can identify. The same operand (with different separators) can be used more than once with the same information type as well.

3.1.1.2 The Algorithm

The rule-based entry tagging algorithm proceeds as follows. First the entry is divided into segments using the font styles and separators. A *segment* is a token or a group of tokens with the same font style, or it consists of given keywords and/or is separated by separators from other segments. In practice, each segment corresponds to a single word or phrase. Then, each segment is scored for each information type defined for that dictionary. During this process, matching font style, keywords, and each separator contributes to the score, but some information is more trustworthy. For instance, keywords seem more reliable than font style given poor OCR performance for font style of short words. The information type with the highest score is assigned to that particular segment. This reiterates for each segment, ending with each segment assigned to a single information type. Given the uncertainty in the document image analysis process can lead to errors in the segmentation, several rules can address each information type, allowing for a relaxed matching of OCRed output to information types. In some cases, the separators are recognized incorrectly, so we may denote the pronunciation beginning with either '(' or '['. When multiple rules exist for one information type with the same operand, disjunction occurs while the score is computed. For different operands defined for one information type, conjunction is used.

The user may change the configuration of the dictionary several times to improve the accuracy and achieve the best the entry tagging results.

As an illustration, the entry's configuration in Figure 3.4 appears in Figure 3.5, and Figure 3.6 presents the resulting tagged entry. The six information types in the dictionary are described by their distinguishing features. The tagged entry associates each token or group of tokens with one of the given information types. A few important points need to be addressed. First, the headword is replaced with the sign '∼' in examples of usage. Second, the gender keyword *f* is replaced by the word *feminine*, an option that can be turned on or off. Finally, the output order is important for linking related information and keeping the hierarchy among and within the entries. For instance, the example translation *life jacket* is the translation of *brassiére de sauvetage*, not another example of usage in the dictionary.

3.1.2 The Stochastic Entry Tagging

Rule-based entry tagging resulted in good performance, as discussed in Section 3.2. One disadvantage of the rule-based tagging, however, comes from its dependency on the human provided configuration. If the human input does not cover the majority

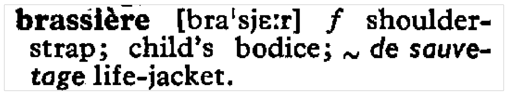

brassière [bra'sjɛːr] *f* shoulder-strap; child's bodice; ~ *de sauvetage* life-jacket.

Figure 3.4: A Sample Entry from a French-English Dictionary

Information Types	**Font**	**Keywords**	**Separators**
Headword	Bold		Headword *InPlaceOf* ∼
Pronunciation	normal		Pronunciation *StartsWith* [
			Pronunciation *EndsWith*]
Gender	Italic	f = feminine	
Translation	Normal		Translation *EndsWith* .
			Translation *EndsWith* ;
			Translation *PrevEndsWith* ;
Example	Italic		Example *PrevEndsWith* ;
Example Translation	Normal		Example *Translation* EndsWith ;
			Example Translation *EndsWith* .

Figure 3.5: Sample Configuration of the French-English Entry Given in Figure 3.4

Headword	brassiére
Pronunciation	bra'sjɛ:r
Gender	feminine
Translation	shoulder-strap
Translation	child's bodice
Example	brassiére de sauvetage
Example Translation	life jacket

Figure 3.6: Sample Tagged Output of the French-English Entry Given in Figure 3.4 Using the Configuration Given in Figure 3.5

of features that distinguish information types, the performance degrades. As an alternative to the rule-based entry tagging, we also applied a stochastic method, based on Hidden Markov Models (HMM), to entry tagging. It also relies on human input, but in the form of correctly tagged entries. This section, first briefly describes HMMs, then explain how HMMs apply in the context of the entry tagging problem.

3.1.2.1 Hidden Markov Models

A Hidden Markov Model[3] offers a statistical model assumed to be a Markov process with unknown parameters, the challenge being the determination of "hidden" parameters from the observable parameters, based on this assumption. In NLP, HMMs were applied in several areas including speech recognition and synthesis (Jelinek et al., 1992; Rabiner and Juang, 1989), POS tagging (Church, 1988; Cutting et al., 1992; Merialdo, 1994), named entity recognition and classification (Bikel et al., 1999), word sense disambiguation (Segond et al., 1997), and word alignments (Vogel et al., 1996; Och and Ney, 2000; Toutanova et al., 2002). These applications employ large data sets. In this work, we apply HMMs to smaller data sets.

A HMM consists of a finite set of states in which each state associates with a probability distribution. Transitions among the states are governed by a set of probabilities, called transition probabilities. In a particular state, an outcome or observation can be generated according to the associated probability distribution. An HMM is characterized by five elements:

1. N, the number of states in the model. The states are denoted as $S = S_1, S_2, ..., S_N$.

2. M, the number of observation symbols. If the observations are continuous, then M is infinite. Individual symbols are denotes as $V = v_1, v_2, ..., v_M$.

3. The state transition probability distribution $A = a_{ij}$.

[3]Rabiner and Juang (1986, 1989) provides a general introduction to HMMs. Charniak (1993) describes the applications of HMM in NLP.

4. The observation symbol probability distribution in each of the states. In state j, it is defined as $B = b_j(k)$.

5. The initial state distribution $\pi = \pi_i$.

Given appropriate values to these five parameters, the HMM can generate an observation sequence $O = O_1 O_2 ... O_T$ with an iterative procedure. In this sequence, each observation O_t is one symbol from V, and T is the number of observations in the sequence.

An HMM is usually specified by two model parameters, N and M, and three probability measures:

$$\lambda = (A, B, \pi)$$

In an HMM, no two transitions can have the same starting and ending states and the same output value. Furthermore, the state sequence followed by an HMM cannot be determined simply from the input sequence: it is *hidden*. This occurs because a particular state can be the starting state for several transitions with the same output symbol, but end in different states. In fact, HMMs are a generalization of Markov chains in which a given state may have several transitions generate, all with the same symbol. This cannot happen in Markov chains.

Given an HMM model $\lambda = (A, B, \pi)$ and a sequence of observations $O = O_1 O_2 ... O_T$, three problems of interest arise (Rabiner and Juang, 1986):

1. **The Evaluation Problem:** What is the probability that the model generates the observations, i.e., $P(O|\lambda)$?

2. **The Decoding Problem:** What is the most likely state sequence $Q = q_1q_2...q_T$ in the model that produced the observations?

3. **The Learning Problem:** How should we adjust the model parameters in order to maximize $P(O|\lambda)$?

The evaluation problem can be solved by a *forward algorithm*, which is recursive. With this algorithm, it becomes straightforward to determine which given HMMs best describe a given observation sequence: the forward algorithm is evaluated for each, and the one that gives the highest probability is selected.

The decoding problem concerns how to determine the sequence of hidden states that most probably generated an observed sequence. Unlike the evaluation problem, no given solution exists. One way involves using *Viterbi algorithm* (Viterbi, 1967; Jr., 1973) based on dynamic programming. The intuition behind the Viterbi algorithm computes the most likely trajectory, starting with the empty output sequence, then using this result to compute the most likely trajectory with an output sequence of length one. The recursion ends when it obtains the most likely trajectory for the entire sequence of outputs.

The learning problem tackles how to adjust the HMM parameters, so the given set of observations (called the training set) is best represented by the model for the intended application. The learning problem, the most difficult one for HMMs, must be solved. HMMs have no optimal way to estimate the model parameters. However, the model can be chosen such that $P(O|\lambda)$ is maximizes locally by adjusting the weights of the transitions to better model the relationship of the training samples.

Two main optimization criteria can accomplish this: Maximum Likelihood (ML) (Baum, 1972; Dempster et al., 1977) and Maximum Mutual Information (MMI) (Bahl et al., 1983).

The Baum-Welch algorithm is an Expectation-Maximization (EM) algorithm, also known as the *forward-backward algorithm.* The intuition adjusts the lambda parameters to maximize the likelihood of the training set. The algorithm can compute an HMM's maximum likelihood estimates and posterior model estimates for the parameters (transition and emission probabilities), given only emissions as training data. The algorithm contains two steps: (1) calculating the forward probability and the backward probability for each HMM state; (2) on the basis of this, determining the frequency of the transition-emission pair values and dividing it by the probability of the entire string, which amounts to calculating the expected count of the particular transition-emission pair. Each time a particular transition is found, the value the transition's quotient divided by the probability of the entire string rises, and this value can then become the new value of the transition.

Another parameter estimation algorithm is segmental k-means (Rabiner et al., 1986; Juang and Rabiner, 1990) which uses the state-optimized joint likelihood for the observation data and the underlying Markovian state sequence as the objective function for estimation.

To build an HMM, the set of relevant states and a set of all possible transitions must be defined. However, a complete set of transitions can be created and allow the algorithms to decide the ones with a transition probability of 0. The initial transitions are assigned random probabilities, then the model is trained using these initial probabilities.

3.1.2.2 Application of HMM to Entry Tagging

The goal of the stochastic entry tagging aims, as before, to determine the information type, i.e., the tag, of each token in a dictionary entry to annotate the dictionary. If an entry is treated as a sequence of tokens, then the problem resembles the decoding task in standard HMM approaches, in which the observation states correspond to the tokens in a lexical entry and the hidden states correspond to the information types associated with those tokens.

The decoding phase uses a standard Viterbi decoding algorithm, which determines the highest likelihood of a given state based on the *entire* input sequence is used. To apply this algorithm, the HMM must first be trained on enough data to induce probability matrices. The stochastic entry tagging trains using DeMenthon and Vuilleumier (2003)'s HMM package. This software facilitates the implementation of entry tagging for three reasons:

1. Observations encode as vectors, allowing for the representation of several features at once.

2. Components of an observation vector can use different numbers of symbols—a desirable characteristic given the features for tokens have different dimensions.

3. Training accommodates multiple observation sequences—an important property because the entire dictionary can be used as the training set.

The HMM package presents three choices for training: Baum Welch, segmented K-means, or the hybrid method. In the stochastic entry tagging, we used the hybrid method that combines the Baum-Welch algorithm with a segmental k-means algorithm for training. The hybrid method finds local maxima by applying ten iterations of the slower Baum-Welch algorithm; then the final (smaller) hill-climbing steps of the faster segmental k-means algorithm apply until no improvement occurs or until the system converges. The input to the system includes observation sequences, number of components that feature vectors have, numbers of symbols used for each feature vector components, number of states used by HMM, and switches that specify some parameters for the software.

The entry tagging input can either be non-tokenized or tokenized, as described in Section 3.1. To handle both types of input, two observation sequences (or observation *vectors*) are designed. Each token in a dictionary entry is represented by this observation sequence. Table 3.1 shows the observation vectors for non-tokenized input, consisting of a set of seven features. Table 3.2 shows the observation vectors for tokenized input, consisting of a set of four features. In the non-tokenized input, *Content* associates with one of four values: Information type if the token is a keyword; *Sym* if the token is a symbol; *Num* if the token consists only of numeric characters; otherwise, has the value *null*. In the tokenized input, *Content* associates with one additional value: the actual value of the token if the token is a separator. For non-tokenized input, the separators are represented as a separate feature, while for tokenized input no need exists for these features as they can be shown in *Content* feature. *Font* is the font style (normal, bold, italic, bolditalic, underline) of the to-

No	Feature	Value
1	*Content*	Information type if the token is a keyword *Sym* if the token is a symbol *Num* if the token consists only of numeric characters *null* otherwise
2	*Font*	normal, bold, italic, bolditalic, underline
3	*Starting Symbol*	separator at the beginning of the token
4	*Ending Symbol*	last character of the token
5	*Second Ending Symbol*	second-to-last character of the token
6	*Is-First-Token*	boolean (whether the first token of an entry)
7	*Is-Latin*	boolean (whether script is Latin or not)

Table 3.1: Features Used in HMM with Non-tokenized Input

No	Feature	Value
1	*Content*	Information type if the token is a keyword *Sym* if the token is a symbol *Num* if the token consists only of numeric characters value of the token if the token is a separator *null* otherwise
2	*Font*	normal, bold, italic, bolditalic, underline
3	*Is-First-Token*	boolean (whether the first token of an entry)
4	*Is-Latin*	boolean (whether script is Latin or not)

Table 3.2: Features Used in HMM with Tokenized Input

ken. *Starting Symbol* takes the value of the special punctuation symbol the token's beginning, if such a symbol exists. *Ending Symbol* and *Second Ending Symbol* takes the value of the last and second-to-last characters of the token, respectively, if are punctuation symbols. *Is-First* indicates whether this is the first token of an entry (a boolean value). Finally, *Is-Latin* corresponds to whether the characters in the token consist of Latin based characters or not (a boolean value).

Each token in the dictionary transforms into an observation vector before operating the HMM. As an example, Figures 3.8 and 3.9 show the observations vectors for the first line of the entry in Figure 3.7, in the non-tokenized and tokenized

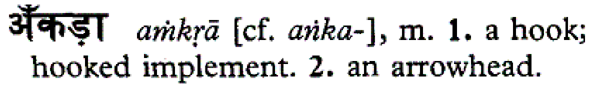

अँकड़ा *aṁkṛā* [cf. *aṅka*-], m. **1.** a hook; hooked implement. **2.** an arrowhead.

Figure 3.7: A Sample Entry From the Hindi-English Dictionary

1	[null	Normal	null	null	null	True	False]
2	[null	Italic	null	null	null	False	True]
3	[Abbreviation	Normal	[	.	null	False	True]
4	[null	Normal	null	,	]	False	True]
5	[Gender	Normal	null	.	null	False	True]
6	[Num	Bold	null	.	null	False	True]
7	[null	Normal	null	null	null	False	True]
8	[null	Normal	null	;	null	False	True]

Figure 3.8: The Observation Vectors for the First Line of the Entry in Figure 3.7 in Non-tokenized Version

1	[null	Normal	True	False]
2	[null	Italic	False	True]
3	[[	Normal	False	True]
4	[Abbreviation	Normal	False	True]
5	[.	Normal	False	True]
6	[null	Italic	False	True]
7	[ ]	Normal	False	True]
8	[,	Normal	False	True]
9	[Gender	Normal	False	True]
10	[.	Normal	False	True]
11	[Num	Bold	False	True]
12	[.	Bold	False	True]
13	[null	Normal	False	True]
14	[null	Normal	False	True]
15	[;	Normal	False	True]

Figure 3.9: The Observation Vectors for the First Line of the Entry in Figure 3.7 in Tokenized Version

versions respectively. These observation vectors provide training data for the HMM. The Viterbi algorithm then applies to find the most probable state sequence for this given input. After the training, an one-to-one mapping occurs from the observation vectors of tokens to the states of this sequence.

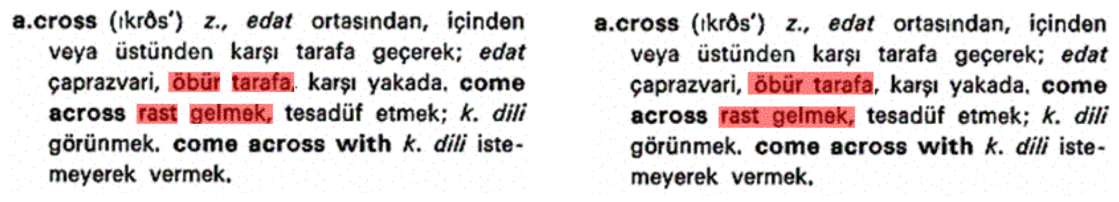
The main challenge in the HMM-based entry tagging involves the mapping of the states to the actual information types for which we are looking. A small training sample from the dictionary is used for this mapping. Around 400 randomly selected tokens are tagged manually for each dictionary. To find the information types corresponding to the states, the number of manually assigned information types in the training data that fall into each state are computed, then the information type with the highest count becomes the tag of the state. We should emphasize that the ground-truth data are not used during the training of HMM; it only maps the produced states to information types. This is a novel way of using the training data in HMMs. One problem arises if the training data miss some information types found in the dictionary, the system cannot assign these missing information types to a state.

3.1.2.3 Post-Processing

Algorithm 3.1: Post-processing Stochastic Entry Tagger Algorithm

```
if two consecutive tokens in a dictionary entry are tagged with the same
information type then
    if there is no separator at the end of the first token or at the beginning of
    the second token then
        merge these two tokens to form a phrase
```

When we analyzed the results of the HMM-based stochastic method, we discovered that although the accuracy of the results for tagged tokens compare to those of the rule-based approach, the identification of phrases is not as robust as that of the rule-based approach. To increase the performance of phrase identification, the results



a.cross (ıkrôs') *z., edat* ortasından, içinden veya üstünden karşı tarafa geçerek; *edat* çaprazvari, öbür tarafa. karşı yakada. **come across** rast gelmek, tesadüf etmek; *k. dili* görünmek. **come across with** *k. dili* istemeyerek vermek.

a.cross (ıkrôs') *z., edat* ortasından, içinden veya üstünden karşı tarafa geçerek; *edat* çaprazvari, öbür tarafa, karşı yakada. **come across** rast gelmek, tesadüf etmek; *k. dili* görünmek. **come across with** *k. dili* istemeyerek vermek.

Figure 3.10: Post-processing Example. Two Consecutive Tokens With Same Tags and Without Any Separators Between Them is Merged to Form a Phrase.

of the stochastic method are post-processed using keywords and separators defined with the operands *StartsWith*, *EndsWith*, and *PreviousEndsWith*. The idea is to merge consecutive tokens with same tags and without any separators between them into a phrase. Figure 3.10 shows one case from a Turkish-English dictionary. Here, two pairs of tokens, the pair *öbür* and *tarafa* and the pair *rast* and *gelmek*, merge to form the phrases *öbür tarafa* and *rast gelmek*. The Algorithm 3.1 demonstrates how the post-processing proceeds. Post-processing increases the phrase accuracy, as the next section exhibits.

3.2 Experiments

Two experiments evaluate the performance of the entry tagging. The first experiment measures *dictionary adequacy*, the degree to which the system adequately tags entries of three printed, bilingual dictionaries (Section 3.2.1). In the second experiment, we examined the degree of dictionary adequacy in more detail (Section 3.2.2).

3.2.1 Dictionary Adequacy

The entry tagging methods described above were evaluated on three of the dictionaries shown in Figure 3.1: French-English (Urwin, 1988), English-Turkish (Avery et al., 1974), and Hindi-English (McGregor, 1993). These dictionaries have different characteristics that affect the noise rate of the OCR font, which may result in low OCR accuracy. Font plays a different role in each one. In the French-English dictionary, font presents a very important feature, whereas in the English-Turkish dictionary font has less importance, but is still necessary. In the Hindi-English dictionary, font is entirely unimportant. Moreover, the French-English dictionary contains symbols that reduce the OCR accuracy, and the Hindi-English dictionary has non-Latin fields.

The adequacy of the resulting dictionaries were compared with respect to ground-truth data, generated manually for five pages worth of randomly selected entries from each dictionary. Performance was rated using the standard precision, recall, and f-measure metrics. Precision (P) measures how accurately the entries are tagged, and recall (R) gives a measure of coverage. F-measure (F) represents the weighted harmonic mean of precision and recall. The formulas for these metrics are as following:

$$\begin{aligned}
Precision &= \frac{number\ of\ correctly\ tagged\ tokens}{total\ number\ of\ tagged\ tokens} \\
Recall &= \frac{number\ of\ correctly\ tagged\ tokens}{total\ number\ of\ tokens\ in\ the\ ground-truth} \\
F-measure &= \frac{2\,(P \cdot R)}{P+R}
\end{aligned}$$

	French-English	English-Turkish	Hindi-English
# of pages	528	1152	1083
# of entries	13537	36747	33020
# of tokens	304601	619715	744722
# of components	[11 4 6 9 7 2 1]	[10 4 6 7 5 2 1]	[12 2 6 10 7 2 2]

Table 3.3: Dictionary Statistics

Statistical information about the dictionaries used in the experiments appears in Table 3.3. The number of components in this table represents the number of different values each feature can take in the non-tokenized observation vector, where the vector represents:

$[Content, Font, Starting\,Symbol, Ending\,Symbol,$

$Second\,Ending\,Symbol, Is-First\,Token, Is-Latin]$

Five pages of ground-truth from the French-English dictionary have 167 entries and 2,918 tokens, and the English-Turkish dictionary has 193 entries and 2,555 tokens, and the Hindi-English dictionary has 136 entries and 2,808 tokens. It is worth noting that the derived word and example of usage have translations for these dictionaries, but these translations have the same properties as the headword translation. Thus, during rule-based entry tagging, the configuration did not have explicit rules for these two types of translations. The same information type was assigned to all translations in the training data for HMM. The type of the translation is identified by the information type of the last token bearing that translation (i.e., headword, derived word, or example of usage). This property that the type of the translation can be identified without user input reduces the amount of user input.

One problem we faced during the mapping concerns the number of states

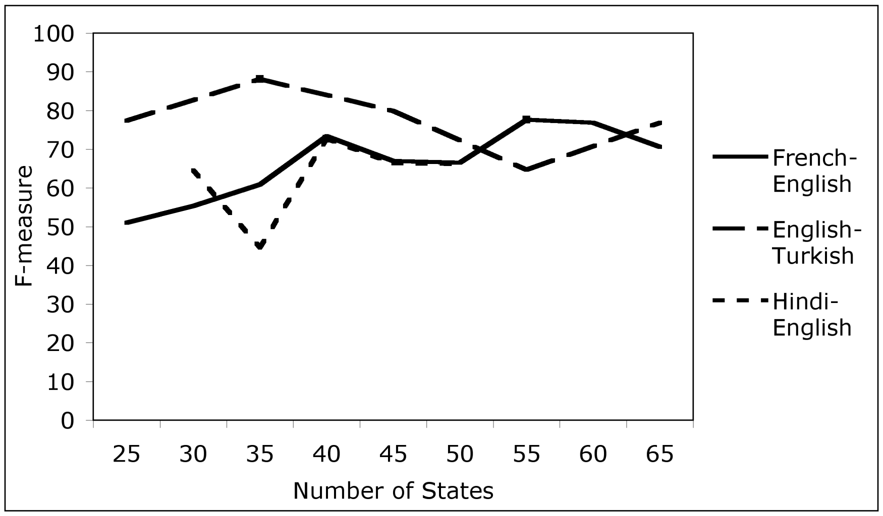


Figure 3.11: Accuracy for Entry Tagging with HMMs Trained on Different Number of States for Three Dictionaries

HMM uses being equal to the number of different information types found in the dictionary, the resulting tags are not accurate. In order to find an optimal number, we experimented with HMMs trained with different number of states. Figure 3.11 presents the token accuracy with different number of states on three dictionaries. The graph marks the number of states that each dictionary used to exhibit its best performance. Unfortunately, the number differs for each dictionary. Moreover, no correlation occurs between the number of information types and number of states giving the best performance. For the French-English dictionary, in which 18 information types were defined, the best accuracy was achieved with 55 states. In the English-Turkish dictionary, 17 information types were tagged best with 35 states. The Hindi-English dictionary with 24 information types performed best with 40 states. In the below experiments, HMMs for each dictionary trained on the number of states giving the best accuracy for that particular dictionary.

During evaluation, we performed two different sub-evaluations. The first evaluation was token-based, in which each token appears as a *single-word* entry, even when part of a phrase. The second was phrase-based, in which we considered *multi-token* entries to be grouped together as a logical phrase. If a multi-token entry is assigned one information type in the ground-truth, we considered the tagging correct only if the same multi-token entry was assigned the same information type by the system. As an example of phrase-based evaluation, consider the French-English dictionary from Figure 3.1. Here, the correct translation for *brasure* is the phrase *brazed seam*. If the system produces the translation *brazed seam* (as a unit), then this is rates a correct entry. If, on the other hand, the system produces two independent words *brazed* and *seam*, this result counts as incorrect. Phrase-based evaluation is important for translation-aid systems and machine translation, but token-based evaluation also signifies because certain cross-language applications (e.g., CLIR) treat all translations of a word as a list.

Table 3.4 summarizes the precision, recall, and f-measure for *all information types* in the dictionary for the rule-based method, and the stochastic method before and after post-processing. The results specify an average value over the ground-truth for each dictionary. The best results for each metric for each evaluation method are highlighted in bold face.

Stochastic method resulted in better token-based performance for the French-English and the English-Turkish dictionaries. For all the other evaluations, the performance of rule-based entry tagging is better. We concluded that when the font represents a distinguishing feature, as in the French-English and the English-Turkish, the stochastic method usually gives better results than the rule-based method. How-

French-English Dictionary

		All Information Types		
Evaluation	**System**	**P**	**R**	**F**
Token-based	Rule-based	72.55	72.55	72.55
	Stochastic & Post-pr. St.	**77.62**	**77.62**	**77.62**
Phrase-based	Rule-based	**74.73**	**75.19**	**74.96**
	Stochastic	55.78	69.97	62.08
	Post-pr. st.	67.59	72.86	70.13

English-Turkish Dictionary

		All Information Types		
Evaluation	**System**	**P**	**R**	**F**
Token-based	Rule-based	86.97	86.97	86.97
	Stochastic & Post-pr. St.	**88.14**	**88.14**	**88.14**
Phrase-based	Rule-based	**89.04**	**87.93**	**88.48**
	Stochastic	40.03	62.86	48.91
	Post-pr. St.	84.55	85.10	84.83

Hindi-English Dictionary

		All Information Types		
Evaluation	**System**	**P**	**R**	**F**
Token-based	Rule-based	**85.93**	**85.93**	**85.93**
	Stochastic & Post-pr. St.	72.69	72.69	72.69
Phrase-based	Rule-based	**85.99**	**85.07**	**85.53**
	Stochastic	51.62	50.45	51.03
	Post-pr. St.	56.69	64.55	60.37

Table 3.4: Experiment Results for All Information Types on Three Dictionaries

ever, the rule-based method outperforms stochastic method when the font is not a distinguishing feature, such as in the Hindi-English dictionary.

Rule-based method was particularly successful in identifying phrases and the stochastic method alone did not succeed well in identifying phrases regardless of the dictionary structure. Post-processing improved the f-measure of the phrase-based results between 13-73%, but rule-based method gave higher precision, recall, and f-measure for all three dictionaries in phrase-based evaluation.

The phrase-based f-measure for French-English dictionary was 75% while for the other dictionaries it was 88% and 86%. We believe the main reason the performance for the French-English dictionary is lower than the other dictionaries is its more complex structure.

French-English Dictionary

		Hw/Derived Word Trans.		
Evaluation	**System**	**P**	**R**	**F**
Token-based	Rule-based	67.93	**77.27**	**72.30**
	Stochastic	70.71	62.47	66.34
	Post-pr. St.	**76.65**	67.72	71.91
Phrase-based	Rule-based	64.97	**74.51**	69.41
	Stochastic	48.15	54.72	51.23
	Post-pr. st.	**74.46**	67.32	**70.71**

English-Turkish Dictionary

		Hw/Derived Word Trans.		
Evaluation	**System**	**P**	**R**	**F**
Token-based	Rule-based	**84.77**	87.93	86.33
	Stochastic	80.09	85.91	82.90
	Post-pr. St.	84.22	**90.33**	**87.17**
Phrase-based	Rule-based	**84.01**	**89.22**	**86.53**
	Stochastic	17.24	39.14	23.94
	Post-pr. St.	82.25	87.59	84.84

Hindi-English Dictionary

		Hw/Derived Word Trans.		
Evaluation	**System**	**P**	**R**	**F**
Token-based	Rule-based	**78.64**	**78.25**	**78.44**
	Stochastic	45.87	53.15	49.24
	Post-pr. St.	46.93	54.37	50.38
Phrase-based	Rule-based	**74.16**	**78.03**	**76.04**
	Stochastic	23.79	17.85	20.39
	Post-pr. St.	37.91	50.86	43.44

Table 3.5: Experiment Results for Headword-Derived Word Translations for Three Dictionaries

Table 3.5 presents the precision, recall, and f-measure results only for *headword* *and derived word translations*. In this case, the f-measure of the post-processed stochastic entry tagging is 1.3% higher than the f-measure of the rule-based entry tagging in phrase evaluation. In all other phrase evaluations, the rule-based method resulted in better performance. For the Hindi-English dictionary, the performance of stochastic method is very low. For the other two dictionaries, token-based evaluation results of two methods are close. This fact again suggests that when font is not important for identifying information types, rule-based method performs better.

The post-processing stochastic method improved the f-measure of the phrase-based results between 38-254% when considering headword and derived word translations. A similar pattern was observed for all information types, thus we conclude that for dictionaries containing many phrases, post-processing is necessary when using the stochastic method.

For token-based evaluation, no system performed best consistently. Rule-based entry tagging, however, succeeds better in phrase-based evaluation; stochastic method outperforms it only in headword and derived word translations for the French-English dictionary, but the difference is 1.3%. Rule-based method has better performance in phrase identification because of the way it operates: first divides entries into meaningful segments, then assigns tags to these segments.

Another point: both systems achieved higher recall than precision in most cases.

	Cebuano	**POS**	**English**	**Total**
Correct	95.36	95.00	88.12	93.42
Missing	2.06	5.00	4.95	3.54
Extra	0.00	0.00	3.96	1.01
OCR error	2.58	0.00	2.97	2.03

Table 3.6: Cebuano Experiment Results

3.2.2 Detailed Analysis

We also processed a Cebuano-English[4] (Carlsen, 1999) dictionary with a much simpler format using the rule-based approach. We evaluated this dictionary using a different approach and investigated the handling of the POS[5], Cebuano, and English terms in detail. One hundred randomly selected (ground-truth) entries from the original dictionary were the basis of the comparison against the generated lexicon (we acquired the dictionary in Microsoft Excel format as well). The evaluation involved a verification of only these information types. Each token in the result was categorized as one of four types:

1. correct—tagged correctly, with no OCR error
2. missing—not in the generated lexicon
3. extra—not in the original dictionary
4. OCR error—tagged correctly, but the term is incorrect because of OCR noise

[4]Cebuano is a language spoken by about 15 million people in the Philippines, written in Latin script.

[5]This is not traditional POS tagging. The goal is extracting the POS information already present in the dictionary. The data that contains the headwords with POS tags can be used as a seed to build a POS tagger for the non-English language.

Table 3.6 presents the results; the accuracy for Cebuano and POS terms were greater than 95%, and for English terms it reached more than 88%. The overall accuracy exceeded 93%. In addition, among the correct Cebuano terms, 12.89% of the terms had incorrect accents because of OCR noise. The structure of this dictionary appears similar to the Chinese-English dictionary Mao and Kanungo (2001) (described in Section 2.2.3.2) used in their evaluations. The reported 92.21% accuracy in that study is a little less than our overall accuracy of 93.42%. They evaluated their method using only one dictionary, and it is unclear how their method can be generalized for other dictionaries and if is can handle complex dictionaries. Our method, on the other hand, has been applied to several dictionaries easily and produced accurate resources rapidly.

3.3 Generation

The entry tagging assigns the information types each token represents in the dictionary entry. The information in this format, however, does not have many uses, the information in the dictionaries should convert to a more usable format. Unfortunately standards do not exist for this area, so we need a flexible representation. We defined a generic representation for entry tagging results to map into different formats. The generation module must take this representation and create usable information for different types of applications, both for NLP systems and translation aid systems. The generation module produces various result sets. Six formats each have different advantages.

1. Text Encoding Initiative (TEI)[6]: An XML-based format that is well-documented, well-regarded, and widespread.

2. Rosetta: Another XML-based format widely used by some government agencies[7]

3. Lexicon-Translation Pairs: Text-based list of headwords and their translation and compound/derived word with their translations. This sort of output may have uses in NLP applications, such as MT or CLIR systems.

4. Example of Usage-Translation Pairs: Text-based list of examples of usage and their translations. This format can also be used in some NLP applications. In fact, we used this output for the morpheme acquisition described in Chapter 5.

5. HTML: Html-based representation of each tag and value.

6. Color-coded HTML with Entry Images: HTML-based representation of entries with color-coded tags and original dictionary entry images, which is especially useful for visualization purposes.

We should note that if the human operator creates a new information type, it will not be represented in the TEI or Rosetta results. Appendix B shows entry samples represented with these different formats.

[6]More information about TEI can be found at http://www.tei-c.org

[7]One drawback of Rosetta is that, this format does not represent all the information types we used in entry tagging (listed in Appendix A).

3.4 The BRIDGE System

This section briefly describes the prototype Bilingual Resource Inference and Dictionary Generation Environment (BRIDGE) system implemented for online dictionary acquisition. The entry tagging and generation described above are two important parts of BRIDGE, a tool that automates the process of acquiring lexical knowledge of low-density languages. This tool works at the rapid production of a high quality OCRed dictionary that can serve both as an enhanced stand-alone translation aid and as a lexical resource for NLP systems. This tool covers the acquisition of written language by incorporating information obtained from a paper dictionary. BRIDGE enables the transfer of content from print dictionaries to electronic format in a matter of one or two days, resulting in a usable resource in a fraction of the time and cost required to accomplish it manually.

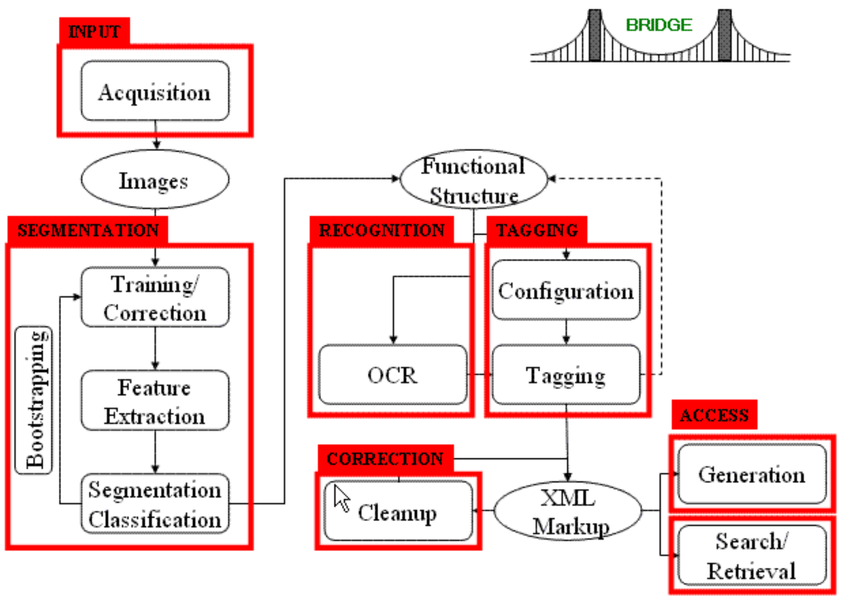


Figure 3.12: The BRIDGE System Architecture

BRIDGE aims at handling different dictionary layouts, structures, and languages, as shown in the architecture schemata in Figure 3.12 shows the system architecture. This thesis focuses on the entry tagging and resource generation parts of the BRIDGE system as implemented by the author. Given an OCRed version of a printed dictionary, it does not follow we have an electronic dictionary because the lexicographic information in the entries is not explicit. We require an explicit encoding of the information in the entries to enable search, which is achieved by the entry tagging module.

The BRIDGE system operator can acquire (via scanning and OCR), segment, tag, and generate a formatted electronic version of a hardcopy bilingual dictionary given minimal knowledge about the secondary language. BRIDGE uses a staged approach to dictionary parsing and tagging. The process of dictionary acquisition consists of: scanning and OCR, segmentation, tagging, and generation, with access as a follow-on process by BRIDGE-View. The system requires is the existence of OCR that supports the dictionary's languages. The performance of OCR systems, however, is far from perfect, and recognition errors significantly degrade the performance of NLP applications. Moreover, low-density languages usually do not have an OCR specifically designed for them. Addressing this issue, Kolak and Resnik (2005) proposed a lexicon-free statistical post-processing method for optical character recognition (OCR) error correction. The method requires minimal resources, and can achieve reasonable recognition accuracy for low density languages altogether lacking an OCR system, to significantly improve on the performance of a trainable commercial OCR system. In another work, Ma and Doermann (2004a) proposed an adaptive OCR trained with character samples extracted from different documents. Using this approach, an OCR for Hindi was developed and trained in less than a month, and it achieved an average recognition accuracy of 88% for noisy images, and 95% for ideal images.

The interface has a goal of providing a seamless integration of the stages, along with the ability to correct errors in the processing at will. The user-friendly system enables the user to interact with the system by providing features and training samples needed at each stage. For segmentation, BRIDGE software allows the operator to identify entries on the image for training, trains the system, and facilitates manual correction and bootstrapping. For the entry tagging, it provides user-friendly environments to define the configuration of the dictionaries, select valid information types, enter keywords, and define separators. Moreover, the BRIDGE tool aids the user in preparing training data and ground-truth. When the system runs, the results appear as color-coded boxes overlain on the original image. The user can also select different output formats to generate and view the result. Appendix C gives several screenshots of BRIDGE for configuration of tagging module and results.

Our group at the University of Maryland performed user and usability tests on BRIDGE (Abels et al., 2005). The user study evaluated whether dictionaries, in the electronic format of BRIDGE, aid foreign language professionals in performing translation tasks. The results of the dictionary use study suggested that foreign language professionals prefer electronic resources for translation activities. The usability study investigated how the ease of transforming a paper dictionary, or other structured resource into an electronic form using BRIDGE. This testing suggested that users of varying backgrounds can successfully use BRIDGE to create resources.

Document Image Analysis The first step in BRIDGE involves physically acquiring the page images of dictionaries by scanning them. Then the document image analysis module processes the images. First, the images segment into individual entries given characteristics such as font properties (bold, italics, size, color, etc) and layout features (indentation, bullets, etc.) are used to indicate new entries in dictionaries. The segmentation is achieved as an iterative procedure that includes three steps: Feature extraction and analysis, training and segmentation, and correction. During the training phase, a Bayesian framework assigns and updates the probabilities of extracted features, such as special symbols, font style, word and symbol patterns, and line structures. The document image analysis module also extracts the functional properties of various elements of entries. The system can classify scripts, font-styles, and font-faces in images. More information about the document image analysis module can be found in Ma and Doermann (2003a,b); Ma et al. (2003); Ma and Doermann (2004b, 2005).

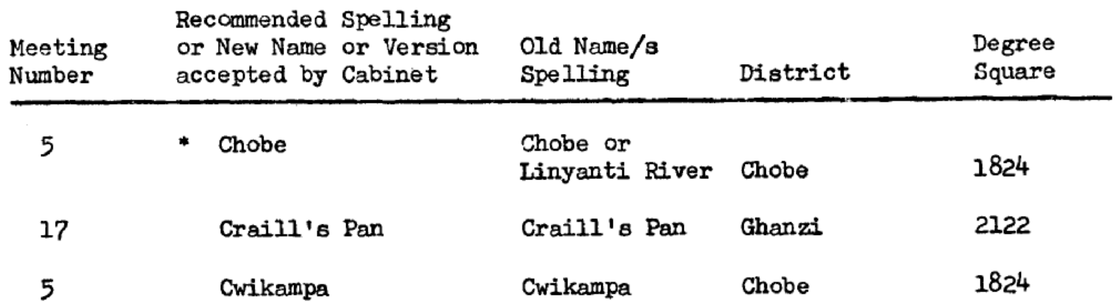
BRIDGE-VIEW BRIDGE-View, the search and retrieval interface, presents one of BRIDGE's the direct applications, a utility that lets the user search for an actual dictionary entry. In other words, this module emulates the process of searching for a particular word in a paper dictionary. This tool converts the results into a format usable by a human user. The TEI output format described in Section 3.3 is used during the indexing of the information in the dictionary. BRIDGE-View allows the user to perform exact or fuzzy search on all or selected information types. The user can view the results, along with the actual dictionary entry image. If the user prefers to view the entire dictionary page so that the context is clearer, this option is also available.

3.5 Summary

In this chapter, we described how to convert dictionary entries of printed dictionaries into annotated electronic resources, including two methods for tagging dictionary entries and generation of different outputs. Both entry tagging methods utilize the repeating structure of the dictionaries and dictionary entries, and the clues publishers use to identify and label the different information types represented in entries. The rule-based method requires a human operator to define each information type precisely and explicitly by a human operator. The HMM-based stochastic method, however, depends on human input in the form of manually prepared ground-truth. Both methods can produce the tagged results quickly, usually in less than in a day.

The entry tagging and generation operate as two modules in BRIDGE, an end-to-end tool implemented by our group at the University Maryland. BRIDGE has been tested by the government organizations and used to convert dictionaries in house, Center for Advanced Study of Language (CASL), MITRE, and National Geospatial-Intelligence Agency (NGA).

In an attempt to process another type of structured documents using BRIDGE, we also worked on gazetteers (Figure 3.13). The system adapted to handle the gazetteer structure easily and rapidly. The only major modification came from the



Meeting Number	Recommended Spelling or New Name or Version accepted by Cabinet	Old Name/s Spelling	District	Degree Square
5	* Chobe	Chobe or Linyanti River	Chobe	1824
17	Craill's Pan	Craill's Pan	Ghanzi	2122
5	Cwikampa	Cwikampa	Chobe	1824

Figure 3.13: Sample Gazetteer Processed by BRIDGE System

addition of new information to the input by document image analysis, which gives the column number of a token, and the introduction of a new operator, in addition to the five operands defined in Section 3.1.1. This new operand, *InColumn*, tells the system in which column of the table a particular information type appears. This operand, used together with the other operands, configures the structure of the table documents. We did not evaluate the results, but the tagging results look very accurate.

Although the performance of rule-based and stochastic methods compare favorably in some cases, we will be using results of rule-based method as this thesis continues, for the following there reasons:

1. During the training of the HMM in stochastic entry tagging, the user must specify the number of states in the HMM. Unfortunately, we were unable to find a magic number for the best performance with each dictionary. Hence, when using stochastic method, the user must try a different number of states, and decide which one operates best for that particular dictionary.

2. The training data provided by the user should include all the information

types found in the dictionary, otherwise these information types will not map to a state in HMM output, and will be missing from the output. If some information types appear rarely in the dictionary, the user may not include these information types in the training data.

3. Rule-based method is usually more dependable in identifying phrases, which are more important in translation aid systems.

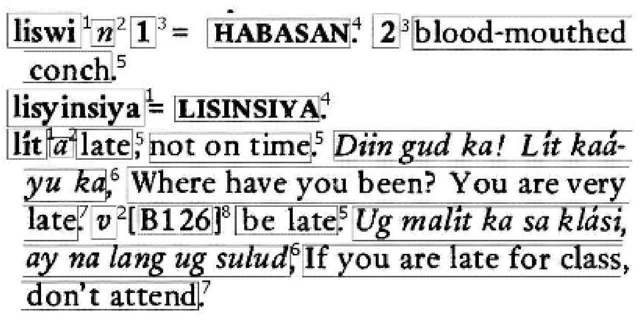
Given these three concerns, the rule-based method will be the default entry tagging method. We achieved quite reasonable tagging results, however, we would like to improve on these initial results. The next chapter describes an adaptive method employed for better tagging performance.

Chapter 4

Adaptive Transformation-based Learning for Improving Dictionary Tagging

The rule-based entry tagger presented in Chapter 3 utilizes the repeating structure of the dictionaries to identify and tag the linguistic role of tokens or sets of tokens. Rule-based tagging uses three different types of clues—font style, keywords, and separators—to identify the information in the entries in a systematic way. The method accommodates noise introduced by the document analyzer by allowing for a relaxed matching of OCRed output to tags. A human operator must specify the lexicographic information used in each particular dictionary, along with the rules for each tag. This process can be performed in a few hours. The rule-based method alone achieved token accuracy between 73%-87% and phrase accuracy between 75%-89% in experiments conducted using three different dictionaries.

The rule-based method has demonstrated promising results, but has two shortcomings. First, the method does not consider the relations between different tags in the entries, i.e., the sequential ordering of the tags. While this may not a problem for some dictionaries, for others the ordering of the relations between tags may be


liswi[1] *n*[2] **1**[3] = **HABASAN.**[4] **2**[3] blood-mouthed conch.[5]
lisyinsiya[1] = **LISINSIYA.**[4]
lít[1] *a*[2] late,[5] not on time.[5] *Diin gud ka! Lít kaáyu ka,*[6] Where have you been? You are very late.[7] *v*[2] [B126][8] be late.[5] *Ug malít ka sa klási, ay na lang ug sulud,*[6] If you are late for class, don't attend.[7]

1 Headword
2 POS
3 Sense number
4 Synonym
5 Translation
6 Example of usage
7 Example of usage translation
8 Subcategorization

Figure 4.1: Sample Tagged Dictionary Entries. Eight Tags are Identified and Tagged in the Given Entries.

the only information that will tag a token correctly, consider the dictionary entries in Figure 4.1. In this dictionary, the word "a" represents a POS when in italic font and part of a translation in normal font. However, if the font is incorrect (and font errors are more likely given short tokens), the only way to mark the tag correctly involves checking the neighboring tokens and tags to determine its relative position within the entry. When the token has an incorrect font or OCR errors exist, and the other clues are ambiguous or inconclusive, the rule-based method may yield incorrect results.

Second, the rule-based method can produce incorrect splitting and/or merging of phrases. Two tokens may merge erroneously into a phrase because either of a font error in one of the tokens or the lack of a separator, such as a punctuation mark. Likewise, phrase may split erroneously either as a result of a font error or an ambiguous separator. For instance, a comma may appear after an example of usage to separate it from its translation or within it as a normal punctuation mark. As a result, the example of usage in Figure 4.1 *Diin gud ka! Lít kaáyu ka,* may be split incorrectly into two separate example of usages as *Diin gud ka!* and *Lít kaáyu ka,*.

The initial results obtained from the entry tagger can be improved by addressing two shortcomings. One great challenge to improving the results of the human aided dictionary tagging application involves using minimal human generated training data. This requirement limits the effectiveness of data-driven methods for initial training. The learning method should work with very limited training data, minimizing human labor.

This chapter presents an adaptive technique that enables users to produce a high quality of dictionary, parsed into its lexicographic components using an extremely small amount of user provided training data. Transformation-based learning (TBL) can adapt as a postprocessor at two points in the entry tagging system to improve performance, especially targeting the shortcomings described above.

TBL (Brill, 1995), a rule-based machine learning method, uses rules to improve the initial results by extracting and comparing the initial results with training data, and then categorizing the errors according to predefined features.

TBL has some attractive qualities that make it suitable for language related tasks, such this problem:

1. It is error-driven, thus directly minimizes the error rate (Florian and Ngai, 2001).

2. It can apply to other annotation systems' output to improve performance.

3. It uses token's features and those in the neighborhood surrounding it, therefore it addresses two shortcomings of rule-based entry tagger.

4. The resulting rules are easy to review and understand.

5. As an iterative algorithm, it can correct its own errors in subsequent iterations.

During the learning process, the transformations are instantiated using a given set of templates. These templates limit the search space for possible transformation rules. Another advantage of TBL is that the templates and instantiated transformations are easy to understand. This property allows the user to figure out the common errors in the initial annotation. Moreover, a non-computational linguist user can add new templates easily if the need arises. One particular case when additional templates are needed might be adapting the system to work with other kinds of structured documents. The templates defined in this work are prepared targeting the dictionary features, and another kind of structured document might need some new templates. Since the templates are easy to understand, we believe that an expert TBL user is not required for such a task.

There are several machine learning algorithms such as Support Vector Machines or Neural Networks. However, these algorithms require large amounts of training data. We need a technique that will improve the performance of an existing system, use the sequential ordering of data and can work with limited training data. TBL has these features. Moreover it is a self-correcting method. These advantages of TBL makes it a good candidate for our purposes.

The rest of this chapter has the following organization. Section 4.1 presents a brief overview of the TBL algorithm. Section 4.2 recounts the application of the TBL to improve the performance of the rule-based entry tagger, and Section 4.3 explains the experiments and results.

4.1 Transformation-based Learning

Transformation-based Learning (TBL), first proposed by Brill (1995), is a rule-based machine learning algorithm. It has been applied since to various NLP tasks, such as:

- Prepositional phrase attachment (Brill and Resnik, 1994),
- Part of speech tagging (Brill, 1995),
- Parsing (Brill, 1996),
- Spelling correction (Mangu and Brill, 1997),
- Segmentation and message understanding (Day et al., 1997),
- Dialog act tagging (Samuel et al., 1998),
- Ellipsis resolution (Hardt, 1998),
- Phrase chunking (Ramshaw and Marcus, 1999; Florian et al., 2000),
- Word sense disambiguation (Lager and Zinovjeva, 2001),
- Constraint grammar tagging (Lager, 2001), and
- Word alignment (Ayan et al., 2005).

TBL has been shown to be fairly resistant to overtraining (Ramshaw and Marcus, 1994). Furthermore, the resulting model is easy to review and understand.

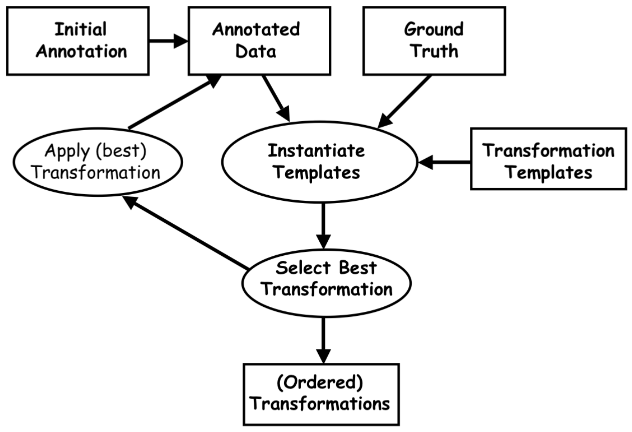


Figure 4.2: TBL Algorithm

Figure 4.2 shows the general architecture of the TBL algorithm, which begins with an initial state and a set of template rules. It requires a correctly annotated training corpus, or ground-truth, for the learning (or training) process. The iterative learning process acquires an ordered list of rules or transformations that correct the errors in this initial state. At each iteration, templates are instantiated, and the transformation which achieves the largest benefit during application is selected. During the learning process, the templates of allowable transformations limit the search space for possible transformation rules. These transformation templates describe valid transformations and must include features that will reliably indicate whether a transformation is applicable. The proposed transformations form by instantiation of the transformation templates in the context of erroneous tags. The learning algorithm stops when no improvement can be made to the current state of the training data or when a pre-specified threshold is reached. The output lists the ordered transformations or rules.

During the application of the learned transformations to new data, the transformations happen deterministically, in the order they are learned to where they can be applied. A transformation may change the result of a previously applied transformation. The output, the final state of the data, applies after all transformations. Algorithm 4.1 gives the pseudo-code for the TBL learning algorithm.

Algorithm 4.1: Pseudo-code for TBL

```
repeat
    Generate all rules that correct at least one error
    for each rule do
        Apply to a copy of the most recent state of the training set
        Score the result using the objective function
    Select the rule with the best score
    Update the training set by applying the selected rule
until score is smaller than some pre-set threshold
```

A transformation modifies a tag when its context (neighboring tags or tokens) matches the context described by the transformation. Two parts comprise a transformation: a rewrite rule—what to replace— and a triggering environment—when to replace. A typical rewrite rule is:

`Change the annotation from` a_a `to` a_b

A typical triggering environment is:

`The preceding word is` w_a

In fact, a transformation can be thought as an if-then statement.

`If triggering environment then rewrite rule`

To apply TBL to a new problem, one should specify:

1. An initial state annotation (i.e., a baseline prediction)

2. A set of allowable templates (i.e., the rewrite rules and the triggering environ-

ments)

3. An objective function for comparing the corpus against ground-truth and choosing the best transformation during the learning process.

The objective function is usually evaluation function, hence, the algorithm can work toward optimizing its evaluation function, as an error-driven approach (Florian and Ngai, 2001). Furthermore, TBL can be termed as greedy error minimization.

TBL differs from conventional symbolic machine learning techniques because (Bringmann et al., 2002):

1. The learning phase of TBL involves an iterative procedure, and TBL performs multiple passes of predictions, and not "oneshot" learning;

2. TBL uses intermediate prediction results;

3. TBL begins with plausible initial baseline predictions and refines on this baseline.

Although TBL presents many advantages and achieves state-of-the-art performance on several tasks, it has one drawback in that it requires a lengthy training time. Several acceleration methods have been proposed. For instance, Ramshaw and Marcus (1994) described a method that makes the update process more efficient, but it requires an enormous amount of memory. The ICA system (Hepple, 2000), another method, introduced independence assumptions on the training samples to reduce training time. Although this method reduces the training time dramatically, the performance also decreases because since the independence assumptions do not hold all the time. Samuel (1998) presented a Monte Carlo approach to TBL. In this method, while determining the "best" transformation at each iteration, only a fraction of the possible rules are randomly selected for estimation. In another study, the μ-TBL system (Lager, 1999) proposed attempts to reduce training time with a more efficient Prolog implementation and an introduction of "lazy" learning. Roche and Schabes (1995) showed that the application of a TBL can be considerably quicker if the rules compile in a finite-state transducer.

Algorithm 4.2: Pseudo-code for *fnTBL*

```
foreach rule r that corrects at least one error do
    store good(r) = the number of errors corrected by r
    store bad(r) = the number of errors introduced by r
Select the rule b with the best score
repeat
    foreach sample s do apply b Considering only samples in the set
    $Us|bchangess\ V(s)$, where $V(s)$ is the set of samples whose tag might
    depend on s: Update good(r) and bad(r) for all stored rules, discarding
    rules whose good(r) reaches 0
    Add rules with a positive good(r) not yet stored; Select the rule b with the
    best score
until the best score is smaller than the threshold t
```

fnTBL (Ngai and Florian, 2001) presents a fast version of TBL that preserves the performance of the algorithm. Instead of regenerating the rules (r) each time, the rules are stored with two values: the number of errors corrected by the rule, *good(r)*, and the number of errors introduced by the rule, *bad(r)*. These values update with each newly selected rule. Algorithm 4.2 presents the *fnTBL* algorithm.

In this thesis, *fnTBL* overcomes the lengthy training time associated with the TBL approach. Our research contribution demonstrates that this method applies effectively to a miniscule set of training data.

4.2 Application of TBL to Entry Tagging

This section describes how TBL adapts to the problem of dictionary entry tagging. TBL applies at two points in the entry tagging system: to render the font style of the tokens correctly and to label the tags of the tokens correctly[1]. Although the ultimate goal aims at improving tagging results, and font style plays a crucial role in identifying tags. The rule-based entry tagger relies on font style, which can be identified incorrectly. Therefore, we also investigate whether improving font style accuracy using TBL will further improve tagging results by applying it in three configurations:

1. To improve font style
2. To improve tagging
3. To improve both, iteratively

Figure 4.3 shows the phases of the TBL application. We applied TBL in six configurations:

- Result1 presents the rule-based entry tagging results with the font style assigned by document image analysis: this includes the initial results and provides the baseline.
- Result2 is obtained by applying TBL only to tagging.

[1]In reality, TBL improves the accuracy of tags and phrase boundary flags. In this thesis, "application of TBL to tagging" denotes the application of TBL to tags and phrase boundary flags together.

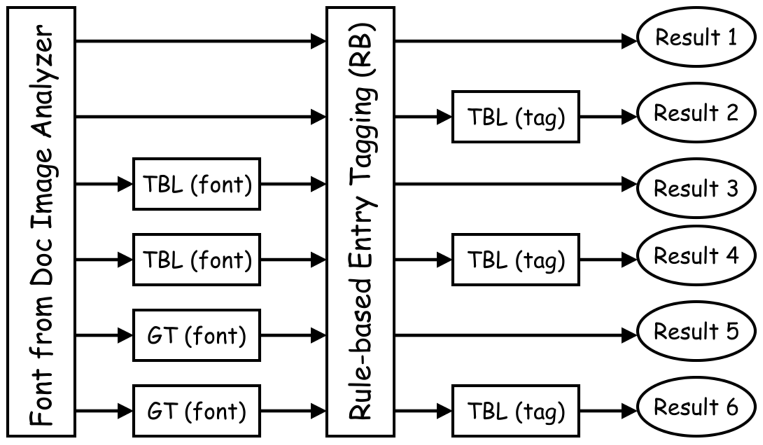


Figure 4.3: Phases of TBL Application to Entry Tagging

- Result3 shows the application of TBL to improve the font style accuracy. These changed font styles then apply to the rule-based entry tagging method.

- Result4 illustrates the application of TBL first to font styles, then to the tagging.

- Result5 derives from feeding the correct font styles to the rule-based entry tagging method and shows the upper bound for tagging given all correct font styles.

- Result6, the application of TBL to tagging given correct font styles, and indicates the upper bound for tagging, and TBL employs the correct font styles.

The last two locate the upper bound when we use the manually corrected font styles in the ground-truth data, relaying the upper bound in tagging accuracy with perfect font style.

In the transformation templates, the tokens themselves become features, i.e., the items in the triggering environment, because the token's content helps to indicate the role. For instance, a comma and a period may differ in functionality when tagging the dictionary. However, allowing transformations to refer to tokens, i.e., *lexicalized* transformations, may lose some relevant information due to sparsity. To overcome the data sparseness problem, a *type* assigns to each token to classify the token's content. We identified eight types for our purposes:

1. uppercase
2. lowercase
3. numeric
4. punctuation
5. symbol
6. non-Latin
7. capitalized
8. other

Each type may play an important role with respect to the identification of information types. For instance, punctuations and symbols are separators in most dictionaries. Some information types, such as examples of usage, may begin with uppercase letters, and a synonym or cross reference may be represented with all capital letters. Non-Latin tokens represent non-English languages. Before applying the TBL algorithm, each token identifies with one of these eight types.

For TBL on font style, the transformation templates contain three features:

1. Token

2. Token's type

3. Token's font style

For TBL on tagging, templates contain four features:

1. Token

2. Token's type

3. Token's font style

4. Token's original tag

Phase	**Transformation Template**
Font	$type_{n-2}$ and $font_{n-1}$ and $type_n$ and $font_{n-1}$ and $font_{n-2}$
Font	$token_{n-2}$ and $token_{n-1}$ and $font_n$
Font	$token_{n-1}$ and $type_n$
Font	$font_{n-1}$ and $token_n$
Font	$font_{n-2}$ and $font_{n-1}$ and $token_n$ and $type_{n-1}$ and $type_{n-2}$
Tag	$tag_{n-7..n-1}$ and $token_{n-1}$ and $font_n$
Tag	$token_n$ and $font_{n-1}$ and tag_{n-1}
Tag	$token_n$ and $font_{n-1}$
Tag	$type_{n-1}$ and $font_n$ and $token_{n-1}$
Tag	$type_{n-1}$ and $font_n$ and $type_{n-1}$
Tag	tag_{n-1} and tag_{n-2} and $font_n$
Tag	tag_{n-2} and tag_{n-1} and $font_n$ and $font_{n-1}$ and $font_{n-2}$
Tag	$type_n$ and $font_n$ and tag_{n-1} and $font_{n-1}$
Tag	tag_{n-2} and tag_{n-1} and $font_n$ and $type_n$ and $font_{n-1}$ and $type_{n-1}$
Phrase	tag_{n-1} and tag_n and $font_{n-1}$ and $font_n$ and $type_{n-1}$ and $type_n$
Phrase	$token_{n-1}$ and tag_{n-1} and tag_n and $font_n$ and $type_{n-1}$

Table 4.1: Sample Transformations Templates.

The parameters needed to apply TBL to a new problem appeared in Section 4.1. For font style, the initial state annotations occur by document image analysis, and, for tagging, the rule-based entry tagging method assigns the initial state of the tokens' tags. The templates for font style accuracy improvement consist of those from studying the data and all templates using all features within a window of five tokens (i.e., two preceding tokens, the current token, and two following tokens). For tagging accuracy improvement, the transformation templates result from studying dictionaries and errors in the entry tagging results. The TBL templates for dictionaries were prepared by the author. The errors in the initial annotation (i.e., font style assigned by the OCR and tags assigned by the rule-based entry tagging) were studied and modeled during the template preparation. Table 4.1 gives examples of transformation templates. For instance, the third template ($token_{n-1}$ and $type_n$) is for font style correction, and it checks the previous token and current token's type.

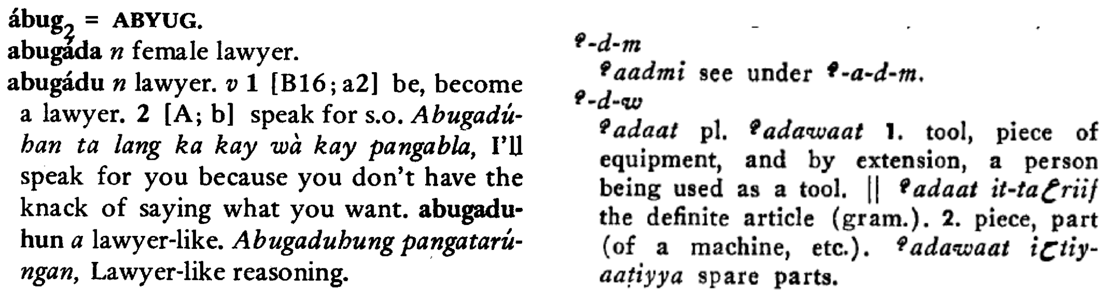
The objective function for evaluating transformations in both cases is the accuracy of the classification, minimizing the number of errors.

4.3 Experiments

	Cebuano	**Iraqi-Arabic**
# of Pages	1163	507
# of Font Styles	4	3
# of Tags	18	26

Table 4.2: Information About the Dictionaries Used in TBL Experiments

This section describes the evaluation of the adaptive TBL application to the entry tagging problem. TBL was evaluated on two bilingual dictionaries, Cebuano-



ábug₂ = ABYUG.
abugáda *n* female lawyer.
abugádu *n* lawyer. *v* **1** **[B16; a2]** be, become a lawyer. **2** **[A; b]** speak for s.o. ***Abugadúhan ta lang ka kay wà kay pangabla,*** **I'll speak for you because you don't have the knack of saying what you want. abugaduhun** *a* **lawyer-like.** ***Abugadubung pangatarúngan,*** **Lawyer-like reasoning.**

ʔ-d-m
ʔaadmi see under ***ʔ-a-d-m.***
ʔ-d-w
ʔadaat pl. ***ʔadawaat*** **1.** tool, piece of equipment, and by extension, a person being used as a tool. || ***ʔadaat it-taʕriif*** the definite article (gram.). **2.** piece, part (of a machine, etc.). ***ʔadawaat iʕtiyaaṭiyya*** spare parts.

Figure 4.4: Sample Entries from the Cebuano-English (Left) and Iraqi Arabic-English (Right) Dictionaries.

English (Wolff, 1972) and Iraqi Arabic-English (Woodhead and Beene, 2003). Figure 4.4 shows sample entries and Table 4.2 presents information about these dictionaries, both of which have complex structures with many information types. For the experiments, we used a publicly available implementation of TBL's fast version, *fnTBL*[2], described in Section 4.1. The default stopping threshold in the learning algorithm was defined in the toolkit.

	Cebuano		**Iraqi-Arabic**	
	Training	**Testing**	**Training**	**Testing**
# of Pages	8	6	8	6
# of Entries	156	137	232	175
# of Tokens	8370	6251	6130	4708
# of Non-punctuation Tokens	6691	4940	4621	3467

Table 4.3: Information About the Training and Testing Data Used in TBL Experiments

From each dictionary, we employed eight randomly selected pages from the dictionaries to train TBL, and six additional randomly selected pages for testing. The font style and tag of each token on these pages were corrected manually from

[2]http://nlp.cs.jhu.edu/rflorian/fntbl

an initial run[3], and the data were tokenized. In tokenized data, the punctuations tags do not have much weight at the end, so we are particularly interested in tokens from other tags, i.e., non-punctuation tokens. Table 4.3 presents the number of entries, tokens, and non-punctuation tokens the training and test data contain for each dictionary.

The evaluation metric represents the percentage of accuracy for non-punctuation tokens, i.e., the number of correctly identified tags for non-punctuation tokens divided by the total number of non-punctuation tokens/phrases. The running times were fast. The learning phase of TBL took less than one minute for each run, and application of learned transformations to the whole dictionary took less than two minutes.

We report how TBL affects accuracy of tagging when applied to font styles, tags, and font styles and tags together, iteratively. To find the upper bound tagging results with correct font styles, we also conducted a rule-based entry tagger using manually corrected font styles and applied TBL for tagging accuracy improvement to these results. We should note that feeding the correct font to the rule-based entry tagger does not necessarily result in totally correct data, as it may still contain noise from document image analysis or ambiguity in the entry. Only the font style information remains correct.

Three sets of experiments were conducted to observe:

1. The effects of TBL (Section 4.3.1),

[3]BRIDGE tool described in Section 3.4 provides ground-truth preparation mechanisms.

2. The effects of different training data (Section 4.3.2), and
3. The effects of training data size (Section 4.3.3).

4.3.1 TBL on Font Styles and Tags

In the first set of experiments, TBL addressed font style and tagging on the data from two dictionaries, Cebuano-English and Iraqi Arabic-English.

	Cebuano	**Iraqi Arabic**
Original	84.43	94.15
TBL(font)	97.07	98.13

Table 4.4: Font Style Accuracy Results for Non-punctuation Tokens for Two Dictionaries

Table 4.4 presents the accuracy of font styles on six pages of test data before and after applying TBL to the font style of the non-punctuation tokens. TBL improved the accuracy of font styles and resulted in very accurate font styles. Although the initial font style accuracy of the Cebuano dictionary was 10% less than the Iraqi Arabic dictionary, after TBL both dictionaries possessed similar font style accuracy (97% and 98%). The remaining errors occurred usually either because of OCR error or a lack of representation in the training data.

Table 4.5 summarizes the results of our tagging accuracy experiments for two dictionaries. In the tables, RB indicates the rule-based entry tagging method, TBL(tag) denotes the TBL run on tags, TBL(font) represents the TBL run on font style, and GT(font) is the ground-truth font style. In each case, the initial font style information consists of the one provided by document image analysis. We tabulated

	Cebuano		**Iraqi Arabic**	
	Token	**Phrase**	**Token**	**Phrase**
RB	83.25	64.08	90.89	82.72
RB + TBL(tag)	91.44	87.37	94.05	92.33
TBL(font) + RB	87.99	72.44	91.46	83.48
TBL(font) + RB + TBL(tag)	93.06	90.19	94.30	92.58
GT(font) + RB	90.76	74.71	91.74	83.90
GT(font) + RB + TBL(tag)	95.74	92.29	94.54	93.11

Table 4.5: Tagging Accuracy Results for Non-punctuation Tokens and Phrases for Two Dictionaries

percentages of tagging accuracy of individual non-punctuation tokens and phrases. In phrase accuracy, if a group of consequent tokens comprises one tag as a phrase in the ground-truth, the tagging of the phrase is correct only if the same group of tokens was assigned the same tag as a phrase in the result (as in Chapter 3). The results for token and phrase accuracy are presented for three different sets:

1. The entry tagger using the font style provided by document image analysis.
2. The entry tagger using the font style after TBL is applied to the font style.
3. The entry tagger using the manually corrected font style (i.e., the ground-truth font style).

All reported results, with the exception of the token accuracies for two cases for the Iraqi Arabic dictionary, namely using TBL(font) vs. GT(font) and using TBL(font) and TBL(tag) together vs. using GT(font) and TBL(tag), are statistically significant within the 95% confidence interval with two-tailed paired t-tests[4].

[4]The t-tests are performed on the results of individual entries.

Using TBL(font) instead of the initial font styles improved initial accuracy as much as 4.74% for tokens and 8.36% for phrases in the Cebuano dictionary, which has a much lower initial font style accuracy than the Iraqi Arabic dictionary. Using the GT(font) further increased the tagging accuracy by 2.77% for tokens and 2.27% for phrases for the Cebuano dictionary. For the Iraqi Arabic dictionary, using TBL(font) and GT(font) resulted in an improvement of 0.57% and 0.85% for tokens and 0.74% and 1.18% for phrases, respectively. The improvements in these two dictionaries differ because the initial font style accuracy for the Iraqi Arabic dictionary appears very high, but for the Cebuano dictionary, potentially useful font style information (namely, the font style for POS tokens) is often incorrect in the initial run.

Using TBL(tag) alone improved rule-based method results by 8.19% and 3.16% for tokens, and by 23.25% and 9.61% for phrases in Cebuano and Iraqi Arabic dictionaries, respectively. The last two rows in Table 4.5 show the upper bound. For the two dictionaries, the results using TBL(font) and TBL(tag) together is 2.68% and 0.24% for token accuracy, and 2.10% and 0.53% for phrase accuracy less than the upper bound of using the GT(font) and TBL(tag) together. Applying TBL to font styles resulted in a higher accuracy than applying TBL to tagging. The number of tag types (18 and 26) is much greater than that of font style types (4 and 3), so the TBL application on tags may require more training data than the font style to perform as well as the TBL application on font style.

In summary, applying TBL with the same templates to two different dictionaries given limited training data resulted in a performance increase in all cases, and the greatest increases occurred in phrase accuracy. Applying TBL first to font style further increased the accuracy, especially for the dictionaries in which font style plays an important role to distinguish different information types. For the Cebuano dictionary, in which the initial font style accuracy did not rate high, applying TBL to font styles first will yield better entry tagging performance.

4.3.2 Effect of Training Data

	Cebuano		**Iraqi Arabic**	
	Token	**Phrase**	**Token**	**Phrase**
RB	81.46±1.14	62.38±1.09	92.10±0.69	85.05±1.64
RB+TBL(tag)	89.34±0.96	85.17±1.55	94.94±0.56	93.25±0.87
TBL(font)+RB	87.40±1.69	71.97±1.26	93.20±1.02	85.49±1.13
TBL(font)+RB+TBL(tag)	93.13±1.58	90.48±0.80	94.88±0.56	93.03±0.70
GT(font)+RB	89.25±1.57	73.13±1.02	93.02±0.58	85.03±2.28
GT(font)+RB+TBL(tag)	95.31±1.43	91.89±1.80	95.32±0.65	93.36±0.81

Table 4.6: Average Tagging Accuracy Results with Standard Deviation for Ten Runs, Using Different Eight Pages for Training and Six Pages for Testing

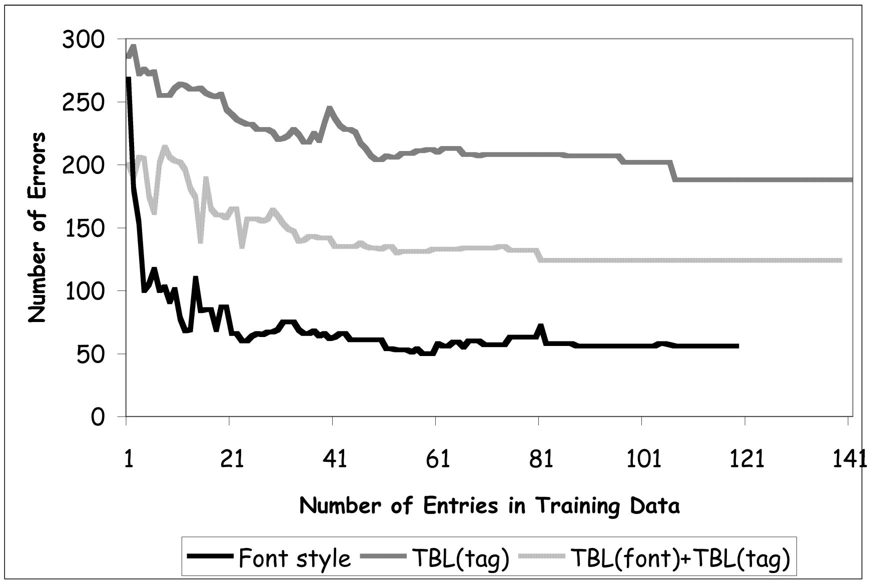
This experiment measures the robustness of TBL application given different training data, using *k*-fold cross validation. For each fold, TBL trained for learning transformation rules on eight pages randomly selected from the 14 pages for which ground-truth exists, and the remaining six pages were reserved for testing. This process was repeated ten times, calculating average accuracy and the standard deviations results, which Table 4.6 presents. The accuracy results coincide with the results presented in Table 4.5, and the standard deviation rests between 0.56–2.28. This consistency and small standard deviation suggest that using different training data does not dramatically affect the performance.

4.3.3 Effect of Training Data Size

The entry tagging problem faces one important challenge, differing from other tasks to which TBL has been applied. Each dictionary has a different structure and different noise patterns. TBL must train for each dictionary, which requires preparing ground-truth manually for each dictionary before TBL application. Moreover, although each dictionary has hundreds of pages, it is not feasible to employ a significant portion of the dictionary for training. This increases the necessary human labor and makes the approach more similar to manual construction. We aim, however, to minimize human effort during the dictionary acquisition, so the training data should be small enough for someone to annotate the ground-truth quickly for each dictionary.

This experiment calculates the minimum quantity of training data necessary for a reasonable improvement in font style and tagging accuracy. For this purpose, the effect of the training data size is investigated by increasing the training data size for TBL training one entry at a time. The entries were added in the order of the number of errors they contain, starting with the entry with maximum errors. Then, the TBL, trained with these entries, tested on two pages. The reason for using only two test data pages is that, if such a method determines the minimal training data required to obtain reasonable performance, then the test data should also be extremely limited to reduce the amount of human provided data and effort.

Figure 4.5 shows the number of font style and tagging errors for non-punctuation tokens on two test pages as a function of the number of entries in the training data.



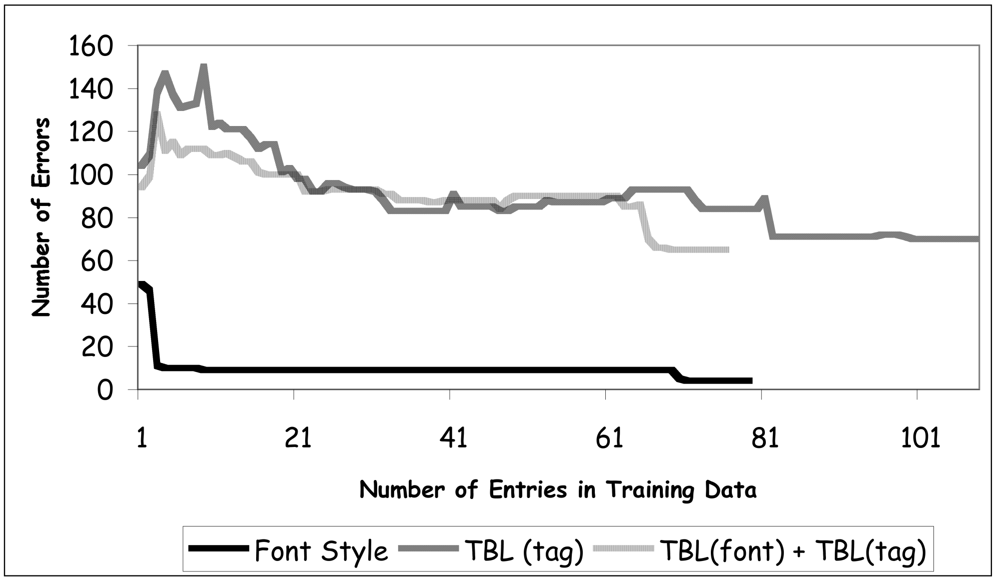


Figure 4.5: The Number of Errors in Two Test Pages as a Function of the Number of Entries in the Training Data for Cebuano-English (top) and Iraqi Arabic-English (bottom) Dictionaries

The first graph look at for the Cebuano-English dictionary, and the second graph represents the Iraqi Arabic-English dictionary. The tagging results appear when using font style from document image analysis and font style after TBL. In these graphs, the number of errors declined dramatically with the addition of the first entries. For the Cebuano dictionary, after 108 entries all stabilized. For the Iraqi-Arabic dictionary, the stabilization differed for each case, but, after 99 entries were added to the training data, all stabilized.

For tagging, the number of errors do not decline as steeply as that for font style. We believe this difference occurs because of the number of tags (18 and 26) and font styles (4 and 3). By adding entries to the training data singly, and by finding the point when the number of errors on selected entries stabilizes, we can determine the minimum training data size to obtain a reasonable performance increase with TBL.

No	**Triggering Environment**	**Change To**
10	$type_{n-2}$ = lowercase and $type_{n-1}$ = punctuation and $type_n$ = capitalized and $font_{n+1}$ = normal and $font_{n+2}$ = normal	normal
15	$font_{n-1}$ = italic and $type_n$ = lowercase and $type_{n+1}$ = lowercase and $font_{n+2}$ = italic	italic
18	$token_n$ = the first token in the entry	bold
1	$token_n$ = a and tag_{n-1} = translation and tag_{n+1} = translation	translation
4	$tag_{n-7..n-1}$ = example and $token_{n-1}$ = , and $font_n$ = bold	example translation
2	$type_n$ = lowercase and $font_n$ = normal and tag_{n-1} = translation and $font_{n-1}$ = normal	translation
9	$token_{n-1}$ = , and $font_n$ = italic and $type_n$ = capitalized	example translation
8	tag_{n-2} = example translation and tag_{n-1} = separator and tag_n = example translation and $type_n$ = capitalized	continuation of a phrase
11	tag_{n-2} = example and tag_{n-1} = separator and tag_n = example and $type_n$ = capitalized	continuation of a phrase

Table 4.7: Sample Transformations Instantiated for Cebuano Dictionary. The numbers are the Orders of the Transformations in the Final Transformation List. *Continuation of a Phrase* Indicates This Token Merges with the Previous One to Form a Phrase.

4.3.4 Example Results

This section shows some example results from the Cebuano-English dictionary, and Table 4.7 lists transformation rules discovered for the Cebuano dictionary for font style and tagging. In this table, the *No* indicates the order of the transformation in the final list, and the triggering environment denotes the condition the transformation will be applied and the *Change To* column shows the new font or tag when this condition holds. These transformations were instantiated using the pre-defined templates, as in Table 4.1. The templates were prepared once, and same templates apply to all dictionaries. Appendix D and E present the first 20 transformations discovered by TBL for Cebuano and Iraqi-Arabic dictionaries respectively.

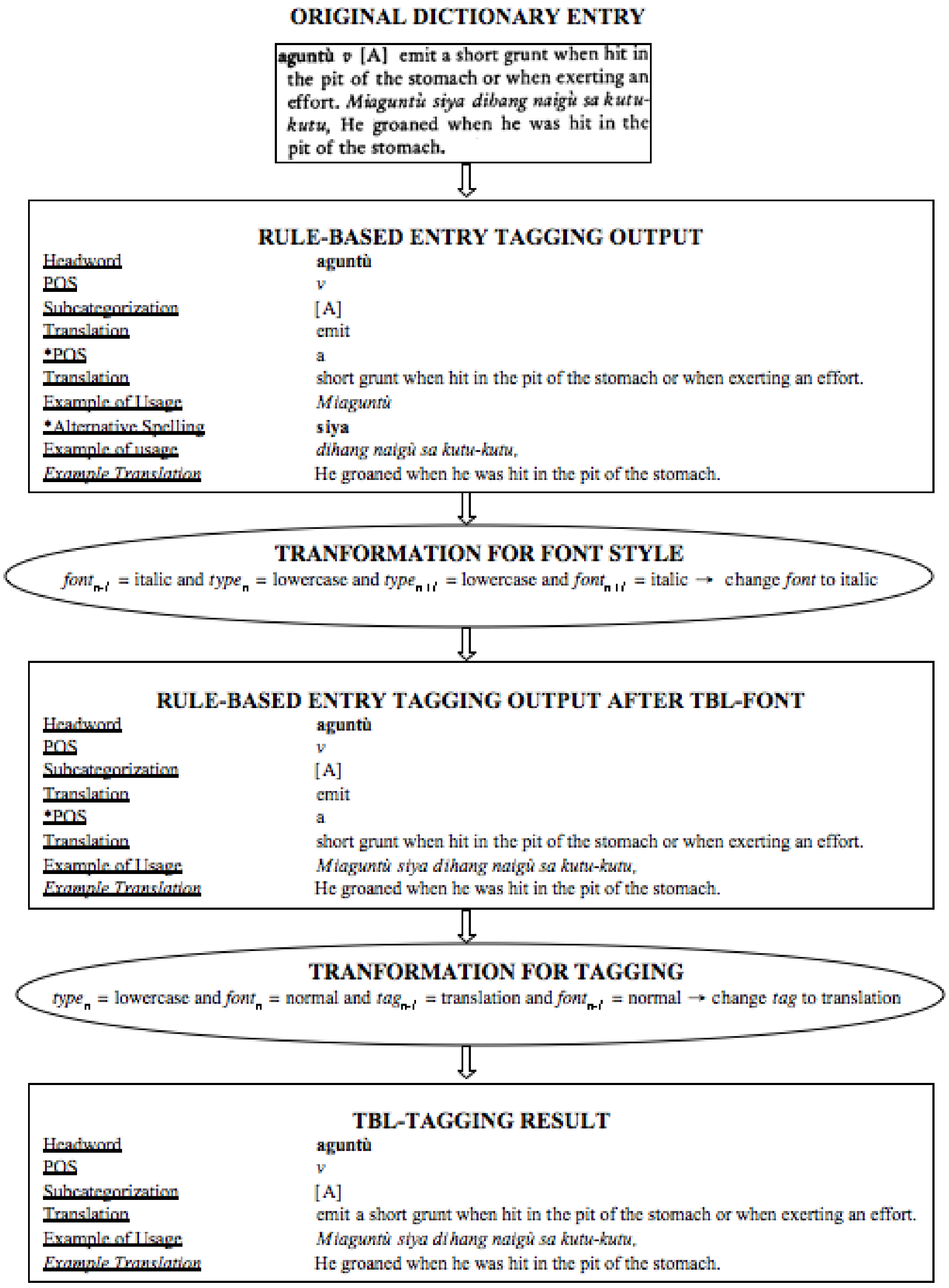
Figure 4.6 presents one Cebuano entry before and after TBL. First the actual dictionary entry is shown. Then the rule-based entry tagging output is given. The font styles assigned by the OCR is also shown (i.e., if a token is bold in the OCR result, it is written in bold style in the figure), and the tags with * indicate incorrect tags. Then TBL is applied to the font style, and one transformation corrects the incorrect bold style and makes it italic. When this corrected font style is used in rule-based entry tagging, the incorrect tag *Alternative Spelling* is corrected as *Example of Usage*. Then TBL is applied to the result of rule-based entry tagger. One transformation corrects the other incorrect tag, and the final entry is tagged correctly.

TBL introduced new errors in some cases. One error we observed occurs when an example of usage translation is assigned a tag before any example of usage tag in an entry. However, TBL introduces fewer errors than the corrected errors, therefore TBL increased the accuracy of the initial results.

4.4 Summary

In this chapter, we introduce a new adaptable approach to improve the initial performance of entry tagging system and present an adaptation of transformation-based learning to the problem of entry tagging. TBL applies at two points during entry tagging, on font style and tagging.

The application of TBL to entry tagging has been evaluated on two different data sets, and the results indicate that application of TBL yielded significant improvements over the initial font style and tagging results, even with limited user provided training data. For two different dictionaries, the application of TBL resulted in an increase from 84% and 94% to 97% and 98% in font style accuracy, from 83% and 91% to 93% and 94% in tagging accuracy of tokens, and from 64% and 83% to 90% and 93% in tagging accuracy of phrases. If the initial font style does not appear accurate, improving the font style with TBL further assisted the tagging accuracy by as much as 2.62% for tokens and 2.82% for phrases, compared to using TBL only for tagging. This result cannot be attributed to a low rule-based baseline as a similar, even a slightly lower base line originates from an HMM-trained system as described in Chapter 3. Results generated using a method to compensate for sparse training data. The similarity of performance between the different dictionaries using the same templates shows the method as adaptive and generic. In another important feature, the approach learns by using extremely limited training data.



ORIGINAL DICTIONARY ENTRY

aguntù *v* [A] emit a short grunt when hit in the pit of the stomach or when exerting an effort. *Miaguntù siya dihang naigù sa kutu-kutu,* He groaned when he was hit in the pit of the stomach.

RULE-BASED ENTRY TAGGING OUTPUT

Headword	**aguntù**
POS	*v*
Subcategorization	[A]
Translation	emit
*POS	a
Translation	short grunt when hit in the pit of the stomach or when exerting an effort.
Example of Usage	*Miaguntù*
*Alternative Spelling	**siya**
Example of usage	*dihang naigù sa kutu-kutu,*
Example Translation	He groaned when he was hit in the pit of the stomach.

TRANFORMATION FOR FONT STYLE

$font_{n-1}$ = italic and $type_n$ = lowercase and $type_{n+1}$ = lowercase and $font_{n+1}$ = italic → change *font* to italic

RULE-BASED ENTRY TAGGING OUTPUT AFTER TBL-FONT

Headword	**aguntù**
POS	*v*
Subcategorization	[A]
Translation	emit
*POS	a
Translation	short grunt when hit in the pit of the stomach or when exerting an effort.
Example of Usage	*Miaguntù siya dihang naigù sa kutu-kutu,*
Example Translation	He groaned when he was hit in the pit of the stomach.

TRANFORMATION FOR TAGGING

$type_n$ = lowercase and $font_n$ = normal and tag_{n-1} = translation and $font_{n-1}$ = normal → change *tag* to translation

TBL-TAGGING RESULT

Headword	**aguntù**
POS	*v*
Subcategorization	[A]
Translation	emit a short grunt when hit in the pit of the stomach or when exerting an effort.
Example of Usage	*Miaguntù siya dihang naigù sa kutu-kutu,*
Example Translation	He groaned when he was hit in the pit of the stomach.

Figure 4.6: Illustration of TBL Application in One Sample Cebuano Entry. * Indicates the Incorrect Tags.

Chapter 5

Morphology Induction from Limited Noisy Data Using Approximate String Matching

Many languages have various printed resources, not have electronic resources. and for such languages, dictionaries and the Bible may offer their first electronic resources. If these resources are transformed into useful form, they can be very valuable. Resnik et al. (1999), for instance, suggested several uses for the Bible as a multilingual parallel corpus. Bilingual dictionaries are essential for the translation process, but to make effective use of a dictionary for translation, users must be able to access it using the root form of morphologically deformed variants. This holds especially importance for morphologically rich languages, such as Turkish and Finnish. Morphological segmentation — i.e., segmenting a word into its morphemes — assists in tasks other than dictionary usage. For instance, data sparseness for morphologically rich languages can be significantly decreased when decomposing words. Finding the root form of a morphologically deformed variant requires such segmentation of a word, which in turn requires either stemming, i.e., to reduce a word to its stem or root form, or morphological analysis of the word, i.e., to parse a word into its component morphemes.

A number of rule-based morphological segmentation systems exist for a range of languages. Traditionally, developing these morphological analyzers or stemming tools requires expert knowledge of linguists. Such methods have good accuracy but are time consuming to develop. More recently, some unsupervised methods have been proposed for learning morphology of a language and morphology segmentation, however, these methods depend on large amounts of corpora. To develop morphological analyzers for languages with limited resources (in terms of experienced linguists or electronic data), we must move beyond data intensive methods developed for rich resource languages that rely on expert knowledge for rule-based methods or large amounts of data for statistical methods. Data driven methods are not suitable given scarce data, and new approaches that can deal with limited, and perhaps noisy, data are necessary for the languages with limited resources.

Motivated by the need for new approaches to learning morphology in languages with scarce electronic resources, this chapter presents a new natural language morphology induction framework, *Morphology Induction from Noisy Data (MIND)*, for discovering morphemes from limited, noisy data found in dictionaries. Dictionary data is limited compared to the large amounts of corpus, and dictionaries are, by nature, terse. Each headword appears once, and morphological variants are often not given, but the examples of usage may include a morphological variant of the headword. The MIND framework is developed employing this characteristic of the dictionaries. In practice, the method described in this chapter can use any electronic dictionary, if the dictionary contains the data the method requires, specifically headwords and corresponding examples of usage. In this thesis, the dictionary data obtained by the entry tagging system described in Chapter 3, and extended in Chapter 4 are used for morphology discovery. The approach introduces the novel application of the longest common substring and string edit distance metrics to the information found in dictionary entries, namely headwords and examples of usage. These string searching algorithms morphologically segment words and identify prefixes, suffixes, circumfixes, and infixes in limited and probably noisy data. Evaluation results show that these algorithms can, in fact, segment words into roots and affixes from the limited dictionary data, and extract the affixes, which, in turn, allow developers to build dictionaries for multilingual tasks rapidly. This also helps users with a limited knowledge of the language in question. In addition, the morphological analysis can feed other NLP tools requiring lexicons. MIND's greatest strength lies in the fact it does not need large amounts of data or expert knowledge of the language.

The rest of this chapter is organized as follows. Section 5.1 presents background on morphology and describes some of the terminology used throughout this chapter. Section 5.2 discusses the importance of morphology in NLP, and Section 5.3 briefly describes previous work on morphology discovery. Sections 5.4 through 5.7 detail the MIND framework and present various experiments that demonstrate the effectiveness of MIND.

5.1 Morphology

Morphology, a field within linguistics, studies the internal structure of words and patterns of word formation. The meaningful parts into which words can be divided are termed the *morphemes*. These building blocks of words represent as the smallest units of a natural language that carry linguistic information about meaning or function. A word, typically composed of one or more morphemes, can contain a *free* morpheme, which can occur as a word by itself, or *bound* morphemes, which appears only in combination with other morphemes. Therefore, a word consists of either a single free morpheme or a combination of a single morpheme with other free or bound morphemes.

The *stem*[1], the main morpheme of the word, belongs to a lexical category, such as noun, verb, adjective, etc. *Affix*es, on the other hand, are bound morphemes that may not belong to a lexical category. Stems contribute the main meaning of the word, and affixes add various meanings, such as part of speech changes (attaching -*ly* to an English stem changes the part of speech from adjective to adverb), or semantic alterations (attaching *un*- in English negates the meaning).

Affixes divide into several types, depending on their position with reference to the stem, i.e., the free morpheme. This thesis is interested in four types of affixes:

- **Prefixes** attached to the front of the stem

[1]Stems are often roots, i.e., is the primary lexical unit of a word and cannot be reduced into smaller morphemes. A stem, however, can also be morphologically complex, such as a compound word or a word with derivational morphemes. In this thesis, stem and root are used interchangeably.

happy — unhappy

- **Suffixes** attached to the end of the stem

  play — playing

- **Infixes** inserted into the stem

  dako [*big*] — dagko [*plural of dako*] (Cebuano)

- **Circumfixes** attached to the front and end of the stem

  baddang [verb, *help*] — kabaddangan [noun, *helpfulness*] (Cebuano)

Affixes are also differentiated as inflectional and derivational. *Inflectional affix*es as highly productive[2] and regular. They modify a word's tense, number, aspect, gender, etc. (e.g., dog [singular] — dogs [plural], ringo-ga [nominative] — ringo-o [accusative] (Japanese)). *Derivational affix*es are less productive, may produce idiosyncratic changes in meaning, and often change the part of speech. Derivational affixes produce new word (e.g., happy [adjective] + -ness → happiness [noun]).

*Allomorph*s are variants of a morpheme and refer to the cases when a morpheme can vary in sound without changing its meaning in different contexts. One example appears in the plural marker in English which is sometimes pronounced as /-z/, /-s/ or /-IZ/.

Languages often separate into four classes, given the richness of their morphology (Spencer, 1991):

1. Isolating languages, such as Mandarin Chinese, usually do not contain bound morphemes.

[2]A morpheme is *productive* if it is found in many combinations.

2. In inflectional languages, such as Spanish or Latin, a single bound morpheme conveys multiple linguistic information.

3. Agglutinative languages, such as Turkish, Finnish, or Swahili, can have multiple bound morphemes attached to a word, and each bound morpheme adds different linguistic information.

4. Polysynthetic languages, such as Inuktikut, use morphology to combine syntactically related components of a sentence together.

Another term used in this thesis, *morphophonemics* (also called morphophonology), refers to the study of phonemic differences between allomorphs of the same morpheme and phonetic modifications of morphemes when combined. In English, different plural markers such as *-s* (cat — cats), *-es* (fox — foxes), and *-ves* (leaf — leaves) present examples of morphophonological alternatives of a morpheme. Inflecting and agglutinative languages may have complex morphophonemics. For instance, Turkish has several morphophonemic rules including (Oflazer, 2003):

1. Vowel harmony: Vowels in suffixes, with very minor exceptions, agree with previous vowels in certain aspects
   (*a, ı, o, u* (i.e., back vowels) are followed by *a*, and *e, i, ö, ü* (i.e., front vowels) are followed by *e*)
   araba [*car*] + PLURAL (-lar/-ler) → arabalar [*cars*]
   ev [*house*] + PLURAL (-lar/-ler) → evler [*houses*]

2. Consonant agreement: Suffix beginning consonants, with very minor excep-

tions, agree with previous consonant in certain aspects, such as voiced

(*ç, f, h, k, p, s, ş, t* are followed by *t*, and the rest of the consonants are followed by *d*)

çicek [*flower*] + FROM (-den/-ten) → çicekten [*from the flower*]

ev [*house*] + FROM (-den/-ten) → evden [*from the house*]

3. Vowel ellipsis: Suffix initial vowels or stem final vowels can sometimes be dropped

   ata [*assign*] + PROGRESSIVE (-ıyor) → atıyor [*is assigning*]

4. Consonant ellipsis: Certain suffix initial consonants can be dropped if preceded by a consonant

   kalem [*pen*] + GENITIVE (-si) → kalemi [*his pen*]

The next section discusses computational morphology and how NLP uses morphology.

5.2 Morphology and NLP

Natural language applications employ morphological knowledge for linguistic resource (corpora, lexical databases) enrichment, inferring semantic properties of words, and terminology acquisition. It also plays a role in parsing, part-of-speech tagging, and lemmatization. Moreover, morphological segmentation benefits NLP tasks such as:

- machine translation (Goldwater and McClosky, 2005),

- speech recognition (Siivola et al., 2003; Hacioglu et al., 2003; Kurimo et al., 2006b),
- information retrieval (Monz and de Rijke, 2002),
- question answering, (Monz, 2003),
- information extraction (Tür et al., 2003), and
- text generation (Levison and Lessard, 1995)

Morphology can assist with issues such as the recognition of unknown words, the acquisition of term variation, or the detection of paraphrases (Daille et al., 2002). For some applications, such as text retrieval, even approximate morphological analysis, being the process of using noisy statistical methods to parse a word into its component morphemes, would deal better with large vocabularies by reducing vocabulary size.

Most research in morphology has focused on English and European languages. With the recent interest and studies in multilingual NLP, the need for morphological descriptions of various languages increases. Especially for morphologically rich languages, where data sparseness presents an important challenge, morphological segmentation can reduce the size of language lexicons, and thus cope with the problem. In Turkish, for instance, millions of word forms can be obtained from one Turkish word (Hankamer, 1989), and in Finnish, a single verb may appear in thousands of different forms (Karlsson, 1987). For such languages, while the lexicon may contain large amounts of words, many valid word forms will not appear due to the high number of possible word forms. Morphological segmentation (or morphological analysis is possible) and utilization of morphemes instead of words as the basic representation units in statistical models seems a plausible approach to the data sparseness problem (Creutz and Lagus, 2002). This representation reduces the size of language lexicons.

A number of rule-based morphological analyzers and morphological segmentation systems exist for most commonly studied languages. However, construction of a comprehensive morphological analyzer requires linguistic knowledge (sometimes the aid of a linguist, most certainly a native speaker) and a considerable amount of time. This costs both time and expense. Moreover, the systems need updating to reflect the changes in the languages, such as new words and their morphological variations (Demberg, 2007). Rule-based systems usually lead to good results; however, when expert knowledge is not present or quick results are needed, unsupervised methods offer attractive solutions. Unsupervised methods, on the other hand, are cheap, because they are language independent and require only unannotated text. Unsupervised methods can also help as an introductory step toward building rule-based systems. An expert can correct the initial results of unsupervised methods to obtain better performance. Unsupervised methods operate using large data, and, for languages without large corpora, we need new approaches for morphology discovery. Motivated by this idea, this thesis introduces an unsupervised method that uses a dictionary instead of unannotated data. Although, perhaps, this does not seem like a true unsupervised method, we classify it as such because it is language independent and requires no expert knowledge.

5.3 Related Work

This work's' main contribution involves the ability to produce robust morphological analysis using sparse data and minimal supervision. Despite a lack of resources, we can discover the affixes and handle the phonological variation among morphemes (i.e., morphophonemics).

Much of the previous work on computational morphology has focused on automatically discovering affix lists, or morpheme-like units, and morpheme segmentation. This section first discusses the general methods used in morpheme discovery, and then describes some of this work in more detail. Most of the methods discussed presuppose larger data sources than we envision for our application.

We can group the studies on morpheme discovery based on the general approach they take. The widely used approaches are as follows:

- "Letter Successor Variety" (LSV) (Harris, 1955) is one of the earliest and most popular concepts used for morpheme discovery. It is the number of different characters that follow the string in words in the collection being considered. LSV has inspired several approaches to finding a list of frequent affixes in natural languages and morpheme segmentation (Hafer and Weiss, 1974; Dejean, 1998; Monson, 2004; Bernhard, 2006; Bordag, 2006; Keshava and Pitler, 2006; Demberg, 2007).

- Using generative models is another approach to discover regular morphological patterns. Some work are based on generative models formulated in a Bayesian framework (Brent, 1999; Creutz, 2003) while some are using "Minimal Description Length" (MDL) (Brent, 1993b; Brent et al., 1995; Goldsmith, 2001; Creutz and Lagus, 2002; Argamon et al., 2004; Creutz and Lagus, 2005; Hu et al., 2005b).

- Clustering of affixes or regular transformational patterns for the same stem is another widely used approach in affix discovery (Jacquemin, 1997; Gaussier, 1999; Schone and Jurafsky, 2000; Yarowsky and Wicentowski, 2000; Goldsmith, 2001; Neuvel and Fulop, 2002; Monson, 2004; Freitag, 2005; Demberg, 2007).

- A few approaches also take into account syntactic and semantic information from the context in which the word occurs (Jacquemin, 1997; Schone and Jurafsky, 2000; Yarowsky and Wicentowski, 2000; Baroni et al., 2002; Neuvel and Fulop, 2002; Bordag, 2006). Exploiting semantic and syntactic information is very attractive because it adds an additional dimension. However, these approaches have to cope with more severe data sparseness issues than approaches that emphasize word-internal cues and they are usually computationally expensive.

Nearly all of the work in morpheme discovery specializes in prefixes and suffixes. We only aware of one work (Schone and Jurafsky, 2001) that can explicitly deal with circumfixes, and none with infixes. Wicentowski (2004) states that his WordFrame model was "successful in handling infixation without explicitly modeling these phenomena", however, no specific results or samples are given for infixation.

Most of the work in morpheme acquisition is done for English or other European languages. Only some of the approaches proposed are applicable to agglutinative or morphologically rich languages (Wicentowski, 2004; Creutz and Lagus, 2002; Creutz, 2003; Argamon et al., 2004; Creutz and Lagus, 2005; Hu et al., 2005a; Demberg, 2007). Also a few recent works concentrated on low-density languages (Sharma et al., 2002; Johnson and Martin, 2003; Wicentowski, 2004; Hu et al., 2005a). However, the techniques described in these studies were only applied to one particular language, with the exception of the work by Wicentowski (2004), which is a supervised model. Our model is particularly suitable for low-density languages since it can work with limited resources. Also the only resource it needs is electronic dictionaries.

5.3.1 Deeper Analysis of the Main Methods for Morphological Analysis

In this section, the main approaches used for the computational analysis of morphology are discussed in more detail.

Harris (1955) proposed LSV which is based on the frequency of letter sequences. The term "successor frequency" refers to the number of distinct letters that appear immediately after the n^{th} letter in a word, in a given corpus. The idea behind LSV is the assumption that the number of variant successor letters, i.e., distinct letters, rises at morpheme boundaries. Therefore, the morpheme boundaries can be discovered by calculating successor varieties. Dejean (1998) extended this idea by first inducing a list of most frequent morphemes and then using those morphemes for word segmentation. He uses successor and predecessor frequencies of letters in a given sequence of letters during the induction step for finding suffixes and prefixes and utilizes frequency limits for accepting morphemes.

The other popular method in morphological studies is MDL (Rissanen, 1989; Barron et al., 1998), which is based on information theory and related to Bayesian inference. It is used to provide a generic model for the model selection problem. MDL is based on the idea that any regularity in the data can be used to compress the data, i.e., to describe it using fewer symbols than the number of symbols needed to describe the data literally (Grünwald, 2005). The more regularities there are, the more the data can be compressed. Brent (1993a) and Brent et al. (1995) were the first to use MDL to find the most data compressing morphemes in a corpus. The first approach (Brent, 1993a) was based on the spellings of the words, while the other (Brent et al., 1995) used the syntactic categories from a tagged corpus as well. However, this latter approach is not knowledge free and requires part of speech tagged data. Although the method was found promising, the exhaustive search proved to be impractical. Snover and Brent (2001) and Snover et al. (2002) later extended these methods with a probabilistic word-generation model and new search algorithms. The hill climbing search in the first study was further improved in the latter study by the introduction of a novel search algorithm, Directed Search.

A tool that relies on MDL is *Linguistica* (Goldsmith, 2001), an unsupervised morphological analyzer that finds morphological paradigms called *signatures*. Signatures are sets of stems and suffixes (or prefixes) grouped together. Linguistica assumes that words are composed of one stem and one possibly empty suffix (or prefix). Although recursive structures are allowed, this method is not well suited for agglutinative languages since words in agglutinative languages can have multiple bound morphemes. The Expectation Maximization (EM) algorithm and triage procedures are used to segment a list of words taken from a given corpus using some predefined heuristics until the length of the morphological grammar converges to a minimum. The source code and executable of Linguistica are freely available on the Internet[3] and Linguistica has been used in the evaluation of many studies, including the work in this thesis. Goldsmith (2006) gives a detailed description of Linguistica.

There are several studies based on Linguistica. In one extension, Hu et al. (2005a) employed a string edit distance algorithm for discovering the finite state automata that constitute the morphologies of languages with rich morphologies. Hu et al. (2005b), on the other hand, proposed combining redundant signatures using syntactic context, therefore minimizing the description length of the model.

The described approaches consider only individual words without regard to their contexts, or to their semantic content. A different approach is using Latent Semantic Analysis (LSA) (Dumais et al., 1988) for in finding affixes (Schone and Jurafsky, 2000). LSA is a technique for capturing the essential relationships between text documents and word meaning, or semantics, by producing a set of concepts related to the documents and terms they contain. Schone and Jurafsky (2000) stated that semantics and LSA can play a key part in knowledge-free morphology induction. In the extension of their work, they incorporated induced orthographic, syntactic, and transitive information and showed that the frequency-based approaches help

[3]http://humanities.uchicago.edu/faculty/goldsmith/Linguistica2000/

semantics-based approach (Schone and Jurafsky, 2001). These two works are shown to improve on the performance of Linguistica. In a conceptually similar approach, Baroni et al. (2002) produced a ranked list of morphologically related pairs from a corpus using orthographic and semantic similarity with minimum edit distance and mutual information metrics.

Morfessor algorithms, proposed by Creutz and Lagus have been proved to be more successful at handling morphologically rich languages, such as Finnish. Three versions have been proposed:

- In the first version, *Morfessor Baseline*[4] (Creutz and Lagus, 2002; Creutz, 2003), two unsupervised methods are used for word segmentation, one based on MDL and one based on maximum likelihood (ML). In this model, a lexicon of morphemes is constructed such that words in the corpus can be constructed by the concatenation of morphemes in the lexicon. The goal is to find the optimal lexicon and segmentation. One drawback of the model is that many frequent word forms remain unsplit, whereas rare word forms are excessively split.

- *Morfessor Categories-ML* (Creutz and Lugas, 2004) is an extension addressing these shortcomings of the baseline model. It is based on ML and it reanalyzes a segmentation produced by the Morfessor Baseline algorithm. Words are represented as Hidden Markov Models (HMMs) in this model, and word frequencies

[4]Morfessor Baseline software is publicly available and can be reached at http://www.cis.hut.fi/projects/morpho/

are not utilized.

- The final extension *Morfessor Categories-MAP* (Creutz and Lagus, 2005) makes use of word frequencies by representing frequent entities in the lexicon directly, thus allowing direct access to such entities. The model is based on maximum a posteriori probability (MAP) (which is essentially equivalent to MDL), and both the meaning and form of morpheme-like units are used in morpheme segmentation task.

Another system, Whole Word Morphologizer (Neuvel and Fulop, 2002), is not entirely a knowledge-free method; it relies on a POS-tagged lexicon as input. It is capable of inducing morphological relationships without attempting to discover or identify morphemes. The similarities and differences between words are discovered by aligning the words, and then used to generate new words beyond the learning sample.

A noise-robust supervised morphological analyzer, the WordFrame model (Wicentowski, 2004), can be useful for co-training with low-accuracy unsupervised algorithms. It uses inflection-root pairs, where unseen inflections are transformed into their corresponding root forms. The model can handle prefixes, suffixes, stem-internal vowel shifts, and point-of-affixation stem changes.

The recent PASCAL Challenge on Unsupervised Segmentation of Words into Morphemes, or Morpho Challenge 2005, certainly escalated the interest in morphology segmentation problem. The objective of the challenge was to design a statistical machine learning algorithm that segments words into the morphemes (Kurimo et al., 2006a). The algorithms of the participants were evaluated on data sets from three languages, English, Finnish, and Turkish in two complementary ways. The first evaluation was the comparison of the proposed segmentations to a text gold standard, with f-measure being the performance measure. In additional speech recognition experiments, the performance measure was phoneme error rate, i.e., the sum of the number of substituted, inserted, and deleted phonemes divided by the number of phonemes in the correct transcription of the data. The Morpho Challenge 2005 contributed to the morpheme segmentation problem by both drawing attention to this problem and providing researchers with corpora and a common ground to compare different algorithms by making data sets (and gold standard sets if you have certain qualifications) publicly available.

In Morpho Challenge 2005, Bernhard (2006) achieved the best f-measure performance in Finnish and Turkish, and became third in English. The algorithm he used first extracts a preliminary set of prefixes and suffixes using transitional probabilities between substrings. Next, the stems are extracted following some constraints. Then, each morphological segment is assigned a category an affix category and new affixes are discovered and validated by aligning word segments for a specific stem. Finally, using the list of extracted segments and their frequencies, the A* algorithm is applied to find the best segmentation for each word that is not used in the learning process.

Bordag (2006) achieved the best precision in Turkish and Finnish, both morphologically rich languages. The recall of the algorithm (therefore f-measure) was low. The two-step algorithm first finds a relatively small number of mostly correct segmentations using an algorithm of the letter successor variety, then trains a trie-based classifier which finds morpheme boundaries in all words using these segmentations.

The best performance in English in Morpho Challenge 2005 was achieved by the RePortS algorithm (Keshava and Pitler, 2006). However, the recall of the algorithm on morphologically rich languages was low (in fact its results were not reported for other languages in the PASCAL results). This simple algorithm uses words that appear as substrings of other words and transitional probabilities together to detect morpheme boundaries. Three modifications which Demberg (2007) proposed to the RePortS algorithm improved results in several languages. These modifications include

1. generation of an intermediary high-precision stem candidate list

2. use of a language model to disambiguate between alternative segmentations

3. learning patterns for regular stem variation

The new algorithm can also identify stem variation through ablauting and umlauting [5].

In yet another extension of the RePortS algorithm, Dasgupta and Ng (2007) relies on the new capabilities of:

1. incorrect attachment detection using relative corpus frequency and suffix level similarity, and

[5]An English example of ablauting and umlauting would be the past and past participle form of the verb *sing*: *sing-sang-sung*

2. orthographic rules and allomorph induction for segmenting words where roots exhibit spelling changes during morpheme attachments.

The improved model showed robust performance for several languages with different levels of morphological complexity. Dasgupta and Ng (2007) and the work in this thesis can handle similar morphophonemic rules; they occur at morpheme boundaries and only at roots (with the assumption that roots change during the attachment, not the morphemes or that when changes occur in morphemes, these morphemes are treated as different morphemes), and the root exhibits either a one character insertion, one character deletion, or one character substitution.

We were not able to impose any of the work described above directly to low-density languages because most of the described methods require clean (i.e., perfect) and most of the time large amounts of data, which may not exist for languages with limited electronic resources. For such languages, morphology induction is still a problem. The work in this thesis is applicable to very limited and even noisy data. It is not entirely unsupervised since it relies on the existence of a dictionary. But then, dictionaries are usually one of the first electronic resources a language has. Although the dictionaries used in this study were digitalized by BRIDGE, any electronic dictionary with example of usage information can be used by our methods.

Our method MIND can be used when resources for a language are scarce. One important strength of MIND is its ability to induce morphophonemics. With the exception of the work of Yarowsky and Wicentowski (2000), Dasgupta and Ng (2007), and Demberg (2007), the above studies are not able to cope with morphophonological changes. Among these studies, Yarowsky and Wicentowski (2000) assumed the existence of a preliminary morphological hypothesis and part of speech tags in order to induce irregular morphology in their “nearly” unsupervised induction of inflectional morphology. MIND, which was developed prior to the other two studies, can identify the stem-final changes in the words when a morpheme is attached. Another strength of MIND is that it can be used for co-training with especially low-accuracy unsupervised algorithms.

The rest of this chapter describes the MIND framework in detail and discusses the experiments and results.

5.4 Approach

Dictionaries are structured documents, and entries in the dictionaries often headwords and many contain the examples of how these words are used in context, i.e., examples of usage. The MIND framework uses the information in dictionaries. The algorithm assumes that each example of usage will contain at least one instance of the headword, either in its root form, or as one of its morphological variants. The intuition behind the method is to find the headword occurrence in the example of usage for each headword—example of usage pair, and then extract the affix if the headword is in one of its morphological variants. We should note that perfect data are not required for this method. The data may have noise such as OCR errors, and the approach described successfully identifies the affixes in such noisy data if the noise is not in the affix itself[6] . Moreover, MIND can identify most of the morphophonemics rules, specifically the ones at the morpheme boundaries.

5.5 Framework

The MIND framework is composed of two stages, *exact match* and *approximate match*, and uses three string distance metrics, the *longest common substring* (LCS), *approximate string matching with k differences* (k-DIFF), and *string edit distance* (SED). The objective of the exact match stage is extracting affixes that do not undergo morphophonological changes, while the approximate match stage is designed to deal with such changes or noisy data. Although approximate match can be used to find exact matches to identify prefixes, suffixes, and circumfixes, it is not possible to differentiate between infixes and OCR errors. For these reasons, two cases are processed separately. Each identified affix is assigned two counts, *exact count* and *approximate count*. Only the affixes with a positive exact count are included in the final affix list.

5.5.1 String Searching Algorithms

The next three sections briefly describe three string distance metrics MIND is based on and the adaptations made to find the edit operations in the SED. Then, we explain how these metrics are used in the MIND framework in Section 5.6.

[6]If the affix in the stem has OCR errors, the affix is extracted, but it is not a correct affix. Correcting the OCR errors in the affixes is beyond the scope of this thesis.

5.5.1.1 Longest Common Substring (LCS)

Given two strings $p = p_1...p_n$ and $q = q_1...q_m$, LCS finds the longest contiguous sequence appearing both in p and q. The longest common substring is not the same as the longest common subsequence because the longest common subsequence need not be contiguous.

There is a dynamic programming solution for LCS[7] that finds the longest common substring for two strings with lengths n and m in $O(nm)$.

Let $L[i, j]$ be the maximum length of common strings that end at $p[i]$ and $q[j]$. Then,

$$L[i,j] \quad = \quad \begin{cases} 1 + L[i-1, j-1] & if \ p[i] = q[j] \\ 0 & if \ p[i] \neq q[j], i = 0, \ or \ j = 0 \end{cases} \tag{5.1}$$

$$\begin{array}{c|ccccc} & A & B & C & D & B \\ \hline A & 1 & 0 & 0 & 0 & 0 \\ C & 0 & 0 & 1 & 0 & 0 \\ B & 0 & 1 & 0 & 0 & 1 \\ C & 0 & 0 & \mathbf{2} & 0 & 0 \\ A & 1 & 0 & 0 & 0 & 0 \end{array}$$

Figure 5.1: Longest Common Substring Example

Using the Equation 5.1, we can calculate the values in the matrix L. At the end, the value in the cell with the maximum value gives us the length of the longest common substring. The indices of this cell are the indices of the last character of LCS in corresponding strings, i.e., if $L[i, j]$ is the cell with the maximum value l, then $p' = p_{i-l+1}...p_i$ and $q' = q_{j-l+1}...q_j$ are same and the LCS of p and q. Figure 5.1

[7]http://www.ics.uci.edu/ dan/class/161/notes/6/Dynamic.html

gives the computed values of L matrix for the strings `ABCDB` and `ACBCA`. The length of LCS for these two strings is 2 (4^{th} row, 3^{rd} column) corresponding to the sequence `BC`.

5.5.1.2 String Edit Distance (SED)

SED, also known as Levenshtein distance (Wagner and Fischer, 1974), is a measure of the similarity between two strings. The SED has been used in spell checking, speech recognition, DNA analysis and plagiarism detection. Given two strings p and q, SED is defined as the minimum number of edit operations which transforms p to q using three kinds of edit operations:

1. *Insertion:* Insert an element to p.
2. *Deletion:* Delete an element from p.
3. *Substitution:* Replace an element of p with another element.

The edit operations may have different costs depending on the application. In the MIND framework, the cost of each edit operation is set to 1. A solution based on dynamic programming computes the distance between strings in $O(mn)$, where m and n are the lengths of the strings p and q respectively (Wagner and Fischer, 1974).

Let D be the distance matrix. The cell $D[n, m]$ gives us the SED between two strings of length n and m. The initialization phase is:

$$
\begin{aligned}
D[0,0] &= 0 \\
D[i,0] &= i \quad \text{for } i = 1..n \\
D[0,j] &= j \quad \text{for } j = 1..m
\end{aligned} \tag{5.2}
$$

And the main algorithm is:

$$
D[i,j] = min \begin{cases} D[i,j-1] + cost_{ins}, \\ D[i-1,j] + cost_{del}, \\ D[i-1,j-1] + cost_{subs} \end{cases} \tag{5.3}
$$

		T	H	U	R	S	D	A	Y
	0	1	2	3	4	5	6	7	8
T	1	0	1	2	3	4	5	6	7
U	2	1	1	1	2	3	4	5	6
E	3	2	2	2	2	3	4	5	6
S	4	3	3	3	3	2	3	4	5
D	5	4	4	4	4	3	2	3	4
A	6	5	5	5	5	4	3	2	3
Y	7	6	6	6	6	5	4	3	**2**

Figure 5.2: String Edit Distance Example.

Figure 5.2 shows the calculated contents of the distance matrix for the strings `THURSDAY` and `TUESDAY`. The SED of these two strings is 2 (shown in bold), the value in the last cell of the matrix.

5.5.1.3 Approximate string matching with k-differences (k-DIFF)

Given two strings p and q, the problem of approximate string matching with k-differences is finding all the occurrences of substrings of q whose edit distance from

$$
\begin{array}{c|cccccccccccccc}
 & & A & B & G & E & A & C & E & F & A & B & C & D & H \\
\hline
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
A & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\
B & 2 & 1 & 0 & 1 & 2 & 1 & 1 & 2 & 2 & 1 & 0 & 1 & 2 & 2 \\
C & 3 & 2 & 1 & 1 & 2 & 2 & 1 & 2 & 3 & 2 & 1 & 0 & 1 & 2 \\
D & 4 & 3 & 2 & 2 & 2 & 3 & 2 & 2 & 3 & 3 & 2 & 1 & 0 & 1 \\
E & 5 & 4 & 3 & 3 & 2 & 3 & 3 & 2 & 3 & 4 & 3 & 2 & \mathbf{1} & \mathbf{1}
\end{array}
$$

Figure 5.3: Approximate String Matching with k-differences Example.

p is less than or equal to k. The k-differences approximate string matching problem received much attention due to its importance in molecular biology and text retrieval. In the algorithm, the three edit operations, namely insertion, deletion and substitution, are allowed. A dynamic programming solution to this problem, which requires quadratic time and space, is the same as the classical string edit distance solution with one difference: the values of the first row of the table are initialized to 0 (Sellers, 1980). This initialization means that the cost of insertions of letters of q at the beginning of p is zero. Then the contents of the distance matrix is calculated using the Equation 5.3. In the end, any value not exceeding k in the last row indicates a position in q where a substring having at most k differences from p ends. Consequently, the minimum value on the last row gives us the distance of the closest occurrence of the pattern. Figure 5.3 illustrates the calculated values of the distance matrix for the strings p = `ABCDE` and q = `ABGEACEFABCDH` with $k = 1$. The substrings `ABCD` and `ABCDH` in q have at most one difference from p (in fact for this example the difference is one since there is no zero value on the last row of the matrix.).

The MIND framework uses the cut-off algorithm proposed by Ukkonen (1985b).

Algorithm 5.1: Cut-off Dynamic Programming k-differences String Matching Algorithm

```
$t = k + 1$;
**for** $i = 0$ *to* $m$ **do**
    $L_i = i$;
**for** $j = 1$ *to* $n$ **do**
    $d = 0$;
    **for** $i = 1$ *to* $t$ **do**
        **if** $p_i = q_i$ **then**
            $e = d$;
        **else**
            $e = min\{L_{i-1}, L_i, d\} + 1$;
        $d = L_i$;
        $L_i = e$;
    **while** $Lt < k$ **do**
        $t = t - 1$;
    **if** $t = m$ **then**
        print $j, L_m$;
    **else**
        $t = t + 1$;
```

		A	B	G	E	A	C	E	F	A	B	C	D	H
	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	1	0	1	1	1	0	1	1	1	0	1	1	1	1
B	2	1	0	1	2	1	1	2	2	1	0	1	2	2
C	3		1	1	2		1	2			1	0	1	2
D	4			2	2			2				1	0	1
E	5												**1**	**1**

Figure 5.4: Cut-off k-differences Example. The Empty Cells are not Calculated.

This algorithm is also based on dynamic programming, but the expected running time is reduced to $O(nk)$ by computing only a part of the dynamic programming array. The work-space is reduced to $O(m)$ as well. The idea behind the cut-off algorithm is based on the diagonalwise monotonicity (Ukkonen, 1985a) of the Levenshtein distance table. Algorithm 5.1 outlines the algorithm. Figure 5.4 depicts the cells of the distance matrix that would actually be computed in the cut-off algorithm for the strings $p =$ `ABCDE` and $q =$ `ABGEACEFABCDH` with $k = 1$. The smallest values in the last row in the Figure 5.3 are also present in the last row in the Figure 5.4.

5.5.1.4 String Edit Distance with Edit Operations (SED-path)

The MIND framework needs to trace back the editing operations performed in achieving the minimum cost alignment between two strings to find which characters are inserted. These inserted characters will then be used to determine the affixes. In order to obtain the sequence of edit operations, one can work backwards from the computed distance matrix. For two strings p and q with lengths n and m respectively, the cell $L[n, m]$ of the distance matrix L gives us the SED between p and q. To get to the cell $L[n, m]$, one had to come from one of:

1. $L[n-1, m]$ (insertion)

2. $L[n, m-1]$ (deletion)

3. $L[n-1, m-1]$ (substitution)

Which of the three options was chosen can be reconstructed given these costs, edit operation costs, and the characters $p[n], q[m]$ of the strings. By working backwards, the entire path can be traced and thus the alignment can be reconstructed. However, there are ambiguous cases; the same minimum cost may be obtained by a number of edit operation sequences. In order to address this issue and deal with ambiguity, the following adaptations are done while tracing the path.

Let *path* be the list of editing operations to obtain minimum distance, and *SED-path* be the SED algorithm that also returns a *path*. The length of the *path* is $max(n, m)$, and $path[j]$ contains the edit operation to change $q[j]$ (or $p[j]$ if $n > m$). *Path* can contain four different types of operations:

1. Match (M) (two characters in the strings are same)
2. Substitution (S) (two characters in the strings are different)
3. Insertion (I)
4. Deletion (D)

Our goal is finding affixes and in case of ambiguity the following heuristics are employed for finding the SED operations leading the minimum distance between two strings:

Case 1: If one string is longer than the other, choose I for extra characters

Case 2: Until an M is found, choose I over M/S in case of ambiguity

Case 3: If an M is found previously, choose M/S over I in case of ambiguity

Case 4: If there is an M between two I's, switch this with the last I

Case 1 ensures that if one word has more characters than the other, these characters are represented by an insertion operation.

If there is an ambiguity, and an M/S or I operation have the same minimum cost, Case 2 gives priority to the insertion operation until a match case is encountered, while Case 3 gives priority to match/substitution operations if a match case was seen previously.

Case 4 helps us to localize all the insertion operations. The example below illustrates this. For the headword—candidate example word pair *abirids* $\rightarrow$ *makaabirı́ds*, the *path* changes from (1) to (2) when Case 4 is applied, and as a consequence correct prefix *maka-* is identified as explained in the next section.

(1) `I M I I I M M M S M M` $\Rightarrow$ Prefix *m-*

(2) `I I I I M M M M S M M` $\Rightarrow$ Prefix *maka-*

5.6 Morphology Induction from Noisy Data (MIND)

Figure 5.5 shows the MIND framework which consists of two stages. Using the headword and corresponding example of usage as input, the MIND framework first checks if the headword occurs without any changes or error in the exact match stage using the longest common substring algorithm. If the headword occurs exactly in the example of usage, and if there is an affix, this affix is extracted. If no such occurrence is found, an approximate match search is performed in the second stage using *k*-difference and SED-path algorithms. The objective of the approximate match search is to find morphophonological changes in the stem when an affix is

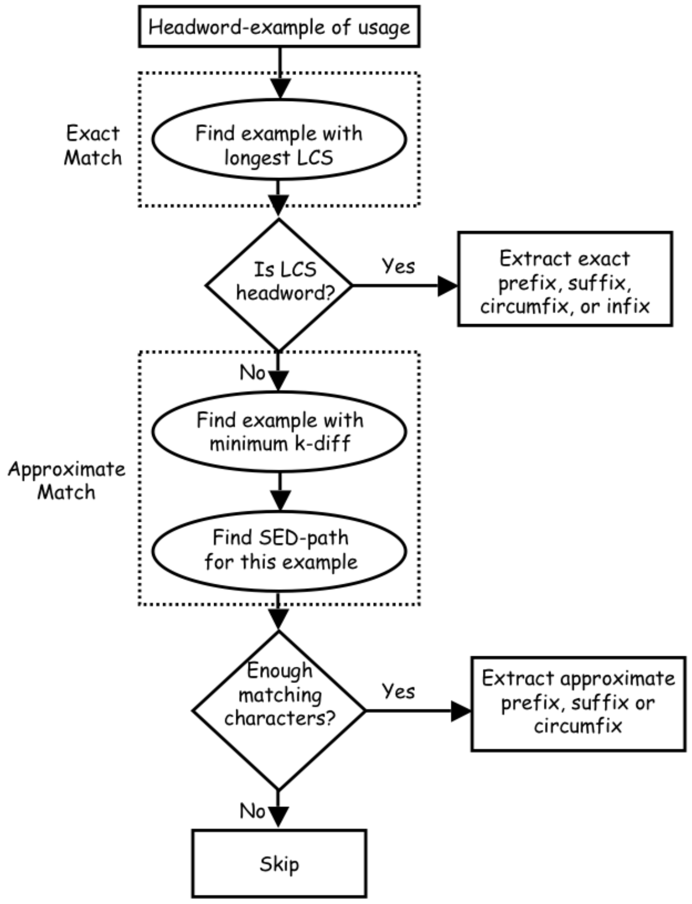

Headword-example of usage
Exact Match
Find example with longest LCS
Is LCS headword?
Yes
Extract exact prefix, suffix, circumfix, or infix
No
Approximate Match
Find example with minimum k-diff
Find SED-path for this example
Enough matching characters?
Yes
Extract approximate prefix, suffix or circumfix
No
Skip

Figure 5.5: MIND Framework Architecture

attached (and to extract morphophonemics rules), and to deal with OCR errors in the stem itself or stem part of the inflected form. Below, these two stages are described in detail.

5.6.1 Exact Match

Given a list of (possibly noisy) headword—example of usage pairs (w, E), the exact match first checks if the headword occurs in E in its root form[8]. If the headword cannot be found in E in its root form, for each e_i in E, the longest common substring

[8]Headwords consisting of one character are not checked in the framework.

of headword and example word, i.e., $LCS(w, e_i)$, is computed. In order to reduce the search space, the example words that are shorter than the headword are skipped[9]. Let e_l be the e_i that has the longest common substring (l) with w. Note that the length of the longest common substring can be at most the length of the headword, in which case the longest common substring is the headword itself. If $w = l$, and for some suffix s and/or some prefix p one of the following conditions is true, then the corresponding affix is extracted.

1. $e_l = ws$ (s is a suffix) or
2. $e_l = pw$ (p is a prefix) or
3. $e_l = pws$ ($p - -s$ is a circumfix)

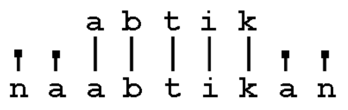
The extracted affixes are added to the induced affix list, and their *exact count*s are incremented.

For the infixes, there is one further step. If $w = w'l$ and $e_l = e'_l l$, $LCS(w', e'_l)$ is computed. If $e'_l = w's$ for some suffix s, then s is added as an infix to the induced affix list. (This means $e_l = w'sl$ where $w = w'l$.)

The following sample run illustrates how the exact match part identifies affixes. Given the Cebuano headword—example of usage pair (*abtik*) — (*naabtikan sad ku sa bátá*), the word *naabtikan* is marked as the candidate that has the longest common substring with headword *abtik*. These two words have the following alignment, and

[9]Although there are some languages, such as Russian, in which headwords may be longer than the inflected forms, such cases are not in the scope of this thesis.

as a result the circumfix *na–an* is extracted. In the illustration below, straight lines represent matches, and short lines ending in square boxes represent insertions.



5.6.2 Approximate Match

There are cases when an exact match of the headword cannot be found in the example of usage. In such cases, there may be an approximate match resulting from an error with OCR or the application of morphophonemic rules[10]. The approximate match stage is designed to deal with such cases separately. For each e_i in E, the difference between headword and example word, $k\text{-}DIFF(w, e_i)$ is computed. The example word that has the minimum difference from the headword is selected as the most likely candidate (e_{cand}). The next step is to find the sequence of the edit operations performed in achieving the minimum distance alignment to transform e_{cand} to w using SED-path algorithm described above[11].

Let $cnt(X)$ be the count of X operation in the computed path. The following conditions are considered as possible errors and no further analysis is done for such cases:

[10]In its current state, MIND does not make any distinctions between noise in the data such as OCR errors and morphophonemic rules.

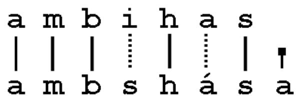
[11]Computing k-difference and the edit path can be done in parallel to reduce the computing time.

$$\begin{aligned}
cnt(M) &= 0 \;\; || \\
cnt(M) &< max(cnt(S), cnt(D), cnt(I)) \;\; || \\
cnt(M) &< cnt(S) + cnt(D) + cnt(I)
\end{aligned}$$

If $cnt(I) = 0$, this case is considered as an approximate root form (with OCR errors). If none of the above is true, then the insertion operations at the beginning and/or at the end of the path are used to identify the type of the affix (prefix, suffix, or circumfix) and the length of the affix (number of insertion operations). The identified affix is added to the affix list, and its *approximate count* is incremented. All the other cases are dismissed as errors. In its current state, infix affixes are not handled in the approximate match case[12].

The following sample shows how approximate match works with noisy data. In the Cebuano input pair (*ambihas*) — (*ambshása pagbutang ang duha ka silya arun makakitá ang maglingkud sa luyu*), the first word in the example of usage has an OCR error, *i* is misrecognized as *s*. Moreover, there is a vowel change in the word caused by the affix. An exact match of the headword cannot be found in the example of usage. In the approximate match stage the k-DIFF algorithm returns *ambshása* as the candidate example of usage word, with a distance 2 with $k = 2$. Then, the SED-path algorithm returns the path $M\ M\ M\ S\ M\ S\ M\ I$ and algorithm successfully concludes that *a* is the suffix as shown below in illustration.

[12]One reason infixes are not handled in approximate match stage is that infixes are usually very short affixes, and it is very hard to distinguish them from OCR errors.



In this example, the substitutions are represented in dotted lines. These are either OCR errors or morphophonological changes occurred in the stem. MIND is capable of extracting *templates* for modeling these substitutions or even insertions and deletions. In the above illustration, MIND extracts the template:

`*i#a#` $\rightarrow$ `*s#á#`

In the templates, * denotes any number of characters and # denotes one character. The meaning of the above template is that a root form ending with an *i* followed by one character followed by an *a* and followed by one character may change when the suffix *a* is added to it. During this change *i* becomes *s* and *a* becomes *á*.

5.7 Experiments

5.7.1 Dictionaries

In our experiments for testing MIND we used two bilingual dictionaries, a Cebuano-English dictionary (Wolff, 1972) and a Turkish-English dictionary (Avery et al., 1974) processed by the BRIDGE system described in Section 3.4. The input to the experiments are the lists of headword—example of usage pairs extracted from these dictionaries. The extracted data are not perfect: it has mistagged information, i.e., it may include some information that is not the headword or example of usage, or some useful information may be missing, and OCR errors may occur. OCR errors can be in different forms: Two words can be merged into one, one word can be split into two, or characters can be misrecognized.

Dictionary	**# of pages**	**# of hw-ex pairs**	**#of words**
Cebuano-all	1163	27129	206149
Turkish-all	1000	27487	111334
Cebuano-20	20	562	4134
Turkish-20	20	503	1849

Table 5.1: Information About the Data Used in Experiments

Along with the headword—example of usage pairs from more than 1000 pages, we randomly selected 20 pages from each dictionary for detailed analysis. Table 5.1 provides details of the data from two dictionaries used in the experiments.

Both Cebuano and Turkish are morphologically rich. Cebuano allows prefixes, suffixes, circumfixes, infixes, while Turkish is an agglutinative language with very productive inflectional and derivational suffixation processes by which it is possible to generate thousands of forms from a given root word. The two dictionaries have different characteristics. The examples of usage in the Cebuano dictionary are complete sentences given in italic font while the Turkish dictionary has phrases, idioms, or complete sentences as examples of usages indicated in bold font. This difference in the structure and representation is nor important for us as long as the information we need is in the dictionaries.

5.7.2 Protocol

For our experiments, MIND was run on all of the data in the headword—example of usage lists.

We evaluated MIND in several ways. The first evaluation is detailed analysis.

For this purpose, we tested the results on a randomly selected 20 pages from each dictionary. As ground-truth, the affixes from each of these 20 pages were manually extracted. During the evaluation, cases where the count of an affix in the ground-truth and result are same as errors if they were extracted from different pairs, i.e., in this detailed analysis both the count and source of the affix should match. We further examined the cause of each error in this data.

We then compared the performance of MIND with the state-of-the-art Linguistica (Goldsmith, 2001) algorithm. Linguistica was trained on two different data sets, and results were analyzed.

The third experiment involves evaluation of the stemming results on a Turkish treebank. The stemming performance using affixes extracted by MIND, and by Linguistica are compared.

Finally, we used the affixes extracted by MIND in morpheme segmentation, and compared our results with the results of Morpho Challenge 2005 participants.

Next four sections describe these experiments and the results.

5.7.3 Analysis

Tables 5.2 and 5.3 show results of MIND runs. Table 5.2 presents the total number of affixes, and Table 5.3 presents the different types of affixes for two dictionaries, Cebuano-English and Turkish-English, and two data sets, the whole dictionary and 20 randomly selected pages. The top portion of the tables provides the exact match results, and the bottom shows the approximate match results. For Cebuano, the

		Cebuano		**Turkish**	
		All pages	**20 pages**	**All pages**	**20 pages**
	Total	26106	542	27314	502
Exact Match	Root form	5727	180	18416	345
	Prefix	10300	197	6	0
	Suffix	1315	16	6983	128
	Infix	25	0	1	0
	Circumfix	717	18	9	0
Approximate Match	App. Root form	1023	14	103	1
	App. Prefix	1697	23	8	1
	App. Suffix	2930	63	168	5
	App. Circumfix	1060	14	20	0
	Couldn't decide	1159	13	765	15

Table 5.2: Total Number of Affixes Extracted from Two Dictionaries Using MIND

		Cebuano		**Turkish**	
		All pages	**20 pages**	**All pages**	**20 pages**
Exact Match	Prefix	180	26	6	0
	Suffix	253	8	447	59
	Infix	11	0	1	0
	Circumfix	221	11	9	0
App. Match	App. Prefix	116	9	8	1
	App. Suffix	199	19	100	5
	App. Circumfix	207	5	20	0

Table 5.3: Number of Different Types of Affixes Extracted from Two Dictionaries Using MIND

approximate match part of the framework finds significantly more affixes than it does for Turkish. The languages possess different characteristics, so in Cebuano, the accents in the vowels frequently change for some affixes. Although MIND incorrectly finds a few prefixes, circumfixes, and infixes for Turkish, these affixes all have count one, using a threshold while accepting the affixes will exclude these incorrect affixes. These affixes occur because of OCR errors (usually merging of two words) or mistagged information during entry tagging process.

Dict.	Affix	Sample words
	mu-	galing/mugaling hikúhíkú/muhikùhíkù
C	nag-	kisdum/nagkisdum kugkugl/nagkugkug
E	mi-	iktin/miiktin kírus/mikárus
B	i-	kunsuylu/ikunsuylu pazíha/iparíha
U	na-	píl/napíl ulatl/naúlat
A	gi-	buga/gibuga dálit/gidádit
N	gi-an	labuk/gilabukan íkug/giikúgan
O	-un	gihay/gihayun gáyung/gayúngun
	-a	pisar/pisara sirnpul/simpúla
	-ı	ad/adı ilaç/ilaeı
T	-i	heves/hevesi ilim/ilmi
U	-a	saz/saza sonsuz/sonsuza
R	-e	deniz/denize zmim/mime
K	-ına	etraf/etrafına kolay/kolayına
I	-ya	hasta/hastaya orta/ortaya
S	-ü	üst/üstü zyüz/yüzü
H	-ini	bel/belini zevk/zevkini
	-ine	derin/derinine iç/içine

Table 5.4: Sample Affixes Extracted from Two Dictionaries

Table 5.4 contains some of the most frequently extracted affixes along with their exact and approximate counts and samples of headword—example of usage word pairs from which they were extracted. Each word segments into one root and one affix, therefore when a word takes multiple affixes, they are all treated as a composite affix (such as -*ına* which is composed of either -*ı* and *na* or -*ın* and *a*).

The extracted templates modeling morphophonemic rules for Turkish are given in Table 5.5. The first template demonstrates the stem final *ç* changing to *c* when attaching the suffix -*u*. Other templates were extracted for Turkish that derived from OCR errors when the headword did not match the morphological variant, as a result of OCR errors either in the headword or in the morphological variant in the example of usage. Fortunately, these templates do not degrade during the stemming

Template	Example
*ç → *c	sonuc + -u → sonucu
e → *i	geçilme + -yor → geçilmiyor
ı# → *D#	akıl + -a → akla
i# → *D#	ilim + -i → ilmi
k → ğ	ilik + -ine + → iliğine
p → *b	muhatap + -ım → muhatabım
t → *d	simit + -i + → simidi
ü# → *D#	gönül + -ünüzden → gönlünüzden

Table 5.5: Correct Morphophonemic Rules Extracted from the Turkish Dictionary. (* Indicates One or More Characters, # Indicates One Character, and D Indicates Deletion of a Character.)

or morpheme segmentation processes, discussed below. In fact, when the presence of a word in the dictionary becomes the stopping condition, some templates indicate a headword with an OCR error. For instance, one common OCR error in Turkish dictionary originates in the misrecognition of superscript[13] 2 as the character z. This is modeled by the template `z*` *rightarrow* `D*`, denoting the first letter z in the headword deletes in the morphological variant. During stemming or morpheme segmentation, this template located the headword in the dictionary.

Dictionary	GT cnt.	Res.cnt.	Misses	Additions
Cebuano	311	314	17	14
Turkish	155	142	8	10

Table 5.6: The Number of Affixes in the Ground-Truth, MIND Results, and the Number of Missed and Added Affixes in the Results of 20 Pages

We analyzed the results of 20 pages in both dictionaries. Table 5.6 shows the number of affixes in the ground-truth and MIND results along with the number of missed and incorrectly added affixes in 20 pages of data. These numbers represent

[13]Superscripts represent the same headword, with different POS.

the count of affixes, not different affixes. MIND missed only 5% of the affixes in the ground-truth in both data sets. When we check the number of different affixes, the results appear lower. For the Cebuano data, five invalid suffixes, three invalid prefixes, and two invalid circumfixes are located (a total of 10), while one valid suffix and one valid circumfix are missed (a total of two). For the Turkish data, three invalid suffixes, one invalid prefix, and two valid suffixes are identified (a total of six), while two valid suffix are missed. The other additions are valid affixes, but their counts are less than the counts in ground-truth. The other missed affixes already included in the extracted affix list, have different counts from those in the ground-truth.

When the invalid affixes in the data are examined, most (six of the Cebuano and all of the Turkish) have count one, and the maximum count in an invalid affix equals five. Therefore, using a low threshold while accepting affixes can eliminate many of the invalid affixes.

Reason	**Cebuano**		**Turkish**	
OCR	8	M→lbi	11	ını→mı or ım
Algorithm	8	(uluy, giuylan)→ not gi-an, -lan is found	7	(alın, alnında)→ not -ında, -da is found
Merge	9	ímung giláug→ímunggiláug	0	-
Split	1	nag-kúgus→nag- kúgus	0	-
Other	5	apr.→april Headword is an abbreviation	0	-

Table 5.7: The Distribution of the Causes of Errors in 20 Pages with Samples

We also examined the cause of each error. Table 5.7 presents the causes of each miss and addition in the output of MIND, with a sample for each cause. We should emphasize that a valid affix such as Turkish suffix *-mı* counts as an error in this detailed analysis, since the suffix *-ını* should be extracted for that particular headword—example of usage pair. An OCR error, such as the misrecognition of *a* as *d*, causes both the missing of the prefix *mag-* and incorrect addition of *mdg-* for Cebuano.

In some cases that, the framework cannot correctly perform identification. These cases usually involve dropping the last vowel because of morphophonemic rules. For the Cebuano dictionary, merge and split causes several errors, and the Turkish data do not have any such errors. Different structure and format of the original dictionaries cause most errors. In the Cebuano dictionary, an italic font indicates example of usages, which results in merges and splits more often.

5.7.4 Comparison to Linguistica

We compared MIND with *Linguistica*, a publicly available, unsupervised, corpus-based morphology learner (Goldsmith, 2001). Linguistica induces paradigms from a noise-free corpus, while MIND employs of string searching algorithms and allows one to deal with noise at the cost of correctness. Linguistica operates on a corpus that may not be available for a low-density language. Moreover, we aim to compare Linguistica with MIND, which uses data in dictionaries. Therefore we trained Linguistica in two different data sets from Turkish dictionary:

- Ling-all: Whole headword-example of usage sentence pairs
- Ling-cand: Headword-candidate example words that MIND returns

In the first case (Ling-all), Linguistica uses more data than MIND, and to avoid any biases resulting from this, we also trained Linguistica using the headword and candidate example word (Ling-cand). During the training, only the suffixes are extracted because Turkish is a suffix-based language. A native speaker conducts the evaluation.

The dictionary data are not noise-free. We used a threshold while deciding which extracted suffixes will appear in the final list. The suffix lists contain suffixes the systems return with counts more than a threshold. The purpose of using a threshold is to decrease the number of invalid affixes caused by the data's noise. For the MIND results, the suffixes over threshold have positive exact counts and total counts (sum of exact and approximate counts) more than the threshold. Although Linguistica is not designed for thresholding, the data used during training is noisy. We explored including only suffixes with a corpus count more than a threshold to eliminate invalid suffixes.

Table 5.8 presents the analysis of the suffix lists produced by Linguistica using two sets of training data and MIND. The results show six threshold values for all of the data. The table gives the total number of suffixes, the percentage of suffixes with a count more than a threshold value, the percentage of invalid suffixes, and percentage of missed suffixes discarded by thresholding for the entire Turkish dictionary. MIND locates a greater number of affixes than Linguistica, with a lower number of invalid affixes. On the other hand, the number of missed affixes is also higher for MIND when given larger threshold values. The cause of this high percentage of missed affixes comes from the particular data containing many affixes with counts

System	**Th.**	**Total**	**% Over Th.**	**% Invalid**	**% Missed**
Ling-cand	0	116	100.00	18.10	0.00
Ling-all	0	274	100.00	34.67	0.00
MIND	0	499	89.58	13.20	3.61
Ling-cand	1	116	98.28	17.54	0.86
Ling-all	1	274	94.89	32.69	1.46
MIND	1	499	50.50	4.37	33.07
Ling-cand	2	116	92.24	16.82	5.17
Ling-all	2	274	87.96	31.12	4.74
MIND	2	499	38.48	4.17	44.49
Ling-cand	3	116	91.38	16.98	6.03
Ling-all	3	274	85.40	31.20	6.57
MIND	3	499	28.86	2.78	53.31
Ling-cand	4	116	81.03	12.77	11.21
Ling-all	4	274	81.39	30.94	9.12
MIND	4	499	25.65	3.13	56.51
Ling-cand	5	116	80.17	12.90	12.07
Ling-all	5	274	79.56	31.19	10.58
MIND	5	499	23.25	2.59	58.72

Table 5.8: Total Number and Percentage of Over the Threshold, Invalid, Missed and Valid Suffixes Found by Linguistica and MIND for Different Threshold Values

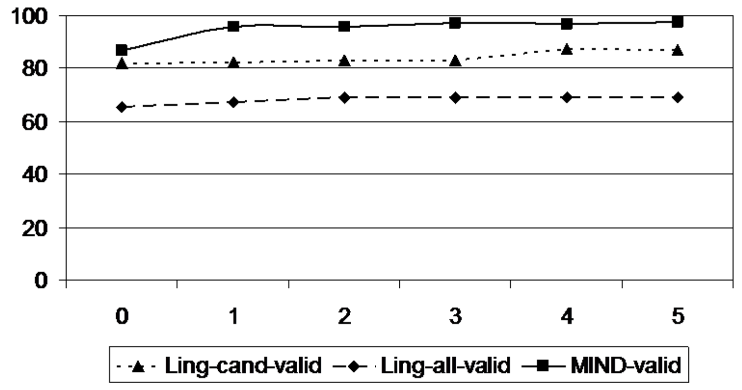


Figure 5.6: Percentage of Valid Suffixes by Linguistica and MIND for Different Threshold Values

less than five. In fact, 41% of the affixes have an exact count of one, which occurs because of the agglutinative nature of Turkish language.

The effect of thresholding can also be examined in Figure 5.6. The graph gives the percentage of valid suffixes as a function of threshold values. MIND takes advantage of thresholding, and the percentage of valid suffixes rapidly decreases for threshold value one. For higher threshold values, the difference remains small. For Turkish data, using threshold value one offers a good choice for MIND.

Table 5.9 presents the same results for 20 pages from the Turkish dictionary for three threshold values. We used only two as the highest threshold value because the amount of data is smaller. MIND performs well even with very little amount of data and finds nearly all valid affixes. Linguistica, on the other hand, locates very few. MIND uses the characteristics of the dictionary data and succeeds at identifying affixes in the data.

System	**Th.**	**Total**	**Over Th.**	**Invalid**	**Missed**
Ling-cand	0	6	100.00	0.00	0.00
Ling-all	0	4	100.00	0.00	0.00
MIND	0	60	96.67	1.72	0.00
Ling-cand	1	6	66.67	0.00	33.33
Ling-all	1	4	100.00	0.00	0.00
MIND	1	60	41.67	0.00	53.33
Ling-cand	2	6	50.00	0.00	50.00
Ling-all	2	4	75.00	0.00	25.00
MIND	2	60	18.33	0.00	76.67

Table 5.9: Total Number and Percentage of Over the Threshold, Invalid, and Missed Suffixes Found by Linguistica and MIND for Different Threshold Values for 20 Pages of Turkish Data

5.7.5 Stemming

In this experiment, our goal is to test the utility of the results. We performed a simple word segmentation with the aim of stripping the inflectional suffixes and finding the bare word form. We implemented a word segmenter, which takes a list of suffixes and their counts from the morphology induction system (Linguistica or MIND), a headword list as a dictionary, a threshold value, and the words from a treebank. For each word in the treebank, a root form (rf) and a usage form (uf) exist. The suffixes with a count more than the threshold index according to their last letters. For each word in the treebank, the first task checks if uf already appears in the dictionary, i.e., in the headword list. If no word is found in the dictionary, we repeatedly attempt to identify the longest suffix that matches the end of uf, and reiterate the dictionary check. The process completes either with a located dictionary word is found or when no suffixes match the end of the word. If the word the segmenter return equals rf in the treebank, the correct count increase, otherwise, this case count as an error. If the suffix associates with a template, the word also converts according to the template. Then it is searched in this format in the dictionary.

We used the METU-Sabanci Turkish Treebank[14], a morphologically and syntactically annotated treebank corpus of 7,262 grammatical sentences (Atalay et al., 2003; Oflazer et al., 2003) for the stemming experiments. The punctuation and multiple parses are skipped,[15] and word segmentation runs on 14,950 unique words. The headword list extracted from the Turkish dictionary operates as the dictionary, which has OCR errors. Therefore, even if the word segmenter returns the correct root form, it may not occur in the dictionary, and the word must be stripped further. In some cases, the templates extracted with an OCR error in the headword help locate the noisy word in the dictionary.

Figure 5.7.5 presents the percentage of correctly segmented words for six threshold values, in which suffixes with counts more than the threshold operate in each case. Again for MIND results, the exact match counts must total more than zero, and the total of exact match and approximate match counts must reach more than the specified threshold. Linguistica uses suffixes with a corpus count more than the threshold. For each threshold value, MIND performed much better than Ling-cand. MIND outperformed Ling-all for thresholds 0 and 1. For the other values, the difference is small. We should note that Ling-all uses much more training

[14]http://www.ii.metu.edu.tr/ corpus/treebank.html

[15]Multiple parses are the cases where a suffix is attached not to a single word, but to a group of words. The suffix *-ti* in *takip etti* is attached to *takip et*.

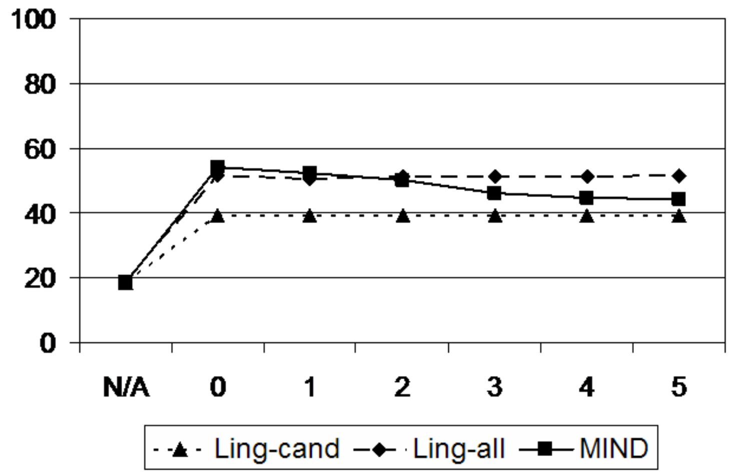


Figure 5.7: Percentage of the Correctly Segmented Words by Linguistica and MIND for Different Threshold Values

data than MIND (503 vs. 1,849 examples of word), and even with this difference the performance of MIND nears Ling-all. We believe this happens in segmentation, despite the huge difference in the number of correct affixes they found, because the affixes Ling-all finds are shorter and more frequent. In its current state, MIND does not segment composite affixes and identifies several long and less frequent affixes. Some of these long affixes can be composed by the shorter affixes Linguistica finds, contributing to the similarities in performance between Linguistica and MIND.

5.7.6 Morpheme Segmentation

We used the Morpho Challenge 2005 data in our final experiment, which provides segmented Turkish data[16]. We used the Morpho Challenge data as the gold standard in a morpheme segmentation evaluation. The test data contain the segmentation, or morpheme analyses, of 60,000 unique Turkish words. The evaluation rates the placement of morpheme boundaries, with precision, recall, and f-measure as the evaluation metrics. Hits (H) are the correctly placed morpheme boundaries; insertions (I) are the incorrect morphemes boundaries; and deletions (D) are the missed morpheme boundaries. Suppose the desired and suggested segmentations for the Turkish word *inanmıyorsan* are:

`Desired:`	*inan mıyor sa n*
`Suggested:`	*inan m ıyor san*

The suggested segmentation contains two hits (the correct boundaries after *inan* and *mıyor*), one insertion (the incorrect boundary between *m* and *ıyor*), and one deletion (the missed boundary between *sa* and *n*). Precision, recall, and f-measure calculate per Kurimo et al. (2006a):

$$\begin{aligned} Precision &= \frac{H}{(H+I)} \\ Recall &= \frac{H}{(H+D)} \\ F-measure &= \frac{2H}{(2H+I+D)} \end{aligned}$$

[16]We thank Morpho Challenge organizers, particularly Mathius Creutz, Murat Saraclar and Ebru Arisoy, for supplying us with the evaluation data

The systems participating in the Morpho Challenge 2005 trained on a corpus composed of 582,923 unique words. The Turkish dictionary for the MIND experiments contains 17,415 headword-example of usage pairs. We are aware that the systems that participated in the Morpho Challenge 2005 and MIND are different characteristics. While the systems in the Morpho Challenge 2005 employ unsupervised algorithms using corpora as input, MIND uses dictionary information. However, we would like to see how MIND, with its noisy and limited data, measures in respect to these systems.

System	**Precision (%)**	**Recall (%)**	**F-measure (%)**
Bordag (2006)	**79.9**	47.9	57.0
Bernhard (2006)	65.4	**65.2**	**65.3**
MIND	79.1	50.3	61.5
MIND - no derivations	74.4	60.0	**66.4**

Table 5.10: The Precision, Recall, and F-measure of the Best Systems in the Morpho Challenge 2005 and MIND for Turkish.

For morpheme segmentation, the same segmentation algorithm described in Section 5.7.5 applies with threshold value zero. Table 5.10 presents the precision, recall, and f-measure values of MIND, and best performances in the Morpho Challenge 2005 for Turkish. MIND scored second in precision and f-measure, and fourth in recall. The best system in precision (Bordag, 2006) scored lower recall and f-measures than MIND. The best system in recall and f-measure (Bernhard, 2006), rated lower in precision than MIND. We should note that these same systems exhibited similar performance in Finnish, but the best system in English performed very poorly in Turkish and Finnish.

The recall of MIND rates low (50.3%) because it is engineered toward inflectional morphology. When the derived words appear as headwords in the dictionary, as with the Turkish dictionary used as the training data, MIND cannot extract derivational affixes. In the gold standard, however, derivational affixes could also segment. The other systems can find derivational affixes since they use corpora as input which includes derived words. To discover how much derivational morphology effects the performance of MIND, we removed the derivational affixes from the gold standard[17], and evaluated MIND using this gold standard without derivational affixes. The last row —"MIND - no derivations"— in Table 5.10 presents the precision, recall, and f-measure values having removed the derivational morphemes from the gold standard. In this case, our f-measure by 4.increased by 4.9%, and recall elevated by 9.7%. The f-measure of MIND is 0.9% higher than the best system in Morpho Challenge 2005. Our system which uses a very small amount of dictionary data achieved a better result than the other systems that were trained on more than a half million unique words.

Another reason for the low recall, and not higher precision, involves the presence of composite suffixes present in the MIND output. Turkish is an agglutinative language, hence words can take multiple suffixes. MIND identifies one affix for each word, therefore composite suffixes, (multiple suffixes used together) do not split into the suffixes that form them. This does not decrease our ability to find a headword in a dictionary, but would decrease the ability to segment the affixes meaningfully.

[17]If a suffix is ambiguous such that it can be both a derivational and inflectional suffix, the suffix was not removed.

We also investigated the percentage of the suffixes in the gold standard also appeared in the MIND results. The gold standard contains 450 unique suffix forms. 155 of the 509 suffixes MIND extracted from the dictionary also exist in the gold standard suffix list, i.e., MIND achieved 34.4% of the gold standard suffixes. The main reason for this low percentage again falls with the composite suffixes. The MIND output also includes incorrect affixes extracted due to OCR errors in the data. We should note that MIND extracted the most common suffixes were extracted.

5.8 Summary

This chapter introduced a framework for morphology induction from noisy data — MIND, which can be especially useful for languages with limited electronic data. It is motivated by the fact that electronic dictionaries often exist for languages previous to large amounts of other electronic text, and they provide a variety of information in a structured format. MIND uses the information in dictionaries, specifically headword and the corresponding example of usage sentences, to acquire affix lists of the language, and it operates with string distance metrics. It is engineered towards inflectional morphology.

The evaluation of the MIND results on two data sets demonstrated that it successfully identifies the prefixes, suffixes, circumfixes, and infixes of two highly inflected languages. When the acquired suffix list from one data set is used in a simple word segmentation process, MIND outperformed a state-of-the-art morphology learner using the same amount of training data. Furthermore, MIND achieved nearly the same precision that the best unsupervised method in the Pascal challenge achieved with much more data. When derivational morphology removed from the test data, the f-measure for MIND was higher than the best system's f-measure.

MIND has several advantages compared to previously proposed methods. Foremost, it does not depend on expert, or native speaker knowledge, and large corpora, but uses only the limited and structured information found in dictionaries. The structure of the data enables us to extract useful information. Second, MIND can locate morpheme boundary morphophonemic rules when they are present in the data. Finally, it can co-train with low-accuracy unsupervised algorithms, offering value for languages where some electronic resources exist.

The performance of MIND depends on the dictionary used for training. If the coverage of the dictionary is poor, or if given a limited number of examples of usage with few morphological variants of the headwords, MIND will be able to acquire only a portion of the affixes.

MIND does not require human presence but it is interesting to ask whether the presence of a human could improve MIND's performance with minimal effort. There a few scenarios. In one case, a linguist knowledgeable about the low-density language in question might be available for a short amount of time. Then it may be possible to get most of the rules about the morphology of the language. Most probably, even in this scenario, we would not get a complete list of affixes if this is a morphologically rich language. A data driven approach supplements the experts knowledge with language data in a dictionary We might also assume the presence of a native speaker of the low-density language for a short amount of time. In this case, it is not realistic to assume that the non-expert user has the analysis skills to provide the morphology of the language. However, such a user may be helpful in correction and identifying the invalid affixes in MIND output, and thus improving the accuracy of MIND. Our method enables a non-linguist user to help in morphology acquisition.

Chapter 6

Conclusions

This thesis presented methods for resource generation for low-density languages, which produce linguistically useful resources with minimal data. Two methods annotate the data in the dictionary entries to represent the contents of printed bilingual dictionaries electronically. These two methods for entry tagging rely on minimal human assistance, and they are fast and generic.

The evaluations show that printed dictionaries can be converted into electronic format with a reasonable accuracy using the proposed methods. The application of transformation-based learning as a post-processor, both to font style accuracy of OCR and tagging accuracy of entry tagging, improved the performance further using very limited human-generated training data. The tagging results also discover the morphemes in the language, providing another kind of resource and increasing the usability of the dictionary by human translators.

The rest of this chapter describes the contributions of this thesis, limitations, and problems of the work presented in this thesis, and directions for future work.

6.1 Contributions

This thesis yields the following contributions:

- Demonstration of the possibility to reduce the need for large amounts of data to produce annotated, searchable resources with machine learning methods, when these methods couple with structured documents.
- Introduction and implementation of an adaptable entry tagging module that annotates dictionary entries rapidly, with minimal non-expert human input.
- Adaptation of Hidden Markov Models (HMMs) to the dictionary entry problem, using minimal training data. Training of HMMs to search for content and information type (headword, etc.) in the dictionary.
- Introduction of a flexible generation component capable of producing different formats, targeting different applications from the same annotated dictionary entries.
- Adaptation of transformation-based learning (TBL) to a new field in NLP, using limited training data.
- Introduction and implementation of a new framework for robust morpheme discovery. The framework finds morphemes and induces morphophonemic rules using limited, noisy dictionary data, and and it needs no knowledge of the language. This presents a novel use of dictionary data are used for morpheme acquisition.

- Application of string algorithms (string edit distance, longest common substring, and k-difference string matching) in morpheme and morphophonemic rule discovery.

6.2 Problems and Limitations

The methods presented in this thesis have the following limitations:

- Entry tagging module relies on the input from OCR and document image analysis. If these have unsatisfactory accuracy, the resulting resource may be poor.

- Rule-based entry tagging depends on the human provided information about the structure of the dictionary entries. If the user misses some information, this will effect the output.

- HMM-based stochastic method relies on human input in the form of the number of states and training data. If the user does not try different number of states or provide the system with the most promising one the performance will be degraded. Moreover, if the training misses some information types or not enough training data are provided for each information types, the accuracy of the output may be low.

- If the user defines new information types, these will not appear in TEI and Rosetta output formats in generation module.

- Adaptive TBL-based post-processing uses manually corrected training data.

It is important that the entries in the training data represent the errors in the initial tagging.

- The performance of the MIND framework is affected by the coverage of the chosen dictionary. If it does not contain many examples of usages, or if not many morphological variants of the headwords appear in the examples of usages, the system's ability to discover morphemes will degrade.

6.3 Future Directions

This section describes possible research directions for future work, which can improve and extend this work.

- English is the most widely studied language in computational linguistics. Its rich resources, either in the form of corpora or different annotated data or techniques for parsing, etc., may prove useful to increase the accuracy of entry tagging, enrich the resulting resource, and discovering derivational affixes in morpheme discovery.

- Using multiple dictionaries may benefit both in lexicon generation and morpheme discovery.

- MIND framework discovers many composite affixes as discussed in Chapter 5. Splitting these composite affixes into their basic morphemes will most probably increase the performance (especially recall) in word segmentation and will provide a better representation of the language.

- When available, incorporating corpora with information given in the dictionaries may be useful during morpheme discovery.

Chapter A

Pre-defined Information Types in BRIDGE

This chapter presents the pre-defined information types, along with what they represent, in BRIDGE Software.

- **Headword:** The main word that defines the entry.
- **Translation:** The translation of the headword.
- **Pronunciation:** The representation of the way a word is spoken, using phonetic symbols.
- **POS:** Part-of-speech, a classification of words according to their functions in a context.
- **Domain:** The region in which a word or translation occurs.
- **Gender:** Grammatical gender (masculine, feminine, or neutral).
- **Gender word:** The form of the headword representing another gender formed irregularly.
- **Number:** Grammatical number (singular or plural).

- **Number word:** The form of the headword representing irregular plural forms.
- **Context:** The framework or perspective in which a word or translation appears.
- **Etymology:** The original language of a word.
- **Alternative Spelling:** Another acceptable spelling of the word.
- **Compound/Derived:** The word lexically related to the headword, such as an adjectival form of a verb entry.
- **Compound/Derived Translation:** The translation of the compound/derived word.
- **Example:** A phrase or a sentence showing the use of the word.
- **Example Translation:** The translation of the example.
- **Idiom:** An idiom including the word.
- **Idiom Translation:** The translation of the idiom.
- **Cross Reference:** A reference made to another entry in the dictionary.
- **Antonym:** The antonym of the word.
- **Synonym:** The synonym of the word.
- **Tense:** The grammatical tense (past, present, future, etc.) associated with a given inflected form.

- **Tense word:** The form of the headword representing an irregularly formed tense.
- **Person:** The grammatical person (1st, 2nd, 3rd, etc.) associated with a given inflected form.
- **Mood:** Information about the grammatical mood of verbs, such as indicative, subjunctive, or imperative.
- **Explanation:** Additional information provided usually in parentheses.
- **Case:** A form of a word indicating its relation to other words.
- **Style:** The way in which the word is actually used.
- **Subcategorization:** A classification of words, such as syntactic patterns of verbs, count/mass distinctions of nouns.
- **Sense Number**: The numerals indicate different POS or different translations of a word.
- **Subject:** A branch of knowledge in which the word is used or that the word indicates.
- **Collocation:** A sequence of words which co-occur with headword more often than would be expected by chance.
- **Abbreviation:** A shortened form of a word or phrase used mainly in writing to represent the complete form.

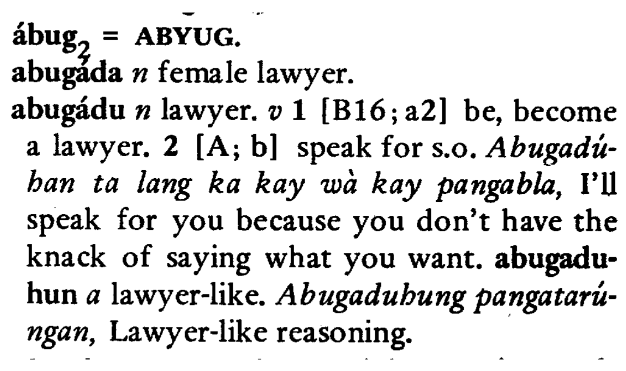
- **Miscellaneous:** Reserved for the words that the tagging process could not identify.

- **Separator:** Shows punctuation when the dictionary entries are tokenized.

Chapter B

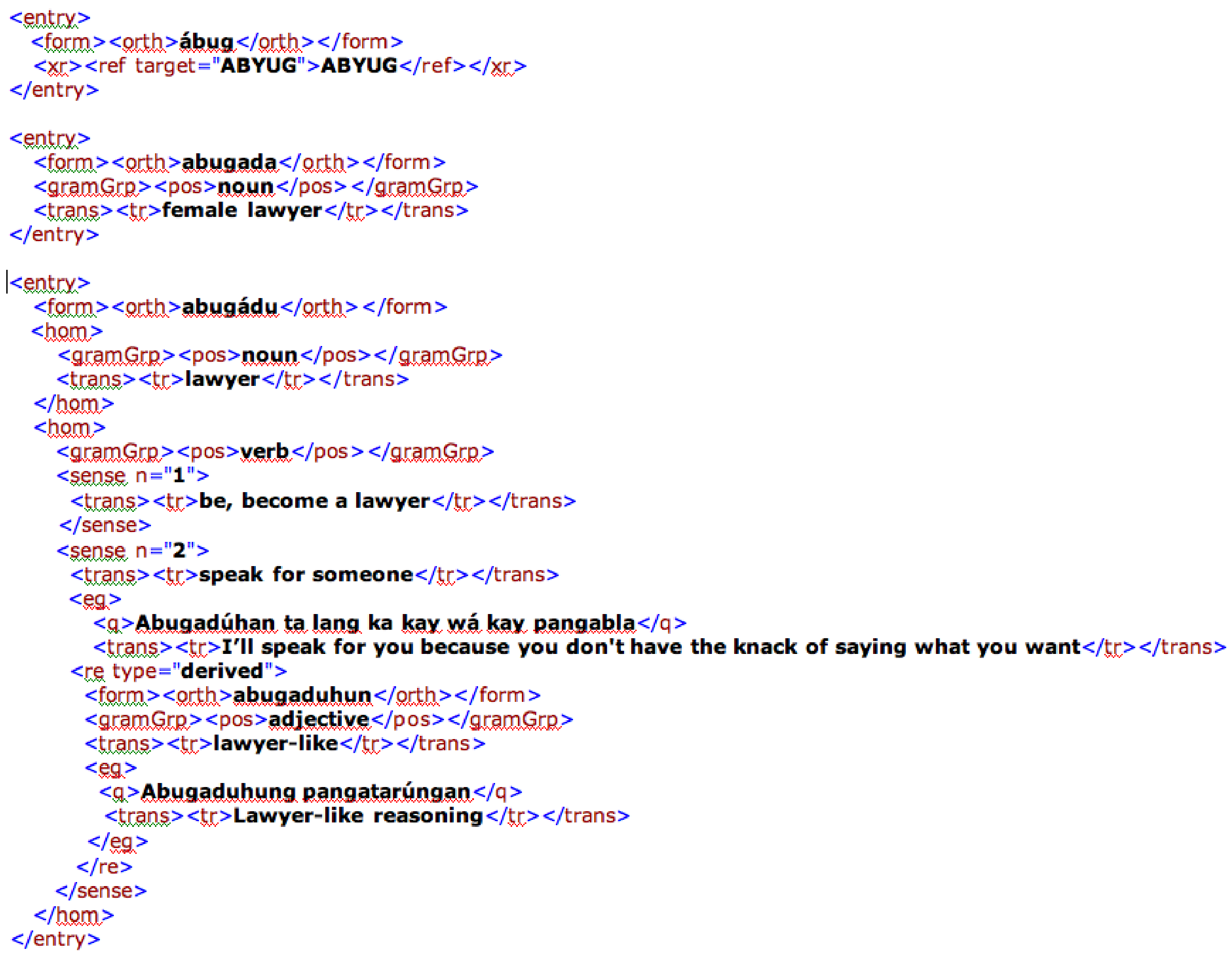
Generation Formats

Figures B.2 to B.7 show outputs for TEI, Rosetta, lexicon-translation pairs, example of usage-translation pairs, HTML, and color-coded HTML with entry images for the dictionary entries in Figure B.1.



ábug$_2$ = **ABYUG.**
abugáda *n* female lawyer.
abugádu *n* lawyer. *v* **1** [B16; a2] be, become a lawyer. **2** [A; b] speak for s.o. *Abugadúhan ta lang ka kay wà kay pangabla,* I'll speak for you because you don't have the knack of saying what you want. **abugaduhun** *a* lawyer-like. *Abugaduhung pangatarúngan,* Lawyer-like reasoning.

Figure B.1: Sample Entries from a Cebuano-English Dictionary

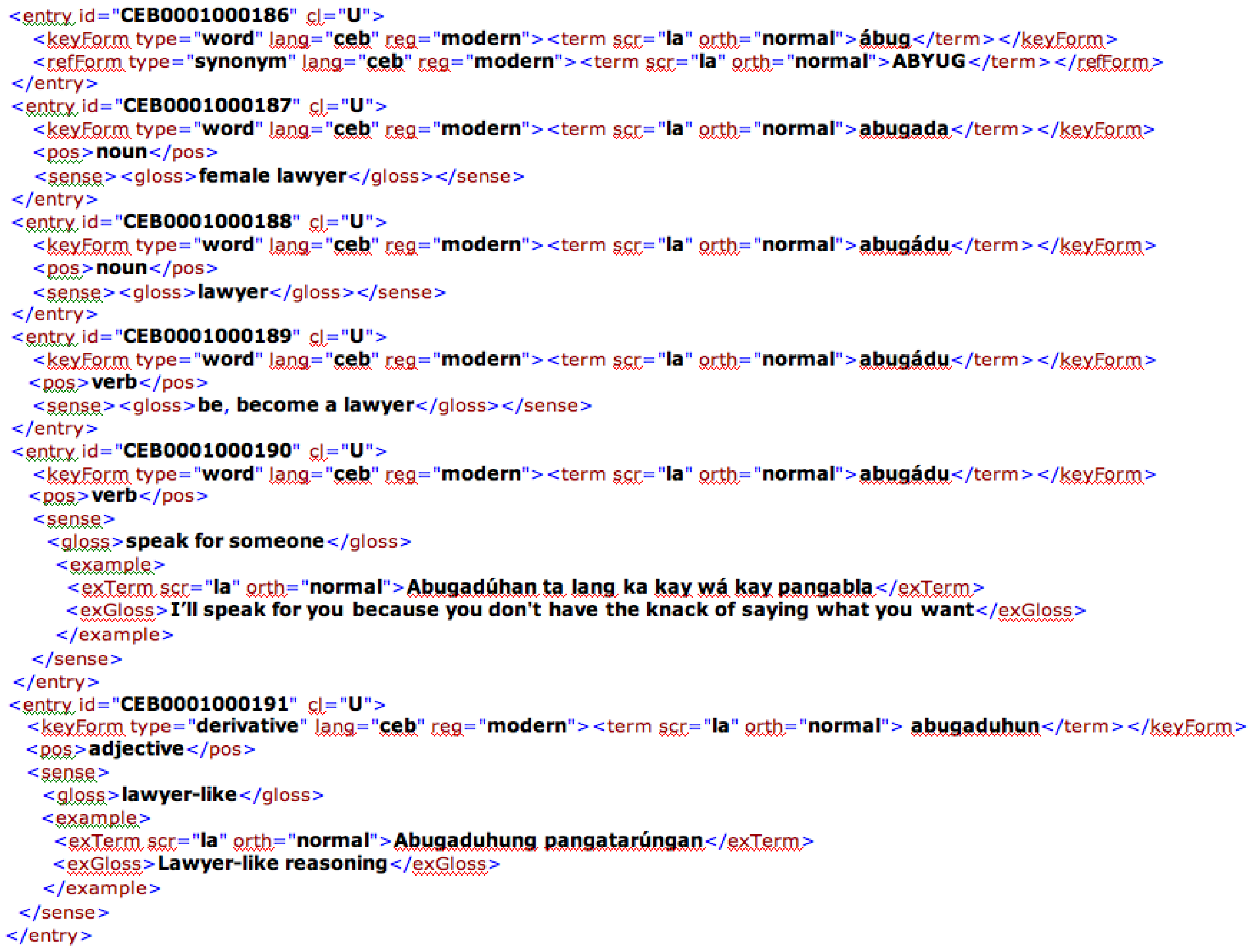


```
<entry>
  <form><orth>ábug</orth></form>
  <xr><ref target="ABYUG">ABYUG</ref></xr>
</entry>

<entry>
  <form><orth>abugada</orth></form>
  <gramGrp><pos>noun</pos></gramGrp>
  <trans><tr>female lawyer</tr></trans>
</entry>

<entry>
  <form><orth>abugádu</orth></form>
  <hom>
    <gramGrp><pos>noun</pos></gramGrp>
    <trans><tr>lawyer</tr></trans>
  </hom>
  <hom>
    <gramGrp><pos>verb</pos></gramGrp>
    <sense n="1">
      <trans><tr>be, become a lawyer</tr></trans>
    </sense>
    <sense n="2">
      <trans><tr>speak for someone</tr></trans>
      <eg>
        <q>Abugadúhan ta lang ka kay wá kay pangabla</q>
        <trans><tr>I'll speak for you because you don't have the knack of saying what you want</tr></trans>
      <re type="derived">
        <form><orth>abugaduhun</orth></form>
        <gramGrp><pos>adjective</pos></gramGrp>
        <trans><tr>lawyer-like</tr></trans>
        <eg>
          <q>Abugaduhung pangatarúngan</q>
          <trans><tr>Lawyer-like reasoning</tr></trans>
        </eg>
      </re>
    </sense>
  </hom>
</entry>
```

Figure B.2: The TEI

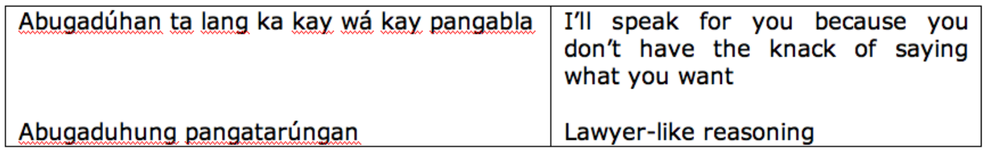


```xml
<entry id="CEB0001000186" cl="U">
  <keyForm type="word" lang="ceb" reg="modern"><term scr="la" orth="normal">ábug</term></keyForm>
  <refForm type="synonym" lang="ceb" reg="modern"><term scr="la" orth="normal">ABYUG</term></refForm>
</entry>
<entry id="CEB0001000187" cl="U">
  <keyForm type="word" lang="ceb" reg="modern"><term scr="la" orth="normal">abugada</term></keyForm>
  <pos>noun</pos>
  <sense><gloss>female lawyer</gloss></sense>
</entry>
<entry id="CEB0001000188" cl="U">
  <keyForm type="word" lang="ceb" reg="modern"><term scr="la" orth="normal">abugádu</term></keyForm>
  <pos>noun</pos>
  <sense><gloss>lawyer</gloss></sense>
</entry>
<entry id="CEB0001000189" cl="U">
  <keyForm type="word" lang="ceb" reg="modern"><term scr="la" orth="normal">abugádu</term></keyForm>
 <pos>verb</pos>
  <sense><gloss>be, become a lawyer</gloss></sense>
</entry>
<entry id="CEB0001000190" cl="U">
  <keyForm type="word" lang="ceb" reg="modern"><term scr="la" orth="normal">abugádu</term></keyForm>
 <pos>verb</pos>
  <sense>
    <gloss>speak for someone</gloss>
     <example>
       <exTerm scr="la" orth="normal">Abugadúhan ta lang ka kay wá kay pangabla</exTerm>
       <exGloss>I'll speak for you because you don't have the knack of saying what you want</exGloss>
     </example>
  </sense>
</entry>
<entry id="CEB0001000191" cl="U">
  <keyForm type="derivative" lang="ceb" reg="modern"><term scr="la" orth="normal"> abugaduhun</term></keyForm>
  <pos>adjective</pos>
  <sense>
    <gloss>lawyer-like</gloss>
    <example>
      <exTerm scr="la" orth="normal">Abugaduhung pangatarúngan</exTerm>
      <exGloss>Lawyer-like reasoning</exGloss>
    </example>
  </sense>
</entry>
```

Figure B.3: The Rosetta Output

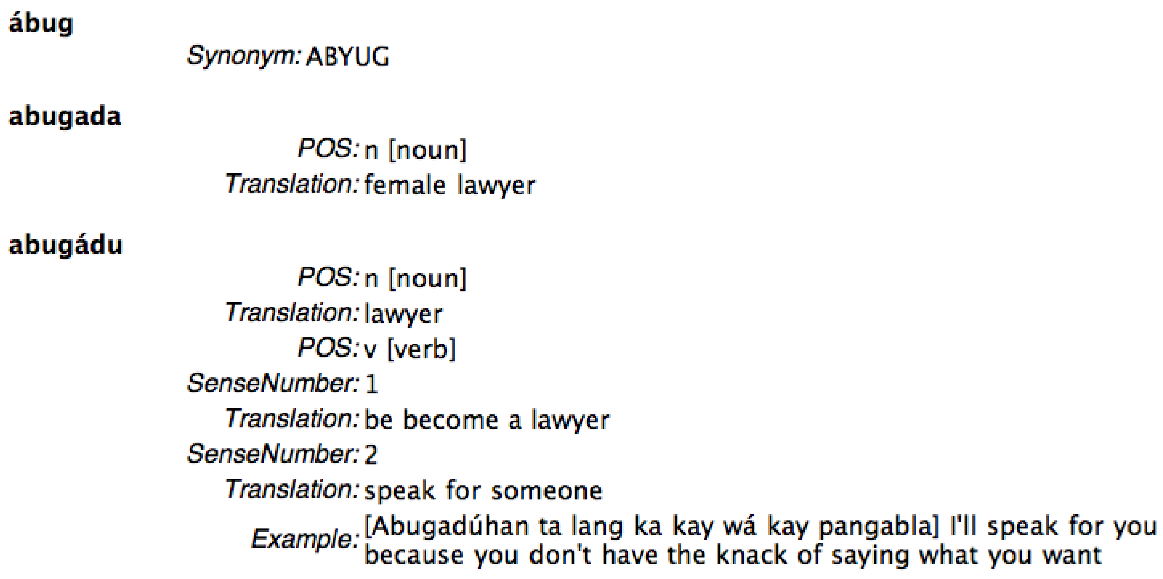
Abugada	female lawyer
abugádu	lawyer
abugádu	be, become a lawyer
abugádu	speak for someone
abugaduhun	lawyer-like

Figure B.4: The Lexicon - Translation Pairs



Abugadúhan ta lang ka kay wá kay pangabla	I'll speak for you because you don't have the knack of saying what you want
Abugaduhung pangatarúngan	Lawyer-like reasoning

Figure B.5: The Example of Usage - Example of Usage Translation Pairs



ábug

Synonym: ABYUG

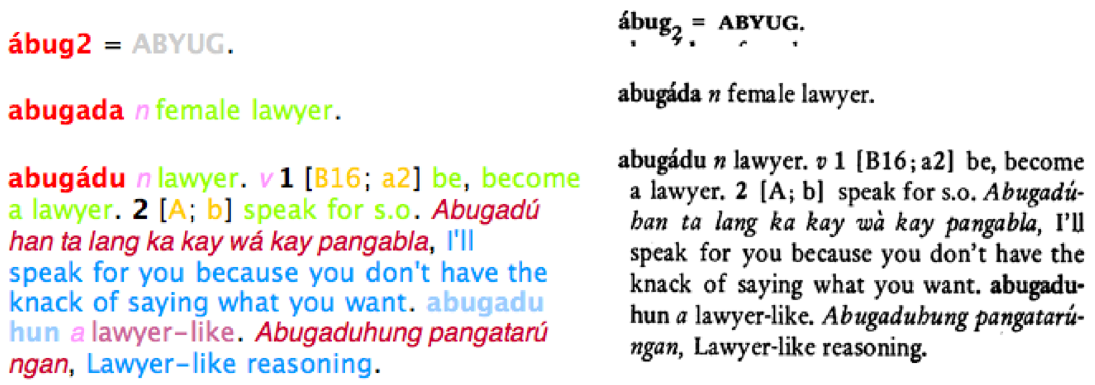
abugada

POS: n [noun]
Translation: female lawyer

abugádu

POS: n [noun]
Translation: lawyer
POS: v [verb]
SenseNumber: 1
Translation: be become a lawyer
SenseNumber: 2
Translation: speak for someone
Example: [Abugadúhan ta lang ka kay wá kay pangabla] I'll speak for you because you don't have the knack of saying what you want

Figure B.6: The HTML Output



ábug2 = ABYUG.

abugada *n* female lawyer.

abugádu *n* lawyer. *v* **1** [B16; a2] be, become a lawyer. **2** [A; b] speak for s.o. *Abugadú han ta lang ka kay wá kay pangabla*, I'll speak for you because you don't have the knack of saying what you want. **abugadu hun** *a* lawyer-like. *Abugaduhung pangatarú ngan*, Lawyer-like reasoning.

ábug$_2$ = **ABYUG.**

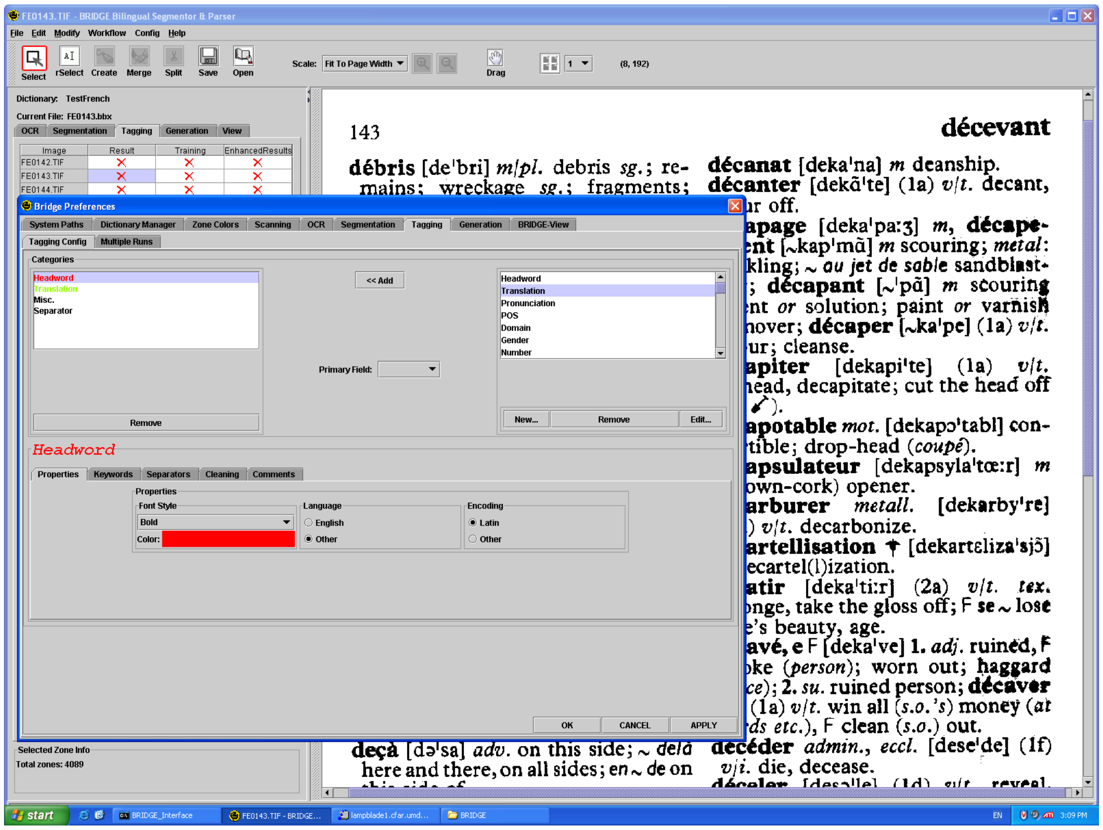
abugáda *n* female lawyer.

abugádu *n* lawyer. *v* **1** [B16; a2] be, become a lawyer. **2** [A; b] speak for s.o. *Abugadú-han ta lang ka kay wà kay pangabla,* I'll speak for you because you don't have the knack of saying what you want. **abugadu-hun** *a* lawyer-like. *Abugaduhung pangatarú-ngan,* Lawyer-like reasoning.

Figure B.7: The Color-coded HTML Output with Entry Images

Chapter C

BRIDGE Screenshots


FE0143.TIF - BRIDGE Bilingual Segmentor & Parser

Dictionary: TestFrench

Current File: FE0143.bbx

Image	Result	Training	EnhancedResults
FE0142.TIF	✗	✗	✗
FE0143.TIF	✗	✗	✗
FE0144.TIF	✗	✗	✗

Bridge Preferences — Tagging — Tagging Config

Categories: Headword, Translation, Misc., Separator

Fields: Headword, Translation, Pronunciation, POS, Domain, Gender, Number

Primary Field:

Headword — Properties

Font Style: Bold

Color: 🟥

Language: ( ) English [✓ Other]

Encoding: [✓ Latin] ( ) Other

Selected Zone Info

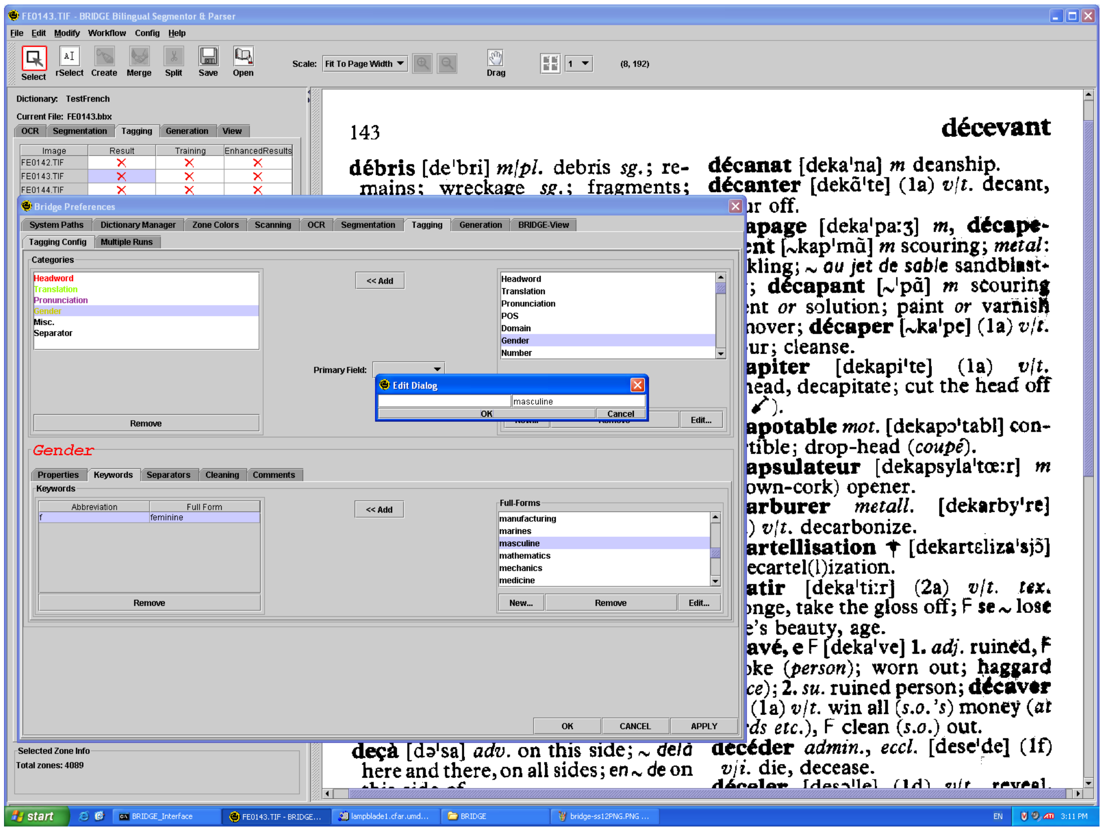
Total zones: 4089

Figure C.1: The Configuration for Entry Tagging in BRIDGE. Properties of an Information Type is Configured.

BRIDGE provides a user-friendly environment for the user to relay the necessary information required for entry tagging. The user first selects the information types found in the dictionary. For each information type, the user also selects the font style, whether the data given in this information type is English [1], or Latin. In Figure C.1, the pre-defined list of information types appear on right-hand side of the upper panel, and the selected information types display on the left-hand side of the upper panel. The user can define new information types using the *New* button. The properties of the selected information type (i.e. *Headword* in this case) are shown in the lower panel. The user has selected the *Headword* as given in *bold* font style in the dictionary, non-English, and Latin. *Red* will be used for visualization purposes in BRIDGE. Each word tagged as headword on the dictionary pages will be surrounded by a red box.

Figure C.2 shows how the keywords are provided to the BRIDGE system. A pre-defined list of the most frequently used information appears in abbreviations in dictionaries on the right-hand side of the lower panel. When the user selects one, the system requires the abbreviation used for the selected information. In the figure, the selected information type is *gender*, and the user has already entered that *f* as an abbreviation for *feminine*, followed by the abbreviation for *masculine*.

[1]Right now English or non-English information is not used, it is preserved for future extensions.



FE0143.TIF - BRIDGE Bilingual Segmentor & Parser

Dictionary: TestFrench

Current File: FE0143.bbx

Image	Result	Training	EnhancedResults
FE0142.TIF	✗	✗	✗
FE0143.TIF	✗	✗	✗
FE0144.TIF	✗	✗	✗

Bridge Preferences — Tagging — Tagging Config

Categories: Headword, Translation, Pronunciation, Gender, Misc., Separator

Available: Headword, Translation, Pronunciation, POS, Domain, Gender, Number

Edit Dialog: masculine

Gender — Keywords

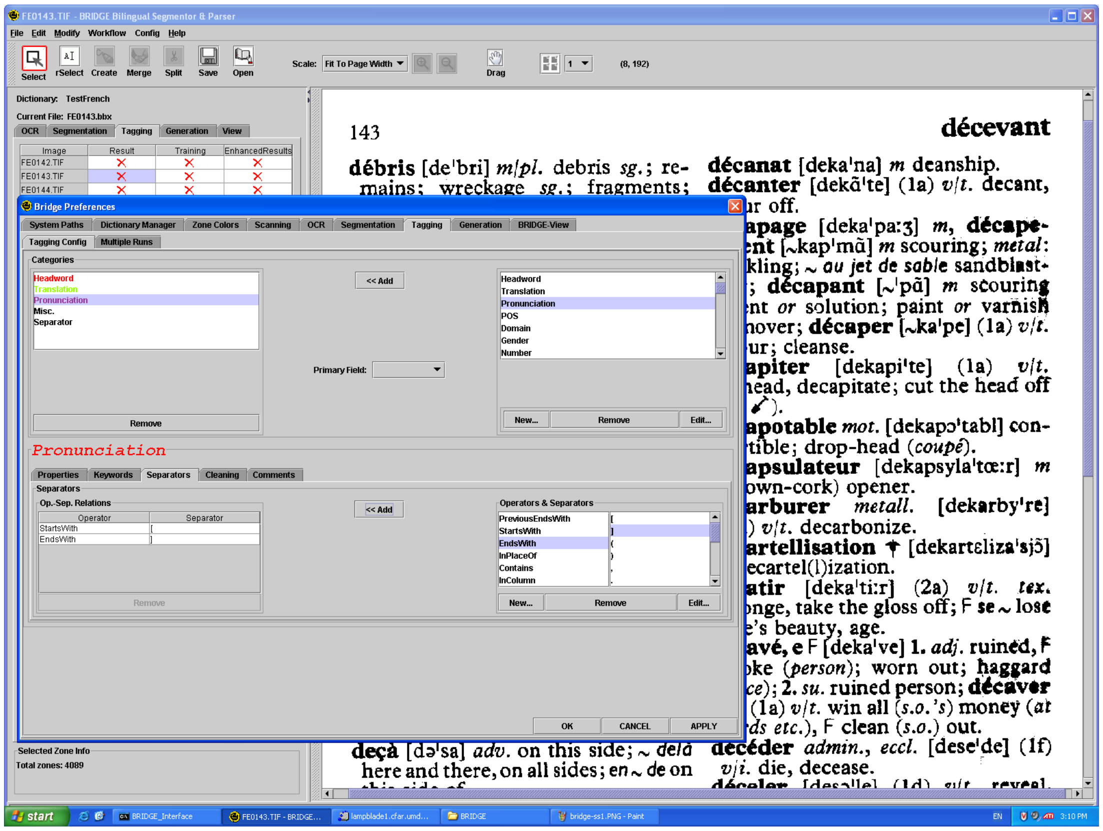
Abbreviation	Full Form
f	feminine

Full-Forms: manufacturing, marines, masculine, mathematics, mechanics, medicine

Selected Zone Info — Total zones: 4089

Figure C.2: The Configuration for Entry Tagging in BRIDGE. The Keywords Associated with an Information Type is Entered.

Figure C.3 shows how the separators communicate to the system in BRIDGE. In the right-hand side the lower panel, a list of available operands and the most frequently used separators in dictionaries exists. Users can define new ones as necessary. The left-hand side of the lower panel shows the separators provided for the current dictionary. In the figure, the user entered that the information type *pronunciation StartsWith* the punctuation *[*, and it *EndsWith* the punctuation *]*.


FE0143.TIF - BRIDGE Bilingual Segmentor & Parser
File Edit Modify Workflow Config Help
Select rSelect Create Merge Split Save Open
Scale: Fit To Page Width
Drag
1
(8, 192)
OCR Segmentation Tagging Generation View
Bridge Preferences
System Paths Dictionary Manager Zone Colors Scanning OCR Segmentation Tagging Generation BRIDGE-View
Tagging Config Multiple Runs
<< Add
Remove
New... Remove Edit...
Properties Keywords Separators Cleaning Comments
<< Add
Remove
New... Remove Edit...
OK CANCEL APPLY
start BRIDGE_Interface FE0143.TIF - BRIDGE... lampblade1.cfar.umd... BRIDGE bridge-ss1.PNG - Paint
EN 3:10 PM

Dictionary: TestFrench

Current File: FE0143.bbx

Image	Result	Training	EnhancedResults
FE0142.TIF	✗	✗	✗
FE0143.TIF	✗	✗	✗
FE0144.TIF	✗	✗	✗

Categories

- Headword
- Translation
- Pronunciation
- Misc.
- Separator

- Headword
- Translation
- Pronunciation
- POS
- Domain
- Gender
- Number

Primary Field:

Pronunciation

Separators

Op.-Sep. Relations

Operator	Separator
StartsWith	[
EndsWith	]

Operators & Separators

PreviousEndsWith	[
StartsWith	]
EndsWith	(
InPlaceOf	)
Contains	,
InColumn	.

Selected Zone Info

Total zones: 4089

143 décevant

débris [de'bri] *m/pl.* debris *sg.*; re-

décanat [deka'na] *m* deanship.

décanter [dekɑ̃'te] (1a) *v/t.* decant,

deçà [də'sa] *adv.* on this side; ~ *delà* here and there, on all sides; en ~ de on

Figure C.3: The Configuration for Entry Tagging in BRIDGE. The Separators Associated with an Information Type is Entered.

Chapter D

TBL Transformations on Cebuano-English

This chapter lists the first 20 transformation rules for font style and tags that were discovered by TBL on Cebuano-English dictionary.

Font style

1. $font_n$ = italic and $type_{n-1}$ = lowercase and $font_{n-2}$ = normal $\Rightarrow$ $font$ = normal
2. $type_{n-1}$ = punctuation and $type_n$ = uppercase $\Rightarrow$ $font$ = normal
3. $type_{n-1}$ = lowercase and $font_{n-1}$ = normal and $font_{n-1}$ = normal $\Rightarrow$ $font$ = normal
4. $token_n$ = n and $type_{n-1}$ = lowercase and $font_{n-2}$ = normal $\Rightarrow$ $font$ = italic
5. $token_{n-1}$ = . and $type_n = 3$ $\Rightarrow$ $font$ = bold
6. $font_{n-1}$ = bold and $type_n = 4$ $\Rightarrow$ $font$ = bold
7. $token_{n-1}$ = v and $type_n = 4$ and $font_{n-1}$ = normal $\Rightarrow$ $font$ = italic
8. $font_{n-2}$ = normal and $type_{n-1}$ = lowercase and $type_n$ = lowercase and $token_{n-1}$ = . $\Rightarrow$ $font$ = normal

9. $font_{n-1}$ = normal and $type_n = 4 \Rightarrow font$ = normal

10. $token_{n-1}$ = [and $type_{n-2}$ = lowercase $\Rightarrow font$ = normal

11. $type_{n-2}$ = lowercase and $type_{n-1} = 4$ and $type_n$ = capitalized and $font_{n-1}$ = normal and $font_{n-2}$ = normal $\Rightarrow font$ = normal

12. $type_n$ = symbol and $type_{n-1}$ = lowercase $\Rightarrow font$ = bold

13. $font_{n-1}$ = normal and $font_{n-1}$ = italic and $type_{n-1} = 4$ and $type_n$ = capitalized and $type_{n-1}$ = lowercase $\Rightarrow font$ = italic

14. $token_{n-2}$ = ; and $type_{n-1}$ = lowercase and $token_n$ =] $\Rightarrow font$ = normal

15. $font_{n-1}$ = bold and $token_n$ = a and $font_{n-1}$ = normal and $font_{n-2}$ = normal $\Rightarrow font$ = italic

16. $font_{n-1}$ = italic and $type_n$ = lowercase and $type_{n-1}$ = lowercase and $font_{n-2}$ = italic $\Rightarrow font$ = italic

17. $type_{n-2}$ = lowercase and $type_{n-1}$ = lowercase and $type_n$ = lowercase and $font_{n-1}$ = normal $\Rightarrow font$ = normal

18. $type_{n-1}$ = numeric and $type_{n-1}$ = lowercase and $font_{n-2}$ = normal $\Rightarrow font$ = normal

19. $type_n$ = lowercase and this is the first token in the entry $\Rightarrow font$ = bold

20. $font_{n-2}$ = italic and $font_{n-1}$ = normal and $type_n$ = lowercase and $font_{n-1}$ = normal $\Rightarrow font$ = normal

Tagging (After TBL application to font style)

1. $token_n$ = ; and $font_{n-1}$ = normal $\Rightarrow$ tag = separator

2. tag_{n-1} = translation and tag_n = translation and $font_{n-2}$ = normal and $font_{n-1}$ = normal and $font_n$ = 0 and $type_{n-1}$ = lowercase and $type_n$ = lowercase $\Rightarrow$ continuation of a phrase

3. tag_{n-1} = subcategorization and tag_{n-2} = separator and tag_n = separator and $font_n$ = normal $\Rightarrow$ beginning of a phrase

4. tag_{n-2} = example translation and tag_{n-1} = example translation and tag_n = example translation and $font_{n-2}$ = normal and $font_{n-1}$ = normal and $font_n$ = normal and $type_{n-1}$ = lowercase and $type_n$ = lowercase $\Rightarrow$ continuation of a phrase

5. tag_{n-2} = translation and tag_{n-1} = separator and tag_n = translation and $font_{n-2}$ = normal and $font_{n-1}$ = normal and $font_n$ = normal and $type_{n-2}$ = lowercase and $type_{n-1}$ = punctuation and $type_n$ = lowercase $\Rightarrow$ continuation of a phrase

6. $type_n$ = lowercase and $font_n$ = normal and tag_{n-1} = subcategorization and $font_{n-1}$ = normal $\Rightarrow$ tag = example translation

7. $token_n$ = a and $font_{n-1}$ = normal and tag_{n-1} = translation $\Rightarrow$ tag = translation

8. $type_n$ = lowercase and $font_n$ = italic and tag_{n-1} = subcategorization and $font_{n-1}$ = italic $\Rightarrow$ tag = example

9. tag_{n-1} = separator and tag_{n-2} = example translation and tag_n = example translation and $type_{n-1}$ = punctuation and $font_n$ = normal and $font_{n-2}$ = normal $\Rightarrow$ continuation of a phrase

10. tag_{n-1} = derived translation and tag_n = derived translation and $font_{n-2}$ = normal and $font_{n-1}$ = normal and $font_n$ = normal and $type_{n-1}$ = lowercase and $type_n$ = lowercase $\Rightarrow$ continuation of a phrase

11. $type_n$ = lowercase and $font_n$ = normal and tag_{n-1} = example translation and $font_{n-1}$ = normal $\Rightarrow$ tag = example translation

12. tag_{n-1} = separator and tag_n = example and $font_{n-2}$ = italic and $font_{n-1}$ = italic and $font_n$ = italic and $type_{n-1}$ = punctuation and $type_n$ = lowercase $\Rightarrow$ continuation of a phrase

13. tag_{n-2} = example translation and tag_{n-1} = example translation and $font_n$ = normal and $type_n$ = lowercase and $font_{n-1}$ = normal $\Rightarrow$ tag = example translation

14. tag_{n-2} = derived translation and tag_{n-1} = separator and tag_n = derived translation and $font_{n-2}$ = normal and $font_{n-1}$ = normal and $font_n$ = normal and $type_{n-2}$ = lowercase and $type_{n-1}$ = punctuation and $type_n$ = lowercase $\Rightarrow$ continuation of a phrase

15. tag_{n-2} = example translation and tag_{n-1} = example translation and $font_n$ = normal and $type_n$ = lowercase and $font_{n-1}$ = normal $\Rightarrow$ tag = example translation

16. $type_n$ = lowercase and $font_n$ = normal and tag_{n-1} = derived translation and $font_{n-1}$ = normal $\Rightarrow$ tag = derived translation

17. tag_{n-2} = example translation and tag_{n-1} = example translation and $font_n$ = normal and $type_n$ = lowercase and $font_{n-1}$ = normal $\Rightarrow$ tag = example translation

18. tag_{n-2} = example translation and tag_{n-1} = example translation and $font_n$ = normal and $type_n$ = lowercase and $font_{n-1}$ = normal $\Rightarrow$ tag = example translation

19. tag_{n-2} = example translation and tag_{n-1} = example translation and $font_n$ = normal and $type_n$ = lowercase and $font_{n-1}$ = normal $\Rightarrow$ tag = example translation

20. tag_{n-2} = example and tag_{n-1} = separator and tag_n = example and $font_{n-2}$ = italic and $font_{n-1}$ = italic and $font_n$ = italic and $type_{n-1}$ = punctuation and $type_n$ = capitalized $\Rightarrow$ continuation of a phrase

Chapter E

TBL Transformations on Iraqi Arabic-English

This chapter lists the first 20 transformation rules for font style and tags that were discovered by TBL on Iraqi Arabic-English dictionary.

Font style

1. $type_n$ = numeric and $token_{n-1}$ = . $\Rightarrow$ $font$ = bold

2. $token_{n-1}$ = (and $type_n$ = lowercase and $font_{n+1}$ = italic and $token_{n+2}$ =) $\Rightarrow$ $font$ = italic

3. $token_{n-2}$ = see and $type_{n-1}$ = lowercase and $type_n$ = lowercase and $token_{n+1}$ = . and $token_{n+2}$ does not exist $\Rightarrow$ $font$ = italic

Tagging (After TBL application to font style)

1. tag_{n-1} = separator and tag_{n-2} = example translation and tag_n = example translation and $font_n$ = normal and $font_{n-2}$ = normal $\Rightarrow$ continuation of a phrase

2. tag_{n-1} = separator and tag_n = example and $font_{n-2}$ = italic and $font_{n-1}$ = italic and $font_n$ = italic and $type_{n-1}$ = punctuation and $type_n$ = lowercase $\Rightarrow$

continuation of a phrase

3. tag_{n-1} = headword and tag_{n-2} does not exist and tag_n = separator and $type_{n-1}$ = lowercase $\Rightarrow$ beginning of a phrase

4. $token_n = \|$ and $font_{n-1}$ = normal and tag_{n-1} = separator $\Rightarrow$ tag = separator

5. tag_{n-2} = headword and tag_{n-1} = separator and $font_n$ = italic and $font_{n-1}$ = italic and $font_{n-2}$ = italic $\Rightarrow$ tag = Alternative Spelling

6. tag_{n-2} = sense number and tag_{n-1} = separator and $font_n$ = normal and $type_n$ = lowercase and $font_{n-1}$ = normal $\Rightarrow$ tag = translation

7. $tag_{[n-7..n-1]}$ = headword and $token_{n-1}$ = see and $font_n$ = normal $\Rightarrow$ tag = separator

8. $token_n = :$ and $font_{n-1}$ = italic and tag_{n-1} = headword $\Rightarrow$ tag = separator

9. tag_{n-2} = translation and tag_{n-1} = translation and $font_n$ = italic and $type_n$ = lowercase and $font_{n-1}$ = italic $\Rightarrow$ tag = idiom

10. tag_{n-2} = separator and tag_{n-1} = number form and $font_n$ = normal and $type_n$ = lowercase and $font_{n-1}$ = normal $\Rightarrow$ tag = translation

11. tag_{n-2} = example translation and tag_{n-1} = separator and $font_n$ = normal and $type_n$ = lowercase and $font_{n-1}$ = normal and $type_{n-1}$ = punctuation $\Rightarrow$ tag = translation

12. tag_{n-2} = separator and tag_{n-1} = translation and $font_n$ = normal and $type_n$ = lowercase and $font_{n-1}$ = normal $\Rightarrow$ tag = translation

13. tag_{n-2} = translation and $token_{n-1}$ = , and tag_{n-1} = separator and $font_n$ = normal $\Rightarrow$ tag = translation

14. $type_n$ = lowercase and $font_n$ = normal and tag_{n-1} = translation and $font_{n-1}$ = normal $\Rightarrow$ tag = translation

15. $token_n$ = see and $font_{n-1}$ = italic $\Rightarrow$ tag = separator

16. tag_{n-2} = headword and tag_{n-1} = separator and $font_n$ = normal and $type_n$ = lowercase and $token_{n_1}$ = . $\Rightarrow$ tag = case

17. tag_{n-2} = number and tag_{n-1} = separator and $font_n$ = italic and $type_n$ = lowercase and $font_{n_1}$ = italic and $type_{n_1}$ = punctuation $\Rightarrow$ tag = number form

18. tag_{n-2} = separator and tag_{n-1} = Alternative Spelling and $font_n$ = normal and $type_n$ = lowercase and $font_{n_1}$ = normal and $type_{n_1}$ = lowercase $\Rightarrow$ tag = translation

19. $type_n$ = lowercase and $font_n$ = normal and tag_{n-1} = translation and $font_{n-1}$ = normal $\Rightarrow$ tag = translation

20. tag_{n-2} = translation and tag_{n-1} = translation and $font_n$ = normal and $type_n$ = lowercase and $font_{n-1}$ = normal $\Rightarrow$ tag = translation

BIBLIOGRAPHY

Eileen G. Abels, Miriam L. Matteson, Daniel W. Denman, and Amy Weinberg. Bridge combined user and usability final report, 2005.

E. Agirre, X. Arregi, X. Artola, A. Díaz de Ilarraza, K. Sarasola, and F. Evrard. IDHS, MLDS: Towards dictionary help systems for human users. In K. Korta and J. M. Larrazabal, editors, *Semantics And Pragmatics Of Natural Language: Logical And Computational Aspects*, number 1 in Ilcli Series. Donostia, 1995.

T. E. Ahlswede, M. Evens, K. Rossi, and J. Markowitz. Building a lexical database by parsing Webster's Seventh Collegiate Dictionary. In *Advances in Lexicography: Proceedings of the Second Annual Conference of the UW Centre for the New OED*, pages 65–78, Waterloo, Canada, 1986.

Hashan Al-Ajmi. Which microstructural features of bilingual dictionaries affect users' look-up performance. *International Journal of Lexicography*, 15(2):119–131, 2002.

Yaser Al-Onaizan, Ulrich Germann, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Daniel Marcu, and Kenji Yamada. Translating with scarce resources. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI 2000)*, pages 672–678, Austin, TX, August 2000.

Vuthichai Ampornaramveth. Saikam: An on-line dictionary development project. In *Proceedings of the Fourth International Workshop on Academic Information Networks and Systems*, February 1998. http://saikam.nii.ac.jp.

Vuthichai Ampornaramveth and Akiko Aizawa. Saikam: Collaborative Japanese-Thai dictionary development on the internet. In *The 2001 Asian Association for Lexicography Biennial Conference*, Korea, August 2001.

Robert A. Amsler. Computational lexicology: A research program. In *AFIPS Conference Proceedings, 1982 National Computer Conference*, pages 657–663, 1982.

S. Argamon, N. Akiva, A. Amir, , and O. Kapah. Efficient unsupervised recursive word segmentation using minimum description length. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, Geneva, Switzerland, 2004.

Susan Armstrong-Warwick. Automated lexical resources in europe: A survey. In Donald E. Walker, Antonio Zampolli, and Nicoletta Calzolari, editors, *Automating the Lexicon: Research and Practice in a Multilingual Environment*, pages 357–403. Oxford University Press, 1995.

Nart B. Atalay, Kemal Oflazer, and Bilge Say. The annotation process in the Turkish treebank. In *Proceedings of the EACL Workshop on Linguistically Interpreted Corpora*, Budapest, Hungary, April 2003.

Robert Avery, Serap Bezmez, Anna G. Edmonds, and Mehlika Yaylalı. *Redhouse İngilizce-Türkçe Sözlük.* Redhouse Yayınevi, 1974.

Necip Fazil Ayan, Bonnie J. Dorr, and Christof Monz. Alignment link projection using transformation-based learning. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language*

Processing, pages 185–192, Vancouver, BC, Canada, October 2005. Association for Computational Linguistics.

L. R. Bahl, F. Jelinek, and R. L. Mercer. A maximum likelihood approach to continuous speech recognition. In *IEEE Transactions in Pattern Anal. Machine Intelligence*, volume PAMI-5, pages 179–190, 1983.

Timothy Baldwin, Steven Bird, and Baden Hughes. Collecting low-density language materials on the web. In *Proceedings of the 12th Australasian World Wide Web Conference (AusWeb06)*, Noosa Lakes, Australia, 2006.

Marco Baroni, Johannes Matiasek, and Harald Trost. Unsupervised discovery of morphologically related words based on orthographic and semantic similarity. In *Proceedings of the Sixth Meeting of the ACL Special Interest Group in Computational Phonology (SIGPHON 2002)*, Philadelphia, PA, 2002.

Andrew Barron, Jorma Rissanen, and Bin Yu. The minimum description length principle in coding and modeling. *IEEE Transactions on Information Theory*, 44 (6):2743–2760, 1998.

L. E. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. In *Inequalities III: Proceedings of the Third Symposium on Inequalities*, pages 1–8, University of California, Los Angeles, 1972. Academic Press.

Emily M. Bender, Dan Flickinger, Jeff Good, and Ivan A. Sag. Montage: Leveraging advances in grammar engineering, linguistic ontologies, and mark-up for the documentation of underdescribed languages. In *Proceedings of the Workshop on First Steps for Language Documentation of Minority Languages: Computational Linguistic Tools for Morphology, Lexicon and Corpus Compilation (LREC 2004)*, Lisbon, Portugal, 2004.

Vincent Berment. Several directions for minority languages computerization. In *Proceedings of 19th International Conference on Computational Linguistics (COLING 2002)*, Taipei, Taiwan, 2002.

Delphine Bernhard. Unsupervised morphological segmentation based on segment predictability and word segments alignment. In *Proceedings of 2nd Pascal Challenges Workshop*, pages 19–24, Venice, Italy, 2006.

Daniel M. Bikel, Richard L. Schwartz, and Ralph M. Weischedel. An algorithm that learns what's in a name. *Machine Learning*, 34(1-3):211–231, 1999.

Hans C. Boas. Bilingual framenet dictionaries for machine translation. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC 2002)*, Las Palmas, Spain, 2002.

Branimir Boguraev and Ted Briscoe. Large lexicons for natural language processing: Exploring the grammar coding system of LDOCE. *Computational Linguistics*, 13, 1987.

Branimir Boguraev and Ted Briscoe, editors. *Computational Lexicography for Natural Language Processing*. Longman, London, 1989.

Stefan Bordag. Two-step approach to unsupervised morpheme segmentation. In *Proceedings of 2nd Pascal Challenges Workshop*, pages 25–29, Venice, Italy, 2006.

Michael R. Brent. From grammar to lexicon: Unsupervised learning of lexical syntax. *Computational Linguistics*, 19(2):243–262, 1993a.

Michael R. Brent. Minimal generative models: A middle ground between neurons and triggers. In *Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics*, Ft. Laudersdale, FL, 1993b.

Michael R. Brent. An efficient, probabilistically sound algorithm for segmentation and word discovery. *Machine Learning*, 34:71–105, 1999.

Michael R. Brent, Sreerama K. Murthy, and Andrew Lundberg. Discovering morphemic suffixes: A case study in minimum description length induction. In *Proceedings of the 15th Annual Conference of the Cognitive Science Society*, pages 28–36, Hillsdale, NJ, 1995.

Eric Brill. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*, 4 (21):543–565, 1995.

Eric Brill. *Recent Advances in Parsing Technology*, chapter Learning to Parse With Transformations. Kluwer Academic Publishers, 1996.

Eric Brill and Philip Resnik. A rule-based approach to prepositional phrase attachment disambiguation. In *Proceedings of 15th International Conference on Computational Linguistics (COLING 94)*, pages 1198–1204, Kyoto, Japan, 1994.

B. Bringmann, S. Kramer, F. Neubarth, H. Pirker, and G. Widmer. Transformation-based regression. Technical Report TR-2002-08, Vsterreichisches Forschungsinstitut f—r Artificial Intelligence, Wien, 2002.

T. Briscoe. Lexical issues in natural language processing. In E. Klein and F. Veltman, editors, *Natural Language and Speech*, pages 39–68. Springer-Verlag, 1991.

Peter F. Brown, John Cocke, Stephen A. DellaPietra, Vincent J. DellaPietra, Frederick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. A statistical approach to language translation. In *Proceedings of the 12th International Conference on Computational Linguistics*, Budapest, Hungary, 1988.

Peter F. Brown, John Cocke, Stephen A. DellaPietra, Vincent J. DellaPietra, Frederick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85, June 1990.

Peter F. Brown, Stephen A. DellaPietra, Vincent J. DellaPietra, and Robert L. Mercer. The mathematics of machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993.

Peter F. Brown, S. Della Pietra, V. J. Della Pietra, and Robert L. Mercer. Word sense disambiguation using statistical methods. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics (ACL 91)*, pages 264–270, Berkeley, California, 1991.

Ralf D. Brown. Automated dictionary extraction for "knowledge free" example-based translation. In *Proceedings of the Seventh International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 111–118, 1997.

Roy J. Byrd. Dictionary systems for office practice. In Donald E. Walker, Antonio Zampolli, and Nicoletta Calzolari, editors, *Automating the Lexicon: Research and Practice in a Multilingual Environment*, pages 207–219. Oxford University Press, 1995.

Roy J. Byrd, Nicoletta Calzolari, Martin S. Chodorow, Judith L. Klavans, Omneya A. Rizk, and Mary S. Neff. Tools and methods for computational lexicology. *Computational Linguistics*, 3(13):218–240, 1987.

Chris Callison-Burch and Miles Osborne. Bootstrapping parallel corpora. In *HLT-NAACL 2003 Workshop: Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, pages 44–49, Edmonton, Alberta, Canada, May 2003. Association for Computational Linguistics.

Nicoletta Calzolari. Detecting patterns in a lexical database. In *Proceedings of the 22th Annual Meeting of the association for Computational Linguistics and the 10th International Conference on Computational Linguistics (COLING-ACL 84)*, Stanford, California, 1984.

Nicoletta Calzolari and Allessandro Lenci. Computational lexicons for cross-language information retrieval. In *Cross-Language Information Retrieval: A Research Roadmap Workshop at SIGIR-2002*, Tampere Finland, 2002.

R. Canals-Marote, A. Esteve-Guillén, A. Garrido-Alenda, M. I. Guardiola-Savall, A. Itturaspe-Bellver, S. Montserrat-Buendia, S. Ortiz-Rojas, H. Pastor-Pina, P. M. Pérez-Antón, and M. L. Forcada. The Spanish-Catalan machine translation system interNOSTRUM. In *Proceedings of MT Summit VIII*, Santiago de Compostela, Spain, 2001.

Jan Edmund Carlsen. *Cebuano-English-Swedish Lexikon*. JEC Computext, 1999.

J. S. Chang, J. N. Chen, H. H. Sheng, and S. J. Ker. Combining machine readable lexical resources and bilingual corpora for broad word sense disambiguation. In *Proceedings of the Second Conference of the Association for Machine Translation in the Americas (AMTA 96)*, Montreal, Canada, 1996.

Eugene Charniak. *Statistical Language Learning*. MIT Press, Cambridge, Massachusetts, 1993.

Kenneth W. Church. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the Second Conference on Applied Natural Language Processing (ANLP 88)*, pages 136–143, Austin, Texas, 1988.

Ann Copestake, Ted Briscoe, Piek Vossen, Alicia Ageno, Irene Castellon, Francesc Ribas, German Rigau, Horacio Rodríguez, and Anna Samiotou. Acquisition of lexical translation relations from mrds. *Machine Translation*, 9(3–4):183–219, 1995.

Mathias Creutz. Unsupervised segmentation of words using prior distributions of morph length and frequency. In *Proceedings of the 41st Annual Meeting of the*

Association of Computational Linguistics (ACL 2003), pages 280–287, Sapporo, Japan, July 2003.

Mathias Creutz and Krista Lagus. Unsupervised discovery of morphemes. In *Proceedings of the Sixth Meeting of the ACL Special Interest Group in Computational Phonology (SIGPHON 2002)*, Philadelphia, PA, 2002.

Mathias Creutz and Krista Lagus. Inducing the morphological lexicon of a natural language from unannotated text. In *Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning (AKRR 2005)*, Espoo, Finland, June 2005.

Mathias Creutz and Krista Lugas. Induction of a simple morphology for highly-inflecting languages. In *Proceedings of the Seventh Meeting of the ACL Special Interest Group in Computational Phonology (SIGPHON 2004)*, pages 43–51, Barcelona, Spain, July 2004.

Doug Cutting, Julian Kupiec, Jan Pedersen, and Penelope Sibun. A practical part-of-speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing (ANLP 92)*, 1992.

Ido Dagan and Kenneth W. Church. Termight: Identifying and translating technical terminology. In *Proceedings of the Fourth Conference on Applied Natural Language Processing (ANLP 94)*, pages 34–40, University of Stuttgart, Germany, 1994.

Ido Dagan, Kenneth W. Church, and William A. Gale. Robust bilingual word alignment for machine aided translation. In *Proceedings of the Workshop on Very*

Large Corpora: Academic and Industrial Perspectives, pages 1–8, Columbus, Ohio, June 1993.

B. Daille, E. Gaussier, and J. M. Lange. Towards automatic extraction of monolingual and bilingual terminology. In *Proceedings of the 15th International Conference on Computational Linguistics, (COLING 94)*, pages 712–716, Kyoto, Japan, 1994.

Béatrice Daille, Cécile Fabre, and Pascale Sébillot. Applications of computational morphology. In Paul Boucher, editor, *Many Morphologies*, pages 210–234. Cascadilla Press, Somerville, MA, 2002.

Sajib Dasgupta and Vincent Ng. High-performance, language-independent morphological segmentation. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL/HLT 2007)*, New York, 2007.

M. Davel and E. Barnard. Effort and accuracy during language resource generation: A pronunciation prediction case study. In *Proceedings of the 17th Annual Symposium of the Pattern Recognition Association of South Africa*, pages 14–17, November 2006.

David Day, John Aberdeen, Lynette Hirschman, Robyn Kozierok, Patricia Robinson, and Marc Vilain. Mixed-initiative development of language processing systems. In *Proceedings of the Fifth Conference on Applied Natural Language Pro-*

cessing (ANLP 97), pages 348–355, Washington D.C., March 1997. Association for Computational Linguistics.

Johan de Caluwe and Ariane van Santen. Phonological, morphological and syntactic specifications in monolingual dictionaries. In Piet van Sterkenburg, editor, *A Practical Guide to Lexicography*, pages 71–82. John Benjamins Pub., Institute for Dutch Lexicology, Leiden, 2003.

H. Dejean. Morphemes as necessary concepts for structures: Discovery from untagged corpora. In *Workshop on Paradigms and Grounding in Natural Language Learning*, pages 295–299, Adelaide, January 1998.

H. Dejean, E. Gaussier, and F. Sadat. An approach based on multilingual thesauri and model combination for bilingual lexicon extraction. In *Proceedings of 19th International Conference on Computational Linguistics (COLING 2002)*, Taipei, Taiwan, 2002.

Vera Demberg. A language-independent unsupervised model for morphological segmentation. In *Proceedings of 45th Meeting of Association for Computational Linguistics (ACL 2007)*, Prague, Czech Republic, June 2007.

Daniel DeMenthon and Marc Vuilleumier. Lamp_hmm v.0.9. http://www.cfar.umd.edu/∼daniel/LAMP_HMM.zip, 2003. software.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

P. Van der Eijk. Automating the acquisition of bilingual terminology. In *Proceedings of the Sixth Conference of the European Chapter of the Association for Computational Linguistics (EACL 1993)*, Utrecht, The Netherlands, 1993.

Mona Diab and Philip Resnik. An unsupervised method for word sense tagging using parallel corpora. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, Philadelphia, Pennsylvania, July 2002.

Gamallo Inés Diz. The importance of MT for the survival of minority languages: Spanish-Galician MT system. In *Proceedings of the MT Summit VIII*, Santiago de Compostela, Spain, 2001.

S. T. Dumais, G. W. Furnas, T. K. Landauer, S. Deerwester, and R. Harshman. Using latent semantic analysis to improve access to textual information. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 281–285. ACM Press, 1988.

Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database.* MIT Press, 1998.

Radu Florian, John Henderson, and Grace Ngai. Coaxing confidence from an old friend: Probabilistic classifications from tranformation rule lists. In *Proceedings of SIGDAT-EMNLP 2000*, pages 26–43, Hong Kong, October 2000.

Radu Florian and Grace Ngai. Multidimensional transformational-based learning. *Proceedings of the Fifth Conference on Computational Natural Language Learning, CoNLL 2001*, pages 1–8, July 2001.

C. Fluhr. Multilingual information retrieval. In R. A. Cole, J. Mariani, H. Uszkoreit, A. Zaenen, and V. Zue, editors, *Survey of the State of the Art in Human Language Technology*, pages 391–405. Center for Spoken Language Understanding, Oregon Graduate Institute, 1995.

E. Fox, T. Nutter, T. Ahlswede, M. Evens, and J. Markowitz. Building a large thesaurus for information retrieval. In *Proceedings of the Second Conference on Applied Natural Language Processing (ANLP 88)*, pages 101–108, Austin, Texas, 1988.

Dayne Freitag. Morphology induction from term clusters. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 128–135, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

Pascale Fung. A pattern matching method for finding noun and proper noun translations from noisy parallel corpora. In *Proceedings of the 33rd Annual Conference of the Association for Computational Linguistics*, Boston, Massachusettes, 1995.

Pascale Fung. A statistical view on bilingual lexicon extraction - from parallel corpora to non-parallel corpora. In J. Véronis, editor, *Parallel Text Processing - Alignment and Use of Translation Corpora*. Kluwer Academic Publishers, 2000.

Pascale Fung and Kenneth W. Church. K-vec: A new approach for aligning parallel texts. In *Proceedings of the Fifteenth International conference on Computational Linguistics*, pages 1096–1102, Kyoto, Japan, 1994.

Pascale Fung and Kathleen R. McKeown. Aligning noisy parallel corpora across language groups: Word pair feature matching by dynamic time warping. In *Proceedings of the First Conference of the Association for Machine Translation in the Americas (AMTA 94)*, pages 81–88, Columbia, Maryland, 1994.

Pascale Fung and Lo Yuen Yee. An IR approach for translating new words from nonparallel, comparable texts. In Christian Boitet and Pete Whitelock, editors, *Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics*, pages 414–420, San Francisco, California, 1998. Morgan Kaufmann Publishers.

William A. Gale and Kenneth W. Church. A program for aligning sentences in bilingual corpora. In *Proceedings of 29th Meeting of Association for Computational Linguistics (ACL 91)*, pages 177–184, Berkeley, California, June 1991.

William A. Gale and Kenneth W. Church. A program for aligning sentences in bilingual corpora. *Computational Linguistics*, 19(1), 1993.

William A. Gale, Kenneth W. Church, and David Yarowsky. Using bilingual materials to develop word sense disambiguation methods. In *Proceedings of the Fourth International Conference on Theoretical and Methodological Issues in Machine Translation (TMI 1992)*, pages 101–112, Montreal, Canada, June 1992.

E. Gaussier. Flow network models for word alignment and terminology extraction from bilingual corpora. In *Proceedings of the 36th Annual Meeting of the Asso-*

ciation for Computational Linguistics and the 17th International Conference on Computational Linguistics (COLING-ACL 98), Montreal, Canada, August 1998.

Eric Gaussier. Unsupervised learning of derivational morphology from inflectional lexicons. In *The 37th Annual Meeting of the ACL Workshop Proceedings: Unsupervised Learning in Natural Language Processing*, pages 285–292, 1999.

Rayid Ghani, Rosie Jones, and Dunja Mladenic. Mining the web to create minority language corpora. In *Proceedings of 2001 ACM International Conference on Knowledge Management (CIKM2001)*, pages 279–286. Association for Computing Machinery, 2001.

D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, New York, 1989.

John Goldsmith. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–198, 2001.

John Goldsmith. An algorithm for the unsupervised learning of morphology. *Natural Language Engineering*, 12(3):1–19, 2006.

Sharon Goldwater and David McClosky. Improving statistical MT through morphological analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2005)*, Vancouver, BC, Canada, October 2005.

Jr. Gordon, Raymond G., editor. *Ethnologue: Languages of the World.* SIL International, Dallas, Texas, fifteenth edition, 2005.

P. B. Gove, editor. *Webster's Seventh New Collegiate Dictionary*. Merriam-Webster, Springfield, MA, 1969.

Peter D. Grünwald. A tutorial introduction to the minimum description length principle. In Peter D. Grünwald, In Jae Myung, and Mark A. Pitt, editors, *Advances in Minimum Description Length: Theory and Applications*. MIT Press, April 2005.

K. Hacioglu, B. Pellom, T. Ciloglu, O. Ozturk, M. Kurimo, and M. Creutz. On lexicon creation for Turkish lvcsr. In *Proceedings of Eurospeech'03*, pages 1165–1168, Geneva, Switzerland, 2003.

M. A. Hafer and S. F. Weiss. Word segmentation by letter successor varieties. *Information Storage and Retrieval*, 10:371–385, 1974.

Benjamin Han. Building a bilingual dictionary with scarce resources: A genetic algorithm approach. In *Student Research Workshop, the Second Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL 2001)*, Pittsburgh, PA, 2001.

J. Hankamer. Morphological parsing and the lexicon. In W. Marslen-Wilson, editor, *Lexical Representation and Process*, pages 392–408. MIT Press, Cambridge, MA, 1989.

P. Hanks, editor. *The Collins English Dictionary*. Collins, Glasgow, 1987.

Mike Hannay. Types of bilingual dictionaries. In Piet van Sterkenburg, editor, *A*

Practical Guide to Lexicography, pages 145–153. John Benjamins Pub., Institute for Dutch Lexicology, Leiden, 2003.

D. Hardt. Improving ellipsis resolution with transformation-based learning. In *AAAI Fall Symposium*, 1998.

Zellig Harris. From phoneme to morpheme. *Language*, 31:190–222, 1955.

Masahiko Haruno, Satoru Ikehara, and Takefumi Yamazaki. Learning bilingual collocations by word-level sorting. In *Proceedings of 16th International Conference on Computational Linguistics (COLING 96)*, pages 525–530, Copenhagen, Denmark, 1996.

R. Hauser and A. Storrer. Dictionary entry parsing using the LEXPARSE system. *Lexicographica*, 9:174–219, 1994.

Mark Hepple. Independence and commitment: Assumptions for rapid training and execution of rule-based pos taggers. In *Proceedings of 38th Meeting of Association for Computational Linguistics (ACL 2000)*, pages 278–285, Hong Kong, October 2000.

J. H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, MI, 1975.

Yu Hu, John Goldsmith, Irina Matveeva, and Colin Sprague. The SED heuristic for morpheme discovery: a look at Swahili. In *Workshop on Psychocomputational Models of Human Language Acquisition (ACL 2005)*, Ann Arbor, MI, 2005a.

Yu Hu, John Goldsmith, Irina Matveeva, and Colin Sprague. Using morphology and syntax together in unsupervised learning. In *Workshop on Psychocomputational Models of Human Language Acquisition (ACL 2005)*, Ann Arbor, MI, 2005b.

Baden Hughes, Timothy Baldwin, Steven Bird, Jeremy Nicholson, and Andrew MacKinlay. Reconsidering language identification for written language resources. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC 2006)*, Paris, France, 2006. European Language Resources Association.

Howard Jackson. *Lexicography: An Introduction*. Routledge, 2002.

Christian Jacquemin. Guessing morphology from terms and corpora. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'97)*, pages 156–165, Philadelphia, PA, 1997.

F. Jelinek, R. Mercer, and S. Roukos. Principles of lexical language modeling for speech recognition. In S. Furni and M. M. Sondhi, editors, *Advances in Speech Processing*, pages 651–699. Marcel Dekker, New York, 1992.

Howard Johnson and Joel Martin. Unsupervised learning of morphology for English and Inuktitut. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL 2003)*, 2003.

G. David Forney Jr. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, March 1973.

B. H. Juang and L. R. Rabiner. The segmental k-means algorithm for estimating parameters of Hidden Markov Models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(9):1639–1641, 1990.

Hiroyuki Kaji, Yuuko Kida, and Yasutsugu Morimoto. Learning translation templates from bilingual text. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING 92)*, pages 672–678, Nantes, France, 1992.

Fred Karlsson. *Finnish Grammar*. WSOY, Juva, second edition edition, 1987.

Martin Kay and Martin Röscheisen. Text-translation alignment. *Computational Linguistics*, 19(1):121–142, 1993.

R. Kazman. Structuring the text of the Oxford English Dictionary through finite state transduction. Technical Report CS-86-20, University of Waterloo, 1986.

Judy Kegl. Machine-readable dictionaries and education. In Donald E. Walker, Antonio Zampolli, and Nicoletta Calzolari, editors, *Automating the Lexicon: Research and Practice in a Multilingual Environment*, pages 249–284. Oxford University Press, 1995.

Samarth Keshava and Emily Pitler. A simpler, intuitive approach to morpheme induction. In *Proceedings of 2nd Pascal Challenges Workshop*, pages 31–35, Venice, Italy, 2006.

Philipp Koehn and Kevin Knight. Learning a translation lexicon from monolingual corpora. In *Proceedings of ACL-02 Workshop on Unsupervised Lexical Acquisition*, 2002.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the Human Language Technology and North American Chapter of the Association for Computational Linguistics Conference (NAACL/HLT 2003)*, Edmonton, Canada, 2003.

Okan Kolak and Philip Resnik. OCR post-processing for low density languages. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 867–874, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics.

Akira Kumano and Hideki Hirakawa. Building an MT dictionary from parallel texts based on linguistic and statistical information. In *Proceedings of the 15th International Conference on Computational Linguistics*, pages 76–81, 1994.

J. Kupiec. An algorithm for finding noun phrase correspondences in bilingual corpora. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, Columbus, Ohio, 1993.

Mikko Kurimo, Mathias Creutz, Matti Varjokallio, Ebru Arisoy, and Murat Saraclar. Unsupervised segmentation of words into morphemes - Challenge 2005 an introduction and evaluation report. In *Proceedings of 2nd Pascal Challenges Workshop*, April 2006a.

Mikko Kurimo, Mathias Creutz, Matti Varjokallio, Ebru Arisoy, and Murat Saraclar. Unsupervised segmentation of words into morphemes - Morpho Challenge 2005: Application to automatic speech recognition. In *Proceedings of the International Conference on Spoken Language Processing - Interspeech (ICSLP 2006)*, Pittsburgh, PA, September 2006b.

Torbjörn Lager. The μ-tbl system: Logic programming tools for transformation-based learning. In *Proceedings of the Third International Workshop on Computational Natural Language Learning*, Bergen, 1999.

Torbjörn Lager. Transformation-based learning of rules for constraint grammar tagging. In *The 13th Nordic Conference in Computational Linguistics*, Uppsala, Sweden, May 2001.

Torbjörn Lager and Natalia Zinovjeva. Sense and deduction: The power of peewees applied to the Senseval-2 Swedish lexical sample task. In *Proceedings of SENSEVAL-2: Second International Workshop on Evaluating Word Sense Disambiguation Systems*, Toulouse, France, July 2001.

Sidney I. Landau. *Dictionaries: The Art and Craft of Lexicography.* Cambridge University Press, 2nd edition edition, 2001.

Alon Lavie, Katharina Probst, Erik Peterson, Stephan Vogel, Lori Levin, Ariadna Font-Llitjós, and Jaime Carbonell. A trainable transfer-based machine translation approach for languages with limited resources. In *Proceedings of the Ninth Workshop of the European Association for Machine Translation (EAMT 2004)*, 2004.

Alon Lavie, Stephan Vogel, Erik Peterson, Katharina Probst, Ariadna Font-Llitjós, Rachel Reynolds, Jaime Carbonell, and Richard Cohen. Experiments with a Hindi-to-English transfer-based MT system under a miserly data scenario. *ACM Transactions on Asian Language Information Processing, Special Issue on Rapid Deployment Hindi-English Translation Systems*, 2(2), 2003.

Douglas B. Lenat and Edward A. Feigenbaum. On the thresholds of knowledge. In *Proceedings of the 10th International Joint Conference on Artificial lntelligence (IJCAI-87)*, pages 1173–1182, Italy, 1987.

Douglas B. Lenat, Mayank Prakash, and Mary Shepherd. Cyc: Using common sense knowledge to overcome brittleness and knowledge acquisition bottlenecks. *The AI Magazine*, 7(4):65–85, 1986.

Micheal Levison and Greg Lessard. Syntax and morphology in verb phrase generation. *Literary and Linguistic Computing*, 10(4):237–245, 1995.

Huanfeng Ma and David Doermann. Bootstrapping structured page segmentation. In *Proceedings of the SPIE Conference Document Recognition and Retrieval*, Santa Clara, CA, January 2003a.

Huanfeng Ma and David Doermann. Gabor filter based multi-class classifier for scanned document images. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003)*, pages 968–972, 2003b.

Huanfeng Ma and David Doermann. Adaptive Hindi OCR using generalized Hausdorff image comparison. *ACM Transactions on Asian Language Information Processing*, 26(2):198–213, 2004a.

Huanfeng Ma and David Doermann. Word level script identification on scanned document images. In *Proceedings of the SPIE Conference on Document Recogntion and Retreival*, pages 124–135, 2004b.

Huanfeng Ma and David Doermann. Font identification using the grating cell texture operator. In *Proceedings of the SPIE Conference on Document Recogntion and Retreival XXII*, 2005.

Huanfeng Ma, Burcu Karagol-Ayan, David Doermann, Douglas Oard, and Jianqiang Wang. Parsing and tagging of bilingual dictionaries. *Traitement Automatique Des Langues*, pages 125–150, 2003.

Muhtar Mahsut, Ogawa Yasuhiro, Sugino Kazue, and Inagaki Yasuyoshi. Utilizing agglutinative features in Japanese-Uighur machine translation. In *Proceedings of the MT Summit VIII*, 2001.

Lidia Mangu and Eric Brill. Automatic rule acquisition for spelling correction. In *Proceedings of the 14th International Conference on Machine Learning*, pages 734–741, Nashville, Tennessee, 1997.

Song Mao and Tapas Kanungo. Stochastic language models for automatic acquisition of lexicons from printed bilingual dictionaries. In *Document Layout Interpretation and Its Applications*, Seattle, WA, 2001.

Mike Maxwell and Baden Hughes. Frontiers in linguistic annotation for lower-density languages. In *Proceedings of the Workshop on Frontiers in Linguistically Annotated Corpora 2006*, pages 29–37, Sydney, Australia, July 2006. Association for Computational Linguistics.

R. S. McGregor, editor. *The Oxford Hindi-English Dictionary.* Oxford University Press, 1993.

I. Dan Melamed. Automatic discovery of non-compositional compounds in parallel data. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing (EMNLP-97)*, pages 97–108, Providence, RI, August 1997a.

I. Dan Melamed. A scalable architecture for bilingual lexicography. Technical Report MS-CIS-97-01, Dept. of Computer and Information Science Technical Report, University of Pennsylvania, 1997b.

I. Dan Melamed. Models of translational equivalence among words. *Computational Linguistics*, 26(2):221–249, June 2000.

Bernard Merialdo. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2):155–171, 1994.

Adam Meyers, Michiko Kosaka, and Ralph Grishman. Chart-based transfer rule application in machine translation. In *Proceedings of the 18th International Con-*

ference on Computational Linguistics (COLING 2000), Saarbrucken, Germany, 2000.

George Miller. Dictionaries of the mind. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, Chicago, IL, 1985a.

George Miller. Wordnet: A dictionary browser. In *Proceedings of the First International Conference on Information in Data*, Waterloo, Ontaria: University of Waterloo Centre for the New Oxford English Dictionary, 1985b.

Christian Monson. A framework for unsupervised natural language morphology induction. In *Proceedings of the Student Research Workshop in ACL 2004*, 2004.

Christian Monson, Ariadna Font Llitjos, Roberto Aranovich, Lori Levin, Ralf Brown, Eric Peterson, Jaime Carbonel, and Alon Lavie. Building nlp systems for two resource-scarce indigenous languages: Mapudungun and Quechua. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC 2006). Fifth SALTMIL Workshop on Minority Languages: "Strategies for developing machine translation for minority languages"*, pages 15–24, Genoa, Italy, May 2006.

Christof Monz. *From Document Retrieval to Question Answering.* PhD thesis, Institute for Logic, Language, and Computation, University of Amsterdam, 2003.

Christof Monz and Maarten de Rijke. Shallow morphological analysis in monolingual information retrieval for Dutch, German, and Italian. In *Post-Conference*

Proceedings of the Cross Language Evaluation Forum Workshop (CLEF 2001). Springer Lecture Notes in Computer Science, 2002.

J. A. H. Murray, H. Bradley, W. A. Craigie, and C. T. Onions, editors. *The Oxford English Dictionary.* Oxford at the Clarendon Press, Oxford, England, 1928.

K. M. E. Murray. *Caught in the Web of Words; James Murray and the Oxford English Dictionary.* Yale University Press, New Haven, Conn., 1977.

J. Nakamura and N. Makoto. Extraction of semantic information from an ordinary English dictionary and its evaluation. In *Proceedings of the 12th International Conference on Computational Linguistics (COLING 88)*, pages 459–464, Budapest, Hungary, 1988.

Mary S. Neff, B. Blaser, J. M. Lange, Hubert Lehmann, and Isabel Zapata Dominguez. Get it where you can: Acquiring and maintaining bilingual lexicons for machine translation. In *Working Notes, Building Lexicons for Machine Translation, AAAI 93 spring symposium, Technical Report SS-93-02*, page 104, Stanford, CA, 1993.

Mary S. Neff and Branimir K. Boguraev. Dictionaries, dictionary grammars and dictionary entry parsing. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics (ACL 89)*, 1989.

Mary S. Neff and Michael McCord. Acquiring lexical data from machine-readable dictionary resources for machine translation. In *Third International Conference* *on Theoretical and Methodological Issues in Machine Translation of Natural Languages (TMI-90)*, pages 85–90, Austin, Texas, 1990.

John Nerbonne, Lauri Karttunen, Elena Paskaleva, Gabor Proszeky, and Tiit Roosmaa. Reading more into foreign languages. In *Proceedings of the Fifth ACL Conference on Applied Natural Language Processing (ANLP 97)*, pages 135–138, Washington, DC, 1997.

Sylvain Neuvel and Sean A. Fulop. Unsupervised learning of morphology without morphemes. In *Proceedings of the Sixth Meeting of the ACL Special Interest Group in Computational Phonology (SIGPHON 2002)*, Philadelphia, PA, 2002.

Grace Ngai and Radu Florian. Transformation-based learning in the fast lane. In *Proceedings of the Second Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2001)*, pages 40–47, Pittsburgh, PA, June 2001.

Sonja Nießen and Hermann Ney. Statistical machine translation with scarce resources using morpho-syntactic information. *Computational Linguistics*, 30(2): 181–204, June 2004.

Sergei Nirenburg. Project Boas: "A linguist in the box" as a multi-purpose language. In *Proceedings of the First International Conference on Language Resources and Evaluation (LREC 1998)*, 1998.

Sergei Nirenburg and Victor Raskin. Universal grammar and lexis for quick ramp-up of MT systems. In *Proceedings of 36th Annual Meeting of the Association for*

Computational Linguistics and 17th International Conference on Computational Linguistics, pages 975–979. Association for Computational Linguistics, 1998.

Douglas W. Oard. The surprise language exercises. *ACM Transactions on Asian Language Information Processing (TALIP)*, 2(2):79–84, 2003.

Douglas W. Oard and Bonnie J. Dorr. A survey of multilingual text retrieval. Technical Report UMIACS-TR-96-19, University of Maryland, Institute for Advanced Computer Studies, April 1996.

Franz Joseph Och and Hermann Ney. A comparison of alignment models for statistical machine translation. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, Saarbrucken, Germany, 2000.

Franz Joseph Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):9–51, March 2003.

Kemal Oflazer. Dependency parsing with an extended finite-state approach. *Computational Linguistics*, 29(4):515–544, December 2003.

Kemal Oflazer, Bilge Say, Dilek Hakkani-Tür, and Gökhan Tür. Building a Turkish treebank. In Anne Abeillé, editor, *Building and Using Parsed Corpora*. Kluwer Academic Publishers, 2003.

Lineke Oppentocht and Rik Schutz. Developments in electronic dictionary design. In Piet van Sterkenburg, editor, *A Practical Guide to Lexicography*, pages 215–227. John Benjamins Pub., Institute for Dutch Lexicology, Leiden, 2003.

Casey Palowitch and Darin Stewart. Automating the structural markup process in the conversion of print documents to electronic text. In *Digital Libraries '95: The Second Annual Conference on the Theory and Practice of Digital Libraries*, Austin, Texas, 1995.

Michael Paul. Translation knowledge recycling for related languages. In *Proceedings of MT Summit VIII*, pages 265–268, Santiago de Compostela, Spain, September 2001.

C. Peters and E. Picchi. Capturing the comparable: A system for querying comparable text corpora. In *JADT Proceedings*, 1995.

Victor Poznanski, Pete Whitelock, Jan Udens, and Steffan Corley. Practical glossing by prioritised tiling. In *Proceedings of 17th International Conference on Computational Linguistics (COLING 98)*, pages 1060–1066, Montreal, Canada, 1998.

Katharina Probst. Semi-automatic learning of transfer rules for machine translation of low-density languages. In *Proceedings of the Student Session of the 14th European Summer School in Logic, Language and Information, (ESSLLI-02)*, 2002.

Katharina Probst, Ralf Brown, Jaime Carbonell, Alon Lavie, Lori Levin, and Erik Peterson. Design and implementation of controlled elicitation for machine translation of low-density languages. In *Proceedings of the MT Summit VIII*, Santiago de Compostela, Spain, 2001.

Katharina Probst, Lori Levin, Erik Peterson, Alon Lavie, and Jaime Carbonell. MT for minority languages using elicitation-based learning of syntactic transfer rules. *Machine Translation*, 17(4), 2002.

P. Procter, editor. *Longman Dictionary of Contemporary English*. Longman Group, 1978.

Gabór Prószéky. Intelligent multi-dictionary environment. In *Proceedings of 17th International Conference on Computational Linguistics (COLING 98)*, pages 1067–1071, Montreal, Canada, 1998.

Gabór Prószéky and Balázs Kis. Context-sensitive electronic dictionaries. In *Proceedings of the 17th International Conference on Computational Linguistics (COLING 2002)*, Taipei, Taiwan, 2002. Also as "Development of a Context-Sensitive Electronic Dictionary" in Proceedings of EURALEX, 2002 Copenhagen, Vol. I., pp. 281–290.

Lawrence R. Rabiner and Biing-Hwang Juang. An introduction to Hidden Markov Models. *IEEE ASSP Magazine*, pages 4–15, January 1986.

Lawrence R. Rabiner and Biing-Hwang Juang. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2): 257–286, 1989.

Lawrence R. Rabiner, J. G. Wilpon, and Biing-Hwang Juang. A segmental K-means training procedure for connected word recognition. *AT&T Technical Journal*, 64 (3):21–40, May 1986.

Lance Ramshaw and Mitchell Marcus. Exploring the statistical derivation of transformational rule sequences for part-of-speech tagging. In *The Balancing Act: Proceedings of the ACL Workshop on Combining Symbolic and Statistical Approaches to Languages*, New Mexico State University, July 1994.

Lance Ramshaw and Mitchell Marcus. *Natural Language Processing Using Very Large Corpora*, chapter Text Chunking Using Tranformation-based Learning. Kluwer, 1999.

Reinhard Rapp. Automatic identification of word translations from unrelated English and German corpora. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 519–526, College Park, Maryland, 1999.

Philip Resnik. Mining the web for bilingual text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL 99)*, University of Maryland, College Park, Maryland, June 1999.

Philip Resnik and I. Dan Melamed. Semi-automatic acquisition of domain-specific translation lexicons. In *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP 97)*, 1997.

Philip Resnik, Mari Broman Olsen, and Mona Diab. The Bible as a parallel corpus: Annotating the 'Book of 2000 tongues'. *Computers and the Humanities*, 33(1–2): 129–153, 1999.

Philip Resnik and Noah A. Smith. The web as a parallel corpus. *Computational Linguistics*, 29(3):349–380, September 2003.

Jorma Rissanen. *Stochastic Complexity in Statistical Inquiry.* World Scientific Publishing Company, New Jersey, 1989.

Emmanuel Roche and Yves Schabes. Deterministic part of speech tagging with finite state transducers. *Computational Linguistics*, 2(21):227–253, 1995.

Kato Ronald and Etienne Barnard. Statistical translation with scarce resources - a South African case study. In *Proceedings of the 17th Annual Symposium of the Pattern Recognition Association of South Africa*, pages 18–22, November 2006.

Fatiha Sadat, Masatoshi Yoshikawa, and Shunsuke Uemura. Bilingual terminology acquisition from comparable corpora and phrasal translation to cross-language information retrieval. In *Proceedings of 41st Meeting of Association for Computational Linguistics (ACL 2003)*, Sapporo, Japan, 2003a.

Fatiha Sadat, Masatoshi Yoshikawa, and Shunsuke Uemura. Learning bilingual translations from comparable corpora to cross-language information retrieval: Hybrid statistics-based and linguistics-based approach. In *Proceedings of the Sixth International Workshop on Information Retrieval with Asian Languages (with ACL)*, pages 57–64, Sappro, Japan, 2003b.

Ken Samuel. Lazy transformation-based learning. In *Proceedings of the 11th International Florida AI Research Symposium Conference*, pages 235–239, Florida, 1998.

Ken Samuel, Sandra Carberry, and K. Vijay-Shanker. Dialogue act tagging with transformation-based learning. In *Proceedings of the 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics*, pages 1150–1156, Montreal, Quebec, Canada, 1998.

Kevin Scannell. Automatic thesaurus generation for minority languages: an Irish example. *Actes des Traitement Automatique des Langues Minoritaires et des Petites Langues*, 2:203–212, 2003.

Kevin Scannell. Machine translation for closely related language pairs. In *Proceedings of the LREC 2006 Workshop on Strategies for Developing Machine Translation for Minority Languages*, Paris, 2006. European Language Resources Association.

Patrick Schone and Daniel Jurafsky. Knowledge-free induction of morphology using latent semantic analysis. In *Fourth Conference on Computational Natural Language Learning and of the Second Learning Language in Logic Workshop*, pages 67–72, Lisbon, Portugal, 2000.

Patrick Schone and Daniel Jurafsky. Knowledge-free induction of inflectional morphologies. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL 2001)*, pages 183–191, Pittsburgh, PA, 2001.

Frédérique Segond, Anne Schiller, Gregory Grefenstette, and Jean-Pierre Chanod.

An experiment in semantic tagging using Hidden Markov Model tagging. In *Proceedings of the Joint ACL/EACL Workshop on Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, pages 78–81, Madrid, Spain, 1997.

P. H. Sellers. The theory and computation of evolutionary distance: Pattern recognition. *Journal of Algorithms*, 1(4):359–373, 1980.

Utpal Sharma, Jugal Kalita, and Rajib Das. Unsupervised learning of morphology for building lexicon for a highly inflectional language. In *Proceedings of the Sixth Meeting of the ACL Special Interest Group in Computational Phonology (SIGPHON 2002)*, pages 1–10, Philadelphia, PA, July 2002.

Svetlana Sheremetyeva and Sergei Nirenburg. Towards a universal tool for NLP resource acquisition. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC 2000)*, Athens, Greece, 2000.

Sayori Shimohata, Mihoko Kitamura, Tatsuya Sukehiro, and Toshiki Murata. Collaborative translation environment on the web. In Bente Maegaard, editor, *Proceedings of the MT Summit VIII – Machine Translation in the Information Age*, pages 331–33, Santiago de Compostela, Spain, September 2001.

V. Siivola, T. Hirsimäki, M. Creutz, and M. Kurimo. Unlimited vocabulary speech recognition based on morphs discovered in an unsupervised manner. In *Proceedings of Eurospeech'03*, pages 2293–2296, Geneva, Switzerland, 2003.

John Sinclair, editor. *Looking up: An Account of the COBUILD Project in Lexical Computing*. Collins, 1987.

John Sinclair, P. Hanks, G. Fox, R. Moon, and P. Stock, editors. *Collins COBUILD English Language Dictionary*. Collins, 1987.

Jonathan Slocum and Martha G. Morgan. The role of dictionaries and machine-readable lexicons in translation. In Donald E. Walker, Antonio Zampolli, and Nicoletta Calzolari, editors, *Automating the Lexicon*, pages 221–247. Oxford University Press, 1995.

Frank Smatja, Katleen McKeown, and Vasileios Hatzivassiogluou. Translating collocations for bilingual lexicons: A statistical approach. *Computational Linguistics*, 21(4):1–38, 1996.

Matthew G. Snover and Michael R. Brent. A Bayesian model for morpheme and paradigm identification. In *Proceedings of 39th Meeting of Association for Computational Linguistics (ACL 2001)*, pages 482–490, Toulouse, France, 2001.

Matthew G. Snover, Gaja E. Jarosz, and Michael R. Brent. Unsupervised learning of morphology using a novel directed search algorithm: Taking the first step. In *Proceedings of the Sixth Meeting of the ACL Special Interest Group in Computational Phonology (SIGPHON 2002)*, Philadelphia, PA, 2002.

Harold Somers. Machine translation and minority languages. *Translating and the Computer*, 19:1–13, 1997.

Harold Somers. Language resources and minority languages. *Translating and the Computer*, 5:20–24, 1998.

Andrew Spencer. *Morphological theory: an introduction to word structure in generative grammar*. Blackwell Publishing, Oxford, UK; Cambridge, Mass., 1991.

Piet Swanepoel. Dictionary typologies: A pragmatic approach. In Piet van Sterkenburg, editor, *A Practical Guide to Lexicography*, pages 44–69. John Benjamins Pub., Institute for Dutch Lexicology, Leiden, 2003.

Kumiko Tanaka and H. Iwasaki. Extraction of lexical translations from non-aligned corpora. In *Proceedings of 16th International Conference on Computational Linguistics (COLING 96)*, pages 580–585, Copenhagen, Denmark, 1996.

Kumiko Tanaka and Kyoji Umemura. Construction of a bilingual dictionary intermediated by a third language. In *Proceedings of the 15th International Conference for Computational Linguistics (COLING 94)*, pages 297–393, Kyoto, Japan, 1994.

Kristina Toutanova, H. Tolga Ilhan, and Christopher D. Manning. Extensions to HMM-based statistical word alignment models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 87–94, Philadelphia, PA, July 2002.

Gükhan Tür, Dilek Hakkani-Tür, and Kemal Oflazer. A statistical information extraction system for Turkish. *Journal of Natural Language Engineering*, 9(2): 181–210, 2003.

Esko Ukkonen. Algorithms for approximate string matching. *Information and Control*, 4(1-3):100–118, 1985a.

Esko Ukkonen. Finding approximate patterns in strings. *Journal of Algorithms*, 4 (1-3):132–137, 1985b.

K. Urwin. *Langenscheidt's Standard French Dictionary.* Langenscheidt, Germany, 1988.

Takehito Utsuro, Takashi Horiuchi, Yasunobu Chiba, and Takeshi Hamamoto. Semi-automatic compilation of bilingual lexicon entries from cross-lingually relevant news articles on WWW news sites. In *Proceedings of the Fifth Conference of the Association for Machine Translation in the Americas (AMTA 2002)*, pages 165–176, Tiburon, California, 2002.

Takehito Utsuro, Y. Matsumoto, and M. Nagao. Lexical knowledge acquisition from bilingual corpora. In *Proceedings of 14th International Conference on Computational Linguistics (COLING 92)*, pages 581–587, Nantes, France, July 1992.

G. J. van der Steen. A treatment of queries in large text corpora. In S. Johansson, editor, *Computer Corpora in English Language Research*, pages 49–63. Norwegian Computing Centre for the Humanities, Bergen, 1982.

Piet van Sterkenburg, editor. *A Practical Guide to Lexicography.* John Benjamins Publishing Company, Institute for Dutch Lexicology, Leiden, 2003.

A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. In *IEEE Transactions on Information Theory*, pages 260–269, April 1967.

Stefan Vogel, Hermann Ney, and Christoph Tillmann. HMM-based word alignment in statistical translation. In H. Tommola, K. Tarantola, T. Salmin Tolonen, and J. Schop, editors, *Proceedings of the 16th International Conference on Computational Linguistics (COLING 96)*, pages 836–841, Copenhagen, Denmark, August 1996.

Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21(1):168–173, 1974.

Donald E. Walker and Robert Amsler. The use of machine-readable dictionaries in sublanguage analysis. In R. Grishman and R. Kittredge, editors, *Analyzing Language in Restricted Domains*, pages 69–83. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1986.

Hideo Watanabe, Sado Kurohashi, and Eiji Aramaki. Finding structural correspondences from bilingual parsed corpus for corpus-based translations. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, pages 906–912, Saarbrucken, Germany, 2000.

Richard Wicentowski. Multilingual noise-robust supervised morphological analysis using the wordframe model. In *Proceedings of the Seventh Meeting of the ACL Special Interest Group in Computational Phonology (SIGPHON 2004)*, pages 70–77, Barcelona, Spain, July 2004. Association for Computational Linguistics.

Herbert E. Wiegand. Aspekte der makrostruktur im allgemeinen einsprachigen wörterbuch: alphabetische anordnungsformen und ihne probleme. In Franz J. Hausmann et. al., editor, *Wörterbücher / Dictionaries / Dictionnaires, Ein internationales Handbuch zur Lexikografie / An International Encyclopedia of Lexicography / Encyclopédie internationale de lexicographie*, pages 371–409. 1989.

Yorick A. Wilks, Dan Fass, Cheng-Ming Guo, James E. McDonald, Tony Plate, and Brian M. Slator. Machine tractable dictionaries as tools and resources for natural language processing. In *Proceedings of 12th International Conference on Computational Linguistics (COLING 88)*, pages 750–755, Budapest, Hungary, 1988.

Yorick A. Wilks, Dan Fass, Cheng-Ming Guo, James E. McDonald, Tony Plate, and Brian M. Slator. Providing machine tractable dictionary tools. *Journal of Machine Translation*, 5(2):99–154, June 1990.

Yorick A. Wilks, Brian M. Slator, and Louise M. Guthrie. *Electric Words: Dictionaries, Computers, and Meanings.* MIT Press, Cambridge, MA, 1996.

John U. Wolff. *A Dictionary of Cebuano Visaya.* Southeast Asia Program, Cornell University, Ithaca, New York, 1972.

D. R. Woodhead and Wayne Beene, editors. *A Dictionary of Iraqi Arabic: Arabic–English Dictionary.* Georgetown University Press, Washington D.C., 2003.

Dekai Wu and X. Xia. Learning an English-Chinese lexicon from a parallel corpus.

In *Proceedings of the First Conference of the Association for Machine Translation in the Americas (AMTA 94)*, Columbia, MD, 1994.

David Yarowsky and Grace Ngai. Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. In *Proceedings of the Second Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2001)*, pages 377–404, Pittsburgh, PA, 2001.

David Yarowsky, Grace Ngai, and Richard Wicentowski. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the First Human Language Technology Conference (HLT 2001)*, pages 1–8, San Diego, California, 2001.

David Yarowsky and Richard Wicentowski. Minimally supervised morphological analysis by multimodal alignment. In *Proceedings of the 38th Annual Meeting of the Association of Computational Linguistics (ACL 2000)*, pages 207–216, Hong Kong, 2000.

Antonio Zampolli. Introduction. In Beryl T. S. Atkins and Antonio Zampolli, editors, *Computational Approaches to the Lexicon*, pages 3–16. Oxford University Press, 1994.