Automatic Improvement of Machine Translation Systems

Ariadna Font Llitjós
July 2007

CMU-LTI-07-008

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

Thesis Committee:
Jaime G. Carbonell, Chair
Alon Lavie
Lori Levin
Bonnie Dorr, University of Maryland

This work was supported in part by the National Science Foundation under grants IIS #0534217, IIS #0121631 and N66001-99-2-891804. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author and do not necessarily reflect those of the funding agencies.

Keywords: Machine Translation (MT), automatic rule refinement (ARR), interactive learning, lexicon and grammar extension, rule adaptation, constraint learning, structure learning, alignments, Transfer-Based MT (TBMT), Rule-Based MT (RBMT), knowledge acquisition (KA), elicitation, online user interaction, post-editing, MT output corrections, feedback, non-expert users, bilingual speakers, bilingual informants, English, Spanish, Mapudungun, Quechua.

Abstract

Achieving high translation quality remains the most daunting challenge Machine Translation (MT) systems currently face. Researchers have explored a variety of methods for including translator feedback in the MT loop. However, most MT systems have failed to incorporate post-editing efforts beyond the addition of corrected translations to the parallel training data for Example-Based and Statistical systems or to a translation memory database. This thesis describes a novel approach that utilizes post-editing information to automatically improve the underlying rules and lexical entries of a Transfer-Based MT system. This process can be divided into two main steps. First, an online translation correction tool allows for easy error diagnosis and implicit error categorization. Then, an Automatic Rule Refiner performs error remediation by tracing errors back to the problematic rules and lexical entries and executing repairs that are mostly lexical and morpho-syntactic in nature (such as word-order, missing constituents or incorrect agreement in transfer rules). This approach directly improves the intelligibility of corrected MT output and, more significantly, it generalizes over unseen data, providing improved MT output for similar sentences that have not been corrected.

Experimental results on an English-Spanish MT system show that automatic rule refinements triggered by bilingual speaker corrections successfully translate unseen data that was incorrectly translated by the original, unrefined grammar. Improvements on translation quality over a baseline, as measured by standard automatic evaluation metrics, are statistically significant on a paired two-tailed t-test ($p = 0.0051$).

One practical application of this research is extending and refining relatively small translation grammars for resource-poor languages, such as Mapudungun and Quechua, into a major language, such as English or Spanish. Initial experimental results on a Spanish-Mapudungun MT system show that rule refinement operations generalize well to a different language pair and are able to correct errors in the grammar and the lexicon.

.

Dedicat a l'Anna i al Juan Carlos

Table of Contents

List of Figures

List of Tables

Acknowledgements

This dissertation could not have been completed without the guidance and support of many people. First and foremost, I would like to thank Jaime Carbonell for letting me find my own path in the dissertation labyrinth, for supporting my multiple endeavors and for giving me the guidance and inspiration necessary to take this thesis to its conclusion. I would like to thank Alon Lavie for his time and feedback during the past few years, and to the rest of my thesis committee, Lori Levin and Bonnie Dorr, for their invaluable guidance. I would also like to thank Stephan Vogel for the technical discussions and for his support; his collaboration strengthened the decoder results in this dissertation. It has been a great experience working with Stephan and it has given me the chance to take *a walk on the other side*, namely statistical Machine Translation.

Toni Badia introduced me to the field of Computational Linguistics during my Translation and Interpreting university studies in Barcelona; it was in one of the Prolog labs taught by him that I discovered that instead of working as a professional translator I could make machines translate for me! I am forever grateful to him, for opening the door to all my posterior Natural Language Processing endeavors and guiding me through my early steps.

Later, at Carnegie Mellon University, Alan Black took me under his wing. I have fond memories of our collaboration on improving the pronunciation of proper names as well as learning unsupervised text-to-speech models. I am grateful to him for allowing me to spend time creating and implementing the US Pronunciation of Proper Names web site, which is the precursor of the Translation Correction Tool developed in this thesis.

I also wish to thank the Avenue team for providing a great forum of discussion and collaboration (Jaime Carbonell, Lori Levin, Alon Lavie, Robert Frederking, Ralf Brown, Katharina Probst, Erik Peterson, Christian Monson, Alison Alvarez and Roberto Aranovich). I would especially like to thank Kathrin for many valuable discussions, for her constant support and for paving the road for me. I am also indebted to Erik for the implementation and maintenance of the transfer engine. Additionally, I would like to thank William Ridmann for his

help with the implementation of the Automatic Rule Refiner and Kenneth Sin for his help with the implementation of the TCTool.

Many thanks go to my friends and colleagues in Barcelona and Pittsburgh who participated in the English-Spanish user studies.

I am very fortunate to have benefited from the tireless efforts of many administrative assistants and staff at Carnegie Mellon University. I would especially like to thank Radha Rao, Mary Jo Bensasi, Stacey Young, April Murphy, Brook Hyatt and Linda Hager, who assisted me in numerous ways in countless occasions.

Being part of Women@SCS gave me a sense of belonging to the School of Computer Science community at CMU and provided me with plenty of outreach, networking and leadership opportunities. Special thanks to Bernardine Dias, Carol Frieze and Lenore Blum for being such wonderful mentors.

My friends made Carnegie Mellon the most amazing place to earn my PhD, and I owe my inspiration and perseverance to them. Thank you Irina, Sarjoun, Bernardine, Kathrin, Guy, Anya, Alice, Joy, Kornel, Stefanie, Moss, Brett, Katerina, Ayorkor, Vinithra, Carlos and Sarah. It was incredible being surrounded by smart and wonderful people like you.

My Catalan and Spanish friends made Pittsburgh a home away from home, and even though space limitations prevent me from thanking everyone individually here, I would like to mention: Rocio, Fernando, Esther & Jose, Andrea & Benjamin, Fernando & Ana, Esther, Carmen, Silvia, Sandra, Maria, Carlos, Oriol, Eli & Carles, Helena & Aitor and Mamen & Iñaki.

At the same time, and despite the distance, my friends from *back home* managed to always be there for me, and they have shown me that true friendship prevails over time and space. Gràcies Elisenda, Saskia, Gemma, Oriol, Jordi, Euge, Ferran, Pau, Roger, Joaquim and Lluís.

Last and most importantly, I am forever grateful to my family for always being so loving and supportive of everything I do, even if it involves spending many years away from them. I will always be indebted to my mother, Anna Llitjós Viza, for being a great role model and for always encouraging me to follow my own path. I would like to thank my father, Pep Font Cisteró, for teaching me how to face and fight adversity; he made me strong. I want to thank my sister, Mariona, for making me laugh like nobody else in this world can and for making the early years of my life the best. I would also like to thank my aunts Rosa and Helena Llitjós Viza for their love and dedication. The memories of my grandfather, Joan Llitjós Armengou, are always with me and are an endless source of inspiration.

I could not have completed this thesis without the love, support, and patience of my husband, Juan Carlos Rivera Guerrero. Thank you for always believing in me and for being my

source of strength. You made my life as a PhD student a happy and privileged one. I cannot end this section without thanking the person that is growing inside me. Mar, thanks for cheering me up with your soft kicks and movements, especially during the long thesis-writing days. It has been quite an experience; one that I will never forget. Thank you!

Chapter 1

Introduction

This dissertation presents a largely automated approach to generalize post-editing efforts and improve Machine Translation (MT) systems. This is an alternative to traditional automated post-editing approaches, which modify MT output but not the core MT technology, where the MT systems do not learn from their past errors. First, let us address two basic questions: what is Machine Translation? And is it really necessary?

Natural language is ever changing, ambiguous and structurally complex. Yet communication between human beings is mainly achieved using language. If communication occurs between two different languages, then translation from one language (source language) into the other language (target language) is required. Professionally trained translators spend their days struggling to find the precise word or phrase that will best express the exact meaning of the source language text and the intention of the author.

Companies and government institutions have typically hired a large number of translators and post-editors to produce high-quality translations at prodigious aggregate costs (US: $10 Billion in 2000; EU: over $1 Billion in 2006 (EUROPA, 27/04/2006)), producing only limited volumes of translated data. In this new information era, increasing volumes of data need to be translated daily. In this context, Machine Translation emerges as the only feasible answer.[1]

[1] The current cost of a professional translation service is estimated at 20 cents per word, compared to a one-time investment of approximately $15,000 per language pair with a commercially available MT system (IBM Websphere Translation Server). This represents a good investment after 75,000 words. http://www-306.ibm.com/software/globalization/expertise/faq_machinetranslation.jsp (3/21/07)

Machine Translation systems get more translations done than is possible manually, and they can deliver translations instantly for time-sensitive content.[2] In addition to cost, volume and speed, MT systems also beat human translation in consistency; when a term is entered in an MT dictionary, it will typically translate it the same way every time, unlike human translators who may choose different translations at different times (Loffler-Laurian, 1996). Consistency might not be highly valued in literary translation, and rightfully so, but it is extremely important in technical translation.

The incessant globalization trend not only favors Machine Translation, but it crucially relies on it. The number of applications that depend on MT technology to function keeps increasing every day. From international e-commerce and advertising to technical support and correspondence within and among multinational companies (email, chat, etc.), not to mention multilingual information retrieval (Zhu & Wang, 2006).

Nevertheless, if human translation is a daunting task, the goal to develop software that can translate correctly is even more challenging and elusive. To translate adequately, MT systems must encode knowledge of hundreds of syntactic patterns, variations, and exceptions, as well as relationships among these patterns. They must include evolving vocabulary and specific semantic knowledge about the usage patterns of tens of thousands of words. They must accurately identify the parts-of-speech and grammatical characteristics of words that may, in different contexts, be nouns, verbs, or adjectives, each having many possible translations. Translation also requires an ample amount of knowledge about the world, the intent of the communication, and the subject. A human translator prioritizes and selectively applies linguistic rules based on this knowledge. MT software cannot do so, as it would require solving the general natural language comprehension challenge first, an AI-complete problem. Despite all these difficulties, which will probably prevent MT from ever attaining the overall quality of human translation, MT technology has already been established as the only solution to quickly getting rough translations of large quantities of text. Such less-than-perfect MT output allows information gathering from foreign texts (assimilation) so that monolingual speakers can perform an initial screening and find texts that are worthwhile sending to a human translator. Dissemination, on the other hand, requires

[2] A translator can typically translate 250 words per hour; accomplished human translators can manage about 650 words an hour (at most 0.2 words/second). On the other hand, a commercial MT system (IBM WTS) translates from approximately 200 words per second to 500 words per second (720,000 to 1,800,000 words an hour).

translation of texts from English to other languages for end-user consumption (government documentation, health services, customer support, e-commerce, etc.), and will always require post-editing to make MT output accurate and fluent.

The field of Machine Translation has been around for more than fifty years and researchers have come a long way since Weaver's memo in 1947, when inspired by successes in cryptography, he suggested treating translation as a decoding problem.[3] Despite recent advances in the field that have made MT systems available for new language pairs in a matter of months, a few important challenges remain unsolved. In particular, even though the quality of MT output for language pairs with plentiful data has increased substantially in recent years, mostly due to more sophisticated utilization of statistical learning methods and objective evaluation methods, state-of-the-art MT systems have not yet reached acceptable publishable quality (Dillinger & Lommel, 2004; Hutchins, 2001). To this date, MT output of even the best performing commercial and research systems require human post-editing to achieve acceptable standards of accuracy and fluency. Therefore, despite the existence of good MT technology, human post-editing has become the bottleneck for unrestrained and effective globalization (Lommel, 2007). (There are as few professional post-editors to manually correct all the MT output required every hour, as there are professional translators to translate).[4]

The main problem is that post-editing efforts are not reused. Therefore, in spite of many human-hours of work on MT output, MT systems keep making the same mistakes over and over again. Adding specific corrections to a database is straightforward and somewhat useful, but nothing outside these examples is learned by the system. Finding a way to utilize such post-editing information to automatically improve and extend the underlying MT technology, without the need of computational linguists or MT experts constitutes a new promising research direction that deserves attention. Such an approach is particularly relevant for resource-poor scenarios, where there is typically no data or experts available. Moreover, MT systems are only available for languages with abundant resources, and the challenge of how to build MT systems for languages with no parallel or comparable data available has so far attracted little attention.

---

[3] "One naturally wonders if the problem of translation could conceivably be treated as a problem in cryptography. When I look at an article in Russian, I say: 'This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode'." Warren Weaver, March 1947.

[4] Post-editing costs are estimated to range from 1/3 of human translation cost for light, brief post-editing, which involves mostly replacements (See footnote 1) to 2/3 of human translation cost for full post-editing (Martin Roberts, president of Linguistics Systems, Inc., personal communication at AMTA, August 2006).

This thesis introduces an automated method to generalize from specific user corrections from non-experts. In rule-based systems, such corrections can be traced back to specific problematic rules and lexical entries, which can then be modified or augmented to fix errors. This way, post-editing efforts are not lost or diluted; instead, they are generalized to enable MT systems to correctly translate similar examples in the future.

One of the key aspects of this work is that, even though non-expert corrections might not be as reliable as expert feedback, many non-expert bilingual speakers are available even in resource-poor contexts, and our approach can benefit from feedback provided by one, two or multiple non-experts.

1.1 Thesis Statement

This thesis tackles the problem of automatically extending the coverage and improving the accuracy of Transfer-Based Machine Translation systems, given post-editing corrections and some additional error information. We show that non-expert bilingual speakers can identify the presence of an error, correct it and provide useful information for generalizing the correction in a structured way, via a user-friendly online graphical user interface. Such error correction information, which affects both words and alignments, can be used to automatically refine the translation grammar and lexicon, via blame assignment, feature detection and rule modification. Automatic evaluation metrics ensure that automatic refinements improve coverage and overall Machine Translation quality.

1.2 Research Goals

The research described in this dissertation encompasses six main objectives:

- Developing an effective method to elicit corrections, alignments and contextual clues from non-expert bilingual speakers via an online graphical user interface (Translation Correction Tool).
- Developing a functional MT error typology and validating the granularity of MT error classifications suitable for automatic rule refinement.
- Developing an automatic blame assignment process to identify the rule(s) and lexical entry(ies) responsible for the error(s).

- Developing an automatic approach to refine transfer rules and lexical entries based on elicited error information and diagnosed blame.
- Evaluating the performance of the resulting refined grammar rules on unseen texts using standard measures such as BLEU.
- Assessing the language independence of the automatic rule refinement approach by testing the same methods on a radically different language pair, i.e. Mapudungun-Spanish.

1.3 Thesis Summary

1.3.1 Background: Overview of AVENUE

A recent growth of parallel corpora has resulted in a proliferation of corpus-based direct MT systems, such as Statistical MT systems and Example-Based MT systems, that have reached translation accuracy rates comparable to those of traditional Rule-Based MT systems (RBMT), developed over many years by a large number of experts. Indeed, the general feeling in the community is that large quantities of parallel data and co-occurrence frequency information make a fine substitute for linguists and computational linguists.

Such corpus-based systems, however, are only viable for a small number of language pairs where there are large amounts of parallel data. When there is little or no electronic parallel text available, only Rule-Based MT systems are viable. On the other hand, traditional RBMT systems can also be cost prohibitive, since they require computational linguists with knowledge of the languages involved to write, test, refine and extend translation rules. Moreover, in resource-poor contexts, there might be very few or no native speakers with computational linguistics expertise. Such resource-poor scenarios require a different approach.

The AVENUE project at Carnegie Mellon University is devoted to the rapid and affordable development of MT Systems for resource-poor languages, such as Mapudungun (spoken in Chile and Argentina), Quechua (spoken mainly in Peru, Bolivia and Ecuador) and Inupiaq (spoken in Alaska).[5] An automatic rule learning approach infers structural mappings between the two

[5] The National Science Foundation has supported AVENUE under grant IIS #0121631, formerly called NICE (N66001-99-2-891804) and currently continuing as LETRAS (IIS #0534217).

languages involved in the translation process from a limited amount of parallel text. An Elicitation Corpus was initially designed (in English and Spanish) to cover a wide range of linguistic phenomena (on the order of magnitude of a few hundred sentences; Probst et al. 2001), and has recently been redesigned so that it can be generated dynamically, according to the phenomena already detected for a particular language pair (Alvarez et al., 2005; Levin et al., 2006). In the AVENUE approach, the Elicitation Corpus (EC) is typically given to a small set of native speakers of the resource-poor language to translate. Next, the translated EC is used to build MT grammars, either manually (Font Llitjós et al., 2005a) or automatically (Probst, 2005). Being able to learn grammar rules automatically is key, since it allows for building an MT system for a new language pair, even when large amounts of parallel data and native technical experts are lacking. The translation quality of such systems, however, is still below that of hand-crafted MT systems (Lavie et al. 2003). In this context, automatic refinement of translation rules becomes a crucial component of MT.

MT errors tend to be repetitive, which makes post-editing of MT output a rather tedious task, as well as one that is suitable for automation. This dissertation advances the state of the art in minimal post-editing, where the number of changes to MT output to make it acceptable is as small as possible (Allen, 2003). Instead of requiring professional post-editors to revise and amend all MT output, we introduce an online tool that allows non-expert bilingual speakers to minimally correct part of the MT output and reuse such corrections to fix similar errors in the future. The main insight is that correction information and additional error information available to the MT system can be used to automatically retrieve the rules that are responsible for the error (blame assignment) and modify them (rule refinement), to prevent this type of error from ever occurring again. Beyond just improving MT output, this method goes to the core of the problem by extending and improving the underlying MT technology in use.

If an MT-produced translation is incorrect, a non-expert bilingual speaker can easily identify the presence of an error (Font Llitjós & Carbonell, 2004). However a non-expert informant cannot determine which translation rules produced the error and even less how to improve such rules. The objective of this work is to automate the Rule Refinement process based on just explicit *error-locus* and implicit *error-type* information from non-expert bilingual speakers. Figure 1.1 shows the interaction between an MT system (with its grammar and lexicon) and the Automatic Rule Refinement system, which includes the Translation Correction Tool (TCTool) and the Automatic Rule Refiner. The next sections summarize them in turn. Note that while a computational linguist is typically required at the beginning of the process to write the

seed grammar rules (bottom of Figure 1.1), non-expert bilingual speakers suffice to provide corrections of MT output (upper right corner).

Chapter 2 describes related work on expert knowledge elicitation, automated post-editing, MT error classification and automatic rule adaptation. It concludes with recent research on learning corrections of MT output.

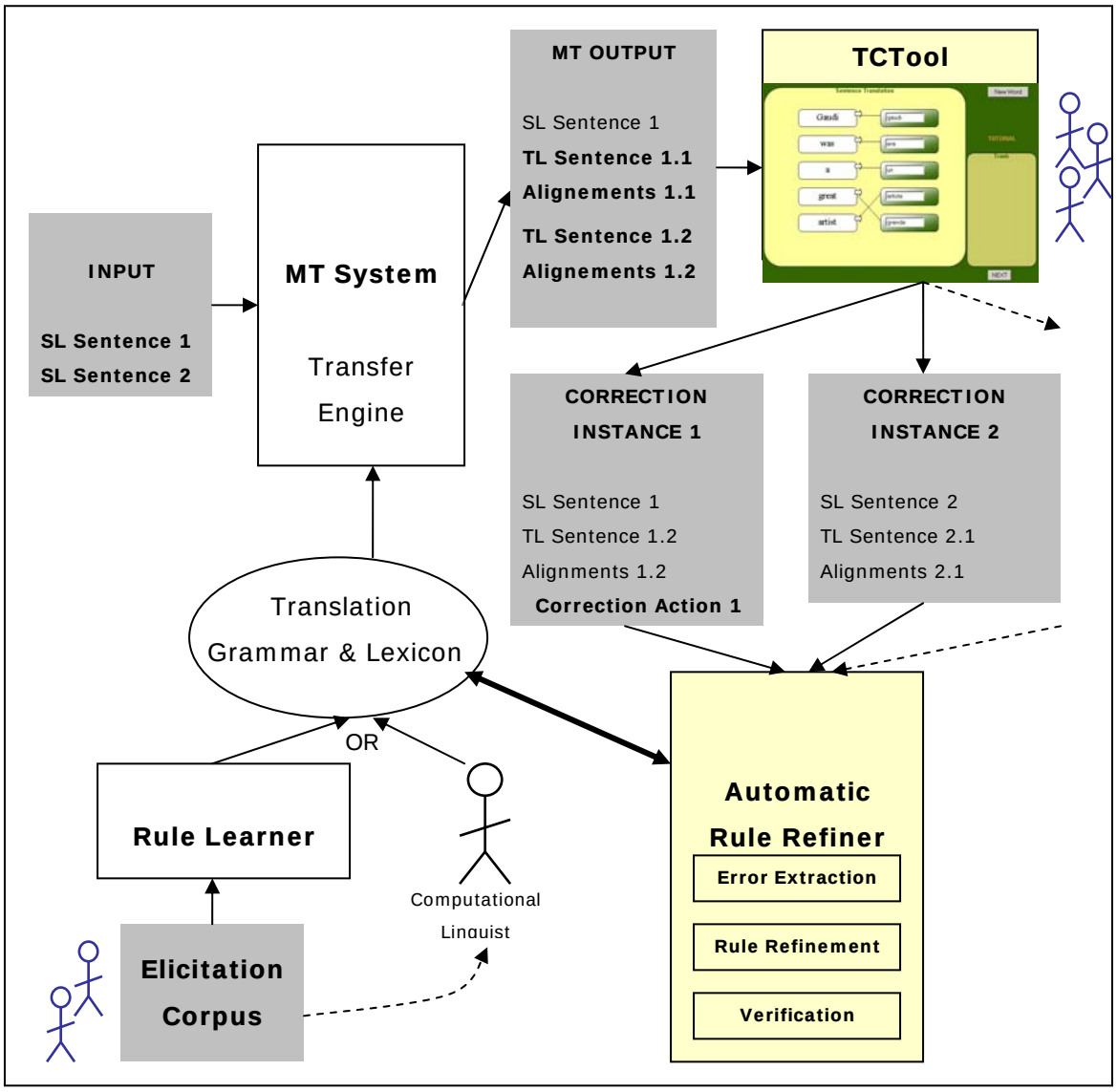


Figure 1.1. Simplified Data Flow Diagram illustrating the interaction between a Machine Translation system and the Automatic Rule Refinement system.

1.3.2 Elicitation of MT Error Information from Non-experts

The automatic rule refinements proposed by this work are guided by post-editing information obtained from non-expert bilingual speakers. To gather such error correction information reliably, we designed, implemented, and tested an online graphical user interface, called the Translation Correction Tool (TCTool). The TCTool allows bilingual informants to easily and reliably correct machine translations by adding, deleting or modifying words and word alignments, and changing their order in the translation, as well as optionally identifying clue words (Section 3.4).

As shown in Figure 1.1, given a set of source language (SL) sentences translated by the MT system, the TCTool takes the MT output containing a set of target language (TL) sentences and the alignments between source and target words (extracted from the grammar rules) and presents them to bilingual informants. The user-friendly GUI allows non-expert users to minimally correct the TL sentence by clicking on the words and alignments that they want to modify and by dragging and dropping them around, simulating a game. For each SL sentence corrected by bilingual informants, the TCTool stores a log file (Correction Instance) containing all the MT output information plus the set of Correction Actions performed by informants, as well as the Corrected TL sentence (C_TL) and the final word alignments (C_Alignments).

For example, given the SL sentence "*I saw the woman*", the AVENUE English-Spanish MT system outputs the following TL sentences: "*vi la mujer*" and "*yo vi la mujer*" with alignment information. Both Spanish translations are missing an animate direct object marker (*a*), and so bilingual informants will select either of them for correction and will add *a* as a new word preceding "*la mujer*" to obtain the correct Spanish translation of the SL sentence (i.e., "*(yo) vi a la mujer*"). The Correction Instance resulting from this user interaction is shown in Figure 1.2.

SL: I saw the women
TL: vi la mujer
Alignments: ((1,1),(2,1),(3,2),(4,3))
Correction Action 1: add a word (W_2'=a)
CTL: vi *a* la mujer
CAlignments: ((1,1),(2,1),*(3,3),(4,4)*)

Figure 1.2. Example of Correction Instance for the English-Spanish MT system.

In comparison with English, Spanish is a highly inflected language and has a more complex verb tense system (17 verb tenses, not counting impersonal forms like progressive), with no one-to-one mapping to the English tenses, especially for the past and the subjunctive tenses. Tense errors are thus common when translating English to Spanish. Furthermore, Spanish has

agreement between nouns and their modifiers (determiners and adjectives), some case marking and different word order than English.

Chapter 3 introduces the MT error typology that underlies the TCTool and the Automatic Rule Refiner, and describes the design and implementation of the TCTool (Section 3.2). The effectiveness of the TCTool to elicit MT corrections from non-expert informants is validated by an English-Spanish user study (Section 3.3).

1.3.3 Automatic Refinement of Translation Rules

Researchers have explored a variety of methods to include user feedback in the MT process. However, most MT systems have failed to incorporate post-editing efforts beyond the addition of corrected translations to the parallel training data for Statistical and Example-Based systems or to a translation memory database. In our approach, precise error correction information that is relevant to the system allows the Automatic Rule Refiner (ARR) to trace the errors back to incorrect lexical and grammar rules responsible for the errors and to propose concrete fixes to such rules. For the most part, refinements involve adding morpho-syntactic information to existing rules.

Correction Instances output by the TCTool, such as the one shown in Figure 1.2, allow the ARR to propose modifications that result in expansions and improvements of the grammar and the lexicon, yielding an improvement on overall translation quality of the MT system, even on unseen data. The ARR can automatically add missing lexical entries, structural modifications of existing grammar rules, and fix incomplete or incorrect rules that applied during the generation of MT output.

First, the ARR parses and stores Correction Instances for specific translation pairs as provided by one or more bilingual speakers. Next, it proceeds to perform blame assignment based on the translation tree produced by the MT system (see Figure 4.13 for an example). At this stage, the system retrieves the error-causing rules and lexical entries and it proposes specific refinement operations. For the example introduced above, the ARR will determine that the verb phrase rule has to be refined so that the grammar also allows Spanish direct objects to be preceded by the preposition *a* (Section 4.7.1).

The ARR comprises an expandable set of refinement operations to modify inaccurate translation rules in both learned and hand-crafted grammars and lexicons. Each correcting action allowed by the TCTool corresponds to a set of refining operations that perform the appropriate modifications in the translation rules, according to the error information available. In a nutshell,

the ARR uses the error information available to determine whether it is necessary to add a lexical entry, modify a current lexical entry, bifurcate a rule and modify the copy, usually making it into a more specific rule, or constrain a rule that is too general.

Chapter 4 introduces the Rule Adaptation Framework, the main Rule Refinement operations and how they can be combined to form rule refinement schemata (Section 4.3) (Font Llitjós et al., 2005b). In Section 4.5, we describe the ARR algorithm in detail and illustrate it with examples for the four main error cases (Font Llitjós & Ridmann, 2007). We end with discussion and conclusions.

1.3.4 Summary of Results

Our approach to improving MT output attacks the problem at its core, generalizing beyond the input sentences corrected by bilingual speakers, thus allowing for correct translation of unseen data. Evaluation results on an English-to-Spanish Transfer-Based MT system show that by applying automatic refinements, higher translation accuracy can be achieved as measured by standard automatic evaluation metrics.

After the refinement process is complete, the ARR has to make sure that, given the original source language sentence, the MT system with the refined grammar and lexicon produces the translation as corrected by the bilingual informant, and that changes in the translation rules increase translation quality over a large enough regression test set.

In particular, we test our Automatic Rule Refinement approach on its ability to improve coverage and overall translation quality over unseen data as measured by automatic evaluation metrics such as BLEU (Papineni et al., 2001) and METEOR (Banerjee & Lavie, 2005). In our experiments, we found that an automatically refined English-Spanish Transfer-Based MT system outperforms the baseline system, after the ARR has processed just a few Correction Instances (Section 5.4.3.4, Table 5.13). Improvements on translation quality over the baseline on unseen data are statistically significant on a paired two-tailed t-test ($p=0.0051$) (Section 5.4.4).

Chapter 5 encompasses multiple experiments on three different data sets, where two different size Correction Sets are used to obtain a refined MT system (Font Llitjós et al., 2007). Chapter 6 describes the issues faced when porting the Automatic Rule Refinement system to a different language pair, namely, Mapudungun-Spanish. A small scale experiment that successfully corrects the grammar and extends the lexicon provides initial evidence to advocate for the language independence of our approach.

1.4 Thesis Contributions

This dissertation introduces a general framework that provides a complete prototype solution to automatic rule refinement in Rule-Based MT systems, which consists of the following:

- A system and language independent online GUI to display Machine Translation output and solicit pinpoint fixes from non-expert bilingual users. This constitutes a new method to generate manually corrected and aligned bilingual corpora.

- An MT error typology and the validation of the appropriate amount of error information that can be provided by non-expert users reliably, abstracting away from linguistic and technical translation issues.

- A mapping between error corrections and rule refinement operations.

- A language independent mechanism to extend and improve translation grammars and lexicons with an expandable set of rule refinement operations and schemata.

- A mechanism to evaluate the effect of automatic rule refinements.

- A correction instance management system, able to keep different candidate rule refinements for later confirmation or rejection.

- Improvement of MT output quality on unseen data, as measured by standard automatic evaluation metrics (BLEU, NIST and METEOR).

- Confirmation of language independence of automatic rule refinement approach.

Chapter 2

Literature Review

In Chapter 1 we discussed the need for accurate Machine Translation. In this chapter, we first review translation productivity tools, such as memory translations, and expert knowledge acquisition approaches, and then we review past work of relevance to different parts of our automated rule refinement method. Figure 2.1 illustrates different research areas relevant to the approach presented in this dissertation that are discussed in this chapter.

In order to improve Machine Translation quality, there are at least three approaches one can take. The first option is to modify the input of an MT system in order to make it more MT friendly (pre-editing), for example by splitting a long sentences into shorter, easier to translate chunks that, together, preserve the meaning of the original sentence (Mellebeek et al., 2006). A second option is to modify the MT output in order to correct all the errors introduced by the automatic translation process. This activity is widely known as post-editing and is common practice at translation services and companies that use MT. Pre and post-editing are not necessarily independent and are often used in conjunction. Krings (2001) confirmed that improvements in machine-translation quality by simplifying the linguistic input into the MT system has a direct influence on the post-editing processes.

In addition to pre and post-processing of MT output, the third option is to look into the cause of occurring errors and try to fix the source of the problem within the MT system itself. Most MT system developers and researchers devote much of their time working on precisely that; trying to find what caused the problem and fixing it so that the MT error does not occur again. However, not much attention has been devoted to automating such detection and remediation, with the exception of some automated post-processing to alleviate the tedious task of manual post-editing by automating the correction process for the simplest and the most frequent errors beforehand (Allen & Hogan, 2000; Knight & Chander, 1994). In this thesis, we describe one way

to automate the error detection and remediation process of the underlying MT system with the aid of bilingual post-editing information from non-experts.

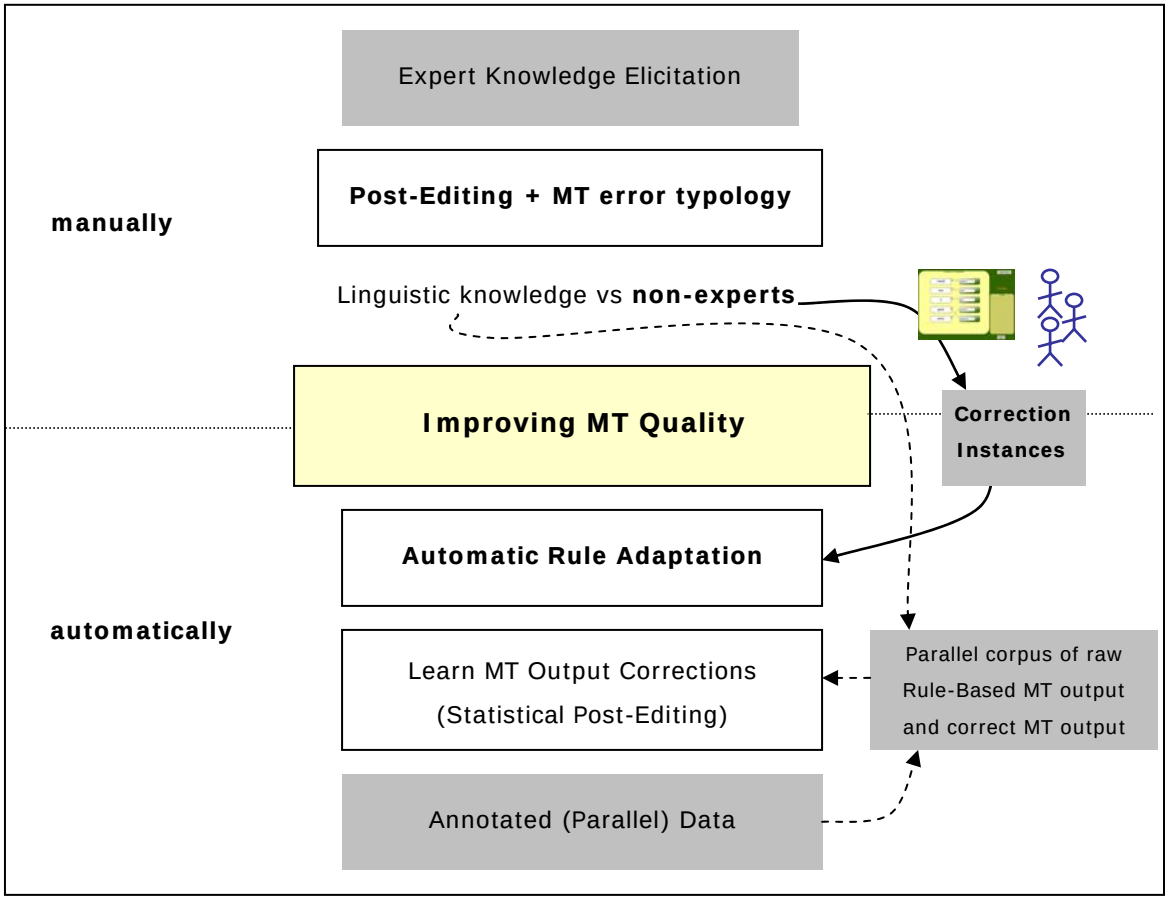


Figure 2.1. Diagram showing related areas of research relevant to this dissertation.

2.1 Translation Productivity Tools: from Translation Memories to EBMT

Translation memories are a type of computer-assisted translation (CAT) tool, one that enjoys huge popularity among professional translators, as they ensure that no sentence ever needs to be translated twice. Translation memories increase accuracy while dramatically slashing globalization times by recycling up to 80% of previous translation work when translating new versions of the same documents (e.g., the manual for software 3.0 after having the manual for software 2.0 loaded in a translation memory). Smaller improvements in productivity are also recorded when translating more dissimilar texts (SDL & TRADOS, 2005).

The concept of a Translation Memory (TM) has been around for more than twenty years, but only recently has it become a significant commercial entity.[6] A TM is a system which scans source text and tries to match strings (a sentence or part thereof) against a database of paired source and target language strings with the aim of *reusing* previously translated materials. Some translation memories attempt only literal matching, i.e., can only retrieve the exact match of a sentence, while others employ fuzzy matching algorithms to retrieve similar target language strings, flagging differences. The flexibility and robustness of the matching algorithm largely determine the performance of the system, although for some applications (i.e., highly repetitive material) the recall rate of exact matches can be high enough to justify the literal approach. Translation memories are typically integrated into translation workstation packages, where they can be used in tandem with a terminology management system, a multilingual dictionary, and even raw MT output.

Despite being widely used in localization, most TMs cannot create new translations different from previously existing examples in the system database; rather, the closest translation matches are proposed to the user for post-editing into the correct translation. For readers more familiar with Example-Based Machine Translation (EBMT) systems, TM is a special case (Brown, 1996; Hodász et al., 2004; Somers, 2001). Namely, TM typically has storage and retrieval functions (exact match) and leaves adaptation to human translators. Adaptation is in fact the most taxing aspect of the EBMT paradigm, since it is difficult to derive any knowledge about how to adapt retrieved translations from examples alone. In the EBMT paradigm, adaptation requires general rules to combine partial transformations from multiple examples. EBMT systems including such rules are often referred to as Generalized EBMT, and are a hybrid between EBMT and RBMT (Brown, 1999; Brown, 2000; 2001; Somers, 2001).

The line between TM and EBMT has in reality also become rather blurry, since many TM systems now allow phrase matching in addition to full-sentence matching (Planas, 2000). Some researchers have suggested that a TM-EBMT hybrid approach outperforms either of the single systems (Carl, 2000; Carl & Hansen, 1999; Schäler, 2001).

---

[6] Commercial translation memory systems include CATALYST (Corel), Déjà Vu (Atril), Eurolang Optimizer, IBM Translation Manager (IBM), Trans Suite 2000 (Cypresoft), IBM TransLexis, Loc@le 2.0, MetaTexis, MultiTrans of MultiCorpora, Passolo, RC-WinTrans Lite, SDLX 2.0 Interactive Translation System (SDL, TRADOS), TR-Aid, TRANSIT & TermStar and Translator's Workbench (TRADOS).

In our work, we introduce an approach that allows for automatic adaptation of translation rules from post-editing feedback given by non-expert bilingual speakers (Section 2.3). In the next section, we discuss different methods that have been proposed to acquire expert knowledge.

2.2 Expert Knowledge Acquisition

Knowledge acquisition includes the elicitation, collection, analysis, modeling and validation of knowledge for knowledge engineering and knowledge management projects. Expert knowledge has typically come from two main sources: expert users and annotated data. More recently, researchers have found ways to also elicit knowledge from non-experts, via user-friendly interfaces (plus mappings of simple sentences or user actions into underlying system representations that render themselves to automatic learning) and from large amounts of naturally occurring data (unannotated or unlabelled data). In this section, we describe some related work with varying amounts of annotated data and of expert user intervention.

Delisle and colleagues (1994) describe a system that extracts knowledge, in this case semantic interpretations, from surface syntax of technical English texts. The linguistic component of the system uses a broad-coverage, domain independent parser of English, as well as a user-assisted semantic interpreter that memorizes its experience. The resulting semantic structures are translated into Horn clauses, a representation suitable for Explanation-based Learning (EBL). An EBL engine performs symbol level learning on representations of both the domain theory and the example provided by the linguistic part of the system. Their approach has been applied to the Canadian Individual Income Tax Guide.

Delisle and Szpakowicz (1997) extract predicate-argument structures from a single text with a substantial narrative part. Working with such texts rather than large corpora requires detailed syntactic analyses, a learning mechanism and a cooperating user who confirms automatically generated results. The goal of their work is to reduce user's participation gradually, as the system learns from preceding analyses. Systematic performance evaluation of their text analysis system (TANKA) showed that their supervised text analysis procedures allow the system to learn to make better analyses, that knowledge acquisition is possible even from erroneous or fragmentary parses and that the process is not too onerous for the user (Barker et al., 1998).

Within the area of linguistic knowledge acquisition and representation, there is an extensive literature about bootstrapping from annotated language data (Ayan & Dorr, 2002; Belz, 2002; Brewster et al., 2001; Brill, 1993; Jiménez, 2002; Xi & Hwa, 2005). On the other hand, in recent years, an ever growing number of researchers have started to exploit large quantities of unlabelled data with unsupervised learning techniques (Clark et al., 2003; Meng & Siu, 2002;

Smith & Eisner, 2004; Turian et al., 2006). Such is the case of Machine translation and the availability of large amounts of parallel data. Two ubiquitous examples of this can be found in automatic word and phrase alignment techniques, which are used to build bilingual translation lexicons (Brown et al., 1990; Koehn, 2004; Koehn et al., 2007; Lopez et al., 2002; Och & Ney, 2003; Vogel, 2005), and in *n*-gram language models, which are widely used to model word order for the target language (SRI, 2006; Zhang & Vogel, 2006). Such corpus-based approaches are not viable in resource-poor language contexts, however, as there are typically no large amounts of data available.

Several systems leverage expert user feedback to elicit the knowledge required by intelligent systems (Brewster et al., 2001; Dupplaw et al., 2004; Kim, 2005; Kim & Gil, 2003). Nevertheless, for our task of translation from resource-poor languages, where there might not be any experts available, the real challenge is to find a way to elicit expert knowledge from end users, who need not be experts in the domain and are typically not programmers.

For expert knowledge elicitation systems to learn effectively without shifting the entire burden to the user or requiring end users to become domain and programming experts, there are two constraints that need to be observed. Namely, providing an interface that both (1) allows non-expert users to express what they mean in simple terms (mostly informal, free text), and (2) is sufficiently structured to allow machine processing and thus learning to occur (Chklovski et al., 2005; Font Llitjós, 2006). The problem faced by our approach is precisely to find the right balance between what and how information is elicited from users without requiring any technical terms or complicated procedures, while enabling the underlying system to learn from it. This aspect of our work is addressed in Chapter 3.

One approach that has been adopted to solve this problem is to build systems that provide strong guidance to end users, so that it becomes intuitive and easy to add new knowledge to intelligent systems (Blythe & Gil, 2004; Blythe et al., 2001). A different approach is to allow end users to enter short, simple sentences that are mapped by the system into valid, plausible modifications (Blythe, 2005). An interesting example of such approach is the one adopted by Learner, a system that interactively acquires knowledge about the everyday world objects from volunteers (Chklovski, 2005). Everyday facts are knowledge that computers do not have and that are not easily obtainable with current text extraction methods. An Artificial Assistant for To-Do lists based on these everyday facts has recently been implemented (Gil & Chklovski, 2007).

Simplifying the elicitation task to the extent that even non-experts can provide the system with learnable information, however, is a very challenging task. The BOAS project provides an example of the difficulty of maximally simplifying the elicitation task to build Machine

Translation systems. The goal of the BOAS project is to elicit linguistic knowledge from informants who are not trained linguists, so that the morphology module, lexicon and grammar of a basic MT system can be derived from it (McShane et al., 2002; Nirenburg & Raskin, 1998). Nevertheless, since their elicitation method still requires significant linguistic knowledge, they provide informants with a glossary and examples to learn about linguistic phenomena, so that they are able to provide information about features of their language (whether the value of number of a specific nominal expression is *singular, plural, dual* or *paucal*, for example).

The work presented in this dissertation is driven by a similar question, but the difference is we do not require bilingual informants to have any formal linguistic knowledge. In fact our elicitation process abstracts away from the intricacies of the underlying MT system and allows users to concentrate on error detection and remediation. In the next section, we review related work relevant to the question of how to elicit information about MT output correctness from non-experts.

2.3 Non-Expert Bilingual Post-Editing

Post-editing has been defined as the correction of MT output by human linguists or editors (Allen, 2003; Veale & Way, 1997). However, many MT errors can also be corrected with high precision (90%) by bilingual speakers who are not linguists, editors or translators, as shown in an initial set of user studies (Font Llitjós & Carbonell, 2004).

Post-editing information can simply be added to augment the example database of a translation memory or an EBMT system (Section 2.1), but more interestingly it can also be used to identify the source of the error in the underlying system representation, in our case, translation rules, and to guide the automatic correction of such problematic rules.

Allen et al. (2000) proposed an automated post-editing (APE) prototype module which is meant to automatically fix the highly frequent, repetitive errors in raw MT output before such text is given to human post-editors, in order to speed up their work. The inspiration behind this work is very similar to ours in that they also react to the observation that if an MT system makes a particular error when translating a document, it is very likely to commit the same error each time the same set of conditions are presented. And if the error is fixed in a similar way, then it is possible to capture these modifications and to implement them automatically so that such repetitive errors can be reduced in MT output (Allen, 2003). The advantage of automatic post-editing is that it is system independent. In comparison with the Automatic Rule Refinement approach described in this thesis, however, if two rules are incorrect but are easy to fix

automatically, the combinatorics of the interaction of such rules might result in hundreds or thousands of sentences that need to be post-edited (automatically or manually).

Our approach was originally motivated by the problem of improving inaccurate or partial MT systems in resource-scarce scenarios, where experts who can improve MT systems or even MT output are often not available. For this reason, our solution does not require experts to provide post-editing feedback nor post-editing rules to be applied automatically. The intended users of the online post-editing tool, known as the Translation Correction Tool (TCTool), are non-expert bilingual speakers, and their goal is to minimally edit MT output to make it an acceptable translation of the source language sentence. This is similar to what has been referred to as *minimal post-editing* in the literature (Allen, 2003), which focuses on adequacy. The minimal correction method we are proposing for the task of rule refinement involves fluency (grammar correctness), in addition to meaning preservation. Stylistic changes are not considered minimal post-editing.

Systran, a long-established commercial MT company, has opened up its system to customers, and now allows users to directly manipulate even linguistic rules (McClure & Flanagan, 2003). On the other hand, Seneff and colleagues (2006) have allowed developers to manually provide a small set of domain-dependent "frame rewrite rules" to specify a reorganization of their interlingua representation for the flight domain when translating Chinese to English. However, since linguistic rules are at a level of abstraction that most inexperienced users do not want to deal with, in our work, we ask bilingual informants to perform a much easier task instead: correct translations that have been generated by an MT system. The Automatic Rule Refiner is the one in charge of executing linguistic rule manipulation inside the MT system.

2.3.1 Elicitation over the Web

Online cooperation of users to improve existing databases has recently experienced a boom, and is just another instance of the new trend that gives the power to create and shape Internet content to end users, also known as Web 2.0. Relevant examples of this phenomenon are Wikipedia, LEO dictionary and the Rosetta Project. It is not hard to imagine a setting where translation users on the WWW feel inclined to improve MT output using an online user-friendly tool like the one described in Chapter 3, so that next time they use the system to read foreign news or travel abroad, for example, they are able to obtain better translations.

On the other hand, online games have also proven to be a great source for gathering key information from users to solve computationally complex problems, such as labeling images (von Ahn & Dabbish, 2004). We have already started working on an online Translation Game that

allows users to correct and validate MT output (Section 7.1.5). If such a game is made available through a major web portal, this will result in a unique and extremely valuable collection of annotated data, which, at a large scale, can be used to improve not only Rule-Based MT systems, but also Statistical MT systems. Such an online game can also be very appealing to second language learners, who can test their skills by trying to correct MT output, and have the system score their performance and encourage them to try more sentences.

2.4 MT Error Typologies

There are several ways to classify translation errors, but most translation error typologies described in the literature are either too fine-grained or too coarse for the purpose of Automatic Rule Refinement. Moreover, existing MT evaluation methods focus on system comparison and on ways to measure translation quality from an end-user viewpoint (Flanagan, 1994), or by developers to be used as a reference for manually modifying the grammar (White et al., 1994) or tweaking a few system parameters. Much like White's approach, our MT evaluation method needs to capture translation adequacy as well as fluency. However, our approach does not assume any linguistic expertise from evaluators (bilingual informants).

Loffler-Laurian (1996) classifies post-editing errors into ten classes for English↔French MT output (Systran's), namely: *vocabulary and terminology*, *acronyms and proper names*, *prepositions*, *nominal and verbal modifiers*, *verb forms (tense)*, *voice (passive, impersonal)*, *presence or absence of modals*, *negations*, *order of word*, and *complementizers*.

General Motors, in their project on Controlled Automotive Service Language, used minimal post-editing following the Society for Automotive Engineering (SAE) J2450 standard metric for translation quality (SAE, 2001). This standard specifies the following categories of errors, which are rated as unacceptable in human translated texts: *wrong term*, *syntactic error*, *omission*, *word-structure* or *agreement error*, *misspelling*, *punctuation error* and *miscellaneous error* (Shäfer, 2003).

MT researchers have proposed methods and tools for automatic evaluation. Niessen and colleagues (2000) describe a tool to facilitate access to an MT evaluation database. Their MT evaluation is based on edit distance (number of insertions, deletions and substitutions) as well as a predefined set of error classes (*missing*, *syntax*, *meaning*, *other*), and the application of different evaluation metrics automatically. This tool makes MT evaluation more consistent over time and it can also be used to predict human evaluation judgment. Other automatic evaluation methods are described in Section 5.3.2.

Marrafa and Ribeiro (2001) propose a more fine-grained error classification (for English, French and Portuguese), taking as a starting point the ISLE framework (ISLE, 2000; 2004) and focusing on system external characteristics at the sentence level. This classification summarizes several linguistic aspects that need to be taken into account for error identification. However, it proves to be too fine-grained for the purpose of interactive and automatic rule refinement.

In general, error typologies currently used to classify MT errors in English contain in the order of 21-26 error types (Secară, 2005). Bredenkamp and colleagues (1999) start with a hierarchical fine-grained typology inspired in (Crysmann, 1997) to annotate a German corpus for the development of controlled language and grammar checking applications, but need to narrow it down to 16 error types to facilitate and seed up annotation (*morphological errors*, *syntax errors* such as *word order*, *categorial*, *case*, and three classes for a*greement errors*; *syntactic-semantic selection* (as for fixed verbal structures), four types of *orthographic errors*, *other syntax errors*).

Correa (2003) introduces a framework especially designed to aid MT system development. He proposes nine different features to describe MT errors (i.e., input segmentation error, segmentation error, markup error, unknown word, name entity error, source analysis error, target lexicon error, target grammar error and target style error). Evaluation results on an English-Spanish MT system (McCord, 1989; McCord & Bernth, 1998) showed that 45% of all errors were due to incorrect target language lexical selections and 40% due to incorrect target language grammar. English-Spanish is also the main direction of the work described in this thesis, and the main focus of our automatic repairs are indeed the target language lexicon and grammar. Given that Spanish is more highly inflected than English, though, having such a coarse error taxonomy is not informative enough for our purposes. In the error typology given in Chapter 3 (Section 3.1), we distinguish between unknown words and sense or forms errors as well as between agreement and word order errors. Our error taxonomy can be seen as a practical approach to classify MT errors that are informative and relevant to automatic refinement of MT systems.

With the goal to design an automated MT evaluation system to help developers, as well as users, Elliott and colleagues (2004) proposed a hierarchical classification scheme of fluency errors in English output and show evaluation results for four French to English systems (Systran (Dugast et al., 2007), Reverso, Comprendium and SDL's online FreeTranslation) on TEMCATE (a TEchnical Corpus for MAchine Translation Evaluation). Their main goal is to devise an evaluation method that does not require human reference translations, and thus they only measure adequacy indirectly from target language fluency. Error types were divided according to parts-of-speech and correspond to items that a post-editor would need to amend if s/he were revising the texts to publishable quality. From their 58 main categories, they found eight error types to be the

most common (*inappropriate V | N | PREP | ADJ | NN string content*, *incorrect verb tense/mood*, *incorrect compound NN sequence* and *unnecessary determiner*) for all four systems.

The idea of devising an MT Evaluation Framework only requiring monolingual competence of English, as measured by the amount of post-editing necessary to make MT output correct, has been followed up in GALE (Olive, 2005; Snover et al., 2006).

Back in 1998, Povlsen and colleagues (Povlsen et al., 1998) report an evaluation of how well a specific MT system, in this case PaTrans for English-Danish (Maegaard & Hansen, 1995), would perform in translating new text-types including an assessment of methods to extend the system so that it can deal with new text-types. To our knowledge, they were the first to define evaluation and quality criteria in terms of how much effort it takes to post-edit the text after being automatically translated. For our approach, the most relevant part of the work they report is the use of informal non-technical language to describe error types to post-editors (Lingtech, 1996; 1997). The error types ranged from "*totally incomprehensible and messy word order at sentence level*" to "*missing words*". In order to use the translation error types in evaluating the suitability of a source text, error types were translated by expert MT system developers into phenomena occurring in the source text that give rise to these translation problems (source of the errors). Evaluation results allowed Lingtech to (manually) improve the MT system targeting most serious errors first.

Our initial approach to automatic refinement of translation rules was to request as much information as possible about MT errors from users. However error classification, based on linguistically motivated classes, is harder for non-expert users than error detection (Font Llitjós & Carbonell, 2004). Hence, we devised a non-technical method to elicit post-editing information from non-expert bilingual speakers that allows them to classify MT errors reliably and with high precision. In our approach, post-editing feedback does not rely on technical terminology, or knowledge of linguistics, but rather it is based on what word (or set of words) in the translation provide a clue that indicates to users that they need to correct it (Section 3.4).

Vilar and colleagues (2006) adopted the error typology proposed in (Font Llitjós et al., 2005b) for their statistical Machine Translation system (RWTH, (Vilar et al., 2005)) in the context of the first TC-STAR evaluation (speech-to-speech translation for real life data) for the following three translation directions: Spanish to English, English to Spanish and Chinese to English. In their typology, first level errors fall into one of these five error classes: *missing words*, *word order*, *incorrect words*, *unknown words*, and *punctuation*. Figure 3.1 shows the original error typology proposed in (Font Llitjós et al., 2005b), from which it can be noticed that their typology adds and punctuation errors, but omits agreement errors.

Researchers at Systran (Dugast et al., 2007) have recently classified the types of post-editing changes made by a statistical post-editing (SPE) component into six main classes (*Lexical changes*: words not translated by Systran generating a translation with SPE, slight terminology change with POS and meaning preservation, multiword expression/locution change, lexical modification altering the meaning of the sentence; *Grammar changes*: determiner, preposition, pronoun, verb tense, number/gender, other; *Punctuation/digit/case changes*; *Local word order changes*; *Long distance word order changes*, and *Stylistic changes*). They found that lexical changes with meaning preservation (local choice of word sense or alternative translation of words and locutions) are the ones most improved by the addition of a SPE component. On the other hand, lexical changes that altered the meaning were responsible for most of the degradations (see Section 2.7 for more details). It is interesting to note that, except for the punctuation and stylistic changes, all the different types of MT error corrections are included in the MT error typology described in Chapter 3.

Finally, the error typology developed in our work includes Ayan's error typology of errors made by automatic alignment systems (2005), even though they were independently developed. His work focuses on fixing *missing and extra alignments* as well as *multi-word alignments*. Our typology is more detailed and, in addition to alignment errors, it includes word and order errors and has a rule refinement application that goes beyond Ayan's work.

2.5 Automatic Rule Adaptation

The idea of rule adaptation to correct or expand an initial set of rules is an appealing one. Researchers have looked at rule adaptation for several natural language processing applications.

Lin and colleagues (1994) researched automatically refining models to decrease the error rate of part-of-speech tagging. They used a probabilistic classification model based on selected discriminative features (which include lexicalized context). Model refinement reduced the error rate of the top 10 error dominant words from 5.71% to 4.35%, showing a 23.82% improvement over the unrefined model.

Lehman (1989) worked on adaptive parsing, more specifically on gradual augmentation of a kernel grammar to include each user's preferred forms of expression, when communicating with a computer. In her work, the existing grammar is assumed to be correct and it is subsequently expanded through interactions with users to learn their idiosyncratic style.

In a similar line of research, Gavaldà (2000) provided a mechanism that enables a non-expert end-user to dynamically extend the coverage of a natural language understanding (NLU)

module, just by answering simple clarification questions. In contrast to our work, Gavaldà relied on non-expert users to automatically learn new *semantic mappings* for his NLU system.

Brill (1993) introduced a technique for parsing free text: a transformational grammar is automatically learned that is capable of accurately parsing text into binary-branching syntactic trees with non-terminals unlabeled. The system learns a set of simple structural transformations that can be applied to reduce error. Brill's method can be used to obtain high parsing accuracy with a very small training set. Although small, the learning algorithm does need the training corpus to be partially bracketed and annotated with part-of-speech information, often not available for resource-poor languages. Even if we had such a small initial annotated corpus, transforming translation rules is non-trivial and cannot be done with simple patterns like the ones proposed in Brill's method. Recent transformation-based learning work to improve MT is discussed in Section 2.7.

Attardi and Ciaramtia (2007) have proposed a tree revision learning approach to correct the output of a dependency parser. Revisions are defined as combinations of atomic moves on a dependency graph and are trained using a discriminative classifier. They report relative error reduction of 11% for Swedish and 16% for English (WSJ Penn Treebank) after applying the revision approach.

2.6 Improving Machine Translation Systems

In the field of Machine Translation researchers have also been searching for the holy grail of automatic improvement of MT systems.

In 1988, Nishida and colleagues described a Post-Editing Correction information Feedback system (PECOF) in its early stages, which attempts to improve a Transfer-based MT system. There are several differences between their approach and the one described in this thesis. The main differences are: 1) the use of expert post-editors, whose work is not only to correct MT output but also to formulate correcting procedures corresponding to unseen error patterns, which are then executed by the PECOF system, and 2) the use of two MT systems in order to detect discrepancies between intermediate representations of the source language and the target language side, namely an *original* MT system (Japanese to English) and a *reverse* MT system (English to Japanese) which is applied to the post-edited English translation (Nishida & Takamatsu, 1990).

The grammar rules of our Transfer-Based MT system integrate information from the three components of a typical transfer system: syntactic analysis (parsing), transfer and

generation. Thus, in comparison with the PECOF system, blame assignment and correction become highly simplified.

More recently, some researchers have looked at other ways of including user feedback in the MT loop. Phaholphinyo and colleagues (2005) proposed adding post-editing rules to their English-Thai MT system with the use of a post-editing tool. However, they use context sensitive pattern-matching rules, which make it impossible to fix errors involving missing words. Unlike our approach, in their system, the rules are created by experienced linguists and their approach requires a large corpus. They mention an experiment with 6,000 bilingual sentences but report no results due to data sparseness.

Su and colleagues (1995) explored the possibility of using feedback for a corpus-based MT system to adjust the system parameters so that the user style could be respected in the translation output. They proposed that the distance between the translation output of the system and the translation preferred by the user should be proportional to the amount of adjustment to the parameters involved in the score evaluation function, and should be minimized over time. In the case of languages with limited data, such a system is not feasible, though, since there is not enough data to estimate and train system parameters. In our work, we are interested in improving the translation rules themselves, rather than just tweaking the evaluation parameters, which in their system are conditional probabilities and their weights.

Yamada and colleagues (1995) used structural comparison of parse trees between machine translations and manual translations in a bilingual corpus to adapt a rule-based MT system to different domains. In order for this method to work, a parser for the target language (TL) needs to be readily available, which is typically not the case for resource-poor languages. Moreover, such a parser must have coverage for the manually-corrected output as well as the incorrect MT output to compute the differences. The actual adaptation technique is not described in this paper.

In her Ph.D. dissertation, Naruedomkul (2001) proposed a basic word-to-word MT system, called the Generate and Repair MT system, that *repairs* a non-acceptable translation if it has a different meaning from the SL sentence. In order to repair a translation candidate, the system outputs an HPSG-like semantic representation for both the SL and the TL, detects the part of the TL that causes the mis-translation and replaces it with the corresponding, appropriate SL semantic representation. The system iterates until the semantic information of the SL and the TL are acceptably similar. In a final step, the word ordering module makes sure that the syntax is correct. All the examples given illustrate corrections of sense errors.

Callison-Burch (2004) also proposes to fix the underlying representations of an MT system to reduce error rate, but his is a Statistical system and thus the underlying representations are word-to-word alignments; in our approach, users can fix words, word order and word alignments. Another difference with our research is that while Callison-Burch's method involves developers manually fixing the alignments, we are go one step further by not requiring expert users and by automating the correction process.

Menezes and Richardson (2001) and Imamura et al. (2003) proposed the use of reference translations to *clean* incorrect or redundant rules after automatic acquisition. The method of Imamura and colleagues consists of selecting or removing translation rules to increase the BLEU score (Papineni et al., 2001) of an evaluation corpus. In contrast to filtering out incorrect or redundant rules, we propose to actually refine the translation rules themselves, by editing valid but inaccurate rules that might be lacking a constraint, for example.

Corston-Oliver and Gamon (2003) learned linguistic representations for the target language with transformation-based learning (Brill style) and used decision trees to correct binary features describing a node in the logical form to reduce noise.

In sum, even though adaptation has been extensively researched for MT and other natural language processing applications before, to our knowledge, no attempts have been made to automatically modify translation rules themselves without the help of experts. Therefore this thesis describes an interesting and novel approach to automatically refine and extend Transfer-Based MT systems. Finally, even though the approach proposed is not completely system independent, we believe that it can be easily adaptable to other Transfer-based MT systems (Section 7.1.4).

2.7 Learning MT Output Corrections

Font Llitjós et al. (2005b) sparked other researchers' interest to look at the problem of how to utilize human corrections of MT output to automatically improve such output. Elming (2006) suggests using transformation-based learning (TBL) to automatically acquire error-correcting rules from a parallel corpus of Danish machine translations from a rule-based commercial MT system (PaTrans: English to Danish) and a human-corrected version of these, in the domain of (chemical) patents. The TBL algorithm is, however, not able to create the transformation rules from scratch. It needs a list of rule templates that it can try to instantiate as concrete rules. In his experiment, Elming uses 70 templates that specify different combinations of possible contextual influence on substitution. All templates are based on the six nearest words. The correction rules that were learned exploit information on word forms and part of speech. And even though the

proposed method only applies to lexical choice and punctuation errors, the experiment resulted in a 3 BLEU point increase.

George and Japkowicz (2005) use machine learning techniques to learn corrections for rule-based MT. Focusing on the problem of relative pronoun translation between French and English, they employ different machine learning strategies to detect and correct wrong translations of the relative pronoun. The algorithm is trained on a small corpus of wrongly translated relative pronouns and their correct counterparts. It is also provided with information on part of speech and the semantics of noun phrases. Based on the experience of the learning algorithm, they are able to detect an incorrectly translated relative pronoun with an accuracy of 83.7%; 73.1% of their corrections are successful.

In addition to allowing manual re-write of interlingua rules for their Chinese-English multilingual dialogue system in the flight domain, Seneff and colleagues (2006) also applied two different methods to further improve the output of their interlingua MT system. On one hand, they used an *n*-gram language model and an English grammar to select the best candidate translation from the final set of interlingua outputs. On the other hand, they used a statistical MT (SMT) system to translate bad English into good English. Their experiments indicated that even though both methods increased the number of good translations, the SMT system approach to correct MT output also resulted in an increase of bad translations, whereas the most conservative approach of selecting alternatives guided by a language model and parsing information did not lead to as much output degradation.

The first approach of letting a linguistically-informed MT system overgenerate, and then use a statistical component at the end to aid with the selection of the best alternative translation, is implemented by the latest versions of the AVENUE transfer engine, which we use in our final experiments (Chapter 5).

The second approach of taking the initial MT system as a black box and applying a statistical MT system at a post-processing step to convert raw output into post-edited output has recently been shown to outperform both a commercial rule-based system and a state-of-the-art phrase-based MT system trained on the same data, as measured by BLEU and TER (Snover et al., 2006). For the Canadian Job Bank domain[7], researchers at the National Research Council of Canada (Simard et al., 2007a) built a statistical system (Portage) trained on raw MT output from a

[7] Job Postings submitted to the Job Bank, which is maintained by the Government of Canada, need to be available both in French and in English within 24 hours of submission.

rule-based English↔French MT system (Systran) and its post-edited counterpart, as corrected by professional translators. They found that very little post-editing data is necessary to improve upon the baseline. Their results suggest that such an approach could be used to do domain adaptation.

In the recent shared task of the Second Workshop on Statistical Machine Translation at ACL (June 23 2007), Simard and colleagues explored the impact of statistical post-editing when no manual post-edited data is available for raw MT output (Simard et al., 2007b). Like in their previous experiment, they translated the source text into the target language using Systran's English↔French MT system, and the APE or Statistical Post-Editing (SPE) component was then trained using Systran's MT output as "source" training material and the existing target portion of the parallel corpus as "target" training material. In this case, instead of having the target of the SPE component derived from the rule-based MT output, they were produced independently from the same original source. Their experiments on Europarl (Koehn, 2005) and News Commentary data confirmed that not only can phrase-based post-editing significantly improve a rule-based MT system (in terms of BLEU score), but when training data is scarce, it also outperforms a direct phrase-based MT strategy. Furthermore, their results indicate that the training data for the post-editing component does not need to be manually post-edited translations; it can be generated from standard parallel corpora.

In the same WMT shared task, researchers from Systran also tested the SPE approach for German→English and Spanish→English MOSES MT systems (Dugast et al., 2007; Koehn et al., 2007). Again, the objective was to train a SMT system on a parallel corpus composed of Systran translations with the reference translations provided by the parallel corpus. They observed a large improvement of BLEU scores for the combined system compared to raw translation output (for German-English, around 10 BLEU points for the Europarl test set of WMT2007), as well as a corresponding improvement of the translation fluency.

In summary, this dissertation presents an approach that integrates non-expert bilingual post-editing with automatic refinement of MT system to improve the translation quality of unseen data. With such an approach, we will show in the rest of the document that:

- Corrections generalize beyond post-edited examples (unlike TM and traditional post-editing)
- The underlying MT system can be refined and extended without expert intervention or pre-existing annotated data.

This is achieved by simplifying the MT error elicitation task so it does not require any linguistic knowledge, while at the same time providing enough information for the Automatic Rule Refiner

to perform blame assignment and execute the appropriate rule modifications. Finally, our work has led the way for what we believe is a new area in the field of Machine Translation, namely automated learning of MT output corrections. This has recently resulted in new combinations of Rule-Based MT systems and phrase-based statistical systems to learn automatic corrections from raw MT output.

Chapter 3

Elicitation of MT Error Information from Non-Experts

In times when statistical methods are in vogue to develop Machine Translation systems from a parallel corpus, when there is no parallel or comparable data available electronically, how can we begin to tackle the Machine Translation problem?

We can always resort to the hand-crafting of translation lexicons and grammars, but for some languages there are no computational linguists with knowledge of a specific language pair or native linguists of that language pair who can be trained to write computational resources. And even if we could find such an expert to write a small set of rules for our MT system, how could this system be improved and extended over time?

In resource-poor contexts, there is usually at least one resource available: bilingual speakers. The research described in this thesis seeks to leverage this fact and relies on non-expert bilingual users to obtain as much accurate information as possible to determine error location and cause. Error information is then automatically processed by the Automatic Rule Refiner to extend and improve the lexicon and the grammar of the MT system.

In order to elicit MT error information from bilingual informants reliably, we developed a user-friendly, graphical user interface (GUI), that is intuitive and very easy to use, and that does not assume any knowledge about translation, linguistics or computers.

This chapter is devoted to describing the design, implementation, and testing of this online GUI that we call Translation Correction Tool. First, we describe an MT error typology that we developed to analyze the type of errors typically made by state-of-the-art MT systems. Second, we move to the interface design and implementation of the online tool. Third, we present a set of

user studies that were done for English-Spanish to test the effectiveness of the Translation Correction Tool both for error detection and classification. Finally, we report some data analysis and lessons learned that lead to an improved and simplified interface design and implementation.

3.1 MT Error Typology

In order to design an effective GUI to elicit corrections from MT output, so that the underlying MT representations can be improved automatically, we faced the need to itemize the types of errors typically done by state-of-the-art MT systems. As part of initial research mostly based on MT errors produced by our English to Spanish Transfer-Based MT system,[8] we defined a preliminary MT error typology. Figure 3.1 shows this typology in a simplified form.

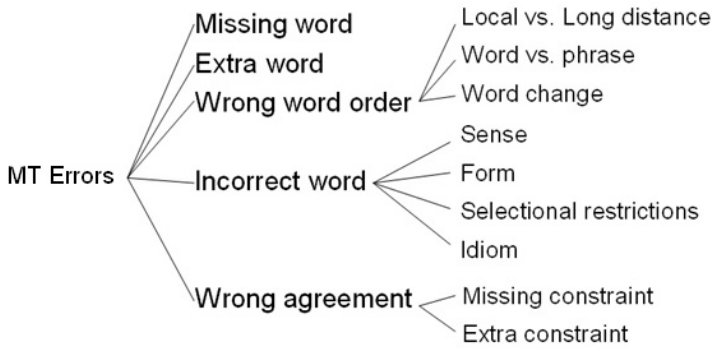


Figure 3.1. Simplified MT Error Typology with classes illustrating implicit user actions on the left, and types/causes of errors on the right.

This MT error classification is not a typology of linguistic phenomena for general error identification or system comparison like the ones discussed in Section 2.4, but rather an error typology that is relevant for automatic rule refinement. Many distinctions that one would probably want to make from a linguistic point of view are not necessary in this context. The task at hand involves tracking the translation rules that generated the incorrect output. Rule blame assignment gives the system most of the linguistic information needed to automatically extend and improve the grammar and lexicon.

[8] Initial error analysis as well as system development were based on English-Spanish MT system output. Chapter 6 is devoted to applying our approach to a resource-poor language pair, namely Mapudungun-Spanish. For the rest of this chapter, English↔Spanish examples are used for broader understandability.

From the error space shown in Figure 3.2, we address the subspace that is tractable fully automatically after initial error information is available. Figure 3.2 illustrates the subspace that is the focus of our work. It shows the error classes (left) into which users implicitly classify the errors, simply by correcting the MT output with the online post-editing tool, followed by finer grained classes that are inferred by the Automatic Rule Refiner in order to fix the appropriate rules and lexical entries (right). Long distance word moves cannot always be reliably refined automatically, in particular when a word is moved to a sentence initial or final position. On the other hand, incorrect constraints (typically learned automatically) make a rule too specific and can prevent it from applying, making such errors very hard to detect and fix automatically.

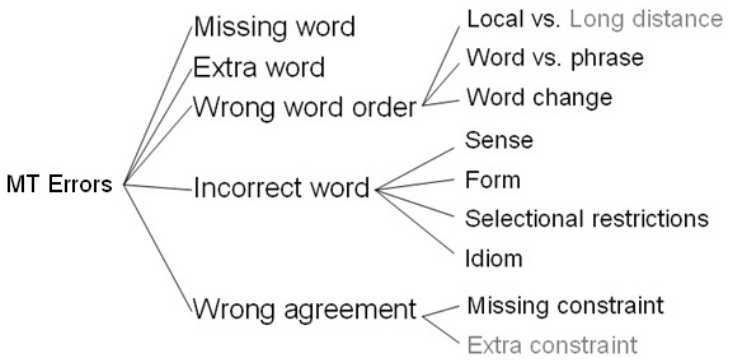


Figure 3.2. Subset of MT errors addressed by the Automatic Rule Refiner.

Below, we provide with a more detailed typology that includes initial indications as to what kind of Rule Refinements (RR) seem appropriate for each error type. Hence, RR below refers to a high-level description of what actions the Automatic Rule Refiner needs to execute to remediate each error type, not to what users must do or tell the system.

The first-level *types* are usually what the literature calls error types. The second-level *types*, however, tend to denote more the cause of the error, and third-level *types* tend to further specify features or constraints for that error/cause. Therefore, given the observation that a word is missing or should not be there, is misplaced or is simply incorrect, one can refer back to this error typology and look up what are the different reasons this might have happened. In combination with the error and system information available to the Automatic Rule Refiner, this error typology guides what correction actions need to be taken by the system (Section 4.5.3). Error types followed by a diamond (♦) are not currently handled by the Automatic Rule Refiner.

The examples shown below assume a fully inflected lexicon, which is the result of adding morphology features to a citation form lexicon.

MT Error Typology

1. **Misspelling**

RR: Add correct spelling to the lexicon (given by user).

2. **No translation (unknown word)**

RR: Add correct translation to lexicon (given by user).

3. **Missing word**

3.0. **Translation not in the lexicon**

Example: I get dressed → *Yo Ø visto → Yo me visto

RR: Add new translation to the lexicon by copying existing lexical entry and updating target language side: ([get dressed] → [me visto]). In this case, this is a missing form (reflexive) of the verb *vestir*.

3.1. **Incomplete lexical entry**

Example: Me gustaría ir de viaje→ *I would like Ø travel → I would like to travel

RR: If user aligns the new word (*to*) to an already aligned SL word (*gustaría*) and it is contiguous to the aligned TL word (*would like*), add the new word to the existing lexical entry: [would like TO] → [gustaría]. If no alignment is added to the newly added word, add a lexicalized rule ([VB VP] → [VB "to" VP]).

3.2. **Preposition missing (subcategorization)**

Example: Let's go home→ *Vamos Ø casa → Vamos a casa

RR: If user aligns *a* with (*let's*) *go*, the lexical entry for *vamos* is extended to include the preposition *a* ([let's go] → [vamos a]).

3.3. **Syntactic restrictions apply**

Example: I saw the woman → *vi Ø la *mujer* → vi a la *mujer*

RR: If no alignment is added between a and *a* SL word, add directly to the rule as a terminal ([V NP]→[V "a" NP]) and if no feature exists for animacy, create one and add it as a constraint to the relevant rules and lexical entries (Section 4.6.1).

4. **Extra word**

4.1. **Literal translation**

Example: He wanted to come → *él quería a venir → él quería venir

RR: Add empty translation for SL word to ([to]→[“”]) or new lexicalized rule to the grammar ([VB "to" VP]→[VB VP]).

4.2. **Overgeneralization**

Example: John read the book → *A Juan leyó el libro → Juan leyó el libro

RR: Add constraint that restricts the application of the rule to the right context, in this case: [NP]→[“a” NP] can only apply in an object position.

5. **Word order**

5.1. **Local — within a constituent**

5.1.1. **Single word**

5.1.1.1. **No word change**

Example: John held me with his arm → *Juan sujetó me con su brazo

→ Juan me sujetó con su brazo

RR: Create a new rule with the order flipped, and add a constraint specifying the features of the word moved; in this case, that *me* is a pronoun ([V NP_{PRON}] → [NP_{PRON} V]).

5.1.1.2. **Word change**

Example: Gaudí is a great artist → *Gaudí es un artista grande

→ Gaudí es un gran artista

RR: Create a new rule with the order flipped and restrict its application to prenominal adjectives ([ADJ N] → [$ADJ_{PRENOMINAL}$ N]).

5.1.2. **Multiple words**

5.1.2.2. **Word change**

Example: I will take him → *Yo llevo a él → Yo le llevo

RR: Add *le* as a translation for *him*, if not already in the lexicon, and create a new rule with the order flipped and a constraint specifying the features of the word moved, in this case, that *le* is an indirect pronoun

(case = acc) instead of a subject pronoun ([V "a" NP_{PRON_nom}] → [NP_{PRON_acc} V]).

5.2. **Long distance — across constituents ♦**

Example: Where are you from? → *Dónde eres tú de? → De dónde eres tú?

RR: Feed back to the Rule Learning module (Probst 2005) to learn a new rule.

6. **Incorrect word**

6.1. **Sense — semantic restrictions apply**

Example: Wally plays the guitar → *Wally juega la *guitarra*

→ Wally toca la *guitarra*

RR: Create a binary feature to distinguish between the two senses of the verb *play* in Spanish; add this feature to the verb as well as the noun that triggers the right sense, in this case *guitar* (the clue word, see Section 4.5), as well as an agreement constraint for that feature to the relevant rule.

6.2. **Form — semantic restrictions apply**

Example: Hay algunas flores y muchos árboles

→ *There were some flowers and much *trees*

→ There were some flowers and many *trees*

RR: Create a binary feature to distinguish between countable and uncountable nouns and mark quantifiers as being able to quantify one or the other noun type. Without all nominal lexical entries being tagged as countable or uncountable, the most the Automatic Rule Refiner can do is enforce *many* to co-occur with *trees*, but not with other countable nouns (cars, girls, insects, etc.).

6.3. **Sense/Form — syntactic restrictions apply**

Example: I'm playing chess → *Soy *jugando* al ajedrez

→ Estoy *jugando* al ajedrez

RR: Without a generalization mechanism, the current implementation of the Automatic Rule Refiner can only add an enforcing constraint between *estoy* and *jugando*, but cannot constraint the appropriate grammar rule to enforce the verb *estar* in the presence of a gerundive verb. The ideal level of refinement for this example would be to duplicate the general VP rule ([AUX V]) and add a constraint

to the specific rule so that, when followed by a gerundive verb, the verb *to be* translates as *estar* instead of *ser* ([AUX_{to_be} V_{gerund}] → [AUX_{estar} V_{gerund}]).

6.4. **Selectional restrictions**

Example: Today we are eating fish → *Hoy vamos a *comer* pez

→ Hoy vamos a *comer* pescado

RR: If user identifies *comer* as a clue word, create a feature to distinguish between *pez* and *pescado* and add the right constraint to *comer* (and the VP rule) so that it only combines with *pescado*.

6.5. **Governed Preposition**

Example: I was worried about you → *Estaba *preocupada* sobre ti

→ Estaba *preocupada* por ti

RR: Add the preposition as part of the lexical entry with the appropriate translation ([worry about] → [preocupada por]), or add a constraint that enforces the preposition following *preocupado/a* to have the right form.

6.6. **Idiom — lexicalized expression**

Example: He bit the dust → *él mordió el polvo → él se murió

RR: Given correct user alignments, enter in the lexicon as a unit ([bite the dust]→[morirse]).

6.7. **Form — morphology restrictions apply**

6.7.1. **Overgeneralization**

Example: *wifes → wives

RR: Add correct form to lexicon

6.8. **Form — phonetic restrictions apply**

Example: *an wife → a wife

RR: Create a binary feature that has value + for *wife* (and eventually all other words starting with a consonant), and add the appropriate constraints to the grammar so that *a* can only combine with such words, and *an* cannot.

6.9. **Other — wrong translation**

Example: I drove to the movies → *Conduje al cine → Fui al cine (en coche)

RR: Add *fui* (*ir*) as an alternative translation for *drove* (*drive*).

6.10. **Translation not in lexicon**

Example: I'm tired of you → *Estoy cansada *de* tú → Estoy cansada *de* ti

RR: Add a lexical constraint to both the clue word (in this case the preposition *de*) and the error word (indirect pronouns (oblique). If the right translation (according to the feature constraints) is not in the lexicon (*ti*), add it.

7. **Agreement (number, gender, person and tense)**

7.1. **Right form not in the lexicon**

Example: They climb the mountain → **Ellos* escala la montaña

→ *Ellos* escalan la montaña

RR: If the right form is not in the lexicon or cannot be generated by the morphology module (Section 6.5), add the correct from to the lexicon and make sure appropriate agreement constraints (between the subject and the verb) are in place.

7.2. **Missing agreement constraint**

7.2.1. Within a constituent

Example: The tall chair was red → *La *silla* alto era roja

→ La *silla* alta era roja

RR: Add a gender agreement constraint to the NP rule between N(*silla*) and the ADJ(*alta*).

7.2.2. **Across constituents**

Example: The chairs were very tall → *Las *sillas* son muy alto

→ Las *sillas* son muy altas

RR: Add a gender agreement constraint to the mother node of the constituent containing *sillas* and the constituent containing *alto.*

7.3. **Extra agreement constraint — Overgeneralization ♦**

Example: John protects plants → *Juan protege la planta

→ Juan protege las plantas

RR: Eliminate overly general constraint. In this case, the automatically learned VP rule [V NP] has an incorrect number agreement constraint between the verb and the direct object.

This error typology is by no means complete, but it contains errors that are useful for other language pairs as well (Section 2.4) and is a good starting point to design an Automatic Rule Refiner. When moving to other language pairs, this MT error typology will most likely need to be expanded to accommodate for new error types.

The actual rule refinement formalization is described in next chapter, and the types of refinement operations that were implemented for each error type (not followed by ♦ above) are addressed in Section 4.6.3.

3.2 Interface Design and Implementation: The Translation Correction Tool (TCTool)

The Translation Correction Tool (TCTool) is a user-friendly online GUI designed to accurately evaluate and minimally correct MT output and to obtain as much information about MT errors as possible from non-expert bilingual speakers. The TCTool presents bilingual informants with a sentence in the source language and up to five alternative translations produced by the MT system, and asks them to select all the correct translations.[9] Alternatively, if none of the alternative translations is correct, the TCTool asks bilingual informants to fix the best translation with the least number of corrections possible. In this context, we tell informants that the best translation is the one requiring the least number of changes to render the same meaning as the original sentence, in a grammatically correct and fluent TL sentence, but not necessarily in the most ideal form of expression (to avoid stylistic changes). This has been referred to as *minimal post-editing* in the

[9] The decision to include the first five translation alternatives produced by the MT system, as opposed to just one or the first fifteen, was motivated by the principle of maximizing user time while not burdening bilingual speakers unnecessarily.

literature (Section 2.3), and the main problem is how to determine what is the minimal amount of post-editing changes that must be made to raw MT output text.

1. Select correct translation form list of alternatives, if any

2. Correct the best translation, if none is already acceptable, by performing any sequence of correction actions (Figure 3.4)

3. Validate corrections (final translation + alignments)

Figure 3.3. Steps bilingual speakers have to take when using the TCTool.

Figure 3.3 summarizes the steps of the translation correction process enforced by the Translation Correction Tool. If bilingual speakers decide that the best translation needs to be corrected, they proceed to edit the translation and make all the necessary changes by performing a sequence of basic correction actions given in Figure 3.4. Once they are done with the correction step, the TCTool asks them to validate the final translation they have produced and all the alignments, by providing the explicit word to word correspondences.

There are three parts to the translations that can be manipulated by users of the TCTool: 1) the words in the translation or target language (TL) sentence, 2) the order of the words and 3) the alignments from the source language (SL) sentence to the TL sentence. The alignments indicate the word-to-word correspondence, namely what word in the SL translates as what word in the translation sentence. The current implementation of the TCTool makes the assumption that if a translation is correct, the alignments for that translation are also correct.

There are six basic correction actions bilingual informants can take to correct a sentence, using the Translation Correction Tool (Figure 3.4).

- Modify a word
- Add a word
- Delete a word
- Change word order
- Add an alignment
- Delete an alignment

Figure 3.4. Basic Correction Actions allowed by the Translation Correction Tool.

When correcting a sentence with the TCTool, bilingual informants are presented with two columns of boxes, each of them containing a word (Figure 3.5). The SL sentence is displayed on the left column, and the TL sentence is displayed on the right. Word alignments between source and target sides are represented as lines that connect boxes containing SL words to boxes containing TL words. For the AVENUE MT system, alignments are directly extracted from the translation rules that generated the TL sentence (Section 4.3, Figure 4.1). White arrows on the SL side are used to create an alignment when dragged to a TL word.

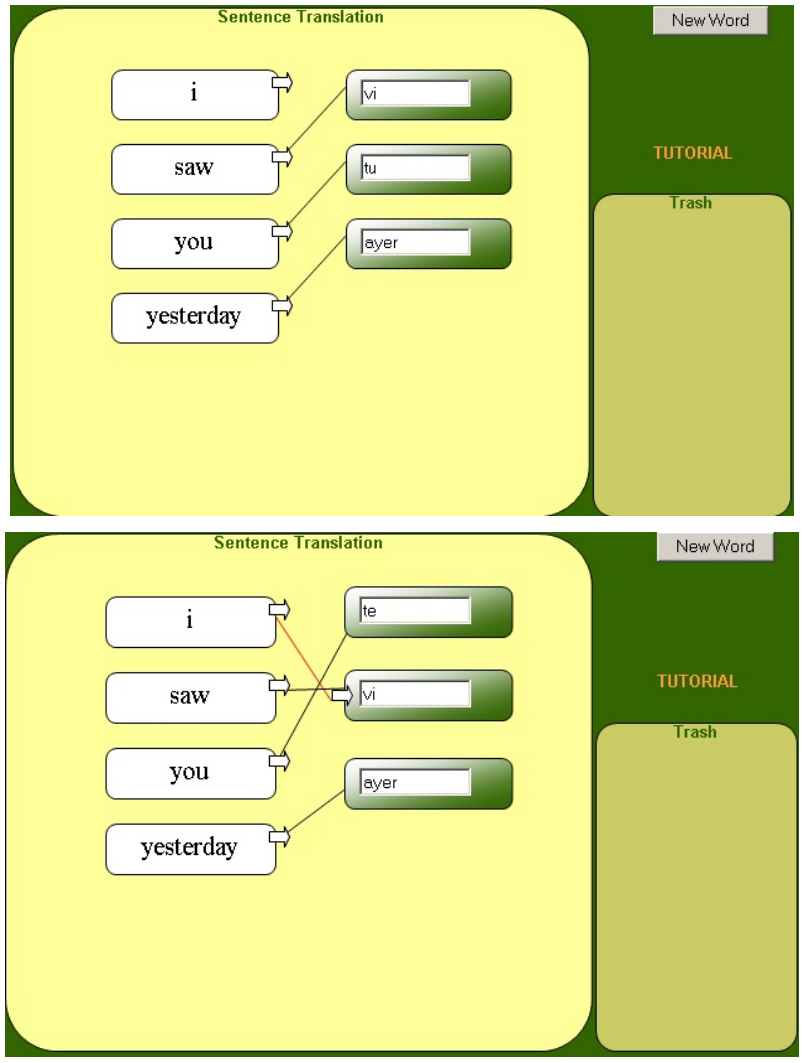


Figure 3.5. TCTool snapshot of the initial screen with the incorrect translation (top), and the final screen with the translation corrected by a bilingual informant (bottom).

Figure 3.5 shows the TCTool interface before and after user corrections for a simple sentence (*I saw you* — **vi tu ayer* → *te vi ayer*). Note that for this example, user corrections included editing a word (*tu*→*te*), moving a word in the right position (*te* before the verb *vi*) and adding an alignment from the English subject to the Spanish verb.

The TCTool interface is designed to allow users to correct errors at a high level, abstracting away as much as possible from the inner intricacies of the MT system.

Since we are aware of the intrinsic difficulty of the task, however, we tried to choose an interface that is simple and fun to use and, at the same time, gives users enough flexibility.

The data flow diagram in Figure 3.6 shows how the core of the TCTool works. HTML pages presented to users are represented in squares and Perl CGI scripts, which process user data, are represented in ovals.

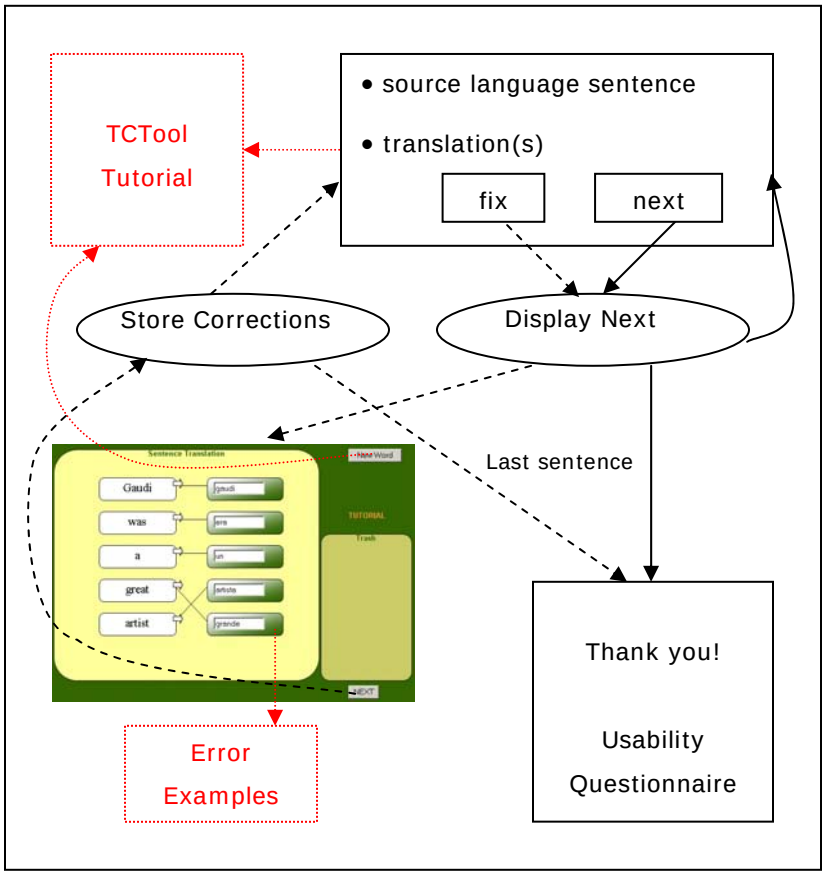


Figure 3.6. TCTool simplified data flow diagram. Discontinuous arrows represent the flow if the translation is not correct and the user decides to fix it; continuous black arrows show the flow if the translation is acceptable; help pages appear in dotted lines.

The initial page is displayed in the upper right and shows the SL sentence and its translations and asks the user to select the best translation. Discontinuous arrows represent the flow if the translation is not correct and the user decides to fix it. In this case, users are presented with the drag and drop GUI shown in Figure 3.5. The editable GUI is implemented in JavaScript to allow users to easily fix translations just by clicking on the words that they want to modify and by dragging and dropping words when they want to change their order.

From this page, users have two help options available: they can go either to the online tutorial or to a page with examples for all the error types. The CGI scripts store log files with the correction actions performed by users. When users decide that a translation is already correct, the next SL sentence, with its translation alternatives, is displayed. Continuous black arrows show the flow if translation is acceptable.

The most important aspect to keep in mind while correcting MT output via the TCTool is that user feedback is not only used to improve the translation at hand, but it is critical for the refinement of the underlying translation rules and will be used to improve the MT system at its core. For this reason, we emphasize to users the importance of only modifying what is strictly necessary to obtain a correct translation of the original sentence from the given translation. For the example shown in Figure 3.5 (*I saw you yesterday*), an unnecessary correction could be the addition of the optional subject in Spanish (*yo*) or changing the order of the adverb (*ayer*) to an sentence initial position (**vi tu ayer* → ***ayer yo*** *te vi*).[10]

To illustrate what it means to correct a translation minimally, as well as how the TCTool can be used, an extensive TCTool tutorial (23 pages) is provided to bilingual informants.[11] The tutorial shows all possible actions to correct a translation and illustrates them in detail through example sentence pairs.

In the next section, we describe a set of user studies done to evaluate the effectiveness of the Translation Correction Tool to elicit MT errors correctly from non-expert bilingual speakers.

---

[10] Spanish is a pro-drop language (the presence of subject pronouns is optional) and can have adverbs either at the beginning or at the end of the sentence.

[11] http://avenue.lti.cs.cmu.edu/aria/spanish/tutorial.html

3.3 Evaluation: English-Spanish User Study (TCTool v.1)

A preliminary usability evaluation presented non-expert bilingual speakers with 32 simple English-Spanish translations. Source language sentences were chosen from the AVENUE Elicitation corpus, designed to cover a variety of linguistic phenomena (Probst et al., 2001), with the goal of exposing users with a wide range of different MT errors.

The purpose of this user study was twofold: 1) to evaluate the TCTool interface for the task of error detection and remediation, and 2) to evaluate the initial MT error classification. For this task, we needed to think of MT error classification in a completely different way, and we needed to find a balance between simplicity and informativeness. Users of the TCTool, i.e. non-expert bilingual speakers, have to be able to understand the different error types and classify them accurately, and, at the same time, we have to obtain the most information about errors possible, in order to be able to automatically refine translation rules.

The TCTool automatically detects word order errors, as well as missing or extra words, and thus informants were not asked to classify them. Errors that involved modifying a word, however, can be classified to help determine the kind of rule refinement to execute automatically. Thus, in version 1 of the TCTool, bilingual speakers were asked to classify errors only when modifying a word. The explicit MT error classification used in TCTool v.1 included the eight error types shown in Figure 3.7. These correspond to error types 6 (incorrect word) and 7 (wrong agreement) in the typology presented above (Section 3.1).

- wrong sense
- wrong form
- agreement error
    - number
    - person
    - gender
    - tense
- incorrect word
- unknown word

Figure 3.7. MT error classification associated with the *modify* correction action in TCTool v.1.

Figure 3.8 shows the screen that displays the different error types listed in Figure 3.7. This window popped up when users edited a word using the first version of the TCTool (v.1). Since we expected some bilingual informants not to know what these types of errors stand for, we

included a brief explanation for each error type with one or two examples (labeled as “Error Examples” in Figure 3.6).

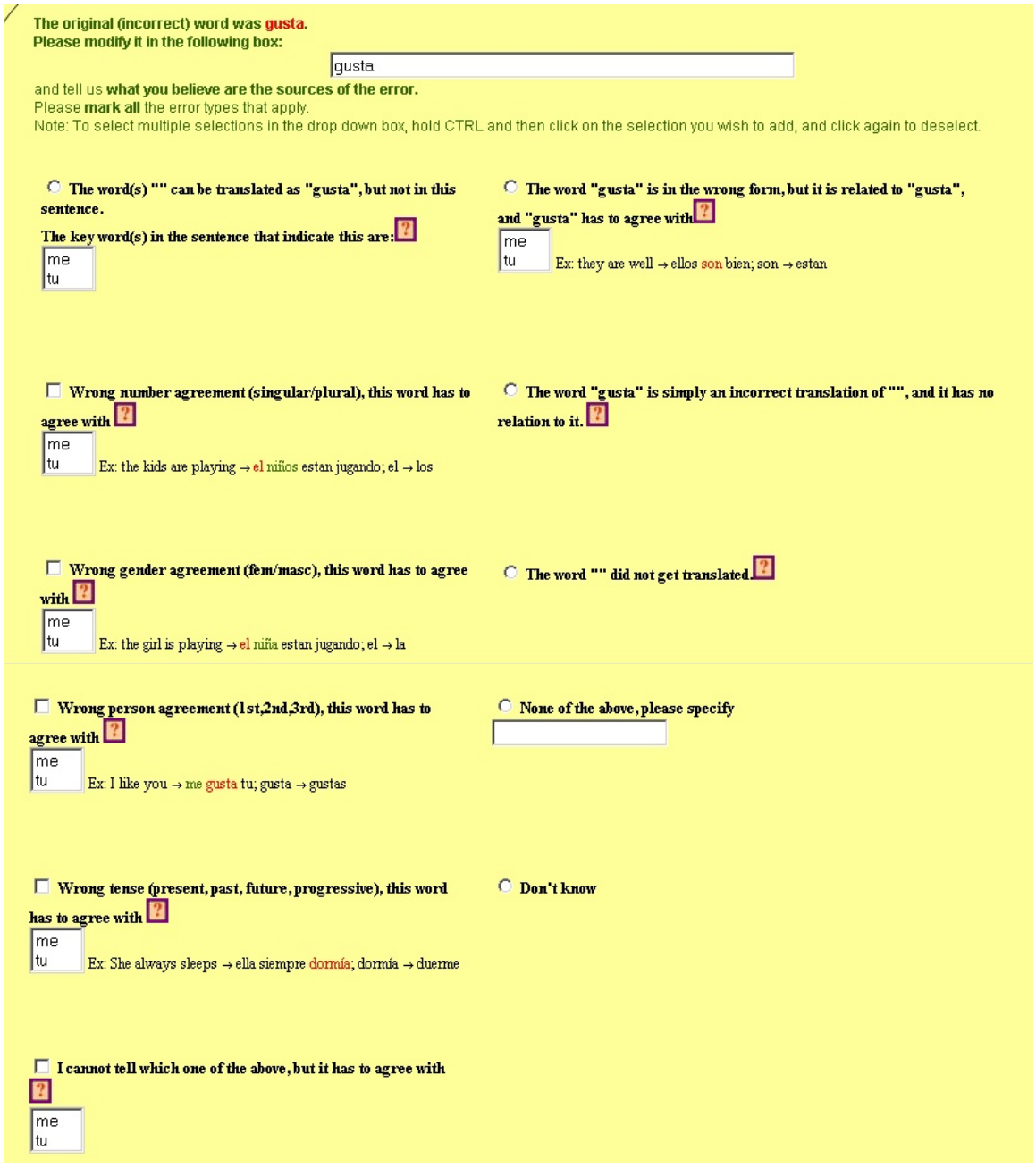


Figure 3.8. Popup window showing the error types associated with editing a word.

The MT system used for this user study consisted of a small manually written English to Spanish grammar containing 12 rules (2 S rules, 7 NP rules and 3 VP rules) and 442 lexical entries, designed to translate the first 400 sentences of the Elicitation corpus.

29 native speakers of Spanish with good knowledge of English completed the evaluation. Most users were from Spain (83%), 33% had a background in Linguistics, 75% had a graduate degree and 25% of the users had a Bachelor's degree.

On average, users took an hour and a half to evaluate the 32 translation pairs and fix 26.6 translations, about 3 minutes per translation pair. But there was a significant variance among users, the duration range being [28min-4:18hours]. For more details, see Font Llitjós and Carbonell (2004).

We found that some of the distinctions between the eight error types are actually not relevant for the purpose of Rule Refinement (i.e. wrong sense and wrong form), but contribute to the task complexity faced by non-expert informants. Thus, a less traditional error classification seems more appropriate for this task, and was implemented for the next version of the TCTool (Section 3.4).

3.3.1 Results and Data Analysis

To measure error correction accuracy, a gold standard was established prior to looking at any of the user corrections. Ten users were selected according to demographics (to reduce possible dialectal differences) as well as levels of education in order to represent as many different education levels as possible.[12]

For these 10 users, we analyzed a total of 300 log files in detail, and manually counted the times that the errors detected by users and the error type associated with each error coincided with the gold standard. This count was very strict, and good corrections that were different from what the gold standard indicated were counted as not being 100% accurate. This might seem more strict than necessary, but the goal of this evaluation was to simulate the process for a language pair we are not familiar with, where we cannot make any judgments.

We found that non-expert bilingual speakers can detect translation errors with reasonably high precision (90%), but have a harder time determining what type of error it is. Given the MT error classification (Figure 3.7), users identified the error correctly 72% of the time, as can be seen in Table 3.1.

[12] They were all from Spain, only 2 had Linguistics background; 2 had a Bachelor's degree, 5 had a Masters and 3 had a PhD.

	Precision	Recall	F1
Error Detection	0.896	0.894	0.895
Error Classification	0.724	0.715	0.719

Table 3.1. Average F1 measures for 10 users and 32 sentences (300 log files).

From the precision-recall viewpoint, we are interested in having bilingual informants correct and classify errors with high precision, even at the cost of lower recall. If informants do not detect or classify all the actual errors, this does not have great impact on the automatic refinement of translation rules. Nevertheless, when informants do detect and classify errors, we need them to do so as precisely as possible, so that the translation grammar and lexicon are not spuriously refined.

User corrections were not always consistent with each other, because there is often more than one correct translation, and different ways to get there when using the TCTool. Most of the time, however, when the final translations differed from the gold standard, they were still correct. On average, users only produced 2.5 translations that were worse than the gold standard (out of 26.6 that they corrected). Users got most alignments correct. To see the way we address user variation when making corrections, see Chapter 4 (Section 4.5.1.2).

Therefore, the challenge for the approach introduced in this dissertation is to classify and fix errors automatically based only on error detection (and alignment information), rather than user-supplied error categories. The latter are less reliable and more time consuming for users. [13]

3.4 Simplified and Improved GUI: TCTool v.2

Results from the user study presented above indicate that bilingual informants are not able to classify errors as accurately as they are able to detect them. Therefore, we simplified the GUI so that bilingual speakers could focus on error detection, and did not have to do any explicit error classification. Instead, TCTool v.2 takes care of error classification implicitly, inferring error classes from error information available and from correction actions themselves.

[13] There is one caveat to the hypothesis that MT error classification is harder than MT error detection, namely that the initial version of the TCTool might have taxed users' capacity to perform two tasks at the same time (error detection plus error classification). Thus, it is possible that a tool designed to just classify errors would yield higher precision.

In TCTool v.2, the eight linguistically-motivated classes associated with editing a word (Figure 3.7) are substituted with a more intuitive non-technical concept, namely what we call clue word(s). Clue word(s) are words in the target language sentence that provide bilingual informants with the clue of what the correction should be. For example, given the English "*plays the guitar*" and the incorrect translation "*juega la guitarra*", non-expert bilingual informants might have a hard time classifying this as a word sense vs. a word form error (if not also as incorrect word). Most users, however, will be able to identify *guitarra* (*guitar*) as the clue word when changing *juega* (*play a game*) into *toca* (*play an instrument*). Figure 3.9 shows the new popup window associated with the edit action, where users can select a clue word, if they wish to.

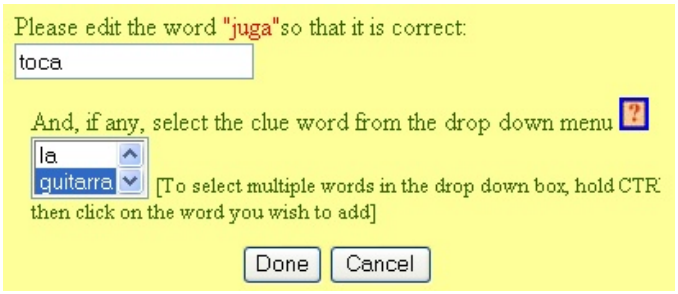


Figure 3.9. Window that pops up in TCTool v.2 when bilingual informants edit a word. Instead of requiring them to classify the error as in the previous version (Figure 3.7), users can optionally select one or more clue words from the target language sentence.

More fine-grained distinctions of the source of the error placed an unnecessary burden on users; the types of rule refinement operations required to remediate word sense and word form errors, for example, are actually the same (See 6.1 and 6.2 in the error typology given in Section 3.1).

In sum, TCTool v.2 simplifies the user correction task and at the same time it allows for higher error information precision.

3.5 Validation for Resource-Poor Language Pair

English and Spanish have a relatively simple morphology, but resource-poor languages such as Quechua and Mapudungun have a more complex morphology. In order to present sentences from such languages to bilingual informants with the TCTool, we need to solve the problem of how to represent words, or in this case stems and suffixes.

To study this problem, a preliminary user study of the correction of Quechua to Spanish translations was conducted in Cusco (Peru). Three Quechua speakers with good knowledge of Spanish evaluated and corrected nine machine translations using TCTool v.2. For this user study, we choose to represent stems and suffixes in separate boxes, treating them as separate words, since Quechua suffixes attached to verbs very often translate into the direct object or into other parts-of-speech in Spanish. Figure 3.10 shows the representation of the Quechua sentence "*pay takishan*", which is depicted in four units (pronoun, verb stem + 2 suffixes). Word segmentations are produced by a morphology analyzer (Section 6.5).

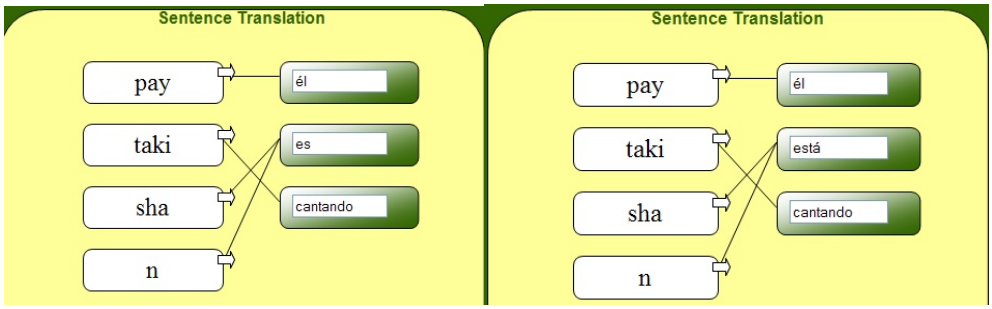


Figure 3.10. Quechua-Spanish output for the sentence "*he is singing*", displayed before (left) and after (right) user corrections.

This user study showed that the Quechua representation of stem and suffixes as separate words does not pose a problem and that it was easy to use for non-technical bilingual informants. This representation was also used to elicit error corrections from the Mapdungun-Spanish system, which we evaluate in Chapter 6.

3.6 Conclusions

User studies indicate that non-expert bilingual speakers can accurately pinpoint and correct MT errors (90% of the time), but have a harder time classifying MT errors according a pre-defined linguistic classification (72% of the time). In addition to that, we discovered that for automatic rule refinement, such a classification is too fine-grained and most of the time does not give any extra information to the Automatic Rule Refiner.

TCTool v.2 offers a non-technical alternative to explicit error classification by optionally allowing bilingual informants to indicate what word(s) in the translation give them a clue with respect to what correction is required. This alternative approach provides all the necessary

information for automatic rule refinement and, most importantly, avoids placing the error classification burden on non-experts.

Chapter 4

Automatic Refinement of Translation Rules

After eliciting the location of the error(s) in MT output from non-expert bilingual speakers, we can now turn to the problem of automatically improving and extending the grammar and the lexicon using such information. This chapter describes how error information provided by bilingual informants can be used to trace errors back to incorrect translation rules and fix them automatically in order to improve translation quality.

One of the biggest technical challenges is mapping simple correction actions onto non-trivial modifications of the underlying rules responsible for causing the error. The core component of the rule refinement process decides what rule refinement operations need to apply to address specific error corrections. This component is also the most sensitive to the set of correction actions currently allowed by the Translation Correction Tool described in the previous chapter. The reason for this is that rule refinement operations that are applied by the system crucially depend on what types of correction actions were chosen by bilingual informants.

The Automatic Rule Refiner (ARR) applies a different refinement operation depending on the correction action type (add, edit, delete, and change word order), the alignment information available and whether the error and correction words detected by bilingual speakers are already in the lexicon or not. In general, the ARR addresses lexical refinements first and then moves on to refinements of the grammar rules, if still necessary. More specifically, the ARR can automatically add missing lexical entries, add missing feature value and agreement constraints to transfer rules, and add, delete, or change the order of constituents in a grammar rule. Table 4.1 summarizes the types of extensions and refinements to both the lexicon and the grammar that can be handled by the approach described in this dissertation. The last type (propagate feature constraints) is applied

in combination with constraint refinements to the grammar, when there are multiple levels of embedment.

LEXICON - Add new lexical entry (no translation) - Add new sense of an existing source language (SL) word - Add a new form of an existing SL word - Add empty translation for an existing SL word, effectively deleting it - Add feature values constraints to an existing lexical entry GRAMMAR - Add a constituent on the target language (TL) side of a transfer rule - Delete a constituent on the TL side of a transfer rule - Change the order of the TL constituents - Add feature values constraints to a constituent on the TL side - Add feature agreement constraints between two constituents on the TL - Propagate feature constraints up to the mother of the transfer rule

Table 4.1. Summary of the types of refinements handled by the Automatic Rule Refiner

In the first section of this chapter, we introduce an example that required two correction actions (i.e., modify a word and change word order) and describe the automatic rule refinements executed to the lexicon and the grammar. The technical concepts and mechanisms required to understand the mode of operation of the Automatic Rule Refiner are formally described in the rest of the Section. First, we introduce the framework for rule refinement or adaptation, followed by a brief description of the underlying transfer rule formalism, so that it becomes clear what are the types of translation rules that are affected by automatic refinements. Section 4.4 discusses the two main types of rule refinement operations (CONSTRAIN and BIFURCATE) and, in Section 4.5, we formalize error information. Section 4.6 outlines the Automatic Rule Refiner algorithm for all correction actions, and Section 4.6.3 breaks it down into the four different correction actions (add, modify, delete, change word order). Section 4.7 gives detailed examples for the correction actions not covered in Section 4.1. Section 4.6.4 discusses the trade-off that exists between accuracy and making refinements too specific, and in Section 4.8, we conclude with a

discussion on the generalization power of the kinds of automatic refinements we have implemented.

4.1 An Example: Change Word Order and Modify Cases

To better illustrate how the Automatic Rule Refiner operates, we first describe a detailed example for two of the four refinement cases: change word order and modify case. We introduce technical concepts as needed, but do not describe them formally until subsequent subsections.

Given the source language sentence "*Gaudí was a great artist*" and the translation generated by our MT system "*Gaudí era un artista grande*", bilingual informants modified the translation into "*Gaudí era un gran artista*" using two correction actions: (1) editing *grande* into *gran* and (2) moving *gran* to a pre-nominal position (*gran artista*), as can be seen in Figure 4.1.

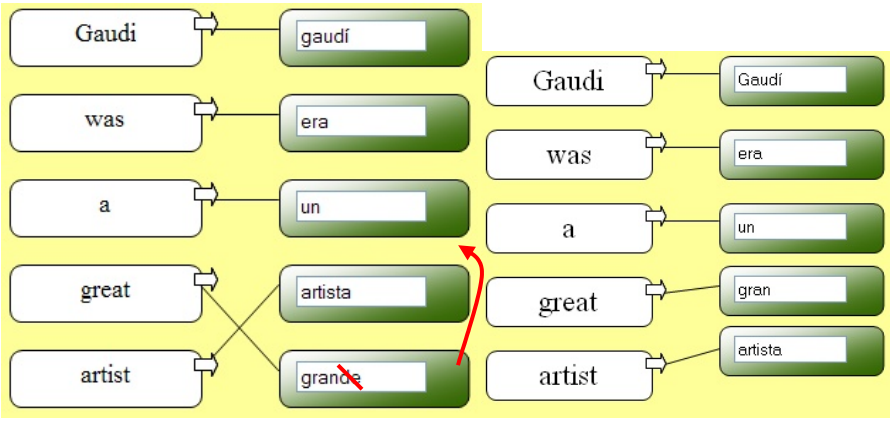


Figure 4.1. TCTool initial and final shots for change word order correction action.

The automatic rule refinement looks up both [great→gran] and [great→grande] in the lexicon and extracts their lexical entries.[14] Next, the ARR compares both lexical entries at the feature level to see if for any of the feature attributes they have a different value. The hypothesis here being that there is a difference between *gran* and *grande*, and in fact this is what triggered the user corrections. In this case, [great→gran] and [great→grande] are identical at the feature

[14] If [great→ gran] is not in the lexicon, the ARR makes a copy of the original lexical entry ([great→ grande]) and adds *gran* as the translation of *great* in the copy.

level, and so in order to discriminate them, the ARR module postulates a new binary feature, say feat1[15], and adds it to each lexical entry with a different feature value (+,-) as shown in Figure 4.2

```
ADJ::ADJ |: [great] -> [grande]
((X1::Y1)
((x0 form) = great)
((y0 agr num) = sg)
((y0 agr gen) = masc)
((y0 feat1) = -))

ADJ::ADJ |: [great] -> [gran]
((X1::Y1)
((x0 form) = great)
((y0 agr num) = sg)
((y0 agr gen) = masc)
((y0 feat1) = +))
```

Figure 4.2. Lexical entries for *grande* and *gran* constrained with a value constraint for feat1, so that the ARR can discriminate them.

When bilingual informants change the order of the TL words (by dragging and dropping them into a different position in the TL sentence using the TCTool), the Automatic Rule Refiner detects which word(s) were moved to a different position and extracts what were their initial (i) and final (i') positions. If the initial and final positions are contained within a the immediate dominating rule, then the ARR algorithm can extract the rule that immediately subsumes the constituents in both positions, make a copy of it and change the order of the appropriate constituents on the target language side of the rule copy. For the example above, if the grammar already has a general NP rule that reverses the order of the adjectives and nouns in Spanish, but is lacking a specific rule for pre-nominal adjectives, given relevant correction feedback (Figure 4.1), the ARR can extract the general NP rule (Figure 4.3) and flip the order of noun (N) and adjective (ADJ) on the TL side of the rule (Figure 4.4).

The next step is to further constrain the newly created rule so that it only applies in the right context, namely to *gran* but not to *grande*, by adding (feat1 $=_c$ +) to the ADJ in the TL side (Figure 4.5).

[15] A more mnemonic name for *feat1* would be *pre-nominal*.

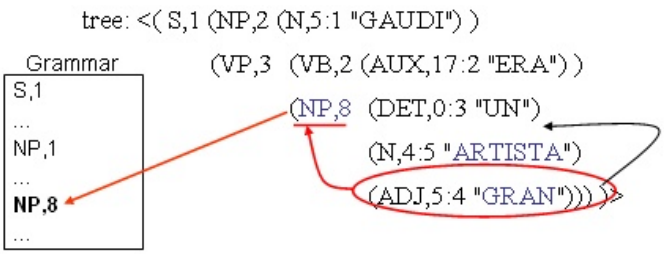


Figure 4.3. Translation Tree as output by the MT system is used for blame assignment. NP,8 subsumes the initial and final positions of *gran*: W_5 and W_4'.

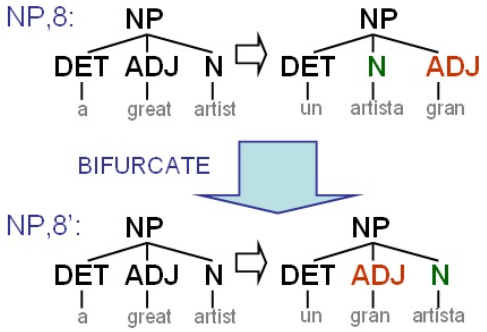


Figure 4.4. The ARR applies the BIFURCATE operation to rule NP,8, by which the order of the noun and adjective constituents is flipped.

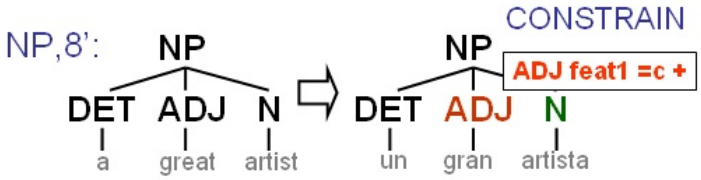


Figure 4.5. The new rule NP,8' is further refined by the ARR to include a value constraint for the adjective; this enforce the refined rule to only apply to adjectives with (feat1 = +).

These two refinements result in the MT system generating the desired translation, namely "*Gaudí era un gran artista*" and not incorrect translations with post-nominal or underspecified adjectives in a pre-nominal position ("**Gaudí era un grande artista*", "**Gaudí era un bueno artista*").

Furthermore, since the system already has the information that "*un artista gran*" is not a correct sequence in Spanish (from user corrections), the grammar can be further refined to also rule out this incorrect translation. This can be done by restricting the application of the general

rule (NP,8) to just post-nominal adjectives like *grande*, which in this example are marked in the lexicon with (feat1 = –). The general rule will also unify with any adjectives that do not have a value specified for feat1 (such as *bueno*).

Next, we move on to the formal description of the Rule Refinement framework and the mechanisms implemented by the Automatic Rule Refiner.

4.2 A Framework for Rule Adaptation

This section introduces the general framework for MT rule refinement or adaptation. The Transfer-Based MT system starts with an original grammar and lexicon[16] as well as the source language sentence that needs to be translated. After the translation process takes place, the system outputs a set of translations or target language sentences and their corresponding translation tree, which serves as a trace of what rules applied during the translation process (Figure 4.3).

In this context, the general goal of our Rule Adaptation or Rule Refinement system is to maximize coverage and translation quality (TQ) of MT output given the following information: the original Grammar (G) and Lexicon (L), the source language sentence (SL), a target language sentence produced by the MT system and selected by bilingual informants (TL), the translation tree output by the MT system (Tree), the corrected target language given by bilingual informants (CTL) and a set of Rule Refinement operations (RRop), which have been especially developed for the purpose of correcting translation rules, and which are described in Section 4.4.

This goal can be divided into two sub-goals:

Sub-goal 1: *Obtain user corrections and optionally some additional error information (CTL).*

Sub-goal 2: *Find optimal set of Rule Refinement operations and apply to current G and L, so that Translation quality after refinements is higher than before refinements:*

$$apply\ (RRop(G, L) \mid SL, TL, CTL, Tree) \rightarrow G'\ L'\ such\ that\ TQ_2 > TQ_1$$

In our approach, the first sub-goal is accomplished with bilingual speaker corrections obtained via the Translation Correction Tool as shown in the previous chapter.

The second sub-goal results in a computable minimal extension of the original grammar and lexicon, which we call G′ and L′, that is consistent with user corrections, namely that given

[16] One of the necessary steps to build an initial MT system is to build a lexicon, and thus even when dealing with resource-poor scenarios, we assume the existence of a basic lexicon or glossary.

the SL sentence, they can generate the correct TL sentence, as indicated by users. Note that even though minimality is a goal of the approach, there is no guarantee of actually reaching it.

In this thesis, we focus on developing a Rule Adaptation system that operates off-line. Having a system that can operate given just initial user corrections is crucial when user availability is limited or when we need to compute refinements with all data gathered so far. When no further user interaction is available, reasonable heuristics can be set as a default (such as always adding the constraint at the most specific level of grammar rules), so that the automatic rule refinement process can still be done reliably.

Moreover, trying to maximize Translation Quality over a regression test set is much more reliable than trying to optimize TQ over just one sentence. Thus, the problem of evaluating the effects automatic refinements have on a specific instance off-line can be reformulated as maximizing translation quality over a corpus ($CP_{Language}$) as measured by standard automatic evaluation metrics (Section 5.3):

$$arg\ max\ Automatic\_Evaluation\_Scores\ (CP_{TL} \mid (G', L', CP_{SL}, RefTrans))$$

where *RefTrans* is a set of reference translations for each source language sentences in CP_{SL} that is needed to calculate automatic metric scores (Section 5.3.3).

In sum, the main goal of this thesis is to find a computable minimal extension of an original grammar and lexicon that is consistent with user corrections, namely that is able to translate the SL sentence into the correct TL sentence, as indicated by bilingual speakers, and that improves MT quality over a test suite.

Before going into details about the formalization of the Rule Refinement operations, the formalism used for the translation rules deserves some attention.

4.3 The AVENUE Transfer Rule Formalism

Most Transfer-Based MT systems require at least monolingual modules for the analysis and generation of sentences, as well as transfer module to map equivalent representations for those sentences. In such transfer systems, a source language (SL) sentence is analyzed into a source language dependent representation (syntactic tree). This SL representation is then mapped to one or more equivalent representations of the target language (TL), as specified by the transfer module mapping. In a final step, one or more sentences are generated from all the legal TL representations of the SL sentence.

In the AVENUE MT system, the translation grammar unifies the three components that were traditionally divided into three modules: analysis, transfer, and generation information. This is similar to the modified transfer approach discussed in the literature (Aho & Ullman (1969) and early METAL system (Hutchins & Somers, 1992)). In this regard, even though the AVENUE translation rules combine information from all three stages, the transfer engine uses that information in several stages (Peterson, 2002).

The fact that all the syntactic information required for translating a SL sentence into a TL sentence is condensed into one single module, namely the translation grammar, is a key element for automatic rule refinement, since this significantly simplifies blame assignment (Section 4.6.2.1).

The AVENUE system translation rules encode six types of information: (1) Unique identifier or rule type, which in most cases corresponds to the top-level source constituent (Sentence, Noun Phrase, Verb Phrase, etc.), plus an integer index into all rules with that constituent name; (2) Constituent sequence for both SL (on the left-hand side) and the TL (on the right-hand side), headed by the mother constituent of the SL followed by the mother constituent of the TL separated by two colons (typically S::S, NP::NP, VP::VP, etc.); (3) Alignments between the SL constituents and the TL constituents; (4) SL-side constraints, which are defined as equalities of grammatical features in the SL sentence (*Analysis*); (5) SL-TL-constraints, which provide information about which feature values or agreements transfer from the source into the target language (*Transfer*), and (6) TL-side constraints, which are defined as equalities of grammatical features in the TL sentence (*Generation*).

In our formalism, source rule constituents are referred to with an X followed by an integer, starting at 0 for the mother of the rule (x0), and ranging from 1 to *n* to enumerate all the daughters (x1, x2, x3,… x*n*); whereas target rule constituents are referred to with a Y followed by an integer (y0; y1, y2, y3,… y*n*). See row labeled 2 in Figure 4.6 for an example. Thus SL-side constituent and constraints are often referred to as X-side constraints and TL-side constraints are often referred to as Y-side constraints; XY-constraints refer to transfer constraints.

Figure 4.6 shows an example of English to Spanish translation rule for noun-phrases containing a noun and an adjective. This translation rule swaps the original English word order, from adjective-noun to noun-adjective, and enforces their agreement in Spanish. The notation NP::NP indicates that a noun phrase (dominated by an NP node) on the SL side translates into another noun phrase on the TL side. In this case, the constituent sequence is comprised by parts-of-speech: [ADJ N]→[N ADJ]. In addition to parts-of-speech, a constituent sequence can also

contain other constituent labels (NP, VP, etc.) and a terminal node, namely a word in the source (left) or target (right) language surrounded by quotes.

```
      ; Rule identifier
1     {NP,9}
      ;  Source and Target Production Rules
2     NP::NP : [ADJ N] → [N ADJ]
      ; x0   y0     x1    x2       y1   y2
      ; Constituent Alignements
3     ( (x1::y2)  (x2::y1)
      ; Analysis Unification Equations
4     ((x0 mod) = x1)                ; the adjective is the modifier
4     (x0 = x2)                      ; the noun is the head
      ; Transfer Equations
5     (y0 == x0)
      ; Generation Equations
6     (y2 == (y0 mod))
6     (y1 = y0)
6     ((y2 agr) = (y1 agr)))
```

Figure 4.6. English-Spanish translation rule for NP; x here means source and y, target.

The remaining portion of the transfer rule specifies alignments of the constituents in the SL side with the constituents in the TL side (3) and feature constraints (4-6).

Transfer rules are based on a context-free Phrase Structure Grammar augmented with feature structures and unification equations. Feature structures are sets of attribute-value pairs, such as ((x0 number) = pl), where x0 is a feature structure associated with a constituent in the phrase structure representation. Unification equations are used in order to combine or compare information. Unification is an operation on feature structures, which has two possible outputs: if the input consists of two compatible feature structures, the output will be the most general feature structure that contains all the information in the inputs. Otherwise, if the input consists of incompatible feature structures, unification will fail. For example, ((x1 number) = (x2 number)) will succeed if the values for these attributes are equal or if either x1 or x2 do not have a value for the attribute number. On the other hand, if x1 and x2 have different values for number, unification will fail (Tomita, 1990; Tomita & Knight, 1987).[17]

[17] The Transfer system implements pseudo-unification instead of full unification. In full unification, the right-hand and left-hand sides of the equation are actually unified, while in pseudo-unification only the left-hand side of the rule is unified with the right-hand side, which is not modified (Tomita, 1990; Tomita & Knight, 1987).

In the transfer rules, the equal sign is the symbol used for unification. If the equal sign is followed by a *c* (=c), it means that the equation is a constraint equation, in which case the left-hand side and right-hand side values have to be non-empty for unification to succeed (Peterson, 2002). For instance, the unification constraint ((x1 number) =c (x2 number)) would fail if either x1 or x2 are unspecified for number, while the assignment equation ((x1 number) = (x2 number)) would succeed under the same circumstances, copying the value for number of x2 into x1's feature structure.

There are two different types of unification constraints in our formalism. *Value constraints* specify a feature value for an attribute in one of the rule constituents. For example, ((x1 number) = pl) enforces that the attribute number of the first constituent on the left-hand side (SL) has the feature value pl. *Agreement constraints*, on the other hand, enforce agreement between the features values of two constituents in the rule for a given attribute, such as number and gender. ((y1 number) = (y2 number)), for example, enforces number agreement between the first and second constituent on the TL side.

The first two feature constraints in Figure 4.6 are analysis or x-side constraints (4), which affect just the SL sentence; next constraint (5) is a transfer or xy-side constraint, and the last three constraints (6) are generation or y-side constraints, which affect the TL sentence.

In the AVENUE MT system, translation rules can be written by hand, or can be acquired automatically by a Rule Learner. The Rule Learner automatically infers hierarchical syntactic transfer rules similar to the one in Figure 4.6. For more details about the AVENUE rule formalism and Rule Learning, see Probst (2005).

Finally, the AVENUE MT system requires one translation lexicon for each language pair, with the option to add morphology modules for both languages, when available. For more information about the overall AVENUE MT system architecture and about the Transfer Engine, see Appendix A.

4.4 Rule Refinement Operations

There are two main types of refinement operations applicable to both grammar rules and lexical entries: CONSTRAIN, which specializes in handling features, and BIFURCATE, which specializes in handling structure. The application of one versus the other entirely depends on the Correction Action taken by bilingual informants. When informants edit a word, the CONSTRAIN operation applies, whereas when informants change the word order, add or delete a word, the BIFURCATE operation applies.

The **CONSTRAIN** operation consists of modifying an existing overly general rule, by adding one or more feature constraints, effectively replacing it with a more specific correct rule. Figure 4.7 shows an example of the resulting rule from a CONSTRAIN operation, R1, which is like the original rule (R0) but with an additional agreement constraint. Even though the rules in this figure (and Figure 4.8) appear lexicalized, this is just for illustration purposes and the rules in our grammar are, for the most part, not lexicalized.

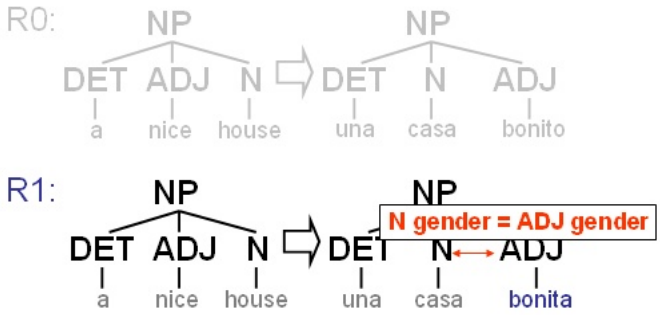


Figure 4.7. The CONSTRAIN operation adds agreement constraints to original rules that are overly general (R0) to achieve the right level of specificity (R1).

In the CONSTRAIN case, the new refined rule R1 needs to translate the same sentences as before plus the corrected sentence. The original general rule R0 becomes inactive and it is not used for further translations.

The **BIFURCATE** operation is used when the original rule is correct in general, but does not handle special cases. BIFURCATE makes a copy of the original rule (R0) and refines the copy (R1) so that it covers an exception to the general rule. For the example shown in Figure 4.8, the initial grammar only contains one correct rule handling post-nominal adjectives, but both post-nominal and pre-nominal adjectives are required to translate into Spanish. Thus, as a result of the BIFURCATE operation, both the original rule (post-nominal ADJ) and the refined rule (pre-nominal ADJ) coexist in the grammar

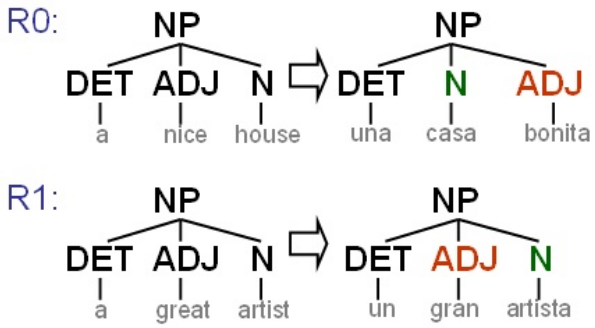


Figure 4.8. Example of the BIFURCATE operation applied to the general NP rule that deals with post-nominal adjectives in Spanish (R0) to also cover the pre-nominal order (R1).

The BIFURCATE operation is appropriate for cases when the general rule has correctly applied before. For example, we know that R0 (ADJ N → N ADJ) is right because it is used in translating *a nice house – una casa bonita*, and we want the grammar to also account for an exception to the general rule, in this case pre-nominal adjectives.

In general, if the new refined rule needs to translate the same sentences as before plus the corrected sentence, the original rule is substituted by the refined rule. However, if the refined rule should only apply to the corrected sentence, then after R0 bifurcates into R1, it can be further constrained via the CONSTRAIN operation by adding a feature constraint, so that the most specific rule only applies to the appropriate set of lexical entries. Figure 4.9 shows the restriction of the most specific rule to pre-nominal adjectives only.

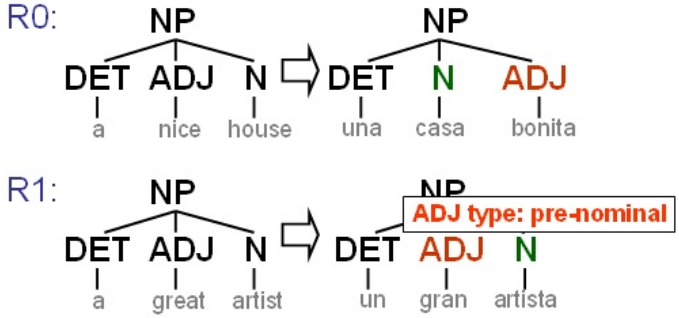


Figure 4.9. Rules result from the BIFURCATE operation can be further constrained with the CONSTRAIN operation to restrict its application to a specific set of lexical entries.

Furthermore, the CONSTRAIN operation can be used to also prevent the application of the general rule R0 to the same set of lexical entries. So that while R1 applies to pre-nominal adjectives, R0 only applies to all other adjectives, marked as post-nominal or underspecified for type.

When bilingual informants add or delete a word, alignment information is required to determine whether the BIFURCATE operation has to be executed at the lexical or at the grammar level. So for example, if a word is added and aligned to a SL word, which is also aligned to a contiguous TL word, then the BIFRUCATE operation is applied to a lexical entry, effectively adding a new form or sense for that SL word. On the other hand, if no alignment is added from the new word to a SL word, the BIFURCATE operation is applied to a grammar rule as shown in Figure 4.8.

Table 4.2 shows the different Rule Refinement schemata that have been implemented by the Automatic Rule Refiner, as well as their effect on Grammar and Lexicon ambiguity (A), namely the total number of alternative translations produced by the MT system. It is important to note that our approach does not create completely new rules that cannot be derived from existing rules in the grammar or in the lexicon. Therefore, if user corrections require a new rule not already in the original grammar (for example when dealing with syntactic divergences), this falls outside the scope of this thesis. Instead, the corrected translation pair can be used as a new training example by a Rule Learner like the one described in Probst (2005), for example.

Grammar	
CONSTRAIN	
RR1: R0 → R1 [+ *constraint*]	A[G] > A[G′]
BIFURCATE + CONSTRAIN	
RR2: R0 → R0 + R1 [≈R0 + *value_constr*]	A[G] < A[G′]
RR3: R0 → R0[+ (*feat* = −)] + R1[≈R0 + (f*eat* $=_c$ +)]	
Lexicon	
CONSTRAIN	
RR4: Lex0 → Lex1[+ *value_constr*]	A[L] ≥ A[L′]
BIFURCATE + CONSTRAIN	
RR5: Lex0 → Lex0 + Lex1[+ ≠ TLword]	A[L] < A[L′]
RR6: Lex0 → Lex0 + [+ (*feat* = −)] + Lex1[+ ≠ TLword + (f*eat* = +)]	
RR7: Ø → Lex1 (adding lexical item)	

Table 4.2. Main types of Rule Refinement schemata (RR) for grammar rules (R) and for lexical entries (Lex).

In the first Rule Refinement schema shown in Table 4.2 (RR1), the original rule is overly general and generates incorrect alternative translations in addition to the correct translation, therefore it needs to be made more specific for all instances of such rule application. In order to make a rule more specific a new constraint equation is added to the rule.

Naturally, CONSTRAIN operations make the rules in the grammar tighter reducing the ambiguity of the grammar (A[G] > A[G′]). This results in the MT system generating fewer translations.

In the second Rule Refinement schema (RR2), the original grammar rule (R0) bifurcates, and the Automatic Rule Refiner leaves the original as is, but modifies the constituent structure of the new copy of the rule (R1) to cover an exception to the general rule. There are multiple ways the constituent structure can be changed: adding a new constituent, deleting a constituent and changing the order of the constituents.

An example of constituent order change can be found in object pronouns in Spanish, which, instead of following the verb like object NPs, often appear in a pre-verbal position (*I saw you* → **Vi te* → *Te vi*), and thus the VP rule ([V NPobj]) would need to be bifurcated and the copy refined in order to generate sentences with the order flipped ([NPobj_pron V]), where the object is realized with a pronoun. Another example of this RR schema is depicted in Figure 4.8.

BIFURCATE operations alone effectively extend the grammar, rightfully increasing its ambiguity (A[G] < A[G′]). This results in the MT system generating more alternative translations. For all cases of BIFURCATE, the copy of the general rule (R1) can then be CONSTRAINed by adding a feature constraint to restrict its application to the specific syntactic and lexical context provided by the example sentence. Combining BIFURCATE and CONSTRAIN operations is meant to only let ambiguity increase to allow higher recall but not to significantly lower precision. See Figure 4.9 for an example.

With the third Rule Refinement schema (RR3), the ARR can further decrease grammar ambiguity by blocking the application of the general rule (R0) to the specific syntactic and lexical context provided by the example sentence, since it is already known that it should not apply in this case. Therefore, in addition to modifying the more specific rule as described for RR2, RR3 constrains the general rule with the addition of a blocking constraint, so that it applies to the earlier cases (where it translated correctly) and not to the new case (where it erred).

To follow up with the example of pre-verbal object pronouns in Spanish, if the ARR has information consistent with the fact that they cannot appear in a post-verbal position, then it can block the application of the general rule (R0) to object pronouns, and thus a constraint requiring the NP not to be of type PRON should be added (pron = −).

Feature constraints added to grammar rules only have an effect if lexical entries are properly tagged with those same features. If a lexical entry is underspecified with respect to the feature of interest, then only the general rule will apply, since refinements to the most specific rule (R1) explicitly require lexical entries to be tagged for that feature. In our transfer formalism, constraint equations use '$=_c$' to enforce that the right and left hand-side values are not empty. Both values must be specified and be equal for a constraint equation to succeed in unification.

At the lexical level, the first RR schema (RR1) is the equivalent of the first grammar RR schema (RR4). An example of RR4 is adding a feature constraint to all animated nouns, such as *woman*, *boy*, *Mary* (in contrast to *houses*, *book*, *feather*), in order to distinguish nouns with animate referents (animate = +) from nouns with inanimate referents (animate = −). The reason for doing something like this is that in Spanish animacy is marked explicitly in the sentence in front of object NPs (e.g. *I saw Mary* → *Vi a María*).[18]

RR5 bifurcates a lexical entry to add a different form or sense of the SL word. This occurs when the translation of an SL word required for a sentence is not the one in the lexicon, but a different one is. In this case, the ARR duplicates Lex0 and changes the TL side to match the translation proposed by the user, effectively adding a missing sense or form to the lexicon. For example, if users were given "*Wally plays guitar* → **Wally juega guitarra*", they would correct the translation of *plays* and change *juega* into *toca*, which is the right sense for playing an instrument in Spanish. If the lexicon only had an entry for [plays]→[juega], then RR5 would apply and generate a new entry ([plays]→[toca]) with the same feature constraints, but with the TL word modified. On the other hand, if users were given "*Mary and Anna fell* → **María y Ana cayeron*", they might correct the TL by adding a *se* in front of *cayeron* ("*María y Ana se cayeron*")[19] and add an alignment from *se* to *fell*. This is a case where RR5 would be adding a new form of *fell* into the lexicon ([fell]→[se cayeron]).[20]

RR6, like RR5, adds new senses and forms of words for a SL word already in the lexicon, and additionally, it adds value constraints to both the original and the copy to allow the ARR to

---

[18] Formal linguistic discussion about the exact use of the *a* marker in Spanish is omitted here purposefully, given that a system like the one described in this thesis would reflect speakers' preferences in terms of language usage, as opposed to any particular linguistic thesis or theory.

[19] "se cayeron" is the reflexive form of *cayeron* (to fall).

[20] Note that the RR schema for adding a new sense and adding a new form of a word is the same, and so there is no need to ask bilingual speakers to distinguish between these two types of errors. In this case, bilingual speakers aligned *se* to *fall* and thus the ARR executes a lexical refinement.

discriminate between the two senses/forms of word. For the *play* (*juega/toca*) example, this would mean that in addition to adding [play]→[toca] to the lexicon, RR6 would also add feature value constraints to both entries to be able to automatically distinguish them ([play]→[juega + (*feat* = −)] and [play]→[toca + (*feat* = +)]).

Finally, RR7 represents the schema required for out-of-vocabulary words, i.e., there is no lexical entry for the SL word aligned to it, and thus the system does not output a translation. In this case, the Transfer engine assigns one of the POS belonging to an open class word (specified in the configuration file), such as N and V, to the out-of-vocabulary word. The MT system's guess is taken as the POS of the new lexical entry. If POS taggers and morphology analyzers are available for the target language, they can be used to create a more reliable lexical entry. In the absence of both, however, reasonable heuristics can still be applied to increase lexical coverage and ultimately translation quality. Such heuristics will likely vary from target language to target language.

The first lexical schema reduces Lexicon ambiguity, together with a properly constrained grammar rule (A[L] > A[L′]), whereas RR5 results in ambiguity increase (A[L] < A[L′]); RR6 and RR7 result in lexicon expansion.

In conclusion, CONSTRAIN operations make the grammar and lexicon tighter, thus reducing the number of incorrect alternatives produced by the MT system, whereas BIFURCATE operations extend the grammar and its coverage, increasing the number of alternative translations to include the correct one.

In general, the number of constraints that are added to refined rules should be determined by the specific goal of the task, as well as by the type of MT system used in combination with the ARR. If the MT system includes a decoder to rank alternative translations and select the best alternative, the CONSTRAIN operation might become less relevant (Section 5.5).

4.5 Formalizing Error Information

The Automatic Rule Refiner represents TL sentences as vectors of words from 1 to *n* (*n*=sentence length), indexed from 1 to *m* (*m*=corpus length): $\overrightarrow{TL_m} = (W_1,...,W_i,...,W_n)$ and the corrected sentence TL_m' as follows:

$$\overrightarrow{CTL_m} = (W_1,...,W_i^{'},...,W_{clue},...,W_{n'})$$

where W_i represents the error, namely the word or multiword phrase that needs to be modified, deleted or dragged into a different position by the user to correct the sentence; and W_i' represents

the correction, namely the user modification of W_i or the word that needs to be added by the user in order for the sentence to be correct. If W_i is deleted, W_i' is empty.

In some cases, bilingual speakers will modify the form of an adjective (red-*roja*) to agree with the head noun of the noun phrase (car-*auto*), for example, which is captured by the system as W_i= roja and W_i'= rojo. In this case, it is essential to capture the information about which word triggered the corrections. Therefore, in addition to the error and correction words, we often also want to capture the word in the TL with which the corrected word has to agree. In the example given above, that word would be *auto*. We call this the clue word, W_{clue} in the formula above.

Clue words represent words that provide a clue with respect to what triggered the correction, namely the cause of the error. This way, instead of expecting non-expert users to determine whether the source of the error is a missing agreement constraint between a noun and the adjective that modifies it, non-experts can much more easily and reliably determine that the word that gives them the clue about *roja* (red+fem) needing to be corrected into *rojo* (red+masc) in the TL sentence is *auto* (car+masc). In general, the notion of clue word is intended to encapsulate information about the source of the error at a non-technical level, so that non-expert bilingual speakers can provide as much error information as possible without having to go into linguistically-motivated classifications of errors (Section 3.3).

W_{clue} can also be a phrase or constituent like a plural subject (eg. *[*Juan y María*] *cayó*, where the plural is implied by the conjoined NP). W_{clue} is not always present and it can be before or after W_i (i>c or i<c). They can be contiguous or separated by one or more words.

Section 4.4 described the different types of rule refinement operations that the ARR can perform to extend and improve a translation grammar and lexicon. But how does the system know what feature constraints are missing? We propose a mechanism for discovering what features triggered the correction in Section 4.6.2.2. Next, we turn to the general Automatic Rule Refinement algorithm.

4.6 Automatic Rule Refiner Algorithm

Our Automatic Rule Refiner algorithm is a three stage process. During the first stage, which we call *Error Information Extraction*, the ARR parses all the error information from the Translation Correction Tool log files and stores them into a Collection of Correction Instances (CIs). After minimizing noise introduced by non-expert users due to hesitation and actual mistakes, it ranks the CIs and process them sequentially. The second stage is the main one, where rule refinements to the lexicon and grammar are executed (*Lexicon and Grammar Refinements*). After initial blame assignment, lexical refinements handle out-of-vocabulary words, finding the triggering

features or postulating a new binary feature (for the modify case), and executing the necessary lexical extensions and adjustments. At this point, the ARR checks whether the translation produced by the refined MT system is already the same as the translation as corrected by the bilingual informant, and if so, it stops. Otherwise, it proceeds to execute grammar refinements. The third and final stage, *Refinement Verification*, loads the refined lexicon and grammar into the Transfer engine and validates the result of automatic refinements, making sure the translation as corrected by bilingual speakers is now in the final list of alternative translations. Otherwise, it flags it, suggesting manual inspection is required to handle that particular correction instance.

Figure 4.10 summarizes the main three stages of the ARR algorithm across correction actions in the sequence they are executed. After the first error extraction stage, and in order to process each Correction Instance, the algorithm splits into four refinement cases, which correspond to the four correction actions users can take with the TCTool (Section 4.6.3).

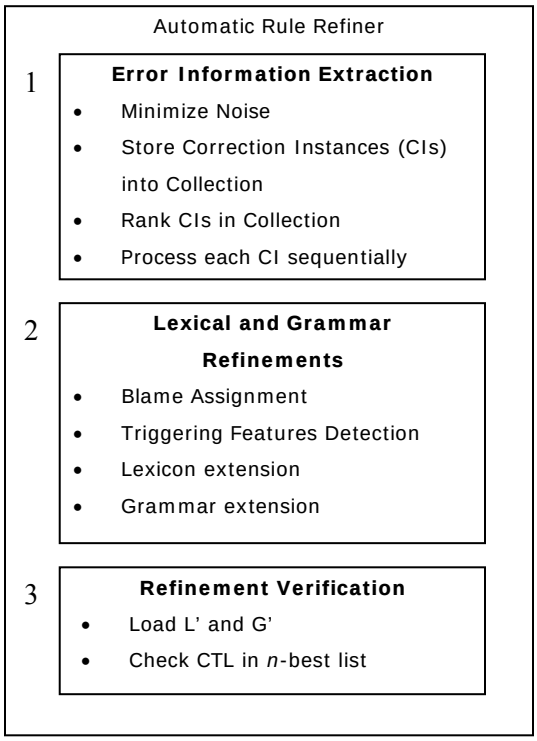


Figure 4.10. Diagram showing the three sequential stages in which the Automatic Rule Refiner processes error and correction information and executes refinements.

Chapter 5 is devoted to the evaluation of the core ARR module, namely the *Lexicon and Grammar Refinements*, focusing on evaluating the effects of a set of manually validated Correction Instances.

4.6.1 Error Information Extraction

Each translation pair corrected via the TCTool generates a log file. Each log file is processed and parsed by the Automatic Rule Refiner to extract all the relevant correction and error information (error and correction words, alignment information, etc.) and it is then stored into a Correction Instance (CI).

CIs store all Correction Actions taken by TCTool users, with related error information (Figure 4.11), into a vector of Actions. Correction Actions are processed by the Rule Refiner one at a time, following the algorithm described in Section 4.6.3.

Since users of the TCTool are not linguists or translation experts, the need to compare different correction instances and filter out noise becomes even more relevant. Since all posterior blame assignment and refinement decisions made by the system fully depend on the correct extraction and processing of error correction information given by non-expert bilingual speakers.

SL: John and Mary fell
TL: Juan y María cayeron
Alignments: ((1,1),(2,2),(3,3),(4,4))

Action 1: add new word (W_4'=se)
Temp_CTL: Juan y María se cayeron
Alignments: ((1,1),(2,2),(3,3),(4,5))

Action 2: add alignment (*fell—se* (4,4))

CTL: Juan y María se cayeron
CAlignments: ((1,1),(2,2),(3,3),(4,5),(4,4))

Figure 4.11. Correction Instance for Add Action storing the source language sentence (SL), the target language sentence (TL) and the initial alignments (AL), as well as all the correction actions done by the user, the corrected translation (CTL) and final alignments.

4.6.1.1 Minimizing Noise

It is crucial that the correction actions stored in our system correspond to the essence of what bilingual speakers did to correct a specific translation pair while using the TCTool. Even with just four correction actions (add, modify, delete, and change word order), users can choose to correct the same mistake in more than one way. For example, instead of modifying a word directly by editing it, deleting the incorrect word and adding a correct word would lead to the same final translation. In addition to user correction variation, bilingual informants often change their mind and sometimes even make mistakes.

Thus the goal of this component is to extract all the post-editing actions taken by bilingual informants and process them while filtering out as much noise as possible at this early stage, so that the error information can be used effectively by the rest of the system.

Even though the TCTool allows bilingual speakers to change alignments, the current system implementation does not consider changing alignments user corrections *per se*. Instead, the ARR algorithm uses alignment information to apply the appropriate refinement operations.

4.6.1.1.1 Spurious Loop Detection

There are several ways in which users change their mind, the first one being to correct a sentence that is already correct. If at some point during the correction session, the user decides to go back and mark the translation as being correct, the ARR ignores any correction actions registered and assumes the translation is correct, effectively filtering out the noise introduced by user's hesitation.

In other cases, users carry out a correction action and then change their mind. Examples of this are when users decide to add a word, but then realize that it is not needed, or modify a word from form1 into form2, and then decide that it was already correct before, and so changes form2 back to form1.

The ARR addresses all this issues with a Spurious Loop Detector. The Spurious Loop Detector operates by iterating over each action (A*i*) and searching for an action (A*i*') that will subsequently have had a reverse effect on the translation correction. Both A*i* and A*i*' are removed from the vector of actions the user performed. Then each action lying in between A*i* and A*i*' is updated to reflect the removal of A*i* and A*i*'. Such updates can result in even more actions removed from the user action history. More specifically, the following user actions can reverse each other:

- **Adding** and **Deleting** the same word (and vice-versa).
- **Editing** a word more than once (first action deleted if last edit on word reverts back to original word, first action change to last edit otherwise).
- **Changing Word Order** to previous order.
- **Adding** and **Deleting** the same SL-TL word alignment (and vice-versa)

Spurious Loop Detection runs in $O(|A|^2)$ time.

Given a source language and target language sentence pair, correctly detecting and discarding spurious loops allows for more reliable comparison of CIs that were parsed from log files generated by different users.

4.6.1.2 Collection of Correction Instances

In our Automatic Rule Refinement approach, having multiple bilingual informants compensates for their lack of technical expertise. Therefore, the idea is that several informants correct the same set of translation pairs, and their Correction Instances are stored in a Collection. This allows the ARR algorithm to further filter out corrections if only a small percentage of all the informants back them up. Inversely, if 90% of all informants agree on a specific correction, we want our algorithm to attach high confidence to it.

Having multiple correction instances affecting the same translation pairs (from multiple users) stored in a Collection allows the ARR to compare all the CIs affecting any given SL-TL pair and, if they contain equivalent information[21], they are stored only once with a weight proportional to the number of CIs that were found to be equivalent. For any given SL-TL pair, its weight directly indicates how much evidence there is in the data to support a correction action set as being more appropriate than another one with less weight. Namely, the relevance of a particular CI can be precisely estimated by its weight, which corresponds to the number of log files (and thus different users) that agree with it.

4.6.1.2.1 Error Complexity

In addition to taking into account the number of informants who agreed on a specific set of correction actions, the ARR also scores CIs according the complexity of their set of correction actions, or error complexity.

To estimate the error complexity of a given CI, both the number of errors addressed (approximated by counting different correction actions) as well as whether there is any dependency among the errors are factored in. The assumption here is that when two different correction actions affect the same word they are targeting the same error, and thus are considered dependent.

More specifically, CIs are sorted first by degree of dependency and then by coefficient, namely the number of clusters with that degree. For example, CIs with one correction action can be codified as (1); CIs with two independent correction actions, as (2), and with two dependent actions, as (1,0); CIs with three independent correction actions, can be encoded as (3), with two

[21] Equivalent CIs are CIs that in addition to having the same SL-TL and Corrected TL, once the spurious loops have been detected and removed, they also have the same set of correction actions affecting the same words.

dependent actions and one independent action, as (1,1), and with three dependent correction actions, as (1,0,0), and so on.

Descendent order of these vectors provides a natural and intuitive way to sort correction instances, since it correctly prioritizes CIs with a larger number of independent errors over CIs with smaller number of errors that are dependent among them: 001, 002, 003, 010, 011, 100, etc.[22]

The ranking algorithm below prioritizes correction instances with more user support and tackles simpler errors first.

Algorithm 1: Ranking CIs in a Collection

For each CI collection:

1. For each SL-TL pair, find the CI with the highest weight (more evidence) → BestCI
2. For each BestCI, compute error complexity
3. Rank BestCIs from lowest to highest error complexity

This algorithm picks the CI with most user support for each SL-TL pair (BestCI) and then computes their error complexity in order to rank simpler CIs higher. This "Tetris" approach is based on the underlying assumption that once simpler errors are fixed, more complex errors will be simplified (thus moving up in the ranking) and become easier to fix automatically.

4.6.1.2.2 Rule Dependencies

In addition to user support and error complexity, there is another aspect that becomes relevant when it comes to processing the Correction Instances in a Collection, namely whether any dependency exists between CIs. In this context, two CIs are not independent if the automatic refinements that they trigger affect the same set of rules or lexical entries, or at least if there is a partial overlap in the affected rules.

To be completely safe, after processing each CI in a Collection, the ARR would need to ask whether the rest of the sentences still need to be corrected, and if so, whether any correction action has changed. In order to do this, we would need to have a fully interactive system, which would refine the grammar and the lexicon at run-time and which would generate the next set of

[22] Currently, the implementation of error complexity does not take alignment correction actions into account.

translations with the newly updated grammar and lexicon (Section 7.1.3). This would make blame assignment a straightforward matter.

On the other hand, this would add time to the user interaction, and a Collection of Correction Instances can be safely executed when detecting rule dependency a priori. Therefore, after the ranking step, the processing of CIs can be executed to avoid undesired rule interactions. The greedy algorithm sketched in Algorithm 2 ensures that only rules that have not been refined before during a batch processing, get refined.

Algorithm 2: Processing CIs in a Collection

For each CI Collection (stack):

1. For each SL-TL pair, find the CI with the highest weight (more evidence) → BestCI
2. For each BestCI, compute error complexity
3. Rank BestCI with lowest error complexity higher
4. **If** first BestCI does not affect a rule that has already been refined by another CI in the Collection, process (see Section 4.6.2).
5. **Else**, push SL sentence from BestCI to a new stack (TCTool_Input), which is saved for the next iteration of user interaction, once G and L have been updated.
6. Pop up BestCI from the CI Collection stack and go to 4.

This principled approach to processing a Collection of CIs ensures that no undesired rule interactions will occur during automatic rule refinement. In some cases, however, it is safe to refine the same rule with multiple agreement constraints, for example DET-N and N-ADJ agreement constraints. In practice, experimental results have shown that a rule can be automatically refined multiple times in one batch without any undesirable rule interactions or negative effects on performance (Figure 5.2).

To address this issue, Algorithm 2 can be extended so that, given a CI in the Collection, in addition to determining whether the affected rule (R0 → R1) has already been refined in this iteration, the ARR could check whether the original translation (TL2) is still being produced by the refined MT system. If the same TL2 is being produced by the system, then it can apply the new refinement, which will lead to R2 (R1 → R2). If the new translation (TL2') is different from what it was at the time the bilingual informant corrected it, the ARR checks if it is the same as the Corrected TL (CTL) initially given by the bilingual informant, and if so, there is no need to process the CI, since the error has already been corrected. If the translation is different from the

CTL, then the SL sentence needs to be saved for further user interaction. Even though this extension of Algorithm 2 has not been implemented in the current version of the ARR, the pseudo-code is given in Algorithm 3 (Figure 4.12 depicts step 5 of Algorithm 3).

Algorithm 3: Possible extension of Algortihm2 (Processing CIs in a Collection)

For each CI Collection (stack):

1. For each SL-TL pair, find the CI with the highest weight (more evidence) → BestCI
2. For each BestCI, compute error complexity
3. Rank BestCI with lowest error complexity higher
4. **If** first BestCI does not affect a rule that has already been refined by another CI in the Collection, process (see Section 4.6.2).
5. Else, check if refinement triggered by a previously processed dependent CI (CI 1) changed TL sentence for current CI (CI 2), TL2.
    a. If TL2 has not been changed, process CI 2.
    b. Else, check whether TL2 = CTL2
        i. If TL2 = CTL2 (error has already been fixed by CI 1), go to 6
        ii. **Else**, push SL sentence from CI 2 to a new stack (TCTool_Input), which is saved for the next iteration of user interaction, once G and L have been updated.
6. Pop up BestCI from the CI Collection stack and go to 4.

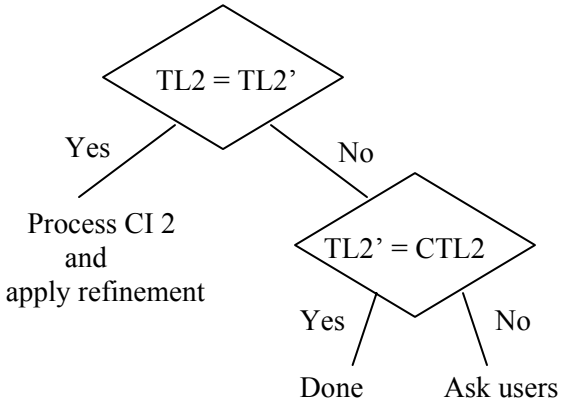


Figure 4.12. Possible way of handling dependent rules, according to the effect they have on final translations. This corresponds to step 5 in Algorithm 3.

Finally, we want to be able to asses the effect of processing any CI empirically. In Chapter 5, we will show that standard automatic evaluation metrics can be used to run the refined MT system on a test set to validate refinements. This way, if a refinement reduces accuracy of translation on held out data, we can choose to reverse the refinements by reverting to the original grammar (Section 4.6.2.1.1) and skip the CI that triggered the refinement.

4.6.1.2.3 Maximizing Corrective Effect

In addition to avoiding rule dependencies, when processing multiple CIs, ultimately we want to optimize the order in which CIs are processed to achieve the most corrective effect.

If we bifurcate a rule that needs to be made tighter, the order in which refinements take place (BIFURCATE vs. CONSTRAIN) will have an effect on how many examples the system needs to process in order to get the bifurcated rule to be tight.

Therefore, when operating off-line and processing several Correction Instances sequentially, executing CONSTRAIN operations before BIFURCATE operations maximizes their corrective effect. For example, if a CI (CI 1) indicates that an agreement constraint needs to be added between N and ADJ for rule NP,8 and another CI in the Collection (CI 2) indicates that the order of the constituents of NP,8 need to be flipped, if the ARR processes CI 2 first, then CI 1 will only apply to the original rule (NP,8); whereas if CI 1 is processed first, and then CI 2, both the original and the refined rule (NP,8') will have the agreement constraint.

The order in which CIs are processed that maximizes their corrective effect is summarized in Algorithm 4. This algorithm ensures that refinements at the grammar level, which have a broader impact, are processed first, and lexical refinements, which are most specific, are processed last. At the grammar level, refinements that make the rules tighter (CONSTRAIN) have precedence over the ones that augment syntactic coverage (BIFURCATE).

Algorithm 4: Maximizing Corrective Effect of a CI Collection

---

1. If there is a (set of) CI(s) in the Collection that triggers an agreement CONSTRAIN operation at the grammar level (edit case with clue word), process first.
2. Else if there is a (set of) CI(s) in the Collection that triggers a BIFURCATE operation at the grammar level (change of word order or add/delete cases without alignments), process next.
3. If there is a (set of) CI(s) in the Collection that only triggers refinements at the lexical level (edit case without clue words or add/delete with alignments), process last.

In addition to refinements to the grammar, step 1 in Algorithm 4 might also lead to refinements to the lexicon, if no feature attribute already exists to discriminate between the error and the correction (Section 4.6.2.2).

4.6.2 Lexical and Grammar Refinements

4.6.2.1 Rule Blame Assignment

After having correctly stored and processed error correction information, rule blame assignment is executed by the ARR. This is a key step of the rule refinement process, and is what differentiates Rule-Based MT systems from most Statistical MT or Example-Based MT systems. Namely, for systems that do not have explicit rules, an approach like the one proposed here cannot be applied directly.

Given the correction information and the transfer tree output by the transfer engine, the ARR can identify the incorrect rules and/or lexical entries, as the case might be, that are responsible for the error. The blame assignment algorithm varies depending on the type of correction action (modify, add, delete, change word order). Below we provide with the pseudo-code for each one of the blame assignment cases. Section 4.6.4 gives a more detailed discussion of blame assignment for add and delete cases. In Section 4.6.3, we describe the automatic refinement algorithm required for each one of these cases. A detailed example for the change word order case was given in Section 4.1; specific examples for the other cases can be found at the end of this Chapter (Section 4.7).

Algorithm 5: Blame Assignment for the MODIFY case

---

1. Extract lexical entry for error word (W_i)
2. Check if correction word (W_i') is in the lexicon, and if so extract
3. Compare lexical entries at the feature level (delta set, Section 4.6.2.2)
4. Apply appropriate refinement operation (LEXICAL REFINEMENT)

Algorithm 6: Blame Assignment for the ADD case

1. Check if added word (W_i') has any alignments to SL word(s)
2. If it does, follow alignments to find and extract the lexical entry that needs to be bifurcated to include W_i' (LEXICAL REFINEMENT)
3. Else, start at the top node of the translation tree output by the MT system and
    a. Check if W_i' can be inserted as a daughter of that node
    b. If W_i' is not embedded within a daughter of the current node, add in the right position to the right hand side (TL-side) of the rule that has the current node as the mother constituent (GRAMMAR REFINEMENT)
    c. Else, descend the tree to the next node (top-down, left-to-right) and go to (a)

Algorithm 7: Blame Assignment for the DELETE case

1. Check if deleted word (W_i) had any alignments to SL word(s)
2. If it does, follow alignments to find if any other TL words are aligned to the relevant SL words. If there is, check if [SL words-TL Words] is already in the lexicon; otherwise add it (LEXICAL REFINEMENT)
3. Else, look up the translation tree output by the MT system to see if W_i was inserted by a lexical entry or a grammar rule
    a. If W_i was inserted from the lexicon, add a new lexical entry for SL word (BIFURCATE), with an empty translation (LEXICAL REFINEMENT)
    b. If W_i was inserted by a grammar rule, extract and bifurcate the grammar rule, deleting the W_i constituent form the TL side of the rule (GRAMMAR REFINEMENT)

Algorithm 8: Blame Assignment for the CHANGE WORD ORDER case

1. Check translation tree output by the MT system and extract the rule that immediately dominates (subsumes) the initial (i) and final position (i') of the error/correction word
2. Bifurcate and flip the order of constituents Y_i and Y_i' on the TL-side of the bifurcated rule (GRAMMAR REFINEMENT)

4.6.2.1.1 Rule handling

In order for the blame assignment algorithm to be effective, the ARR pre-processes the lexicon and the grammar and assigns unique Rule IDs to all the entries.

To ensure fast look up of rules, red-black trees are used to index all rules by their respective Rule IDs. Additionally, lexical entries are indexed by their SL and TL sides, including exact and partial matches. Red-black trees are a balanced-tree data structure that ensures amortized look-up times of $O(\log|R|)$, where $|R|$ is the number of rules. Logarithmic lookup time is vital as the lexicon could potentially have hundreds of thousands of rules.

A Rule Hierarchy is used to keep track of refinements. Either if a rule is bifurcated or constrained, the refined rule is stored as a child of the original rule, indicating that it has been derived from it. In the bifurcate case, both the original and the copy co-exist in the grammar, whereas in the constrain case, the original rule is deactivated, and only the constrained rule is active.

Keeping all the versions of the rules in the Rule Hierarchy is not only useful to see the history of refinements that applied, but crucially it allows the ARR to guide automatic refinement applications according to MT system performance. Namely, if a specific refinement leads to MT system performance degradation, as measured by automatic evaluation metrics (Section 5.3), the ARR has the capability to reverse this refinement, by moving up in he hierarchy and restoring the state of the grammar before the refinement was applied. None of the automatic refinements executed by the ARR for our final evaluation (Section 5.1.1) lead to degradation of MT quality, and so this mechanism has not been fully tested.

Since refined rules are stored in a text file that needs to be parsed by the transfer engine, hierarchy information is stored as meta data encapsulated by comments that are ignored by the transfer engine. In general, all meta data specific to the ARR is stored as comments in the grammar and lexicon text files so as not to disturb transfer engine parsing.

4.6.2.1.2 Translation Trees

The AVENUE MT system outputs a translation tree for each alternative translation. The translation tree contains a precise trace of what translation rules were applied to what lexical entries in order to generate the target sentence that the user corrected. This is done via unique rule IDs displayed by the tree, which are used by the blame assignment process to retrieve the relevant rules that need to be refined (Figure 4.13).

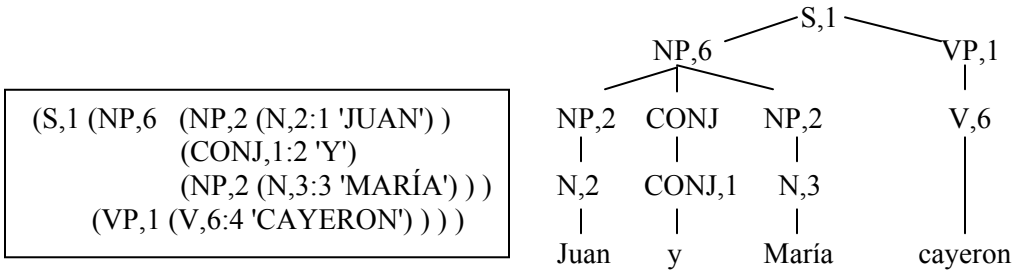


Figure 4.13. Translation tree output by the MT system for the SL sentence "*John and Mary fell*".

Given the Correction Instance in Figure 4.11, where the alignment information links *se* to *cayeron* (through *fell*), and the translation tree shown in Figure 4.13, the ARR determines that it needs to extract V,6 (fell-cayeron) from the lexicon.

Once the ARR extracts the lexical entries and the grammar rules that are responsible for the error, it proceeds to execute lexical refinements. In the modify case, it first proceeds to detect if there are any features that triggered the correction.

4.6.2.2 Correction Triggering Features

For the modify case, the Automatic Rule Refiner compares W_i and its correction W_i' at the feature level to discover which common feature attribute (or set of attributes, in cases where a correction fixes more than one error) has a different value in the lexical entries for W_i and W_i'. The common feature attribute that has different values is considered responsible for the correction and is thus called triggering feature.

We define the difference between an incorrect word and its correction as the set of feature attributes for which they have different values. We can extract the set of features and their values from the lexicon.[23] We call this the feature delta function (δ) and it can be written as

[23] If the lexicon contains roots, a morphological analyzer is needed to extract the features for each word.

$\delta(W_i,W_i')$. The resulting δ set can be one feature attribute, a set of feature attributes, which are all responsible for the correction, or the empty set.

If the δ set has one or more elements, this indicates that there is a missing feature constraint for all the attributes in the set. Examples of this can be found when comparing Spanish variations for *red* δ(*rojo,roja*)={gender} and *eat* δ(*comimos,comia*)={person,number}.

If the δ set is empty (which is always the case when W_i' is not already in the lexicon), that indicates that the existing feature set is insufficient to explain the difference between the error (W_i) and the correction (W_i') and therefore a new binary feature (feat_n) is postulated by the ARR in order to discriminate between the two entries. An example of two words that would not have any attribute with a differing value is δ(*mujer,guitarra*)={Ø}, since the lexical entries in our grammars are not marked for animacy. Note that it does not matter what new features are called, since in order to execute automatic rule refinements, the system only needs to discriminate between two different contexts, not determine the linguistically-motivated reason behind it.

Once the ARR has determined the triggering features, it proceeds to refine the relevant grammar and lexical rules by adding the appropriate feature constraints. If the user was able to identify a W_{clue}, the appropriate feature constraint is added between W_i and W_{clue}.

In the next section, we outline the algorithm for each refinement case. For full-fledged examples of modify, add and delete refinement cases see Section 4.7; A complete change word order example was given in Section 4.1.

4.6.3 Refinement Cases

The ARR algorithm applies different refinement operations depending on the type of correction action performed by users of the TCTool, including whether the user identified a clue word or not for any particular correction action.

Algorithm 9 deals with the different Rule Refinement cases, given one correction action (Section 4.6.1). When users identify a clue word, this allows the ARR to refine the appropriate rules without increasing grammar ambiguity unnecessarily. Automatic refinement of cases where there is no clue word available[24] and involve adding a new lexical entry without constraining its application typically result in a decrease in precision.

[24] Either because the user did not identify it, or because the context that provides the clue about the error/correction is not localized in one word, but rather in the syntactic structure.

For each different refinement case, the corresponding error type from the typology specified in Section 3.1 is indicated, as well as the specific type of Rule Refinement schemata (Table 4.2).

Algorithm 9: Rule Refinement Cases

MODIFY CASE: $W_i \rightarrow W_i'$

If modified word is in the lexicon:

1. a. If **$\delta(W_i, W_i') \neq \{\emptyset\}$**

E.g. *I see the red car* → **Veo el auto roja* → *Veo el auto rojo*

Error type: 7.2.1 (agreement)

RR1: Add constraints to the relevant grammar rule for all the features in the δ set between W_i and W_{clue} (*auto*), if identified.

1.b. Else **$\delta(W_i, W_i') = \{\emptyset\}$**

E.g. *Mary plays the guitar* → **juega la guitarra* → *toca la guitarra*

Error type: 6 (incorrect word: sense, form, etc.)

RR4: Postulate new binary feature (feat_n) and add with value + to the lexical entry for W_i' and with value − to the lexical entry for W_i.

RR1: If a W_{clue} has been identified (*guitarra*), add an agreement constraint to the relevant grammar rule between W_i, and W_{clue} ($(Y_i$ feat_n) = (Y_{clue} feat_n)). Otherwise, add a feature constraint with value + for constituent Y_i (Y_i feat_n = +).

If the W_i' is not in the lexicon as the translation of the SL word aligned to W_i, first add it by copying the lexical entry of W_i (BIFURCATE) and then go to 1.b.

ADD CASE: Ø → W_i'

If new word is aligned to an unaligned SL word, check lexicon for [SLWord] → [W_i'], if not there, add to lexicon.

2.a. If **alignment added to W_i'** (requires look-ahead method)

E.g. *The glass broke* → **el vidro rompió* → *el vidrio se rompió*

Error type: 3 (missing word, translation not in lexicon)

RR5: Add lexical entry with the SL word(s) aligned to W_i' in addition to previous TL words that were part of the lexical entry, if any:

([SL Word] → [(TL W_{i-1}) W_i' (TL W_{i+1})]).

2.b. Else if **no alignment added to W_i'**

E.g. *You saw the woman* → * *viste la mujer* → *viste <u>a</u> la mujer*

Error type: 3.3 (syntactic restrictions apply)

RR2/RR3: Bifurcate original rule and add new word ("W_i'") directly to the copy (Section 4.6.2.1 describes the algorithms for detecting what rule needs to be bifurcated).

If a clue word has been identified, the ARR postulates a new binary feature and adds it to W_{clue} with value +, and as feature constraint with '$=_c$ +' to the appropriate rule for the corresponding constituent (Y_{clue}), so that underspecified lexical entries do not unify with the refined rule.

DELETE CASE: W_i → Ø

3.a If there is an **alignment to another TL word**

E.g. *The boy broke the table* → *El niño <u>se</u> rompió la mesa* → *El niño rompió la mesa*

Error type: 4.1 (extra word)

RR5: If the SL word that used to be aligned to W_i is also aligned to another TL word, bifurcate the lexical entry for that SL word and delete W_i on the TL-side of the copy: ([SL Word(s)] → [$Ø_i$ TL Word]). For example, add new form of the word to the lexicon [broke→rompió] in addition to [broke→ se rompió].

3.b. Else, if there is no **alignment to another TL word**

E.g. *I would like to go* → *Me gustaría <u>que</u> ir* → *Me gustaría Ø ir*

Error type: 4.1 (extra word)

Check the translation tree output by the MT system to see whether the deleted word was inserted by the lexicon (RR5) or by the grammar (RR2).

RR5: If W_i was inserted by the lexicon, the lexical entry is bifurcated and W_i is substituted by the empty string on the TL-side: ([SL Word] → [""]).

RR2: If W_i was inserted by a grammar rule, the rule is bifurcated and the extra word is deleted in the copy.

CHANGE WORD ORDER CASE: W_i (...)W_j → W_j (...)W_i

4.a If **W_i W_j** are **subsumed by** the **immediately dominating rule**

4.a.1. If **W_i is the same** (= **W_i**)

E.g. *I saw you* → **vi* <u>*te*</u> → <u>*te*</u> *vi*

Error type: 5.1.1.1 (single word order change)

RR2/RR3: If the rule with the final order in the TL side does not exist, create one: BIFURCATE + flipped order of appropriate constituents: POS_j POS_i.

4.a.2. Else if **W_i changed (W_i → W_i')**

4.a.2.1. If, [SL W]→[TL W_i'] is not in the lexicon, assume **$POS_i = POS_i'$**

E.g. *Gaudi was a great artist* → **Gaudí es un artista* <u>*grande*</u> → *Gaudí es un* <u>*gran*</u> *artista*

Error type: 5.1.1.2 (word order change + modified word, same POS)

RR2/RR3: Flip POS_i POS_j in the relevant rule and solve as in 1.b (or 1.a if $\delta \neq \emptyset$).

All these cases have been implemented by the ARR. For case 4.a.2, if the POS had changed, as indicated by the lexicon or a POS tagger for the target language, the ARR can detect that the difference is the POS ($\delta = \{POS\}$), and thus can add a feature value constraint with that POS for the appropriate constituent to the grammar rule. However, since we did not have a sentence pair for this case, this was not fully tested.

4.6.4 Accuracy and Specificity Trade-off

The main goal of the ARR is to increase translation accuracy, whereas the secondary goal is to control ambiguity, so that accuracy is not achieved at the expense of ambiguity explosion.

This trade-off points to one of the key issues in automatic rule refinement. If refinements are so specific that they only affect one lexical item, then the lack of generalization prevents any accuracy gains that could have been achieved by such refinement on unseen data. On the other hand, when making refinements to the grammar, the goal is to be as specific as possible, since their generalization effect is greatly magnified due to the compositional and recursive nature of grammars: rules can plug into different slots, thus refining one rule typically affects multiple syntactic structures (and an infinite number of strings).

In an ideal world, we would have plenty of relevant examples available that would allow us to explore the space of all solutions and empirically determine the level of generality appropriate for each refinement. However, in the context of this thesis, such examples are not always readily available to test all possible refinements a system might need to do. Thus we are confined to a smaller region in the solution space. This region involves having a set of pre-

determined principles that govern these decisions. In Section 7.1.2 we discuss how Active Learning can be used to reduce the seeded version space that needs to be explored.

All correction actions that users can take exhibit this trade-off. Here we discuss only the two more complex cases, i.e., the ADD case and the DELETE case. Let us look at each one in detail.

4.6.4.1 ADD case

First the ADD case. Consider, for example, the correction that involves adding the object marker *a* to animate objects. What is the right amount of automatic refinement that needs to be done to the original grammar to strike a good balance between accuracy and specificity so that overall accuracy increases on held-out data, but this is not done at the expense of ambiguity explosion? To answer this we need to address the granularity of refinements as well as the right amount of context encoded by rules that need to be refined.

First, we turn to what is the right granularity required for each refinement. In theory, add and delete refinement operations can be done at the word level or at the POS level. However, without further information, it is impossible to know which level is most appropriate for each case. To avoid overgeneralization, when doing grammar refinements from just one instance, the ARR stays at the most specific level, i.e., the word level. Imagine that for the case in 2.b in the previous section, instead of adding an *a* in front of the NP object, the ARR had assumed that this was a general refinement that held for all prepositions and generalized from the word to the POS level by adding the constituent PREP in front of object NPs. The refined grammar would overgeneralize and license a large number of incorrect sentences (**viste contra la mujer*, **como en peras*, **él lee por un libro*, etc.)[25]. This is clearly undesirable when no further evidence is available, and so the ARR adopted the conservative approach of adding the most specific constraint at the grammar level.

This way, if a refinement is appropriate at the POS-level, a post-processing step can detect redundant rules ([“el” N ADJ], [“un” N ADJ], [“la” N ADJ], [“una” N ADJ], etc.) and generalize them into one single rule, once the ARR has encountered several other examples that validate such generalization ([DET N ADJ]). A fully interactive system could use Active Learning methods to determine what the right level of granularity for each case is at run time (Section 7.1.3).

[25] **You saw against the woman*, **I eat in pears*, **He reads for a book*, etc.)

On the other hand, in the add case, there is still the question of what is the relevant rule that needs to be refined. Since *a* is added between the verb (viste-V) and the determiner (la-DET), there are four potentially relevant rules in this case: VB,1, VP,1, VP,2 and NP,3 (Figure 4.14).

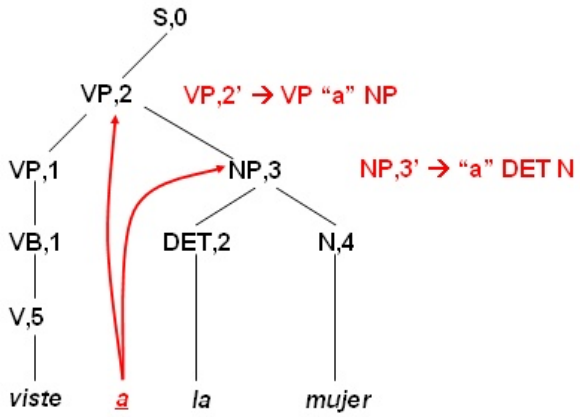


Figure 4.14. Translation Tree showing user insertion ("a") with two potentially relevant rules highlighted (VP,2 and NP,3). Other potentially relevant rules are VB,1 and VP,1.

If the Automatic Rule Refiner picked the NP, the refinement would be too general, since the rule does not contain all the necessary context that validates this refinement, and so all NPs in all sentence positions would appear preceded by an *a* (**A* *María jugó*, **Yo le doy a la manzana a a María*, etc.)[26], which is an incorrect generalization.

In such cases and to avoid overgeneralization, the Rule Refiner algorithm selects the rule that encodes the **most context**. This ensures that the refinement applies to syntactic environments most similar to the original corrected sentence. In this case, this immediately disqualifies VB,1 and VP,1, which only cover *viste*, and ensures that the refinement applies to object NPs only and not to all NPs. Figure 4.15 below depicts the amount of context for the two remaining candidate rules, form which it can be seen that VP,2 encapsulates more context than NP,3, and thus is more specific.

More formally, the ARR algorithm selects the highest rule in the translation tree for which the new word can be added as a daughter of the rule. If the word needed to be inserted within a daughter constituent at the current level, it goes one step down in the tree (See Algorithm 6 in Section 4.6.2.1). In general, an ideal system would create a seeded version space with SeedG

[26] **To Mary played*, **I give to Mary to the apple to Mary*, etc.)

(most general rule, with least context) and SeedS (most specific rule) for future exploration in case SeedS turned out to be more specific than needed (Section 7.1.2).

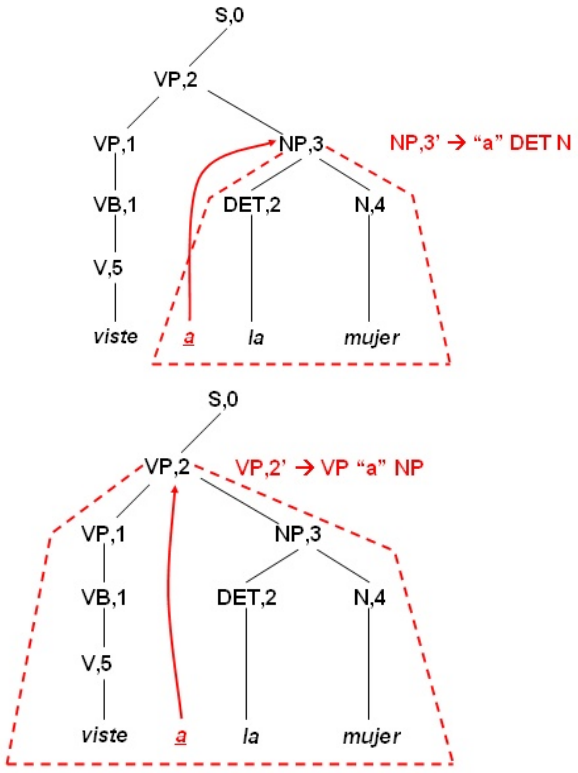


Figure 4.15. The context of NP,3 is depicted on the left tree ("a" DET N) and the context for VP,2 is depicted on the right tree (VP(VB (V)) "a" NP (DET N)).

So for example, the S rule is the highest rule in the translation tree, but *a* cannot be added as a daughter of it, as VP,2 contains it. Thus, at this point, the algorithm goes one step down in the tree and finds that *a* can now be added between the left and right daughters of VP,2, and so it picks VP,2 as the relevant rule for refinement (Figure 4.16).

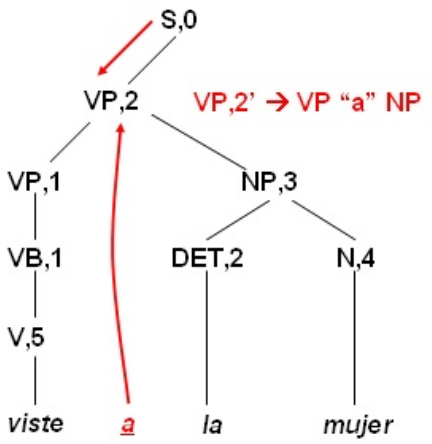


Figure 4.16. The Automatic Rule Refiner algorithm searches the translation tree top-down and selects the first node in the tree where the newly added word (W_i') can be inserted as a daughter.

The refined rule (VP → VB "a" NP) will now apply to all object NPs adding an *a* in front of them. The correct generalization in this case would be to add an *a* in front of all *animate* objects. However, in the absence of semantic features, the best the ARR can do is add an *a* to all objects, thus increasing ambiguity unnecessarily for inanimate objects, but allowing to correctly translate all animate objects in Spanish.

This refinement could be made more specific, if the goal of the ARR were to be as specific as possible and not allow any generalization not supported by data. For this, the ARR would postulate a new binary feature, add a value constraint to the lexical entry for [woman→mujer ((feat_n) = +)] and then add a feature value constraint to the NP in the refined rule (VP → VB "a" NP((feat_n) = +))), so that the refined rule would only unify with one particular lexical entry (most specific refinement possible).

Adding more constraints would increase precision, but then the refinement would not generalize to unseen words and syntactic structures. In this case, if the ARR added a constraint so that the new refined rule only applies to object NPs with the word *woman*, ambiguity would not increase, but then sentences with this structure but a different word (*women*, *girl*, *boy*, *students*, etc.) would not be generated correctly.

So even though ambiguity doubles for sentences with object NPs, when adding an *a* in front of all object NPs, this is justified by an overall increase in translation accuracy, and trying to further reduce ambiguity would hurt MT system's performance. In the absence of a decoder, to make refinements such as this one both general and correct, we would need to appeal to an external ontology, and use active learning to learn the right semantic class of the direct object that determines whether the preposition *a* should be used or omitted. When (semantic) features are

available in the lexicon, adding appropriate feature constraints to refined rules to restrict their application to the right class of lexical entries is the right thing to do.

The current implementation of the Automatic Rule Refiner trades off specificity (lack of ambiguity) with accuracy, since it the AVENUE MT system includes a decoder with a statistical language model whose job it is to select the best translation alternative from the final list of candidates (Appendix A). An alternative approach is to stop adding further constraints before overall translation accuracy on a regression set degrades.

4.6.4.2 DELETE case

The other interesting case, from the perspective of the accuracy vs specificity trade-off, is the DELETE case. Like in the add case, when users make a correction (in this case, delete a word from the TL sentence), the system has to determine what is the appropriate level of generality for the refinement. At the most general level, the rule refiner can delete the constituent (POS) that generated that word from the grammar rule. So for example, given the translation *"me gustaría que ir*" (*I would like to go*) and the correction "*me gustaría ir*", consider a case where the constituent sequence of the rule that generated it was VP [VP PREP VP] → [VP CONJ VP] (as indicated by the translation tree). This option would result in deleting the constituent CONJ from the TL side of the rule. This would imply that all other CONJs that could be generated by the original rule would now not be generated by the refined rule. This might be the appropriate level of generality required by this example; however, it is risky to assume this based on just one correction example.

At the other end of the spectrum, the rule refiner can add a lexical entry for that word with an empty string as the translation (PREP::CONJ :| [to] → [""]), so that effectively, in addition to the other translations given by the lexicon, the word deleted by users (with the POS indicated by the translation tree) can also be skipped over in the translation process.

An ideal middle ground can be achieved if the grammar rule contained the word as a terminal node (instead of a POS), and thus the rule refiner could delete the word in the right context, and not for every instance of that word with that POS (VP [VB "to" VP] → [VB "que" VP]). This option is analogous to adding the word to the rule as a terminal node ("a") as opposed to adding it as a non-terminal node (PREP).

Therefore, for the DELETE case, the right level of generality depends on the characteristics of the grammar rule to be refined. Given just one correction example, if a non-terminal node (CONJ) in the grammar rule generates the word deleted by the user correction, then it is safer to refine at the most specific level, namely the lexicon, and add an empty translation for

the SL word aligned to it. This is the approach implemented by the ARR, and it results into the refined MT system producing double the amount of alternative translations for all sentences where that SL word appears with that POS.

In sum, if the goal of the Automatic Rule Refiner were to increase accuracy and did not care about increasing ambiguity at all, then bifurcating grammar rules and not adding any agreement or value constraints to tighten the rule application to a subset of the lexical entries with a particular POS, would achieve that goal. However that would make the decoder's task much more challenging and in practice could prevent the correct translation from being found in a large *n*-best list of alternative translations, as shown in Section 5.4.3.3.

On the other hand, if the goal of the ARR were to make refinements that are as specific as possible, then all refined rules would have to be constrained to only apply in the same specific syntactic context of the corrected sentence and with the same lexical items that appear in that sentence. Even though this strategy is risk-free, its usefulness is clearly questionable, since refinements become very local and, in the worse case, corrections are required for each new sentence, like in the pure post-editing case.

The current implementation of the Automatic Rule Refiner attempts to find the balance between these two extremes by keeping grammar refinements as specific as possible (at the word level) while selecting the rule with the most context, and lexical refinements as general as possible (not adding feature constraints to restrict the application of a refined rule to only one lexical item).

These principles are language independent in the sense that, for any language pair, the ARR will stay in the space of making safe generalizations, and not make refinements that generalize over all sentences with the same syntactic structure from just one correction example.

4.7 Other Examples of Refinement Cases

In Section 4.1, we described an example of change word order case in detail. The following section describes detailed examples for each one of the three remaining correction cases: add, modify and delete. Empirical evaluation showing the effect of these refinement examples to other sentences is described in detail in Chapter 5.

4.7.1 Add a Word

When users add a word (by clicking on the [New Word] button on the TCTool interface and then by writing the word in the newly created box), there is no error word *per se*, however the ARR can reliably identify a correction word, namely the newly added word (Figure 4.17).

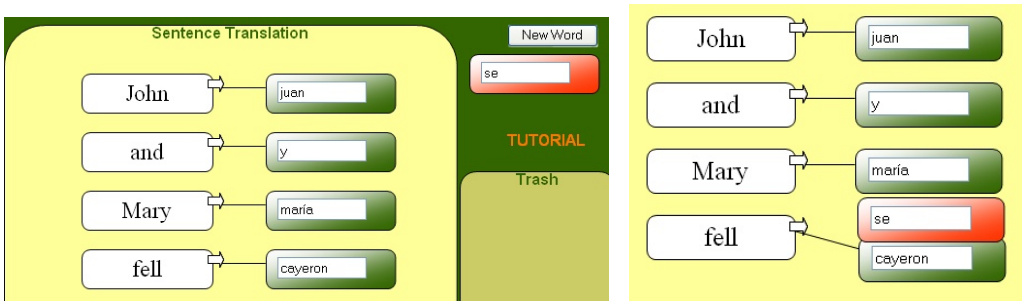


Figure 4.17. TCTool snapshots after having created a new word (*se*) [left], and after having added the newly created word into the right position [right] (Action 1).

After having added the new word, users drag it to the right position in the translation. This operation allows the system to instantiate the new word with a specific position in the Corrected Target Language (CTL) vector. In this case, the ARR instantiates W_i' to W_4' = se (Figure 4.17 right). The next step is to check if the user added any alignments from any word(s) in the SL sentence to the corrected word W_4', and if so, to extract them: in this case (4,4) (Figure 4.18).

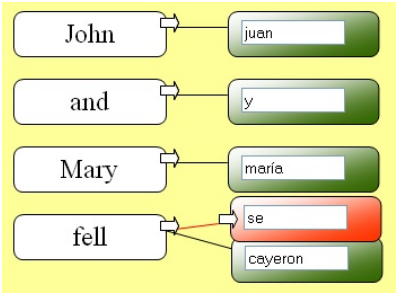


Figure 4.18. TCTool snapshot showing Action 2: Adding manual alignment.

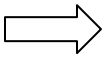
See Figure 4.11 for the Correction Instance that corresponds to the user interaction illustrated in Figure 4.17 and 4.18.

Alignment information is required to retrieve the relevant lexical entries and determine the necessary refinements. In our example, [fell→se] and [fell→se cayeron] are not in the lexicon, however [fell→cayeron] is.

At this point, the ARR BIFURCATES the lexical entry [fell→cayeron] creating a copy of it and extending the TL side to also contain W_i', in addition to the other word aligned to the SL word: [fell→se cayeron]. The resulting refined entry is displayed below in Figure 4.19.

V::V |: [fell] → **["cayeron"]** V::V |: [fell] → **["se cayeron"]**

((x0 form) = fall)
((x0 actform) = fell)
((x0 tense) = past)
((y0 agr pers) = 3)
((y0 agr num) = pl)



((x0 form) = fall)
((x0 actform) = fell)
((x0 tense) = past)
((y0 agr pers) = 3)
((y0 agr num) = pl)

Figure 4.19. Original and refined lexical entry result of BIFURCATE operation.

The new lexical entry is added to the lexicon and the Refined Lexicon is loaded to the MT system to assess the effect of the rule refinement.

The translation alternatives output by the system are now checked against the CTL sentence as corrected by the user. If the ARR finds that CTL sentence is being generated by the MT system, it stops, otherwise, it proceeds to grammar refinements. For this example, the algorithm described above successfully refined the lexicon and the output of the refined MT system already contains the CTL sentence.

If the word added (W_i') is not aligned to any word in the SL sentence, then there is nothing to be done at the lexical level and the algorithm proceeds to grammar refinements (Figure 4.20).

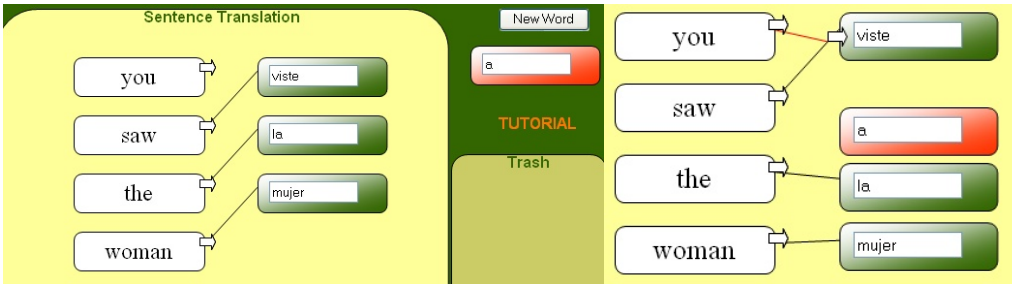


Figure 4.20. TCTool snapshots after having created a new word (*a*) [left], and after having added the newly created word into the right position [right].

The first step is blame assignment from the translation tree. For example, given the translation pair "*you saw the woman − viste la mujer"* and the user correction of adding the word *a* in front of *mujer* (W_2' = a), as shown in Figure 4.20, the ARR detects that *a* is not aligned to

any words in the SL sentence, and it proceeds to look at the translation tree to extract the appropriate rule that needs to be refined.[27]

In this case, since *a* is inserted between *viste* (V) and *la* (DET), there are four candidate rules for refinement, namely VB,1, VP,1, VP,2 and NP,3. Adding an *a* in the right position to any of these rules ([VB "a"], [VP "a"], ["a" DET N] and [V "a" NP]) would have the desired effect for this example. However, only the last option generalizes well to other sentences (Section 4.6.4). In general, to handle these cases off-line, the ARR needs to refine the rule that encodes the most context. Figure 4.21 shows the different amounts of context for VP,2 and NP,3. A top-down algorithm (Algorithm 6 in Section 4.6.2.1) ensures that the refinement applies to syntactic environments most similar to the original corrected sentence. In this case, this means the refinement applies to object NPs only and not to all NPs, namely that the refinement is applied to VP,2.

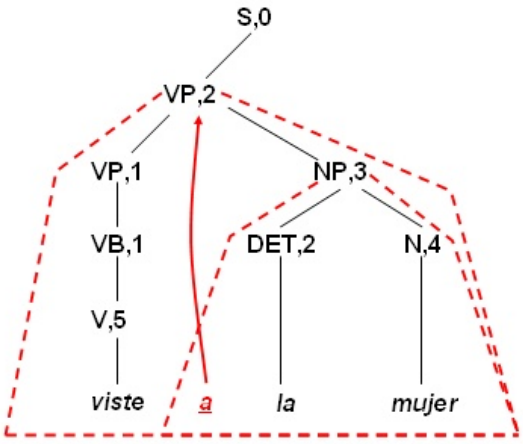


Figure 4.21. Depicting context captured by each candidate rule. VP,2 encapsulates more context than NP,3, and thus is more specific.

This is still not the ideal level of generalization, since one would want to only add an "a" in front of animate object NPs in Spanish. The ARR could further refine the bifurcated rule to have a value constraint that restricts its application to NPs with 'mujer' as a head. However, even if the lexicon is labeled with the appropriate semantic features (such as animacy), in the absence of further user interaction, not adding any further refinements is the best strategy to strive for high accuracy and control unnecessary ambiguity.

---

[27] The reason for this correction is that in Spanish, direct objects with animate referent are marked with an 'a' (*viste a la mujer* vs. *viste la guitarra*).

4.7.2 Modify a Word

When users modify a word (W_i) into a related form or sense (W_i'), there are two possible scenarios. The one most favorable to generalization is that the lexicon already discriminates between these two forms, by giving them a different value for the same feature attribute (example: [*red-roja* gender: fem] and [*red-rojo* gender: masc]). The one with less immediate impact is that the two words are identically defined in the lexicon, namely they have the same POS and the same feature attributes and values (e.g. [women-mujer] and [guitar-guitarra] are both singular feminine nouns in Spanish).

If the lexicon already discriminates between the two lexical entries, the ARR extracts the grammar rule for the immediate common parent of W_i and W_{clue} (as identified by the user) and adds an agreement constraint with the triggering feature[28] between the constituents corresponding to W_i and W_{clue}.

SL: I see the red car
TL: veo el auto roja
Alignments: ((2,1),(3,2),(4,4),(5,3))
Action 1: edit (W_4=roja → W_4'=rojo; W_{clue}=auto)
CTL: veo el auto *rojo*
CAlignments: ((2,1),(3,2),(4,4),(5,3))

Figure 4.22. Correction Instance automatically extracted from user edit action via the TCTool.

For the correction instance represented in Figure 4.22 (*I see the red car)*, the user edits *roja* into *rojo* (by clicking on the word and changing *-a* into *-o*), and the system finds that the difference (delta set) between the lexical entry for *roja* and *rojo* is [agr gen].

At this point, the ARR moves on to the Grammar Refinements.

Since the user identified *auto* as being the clue word (Figure 4.23), the ARR algorithm can now instantiate what variables W_i and W_{clue} correspond to in the relevant rule (**NP,8**: ADJ N → N_{clue} ADJ_i), namely the system internal variables that represent the TL adjective and noun (*Rule Variable Instantiation*).[29]

[28] The triggering feature is the attribute name for which the two lexical entries have a different value (Section 4.4.1).
[29] The relevant rule is extracted from the translation tree output by the MT system, making blame assignment straightforward.

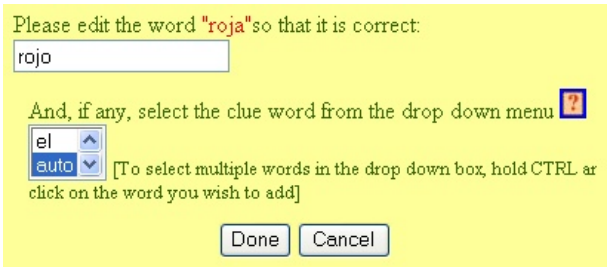


Figure 4.23. Edit Word window eliciting for clue word information.

Next, the Automatic Rule Refiner adds an [agr gen] constraint to rule NP,8 between the noun and the adjective: **NP,8**: ADJ N → **N_{clue} ADJ_i [(N_{clue} agr gen) = (ADJ_i agr gen)]**.

However, if the lexicon does not already discriminate between the two lexical entries (W_i and Wi′), the ARR postulates a new feature attribute and adds a binary value constraint to each lexical entry, in order to allow the grammar to distinguish between the two forms/senses of the same SL word automatically.

For example, given the sentence "*Mary plays guitar*" and its translation as produced by our MT system, "**María juega guitarra*", the user will edit *juega* into *toca*, and since this new sense is not listed in the lexicon, the ARR will BIFURCATE the original lexical entry [play→juega] and CONSTRAIN it by replacing the TL side. Naturally, [play→toca] is otherwise an exact copy of [play→juega] (with the same POS and features), and so the system postulates a new feature (*feat_0*) to distinguish between the two and adds the following constraints to the lexical entries:

[play→toca$_{((feat_0) = +)}$] [play→juega$_{((feat_0) = -)}$]

Note that in the absence of a semantically annotated lexicon, our approach will only be able to solve such errors on a case by case basis.

If no clue word had been identified by bilingual informants, our current implementation of the ARR would not add a constraint to the appropriate rule. Nevertheless, if the triggering feature already existed in the grammar, an alternative is to look for all the words in the TL sentence that have the same value for the triggering feature (in the first example above [agr gen]) and hypothesize agreement constraints between them and W_i'.

4.7.3 Delete a Word

If users delete a word, first the ARR algorithm needs to make sure this is not followed by a word being added in the same position, which is the equivalent to editing a word. After making sure it is really a delete case (Section 4.6.1.1.1) , the ARR algorithm checks if there were any alignments from the deleted word (W_i) to one or more SL words, and if so, it looks ahead to see if there was

any other word in the TL sentence that was aligned to the SL word(s) at a later point in the correction session. If there is a TL word aligned to any of the relevant SL words, then the ARR algorithm checks if [SL words-TL words] is already in the lexicon, and if not, it adds it.

If no alignment is added to the relevant SL word(s), there are two possible scenarios: 1) W_i comes from the lexicon, or 2) W_i is inserted by a grammar rule. In either case, the translation tree will indicate how W_i was generated. In the first scenario, the ARR algorithm adds a new lexical entry for the SL word with an empty TL side ([SL word → Ø]). So for example, given the translation *"me gustaría que ir*" (*I would like to go*) and the correction "*me gustaría ir*", if *que* is generated by the lexical entry [to → que], the rule refiner bifurcates it and substitutes *que* with an empty string as the translation ([to] → [""]), so that effectively, in addition to the other translations given by the lexicon for *to*, it can be skipped over during the translation process.

In the second scenario, the ARR retrieves the appropriate rule and bifurcates it, so that it can delete the word constituent from the copy. Consider a case where the constituent sequence of the rule that generated it was VP [VB "to" VP] → [VB "que" VP] (as indicated by the translation tree output by the MT system). In this case, the automatic refinement would delete "que" from the TL-side of the VP rule (VP' [VB "to" VP] → [VB VP]). This refinement would encode the most context and thus would be preferred to control ambiguity, while increasing recall. However, in the delete case, the level of automatic rule refinement entirely depends on whether the deleted word was originally inserted by the lexicon or by the grammar (See Section 4.6.4.2 for a more detailed discussion).

4.8 Discussion and Conclusions

The main goal of the Automatic Rule Refiner is to extend the lexicon and the grammar to account for new cases or exceptions not originally encoded in the translation rules. A secondary goal is to make overly general rules more specific to reduce grammar ambiguity.

In this chapter, we have described how automatic refinements of translation rules in the grammar and the lexicon can effectively tighten (CONSTRAIN) and extend (BIFURCATE) the original grammar and lexicon to reduce undesired ambiguity and cover new syntactic structures and lexical items. The difference between this approach and mere post-editing is that the resulting refinements affect not only the translation instance corrected by the user, but also other similar sentences where the same error would manifest itself. As an example, after the refinements in the change of word order example introduced above (Section 4.1) have been applied to the grammar, sentences like "*Irina is a great friend*" and "the young professor is a great person" will now correctly translate as "*Irina es una gran amiga*" and "*el profesor joven es una gran persona*"

(instead of "**Irina es una amiga grande*" and "**el professor joven es una persona grande*"). Thus, generalization goes beyond just lexical variation, and applies to constituent generalization. Moreover, the generalization power of this approach is greater when refinements involve information that is already encoded in the lexicon and the grammar (such as gender, number, person and case). In our lexicons, this means mostly errors of lexical and morpho-syntactic nature.

On the other hand, most semantic errors require the system to postulate a new binary feature to distinguish between the different senses of the word. Since the ARR cannot populate other lexical entries with newly hypothesized features automatically, in the absence of a generalization mechanism, this process represents just a first step towards semantic correction. An extension of this work could be to query an external ontology, and derive semantic distinctions from it.

The space of solutions for an Automatic Rule Refiner is large and there is a clear tradeoff between adding constraints to control ambiguity and losing refinement generality.

Adding generation capabilities to the model by always bifurcating, for example, increases ambiguity exponentially. In a compositional translation system, where rules plug into each other, blind bifurcation increases the complexity of the grammar unnecessarily and can pose a serious problem.

Adding more feature constraints will decrease ambiguity, but unless features already exist in the lexicon, refinements will not generalize over unseen words and syntactic structures.

In the ideal oracle case, only the right constraints at the right level of generality, are added to the model. In the practical learning case, the Automatic Rue Refiner learns refinements that increase the generation capabilities, but do not increase ambiguity exponentially (much like a partial constraint oracle). When applying refinements automatically, it cannot always be determined whether a refinement should be as general as possible or, on the contrary, it should only be made as specific as possible. Therefore, choosing between prioritizing generalization versus ambiguity reduction becomes a practical matter and can be decided according to specific needs. In Section 7.1.2, we describe other generalization strategies involving Active Learning methods to reduce the seeded version space, using either an existing corpus or a bilingual speaker as oracle.

In the next chapter, we present experimental results and extensively discuss the last stage of the Automatic Rule Refiner, *Refinement Verification*.

Chapter 5

Experimental Results: English-Spanish

The ultimate goal of the Automatic Rule Refiner is to improve MT output quality on previously unseen text. In order to measure the actual impact of automatic refinements, we evaluate a baseline English-Spanish MT system comparing it to its refined version along several dimensions.

The first question we want to ask is whether the MT system with refined rules is able to generate a correct translation among its output candidates for sentences that did not produce a correct translation before (thus yielding an increase in recall). The second relevant question is whether the output candidate set contains fewer incorrect translations (thus yielding an increase in precision). To answer both these questions, we look at the final list of all candidate translations produced by the system.

Since we seek to increase both precision and recall (i.e., accuracy and coverage), we also provide results for a system that includes a statistical decoder to select the most likely phrases and rank alternative translations in the *n*-best list according to a statistical language model among other features (Appendix A). We refer to a system including a decoder as an end-to-end system, since after initial MT output corrections, there is no other human intervention in the refinement and translation process, and just the first-ranked translation (first best) is considered for evaluation. End-to-end system results are expected to be significantly lower than *n*-best list quality evaluation results. Yet, improvements observed in such a set up constitute solid proof that our approach to Automatic Rule Refinement has a positive impact on final translation quality produced by a MT system, without further human intervention.

In this chapter we report experiments on three different data sets, using a baseline system and two refined systems. For the two development data sets, we focus on *n*-best list quality results, namely does the *n*-best list contain the correct translation (oracle), and for the unseen test data set, we focus on end-to-end system results. We provide results for initial decoder weights as well as results for the decoder with optimized weights; we include both lower-bound and upper-bound results. For all these experiments, we abstracted away from morphology and thus the translation lexicon contains fully inflected forms. Evaluation of an MT system including morphology modules is described in Chapter 6.

5.1 Data

As development data, we used a Diagnostic set designed to exhibit all the error types we want to refine automatically. This Diagnostic set (DSet) allows us to do a detailed glass-box evaluation of each type of refinement and its effect on a small set of sentences, which are similar to the ones that triggered the user corrections (training set in Figure 5.1), but are not the same. Part of the AVENUE Elicitation Corpus (EC) is used as validation data, and a pre-existing evaluation test set from the Basic Travel Expression Corpus (BTEC) is used as blind test data. The BTEC data is a good candidate corpus for our blind evaluation, since parallel data is available for a number of different languages, including English and Spanish, and we had never seen it before, nor did we decide what sentences were in the test set.

Data			English
DSet	Development	Sentence Pairs	55
		Avg. Sentence Length	4.9
		Word Tokens	271
		Word Types	112
EC	Validation	Sentence Pairs	200
		Avg. Sentence Length	3.8
		Word Tokens	766
		Word Types	121
BTEC	Train (decoder)	Sentences Pairs	123,416
		Avg. Sentence Length	7.3
		Word Tokens	903,525
		Word Types	12,578
	Test	Sentence Pairs	506
		Avg. Sentence Length	6.6
		Word Tokens	3,764
		Word Types	776

Table 5.1. Corpus Statistics for the source language

We are interested in seeing how well user corrections generalize over unseen data, but would like to focus on MT quality, rather than on lexical coverage alone, hence we entered lexical entries from these corpora. Corpus statistics for all three data sets are shown in Table 5.1.

First we turn to the description of the Set of Correction Instances that was used to refine the different baseline MT systems used in our experiments.

5.1.1 Training Set: Correction Instances

The English-Spanish refined MT system results from having the Automatic Rule Refiner process a manually validated set of Correction Instances and applying refinements to the baseline grammar and lexicon (Section 5.2). In the evaluations reported in this chapter, we abstract away from multiple user feedback and focus on relevant and correct Correction Instances (CIs) containing one or two errors drawn from the Elicitation Corpus. The set of CIs used in this evaluation were manually selected from the pool of CIs created by the subjects of the user study described in Chapter 3 (see Section 3.3 for details). The final Correction Set (CSet) was chosen since it increased recall without proportional loses in precision on the Diagnostic set. The CIs were tested on both the EC validation set and the BTEC blind test set.

Figure 5.1 sketches the fourteen Correction Instances in the final CSet (CSet 14) in the order in which they were processed by the ARR. CIs in this Cset were ordered according to Algorithm 4 in Section 4.6.1.2.3, namely all CIs leading to refinements involving adding agreement constraints (CONSTRAIN) are processed first, followed by syntactic refinements (BIFURCATE), and finally lexical refinements. This ordering optimizes the effect of correction examples, since, before bifurcating any rules, we want to make them as tight as possible. Otherwise, multiple correction examples would be required to obtain the same corrective effect, as each resultant rule after bifurcation would require its own CONSTRAIN corrections. Lexical refinements are most specific and some times require value or agreement constraints to be added for a specific syntactic context, so they are better dealt with at the end. Processing of CIs to maximize their corrective effect is described in detail in Section 4.6.1.2.3.

The Automatic Rule Refiner stored all CIs in each Correction Set as one Collection and processed the CIs in the Collection one after the other in the order specified in Figures 5.1 and 5.3. Note that errors (in TL) and corrections (in CTL) appear underlined in Figure 5.1, and that clue words (Section 4.5) as identified by bilingual speakers, appear in bold. Refinements triggered by these Correction Instances are only sketched here, as the different cases have already been described in detail in Sections 4.6.3 and 4.7. We have limited evaluation to two errors and two correction actions per sentence.

1. I sleep – ***Yo** duermen – Yo duermo
CONSTRAIN: Add person and number agreement constraints (Subj-Verb)

2. I saw some tall girls – *Yo vi algunas **niñas** alta – niñas altas
CONSTRAIN: Add number agreement constraint (N-ADJ)

3. I love a secret agent – Yo amo **una** agente secreto – una agente secreta
CONSTRAIN: Add gender agreement constraint (DET-ADJ)

4. The girl is tall – ***La niña** es alto – La niña es alta
CONSTRAIN: Add gender agreement constraint (Subj-Compl)

5. The boys are tall - ***Los niños** son alto – Los niños son altos
CONSTRAIN: Add number agreement constraint (Subj-Compl)

6. I see the red car – *Veo el **auto** roja – Veo el auto rojo
CONSTRAIN: Add gender agreement constraint (N-ADJ)

7. I gave John a book – *Yo di un libro Juan – Yo di un libro a Juan
BIFURCATE: Add "a" before indirect object NPs

8. You saw the woman – *Viste la mujer – Viste a la mujer
BIFURCATE: Add "a" before direct object NPs (animate object marker)

9. Gaudi was a great artist – *Gaudí era un artista gran – gran artista.
BIFURCATE: Add NP rule to cover pre-nominal ADJ

10. I see the red unicorn – *Veo el unicorn rojo – Veo el unicornio rojo
ADD NEW ENTRY: Add OOV word to the lexicon ([unicorn] → [unicornio]).

11. John and Mary fell – *Juan y María cayeron – Juan y María se cayeron.
LEX. BIFURCATE: Add reflexive form for "fall" ([fall] → ["se cayeron"]).

12: I would like to go – *Me gustaría que ir – Me gustaría ir
LEX. BIFURCATE: Add empty translation for "to" ([to→ " "]).

13. Mary plays the guitar – *María juega la guitarra – María toca la guitarra
LEX. BIFURCATE: Add new sense of the word "play"

14. I love the boy – *Yo amo el niño – Yo amo al niño
LEX. BIFURCATE: Add alternative translation for ([the→al]).

Figure 5.1. Training Set containing 14 Correction Instances (CSet 14). Clue words (Section 4.5) appear in bold.

Figure 5.2 shows two grammar rules after the Correction Instances in CSet 14 have been processed by the Automatic Rule Refiner; automatic refinements are in bold. For the first rule, three agreement constraints (person, number and gender) were added between the subject and the verb of the original sentence rule yielding S,74. The first Correction Instance in CSet 14 is

responsible for adding the person and number agreement constraints and the fourth Correction Instance triggers the addition of the gender agreement constraint, which will percolate down to the complement of the copulative verb, if any.

```
{S,74}
S::S : [NP VP] → [NP VP]
( ;(P:{S,73})
  (X1::Y1)  (X2::Y2)
  (x0 = x2)
  ((y2 subj) = -)
  ((y1 case) = nom)
  ((y1 agr) = (x1 agr))
  ((y2 tense) = (x2 tense))
  ((y2 agr pers) = (y1 agr pers))
  ((y2 agr num) = (y1 agr num))
  ((y2 agr gen) = (y1 agr gen)) )
```

```
{VP,20}
VP::VP : [VB NP NP] → [VB NP "a" NP]
( ;(P:{VP,4})
  (X1::Y1)  (X2::Y4)  (X3::Y2)
  ((x0 indirect_obj) = x2)
  ((x3 case) = acc)
  ((x0 obj) = x3)
  ((x0 agr) = (x1 agr))
  (x0 = x1)
  ((y0 tense) = (x0 tense))
  ((y0 agr) = (y1 agr)))
```

Figure 5.2. Examples of grammar rules affected by three of the Correction Instances in CSet 14. Figure 5.5 shows the original rules.

The second rule VP,20 shows the result of processing Correction Instance 7, which results in the addition of the indirect object marker on the target language side.

This poses a problem for blame assignment, however, since when the Automatic Rule Refiner tries to retrieve the appropriate rule for refinement, that rule might have been modified by a previous Correction Instance and might not be active anymore. To solve this problem, the ARR navigates the rule hierarchy in order to find the relevant active rule that can be refined (Section 4.6.2.1.1).

5.1.2 Development Data

5.1.2.1 Diagnostic Set

The purpose of the Diagnostic Set (DSet) is to test each error type described in Chapter 4 and see that the ARR successfully corrects them given different instances, and not just the ones for which a correction was given by bilingual speakers.

The DSet contains 55 sentences that exhibit the same types of errors as the ones that were used to elicit corrections (Figure 5.1), but are not the same. It illustrates the impact of the different types of refinements covered in this work, which do not all necessarily apply on unseen data. The main goal of DSet results are to confirm that the automatic RR has the desired effects on such comparable sentences. A step-wise evaluation of this DSet also serves to analyze effects of each type of refinement individually.

The baseline lexicon and grammar used to translate the development data (DSet and EC validation set) were manually written to cover the syntactic structures and the vocabulary of the first 400 sentences of the AVENUE Elicitation Corpus (Section 5.2). Since the EC had previously been translated into Spanish to elicit translations from Mapudungun and Quechua (Font Llitjós et al., 2005a), these Spanish translations were used as reference translations.

The DSet was translated both with an initial grammar and lexicon and with the final refined grammar and lexicon, result of all the refinements triggered by CSet 14 (Figure 5.1). For the step-wise evaluation, we also created a refined system from a subset of seven CIs in CSet 14, namely CSet 7 illustrated in Figure 5.3. The rational behind having another CSet was that we wanted to illustrate each type of rule refinement implemented by the system with just one example. In Figure 5.3, the number in bold next to each corrected sentence belonging to CSet 7 indicates how many sentences in the DSet exhibit that type of error, and thus are expected to be affected by it. Below each CI, we give one of the relevant examples in the DSet as corrected by the automatic rule refinements triggered by the CI.

1. I see the red car – veo el auto roja – veo el auto rojo **(14)**
I meet some didactic professors at the conference –
conocí a algunos profesores didácticos en el congreso

2. You saw the woman – viste la mujer – viste a la mujer **(8)**
I saw the children – vi a los niños

3. I see the red unicorn – veo el unicorn rojo – unicornio **(4)**
The unicorn slept – el unicornio durmió

4. Mary plays the guitar – María juega la guitarra – toca **(11)**
The boy plays the viola – el niño toca la viola

5. John and Mary fell – Juan y María cayeron – se cayeron **(5)**
The little boys fell – los niños pequeños se cayeron

6. Gaudí was a great artist – Gaudí era un artista gran – gran artista **(3)**
Juan is a great person – Juan es una gran persona

7. I would like to go – me gustaría que ir – me gustaría ir **(10)**
They want to contribute – Ellas quieren contribuir

Figure 5.3. Cset 7 with relevant translation examples from Diagnostic Set.

5.1.2.2 Elicitation Corpus Validation Set

Part of the AVENUE Elicitation Corpus (EC) is used as validation set to further test the impact of automatic refinement on sentences that have not been designed for the purpose of showing refinement effects. The AVENUE EC contains sets of minimal pairs in English (e.g., *Mary and Anna are singing* vs *John and Anna are singing*) and it was designed to cover a variety of linguistic phenomena (Section 1.3.1). EC sentences are relatively short and simple (Figure 5.4). Table 5.1 shows corpus statistics for the EC validation set.

Mary and Anna are singing
My mother laughed
They are falling
The trees protect the animal
You are hitting John
John is hitting the tree
…

Figure 5.4. Example sentences from the Elicitation Corpus (EC)

5.1.3 Test Data: BTEC Corpus

The Basic Travel Expression Corpus (BTEC) (Takezawa et al., 2002) was selected as the unseen data set. The purpose of this blind test set is to see if corrections that are made on one error instance also have an impact when facing a totally different set of instances that have not been seen before. Namely, we want to answer the question whether the types of automatic refinements described in this thesis generalize well to unseen data.

The BTEC data has been used in evaluations in the International Workshop on Spoken Language Translation (IWSLT). Besides still being currently used to build real systems (Nakamura et al., 2006; Shimizu et al., 2006), this corpus contains relatively simple sentences that are comparable to the ones initially corrected by users and mostly covered by the baseline manual grammar. Figure 5.5 shows a few example sentences from the BTEC.

As our blind test set, we used a pre-existing test set containing 506 English sentences for which two sets of Spanish reference translations were made available. Table 5.1 shows corpus statistics for the BTEC data. Note that the average sentence length of the BTEC data is longer than for both the DSet and the EC data.

i have a throbbing pain .
it is august fifteenth .
could you send this to japan ?
the rate is one hundred twenty dollars a night .
what newspapers do you have ?
how do i use a safe deposit box ?
where is the boarding gate ?
are these all your personal effects ?
…

Figure 5.5. Example sentences from the Basic Travel Expression Corpus (BTEC)

In order to test on unseen data while trying to take out-of-vocabulary words out of the equation as much as possible, we used phrase alignments to generate a lexicon. This basic statistical lexicon was manually augmented with POS and agreement information and formatted to fit the translation lexicon specifications (Section 5.2.1). It was used in conjunction with the manual lexicon used to translate the development data.

For this test set, we also built a Suffix Array Language Model with the SALM toolkit (Zhang & Vogel, 2006) on the 123,416 Spanish sentences from the training partition of the BTEC corpus (Table 5.1).

5.2 English-Spanish MT System

The Machine Translation system used for all the experiments is a Transfer-Based MT system developed within the AVENUE project (Lavie et al., 2003; Peterson, 2002). Even though the focus of our evaluation is assessing the increase of generation capabilities of the refined grammar and lexicon, we also ran experiments to optimize decoder weights and obtain a better re-ranking of the translation alternatives produced by the grammar and the lexicon. These experiments helped to further enhance the impact of the Automatic Rule Refiner embedded in an end-to-end MT system.

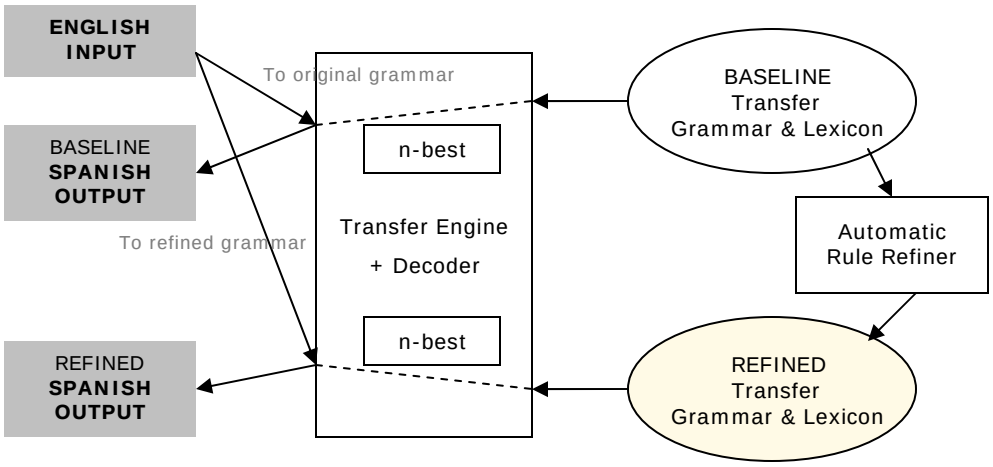


Figure 5.6. Experimental set-up for the English-Spanish MT system. The Automatic Rule Refiner is described in detail Section 4.5.

Figure 5.6 illustrates the experimental set-up of the experiments done for the English-Spanish MT system. Given the same English input (top left), when the baseline grammar and lexicon were loaded to the Transfer Engine, we obtained the Baseline MT output; loading the refined grammar and lexicon, we obtained the Refined Spanish output.

In Appendix A, we describe the MT system language-independent components, such as the Transfer Engine and the decoder. In this section we focus on the grammar and lexicon specific to the English-Spanish MT system used in our experiments.

The number of rules in the English-Spanish original baseline grammar and the refined grammar after CIs were processed by the ARR can be seen on Table 5.2.[30] During training, three BIFURCATE operations were applied to the baseline Grammar, whereas four were applied to the baseline Lexicon (plus one new lexical entry was added). The last column shows the number of constraints in the grammar. A total of 30 constraints were added to the grammar via automatic refinements based on the training examples in Figure 5.1. This means that most of the modifications in the grammar were at the constraint level, and thus resulted from applying the CONSTRAIN operation described in Section 4.4. Even though the number of BIFURCATE

[30] Keep in mind that translation rules in our MT system include parsing, transfer and generation information (Section 4.2), which might otherwise be expressed with three different rules in other Transfer-Based MT systems.

operations that applied was lower, such operations had broader applicability (Sections 5.4.2.2 and 5.4.3.3.1).

Data	System	Lex.	Gram.	Gram. Constr.
DSet	Baseline	247	19	86
	Refined (CSet 7)	251	21	110
	Refined (CSet 14)	252	22	142
EC	Baseline	474	40	201
	Refined (CSet 14)	479	43	231
BTEC	Baseline	1732	40	201
	Refined (CSet 14)	1737	43	231

Table 5.2. Number of translation rules in the grammar and the lexicon.[30]

Figure 5.7 shows examples of translation rules in the baseline grammar and the lexicon, whereas Figure 5.2 shows their refined counterparts.

```
{S,4}
S::S : [NP VP] → [NP VP]
( (X1::Y1)  (X2::Y2)
  (x0 = x2)
  ((y2 subj) = -)
  ((y1 case) = nom)
  ((y1 agr) = (x1 agr))
  ((y2 tense) = (x2 tense)))
```

```
    {VP,4}
VP::VP : [VB NP NP] → [VB NP NP]
     ( (X1::Y1)  (X2::Y3)  (X3::Y2)
((x0 indirect_obj) = x2)
((x3 case) = acc)
((x0 obj) = x3)
((x0 agr) = (x1 agr))
(x0 = x1)
((y0 tense) = (x0 tense))
((y0 agr) = (y1 agr)))
```

Figure 5.7. Basic English→Spanish translation grammar rules.

5.2.1 Adapting to a New Domain

A major bottleneck in developing a Rule-Based MT system for a new translation task (either a new language pair or a new domain) is building the lexicon, and some times extending the grammar. Automatic grammar induction using statistical alignments has been studied in (Probst 2005).

For our experiments on unseen data, we use the same basic manual grammar used in the EC experiments, however we augmented the basic lexicon with entries to cover the vocabulary

for new domain. We explored semi-automatic lexicon generation for fast adaptation to the travel domain.

5.2.1.1 Semi-Automatic Generation of the Transfer Lexicon

The Transfer-Based system relies on the lexicon including part-of-speech information, agreement features, such as gender and number, as well as several other linguistically motivated features. To adjust the system quickly to a new domain, we decided to leverage from statistical alignment models to generate word and phrase alignments as candidates for the transfer lexicon.

In the first step, we trained a statistical lexicon, using the IBM1 word alignment model, for the directions Spanish to English and English to Spanish. As multi-word entries are often needed ([valuables] → [objetos de valor], [reception desk] →[recepción], [air conditioner]→[aire acondicionado]), we used phrase alignment techniques to create translation candidates for words and 2-word phrases (Vogel, 2005).[31] The phrase alignment also generates multi-word translations for single source words. With reasonably tight pruning, a manageable phrase translation table was generated.

The next step, manually cleaning the translation table, annotating entries with parts-of-speech, and with agreement and tense constraints, was initially restricted to those items that overlapped with the vocabulary of our development test set.

The initial manual lexicon contains 479 lexical entries, whereas the one statistically generated comprises 1,727 lexical entries.

5.3 Evaluation Metrics

We use several methods to evaluate MT output accuracy, all of which can be used to guide refinement decisions. This allows the feedback loop to be completed, since we can make sure that automatic refinements lead to real improvements of the MT system and its output.

[31] Unlike standard techniques used in Pharaoh and MOSES, the technique we used (PESA) does not use the Viterbi alignment; it uses a sentence splitting approach that splits the target sentences into a center part, a left and a right part such that the center part matches the phrase in the source sentence and the outer parts in the target sentence match the outer parts in the source sentence.

5.3.1 Precision, Recall and F1

Precision (P) and recall (R) are widely used to evaluate NLP systems. When comparing a set of candidate translations to a set of reference translations, **precision** is a measure of the **proportion of translations generated by the system that are correct**. Namely, for each source language sentence, precision can be measured as the intersection between the set of translations generated by the system (Candidate) and the set of reference translations (Correct), divided by the total number of translations generated by the system (Candidate):

$$P = \frac{|\, Candidate \cap Correct \,|}{|\, Candidate \,|} = \frac{P_{\{0,1\}}}{|\, Candidate \,|}$$

To simplify, we consider the numerator to be binary, with a value of 1 if one of the candidate translation matches one of the correct translations, and 0 otherwise. We use average precision to calculate precision for a test set as follows:[32]

$$AvgP = \frac{\sum_{i=1}^{n} P_{\{0,1\}}}{\sum_{i=1}^{n} |\, Candidate \,|}$$

It is important to note that if automatic refinements are successfully in making the grammar tighter and not allowing overgeneralization, the number of total translations produced by the system (denominator) decreases, and so precision increases.

Recall is defined as the **proportion of correct translations that the system generated**. Namely, for each source language sentence, recall can be measured as the intersection between the set of translations generated by the system (Candidate) and the set of reference translations (Correct), divided by the number of correct translations (Correct):

$$R = \frac{|\, Candidate \cap Correct \,|}{|\, Correct \,|} = \frac{R_{\{0,1)}}{1}$$

[32] Note that this is not the same as the average precision metric used in Information Retrieval.

Which at the sentence level, can be simplified to 1 if one of the candidate translations is correct as specified by the reference translations, and to 0 otherwise. We use average recall to calculate recall for a test set as follows:

$$AvgR = \frac{\sum_{i=1}^{n} R_{\{0,1\}}}{|\,TestSet\,|}$$

It is generally the case that one can trade off precision and recall (one could produce all the sentences in a target language to translate a single source language sentence and get 100% recall but very low precision). For this reason it is convenient to combine precision and recall into a single overall performance measure. One way to do this is the **F measure**, variant of the E measure introduced by van Rijsbergen (1979) for information retrieval, where F = 1 – E. The F measure is defined as follows:

$$F = \frac{1}{\alpha \frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{2PR}{P+R}$$

where α is a factor which determines the weighting of precision and recall. A value of $\alpha = 0.5$ is often chosen for equal weighting of P and R. With this α value, the F measure simplifies to 2PR/ (P+R) (Manning & Schütze, 1999).

5.3.2 Automatic Evaluation Metrics

Even though current automatic evaluation metrics do not always have high correlation with human judgments, they have become prominent to evaluate MT system performance, especially for development of data-driven MT. For the Automatic Rule Refiner to be effectively integrated in an end-to-end MT system, automatic evaluation of its impact on unseen MT output becomes crucial. Therefore, we also report these metrics to provide some insight into system performance evaluation on test data.

The main idea of automatic evaluation comes from the *word error rate* metric successfully used by the speech recognition community, appropriately adapted to account for multiple reference translations and to allow for legitimate differences in word choice as well as word order. Automatic evaluation metrics use a weighted average of variable length phrase matches against the reference translations. Different weighting schemes are used by different metrics. The IBM MT research group proposed a metric called BLEU (BiLingual Evaluation

Understudy). BLEU averages the precision for unigram, bigram and up to 4-grams and applies a length penalty if the generated sentence is shorter in length than the best matching reference translation (Papineni et al., 2001). A variant of BLEU has been adopted by the National Institute of Standards and Technology (NIST) for its MT effort. The NIST metric is derived from the BLEU evaluation criterion but differs in one fundamental aspect: instead of n-gram precision, the information gain from each n-gram is taken into account. The idea behind this is to give more credit if a system gets rare n-gram matches, but to give less credit for frequent n-gram matches (Doddington, 2002). Under this weighting scheme, a system does not get any credit for matching bigrams that appear only once. Therefore, NIST scores tend to be less discriminative when evaluating small data sets.

More recently, it has been shown that a balanced harmonic mean (F1 measure) of unigram precision and recall outperforms the widely used BLEU and NIST metrics for Machine Translation evaluation in terms of correlation with human judgments of translation quality. A Metric for Evaluation of Translation with Explicit word ORdering (METEOR, Banerjee and Lavie, (2005)) performs a maximal-cardinality match between translations and references, and uses the match to compute a fragmentation penalty. This computation is done by assessing how well ordered the words in the translation are. METEOR assigns most of the weight to recall, instead of precision, and can use stemming and synonymy matching along with the standard exact word matching, if a stemmer and an ontology are available for the target language.

At the time of this evaluation, a Spanish stemmer was not available, nor was a Spanish synonym dictionary, hence we ran meteor with "exact" matches, without doing any stemming or synonymy matching.

5.3.3 Reference Translations

In addition to source language sentences and candidate translations or hypothesis, Automatic Evaluation metrics need one or more human translations per source language sentence to act as reference. For all the experiments described in this chapter, independent human reference translations were already available.

For situations with no parallel data, however, the advantage of our approach to improving translation quality is that user corrected translations effectively provide system-relevant reference translations, which can then be used to automatically evaluate both the original and the refined system, much in the same way as described by Snover and colleagues (2006).

Since user corrections of MT output are precisely what we are trying to approximate with the Automatic Rule Refiner, they make ideal reference translations. For each user, each sentence evaluated as correct and all the corrected sentences constitute one set of reference translations. Thus, we can have as many reference translations for a particular data set, as users have corrected it. We can then run all automatic evaluation metrics (BLEU, METEOR and NIST) using as many reference translation sets (user corrections) as are available.

For the experiments described in this section, both the hypothesis and reference files were normalized to lower case to ensure reliable automatic scores.

5.4 Results

In this section we report results first on the two development sets, and then on unseen test data. The refined system achieves higher translation accuracy on all data sets, both according to human judgments and as measured by the three automatic evaluation metrics described above.

5.4.1 Diagnostic Set

The first three rows in Table 5.3 show average recall (percentage of correct translations), precision (recall divided by average number of total translations) and the harmonic mean of the two (F1), *before* and *after* refinements were applied to the MT system. The last three rows illustrate the amount of ambiguity, which corresponds to the total number of translations generated by the system.

		After training with	
	Baseline	CSet: 7	CSet: 14
Avg. Recall	0.218	0.782	0.836
Avg. Precision	0.053	0.128	0.181
F1	0.085	0.220	0.298
Ambiguity			
Avg. Num of Total Tr.	4.09	6.11	4.33
Avg. Increase of Total Tr.		2.02	0.24
Cumulative Increment		49%	6%

Table 5.3. Recall, precision and F1 values achieved before and after the MT system was trained with CSets containing 7 and 14 Correction Instances on the Diagnostic Set of 55 sentences.

DSet results show that the MT system went from not producing a correct translation to producing at least one correct translation for most SL sentences. On the other hand, the number of total translations per sentence went from an average of 4 to an average of 6 translations, when

CSet 7 was processed, but managed to grow minimally when CSet 14 was processed. This shows that the increase in recall is not at the expense of a decrease in precision.

In the data section, Figure 5.3 shows example sentences from the DSet. It is important to note that these example sentences were not being correctly translated by the original MT system and are now being correctly translated by the refined MT system. Most importantly, there is no overlap between the DSet and either of the two Correction Sets.

When processing the seven additional CIs (CSet 14) targeted mostly to control ambiguity (refinements were done by applying the CONSTRAIN operation), ambiguity decreases 43%, which represents an increase of only 6% over the baseline. Therefore, given CSet 14, the number of correct translations increases maximally, and at the same time, the number of total translations only increases an average of 0.24. We now show a detailed analysis of the effect of individual CIs on the DSet.

5.4.1.1 Step-wise Evaluation and Error Analysis

More interesting perhaps is the breakdown of these results to determine what the specific effect of each type of refinement is. For this, we took the lexicon and the grammar resulting from applying each one of the refinements result of processing CSet 7 (Figure 5.3) and run the MT system on the Diagnostic set. Incremental results can be seen in Table 5.4. The last row shows the cumulative relative increment of the total number of translations (ambiguity).

	0	1	2	3	4	5	6	7
Avg. Recall	0.218	0.218	**0.309**	**0.418**	**0.545**	**0.636**	**0.691**	**0.782**
Avg. Precision	0.053	**0.062**	0.062	0.092	0.110	0.125	0.135	0.128
F1	0.085	0.097	0.103	0.151	0.183	0.209	0.226	0.220
Ambiguity								
Avg. Num of *Total Tr.*	4.09	3.49	4.95	4.55	4.95	5.09	5.13	6.11
Avg. Increase		-0.60	1.46	-0.4	0.4	0.14	0.04	0.98
Cum. Rel. Increment		-15%	21%	11%	21%	24%	25%	49%

Table 5.4. Step-wise results for the processing of each CI in CSet 7 on the Diagnostic Set.

5.4.1.1.1 CI 1: Adding a gender agreement constraint

Adding an agreement constraint leaves recall unchanged, since all the possible gender combinations were already being generated by the original system. However, for the sentences containing a noun modified by a determiner and an adjective marked with gender (DET N ADJ),

the total number of translations decreased by half (50%). This constitutes a 15% average decrease on the 55 sentences of the DSet (Column 1 in Table 5.4).

The diagram in Figure 5.8 shows the *total* number of translations (Tr.) produced by the original system (Before) with a dotted line, and the total number of translations produced by the refined system after having processed the first CI in CSet 7 (After) depicted with a solid line. Bars illustrate the number of *correct* translations before (with stripes) and after (solid) the Rule Refiner has processed CI 1.

Note that the total number of translations (ambiguity) is reduced for the 12 sentences that were already being translated correctly by the original system. The peak in ambiguity for the 3rd sentence in the DSet is due to fact that the noun (*agente* - agent) is underspecified for gender and so none of the agreement constraints can apply.

The reason sentences 2 and 3 do not have a correct translation is that they also exhibit a different error, namely they are missing the direct object marker.

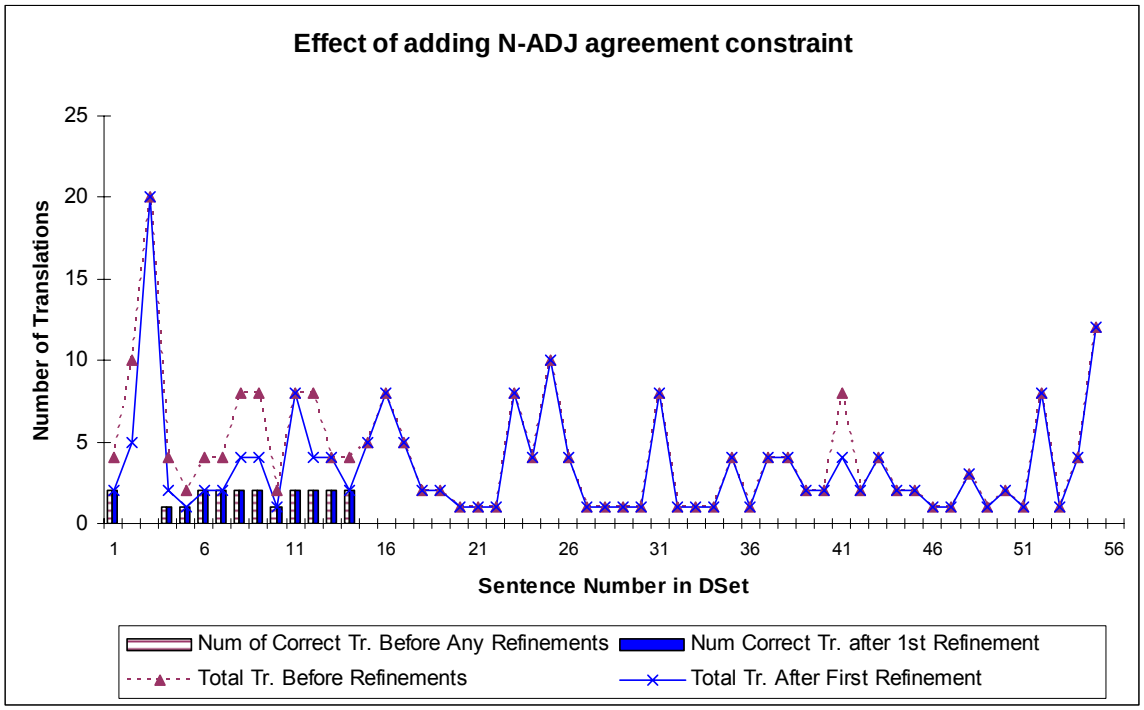


Figure 5.8. Plot showing the effect of the refinement triggered by the first CI illustrated in Figure 5.3 on the number of correct translations (bars) and the total number of translations (lines).

5.4.1.1.2 CI 2: Add "a" to the appropriate Grammar rule

When adding a constituent to a rule, the amount of context expressed by the refined rule has an impact on both recall and precision. In our example, the correction triggers a refinement, which

can be applied to one of two different grammar rules: the NP following the *a* or the VP that subsumes the verb as well as the *a* and the object NP (Section 4.5.4).

Figure 5.9 shows the effect of refining the NP rule. Six sentences, for which the original system was not producing any correct translation, are now correctly translated by the refined system (sentences number 15-20).

There are two observations worth mentioning here. First, when refining the NP rule, the *a* is added in front of the two constituents that are part of the rule, in this case DET N. This refinement does not generalize to other types of NP. For this DSet, sentences 2 and 3 have direct objects, which require an *a*, however their object also has an adjective (DET N ADJ), and thus the refinement does no affect them. Second, *a* is now produced by the refined system in front of every NP containing a determiner and a noun, and not just object NPs. This doubles the amount of alternative translations for sentences containing such NPs in any position (subject, oblique etc.).

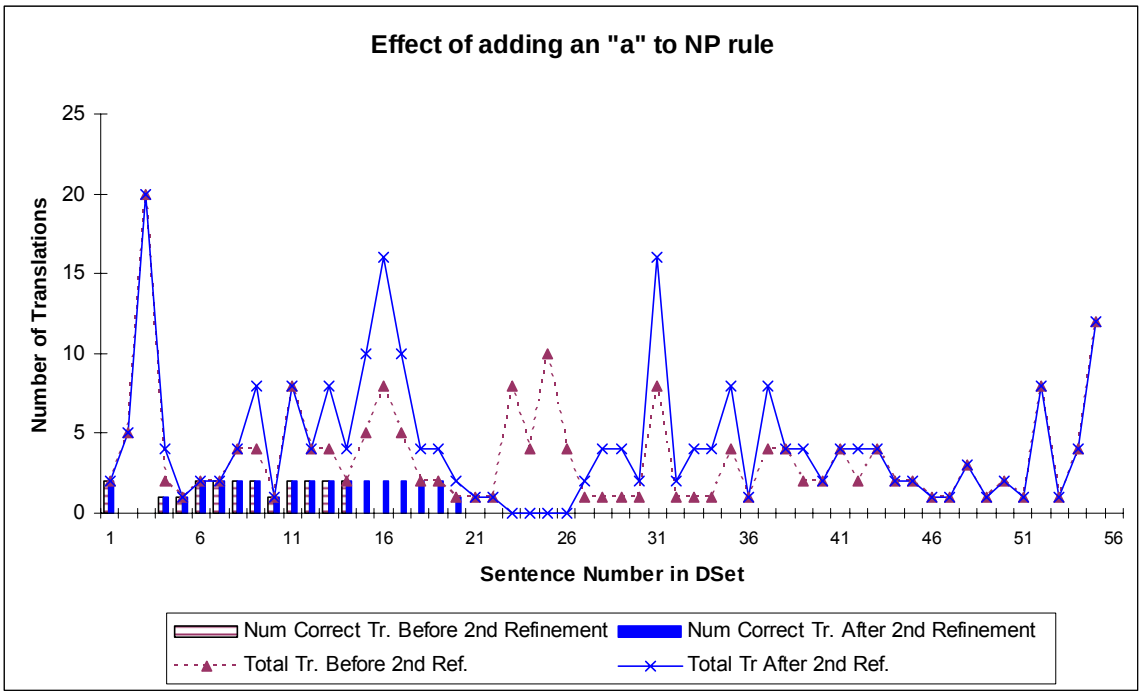


Figure 5.9. Plot showing the effect of the refinement triggered by the second CI sketched in Figure 5.3, if the Automatic Rule Refiner had picked the NP rule for refinement.

The current implementation of the ARR automatically selects the most specific rule for refinement (Sections 4.5.4), in this case, the VP rule. This is the linguistically motivated option,

for it represents a step towards the appropriate solution, which is to add *a* only to animate object NPs. Since this is the right level of generalization, given the current grammar and lexicon, sentences 2 and 3 are now being correctly generated. However, sentences containing V NP NP (sentence numbers 18-20 in Figure 5.10) are not correctly translated. That is fine, since the *a* required in front of the second NP here, is an indirect object marker, and thus it really is a different *a*. The appropriate way to refine this rule is by presenting bilingual speakers with examples of this syntactic structure. This is achieved by processing CI 7 in CSet 14.

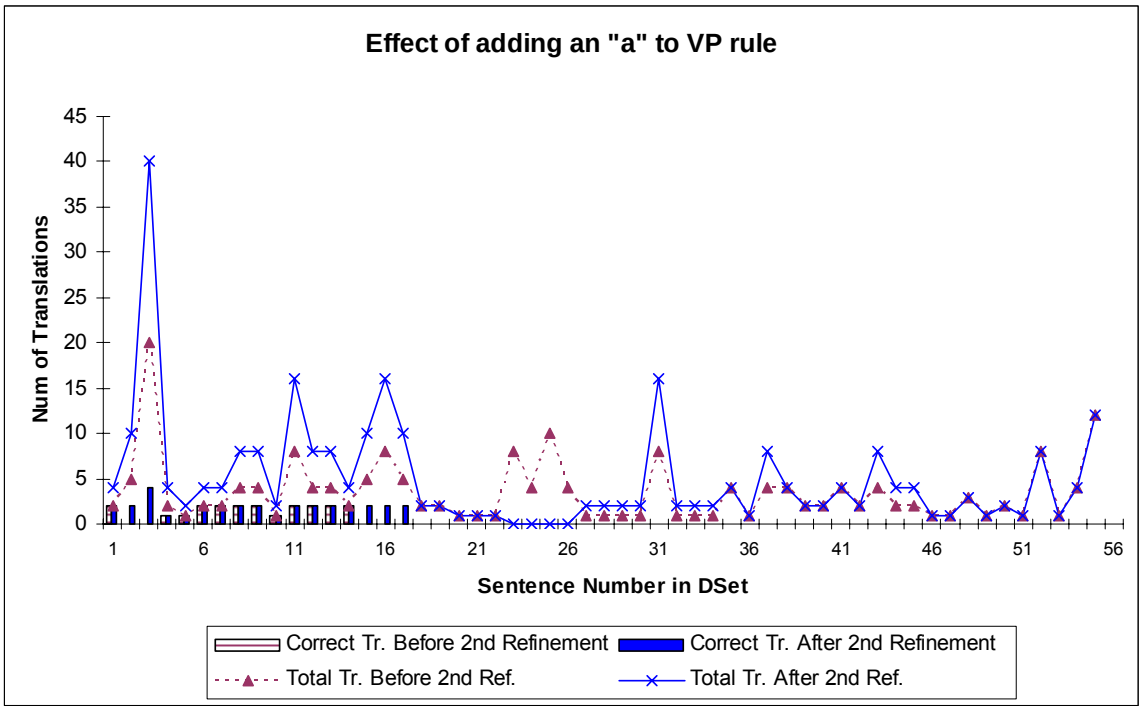


Figure 5.10. Plot showing the effect of the refinement triggered by the second CI sketched in Figure 5.3, when the Rule Refiner picks the VP rule for refinement.

In this particular DSet, adding *a* to the NP rule also accidentally fixed another error (indirect object marker) present in three sentences, causing both precision and recall to be slightly higher than for the VP refinement (Table 5.5). However, we expect that having a much larger regression test set to drive automatic refinements will take care of accidental fixes and properly reward correct refinements.

	(NP)	(VP)
Avg. Recall	0.327	0.309
Avg. Precision	0.076	0.062
F1	0.123	0.103
Ambiguity		
Avg. Num of *Total Tr.*	4.29	4.95
Avg. Increase	0.80	1.45
Cumulative Increment	5%	21%

Table 5.5. Comparing results for different refinement strategies on the DSet.

Adding more constraints would increase precision, but then the refinement would not generalize to unseen words and syntactic structures. In this case, if the ARR added a constraint to restrict the application of the VP rule to object NP with the word *woman*, ambiguity would not increase, but then none of the translations for the eight relevant sentences in the DSet would be generated correctly.

Finally, it is important to note that when the animate object marker (*a*) is followed by a masculine singular definite article in Spanish (*el*), they combine into "*al*". This is commonly dealt with a mere string replacement at a post-processing step, and thus for experiments on the DSet, translations that contained "*a el*" were post-processed to "*al*" and considered correct.

5.4.1.1.3 CIs 3, 4 and 5: Adding new lexical entries

Adding a new lexical entry, be it a new sense or a new form, has the same effect on recall and precision. For systems with fully inflected lexicons, recall increases for sentences where the exact same word appears and if no further constraints are added to the lexicon or the grammar, the number of total translations for such sentences doubles, which results in an overall slight increase of ambiguity. See Columns 3, 4 and 5 in Table 5.4.

5.4.1.1.4 CI 6: Add NP rule to cover pre-nominal adjectives

Changing the order of the constituents of a rule to cover a new syntactic structure increases recall (Table 5.6). Furthermore, if constraints are added to the lexical entries and the rules affected by the order change, ambiguity decreases significantly, and thus prevision increases (++ on Table 5.6 indicates that constraints were also added to the rules). For the NP (DET ADJ N) example, before adding the appropriate constraints, the total number of translations increases 39% (Column 6 on Table 5.6); whereas after adding the constraints, it only increases 1% on average (Column 6++ on Table 5.6), as you can graphically see in Figure 5.11 and 5.12 below. Note the difference in the y-axis scale and ambiguity reduction for sentences 1-15.

Recall	5	6	6++	7	7++
Avg. Recall	0.636	0.691	0.691	0.782	0.782
Avg. Precision	0.125	0.103	0.135	0.102	0.128
F1	0.209	0.179	0.226	0.180	0.220
Ambiguity					
Avg. Num of *Total Tr.*	5.09	6.67	5.13	7.65	6.11
Avg. Increase	0.14	1.58	0.04	0.98	0.98
Cumulative Increment	24%	63%	25%	87%	49%

Table 5.6. Comparing results for different refinement strategies on the DSet.

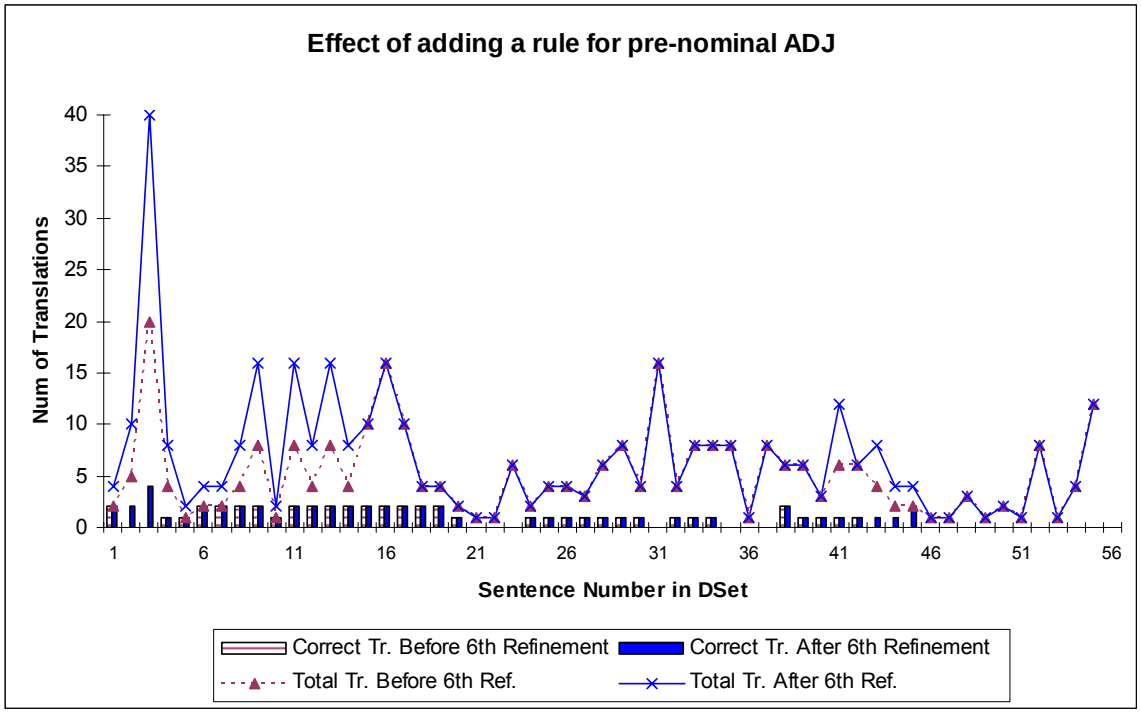


Figure 5.11. Plot showing the effect of the refinement triggered by the CI 6 (Figure 5.3).

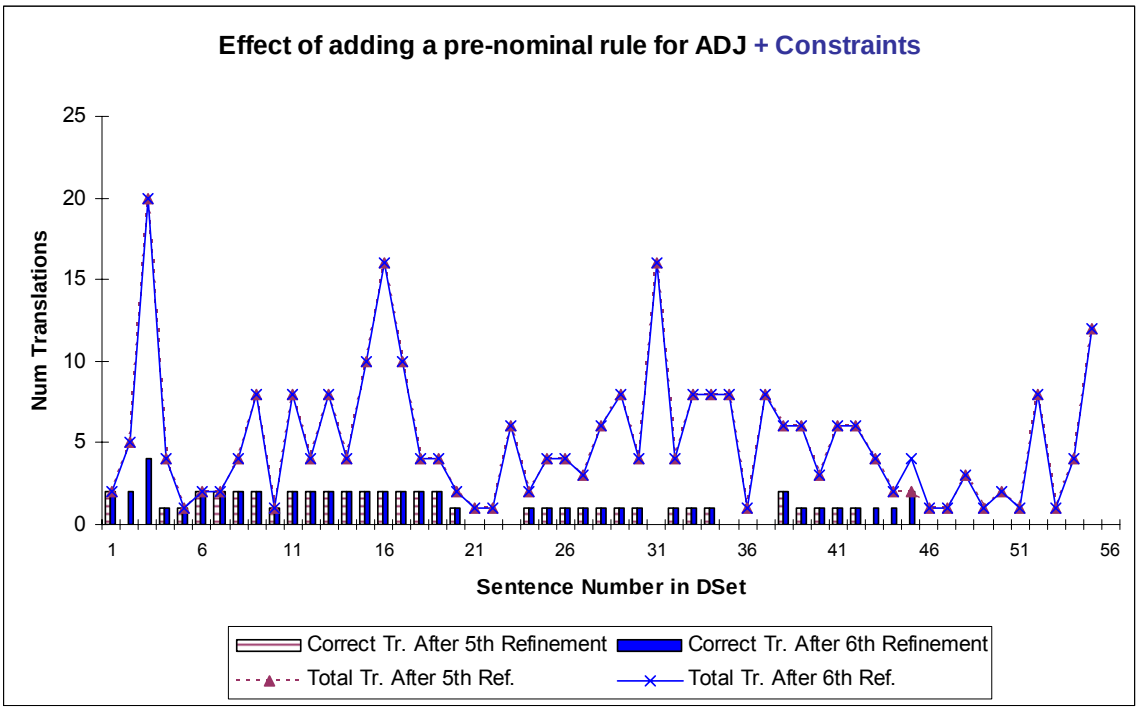


Figure 5.12. Plot showing the effects of adding constraints to the newly created rule (Figure 5.11). This plot corresponds to Column 6++ in Table 5.6.

5.4.1.1.5 CI 7: Permit empty translation for words in the source lexicon

Finally, recall for the last few sentences of the DSet increases when we allow the MT system not to translate a source language word into the target language, effectively deleting a word in the translation. In this case the word *to* in "*I would like to go*" does not correspond to any Spanish word ("*me quiero ir*"), and so the Automatic Rule Refiner adds a new lexical entry for *to* with an empty string as its translation ([to→ " "]). Figure 5.13 depicts to the output of the Refined system once all the CIs in CSet 7 have been processed (Figure 5.3). It also corresponds to the results shown in Table 5.6, Column 7++ after having processed CIs 6 with constraints (6++).

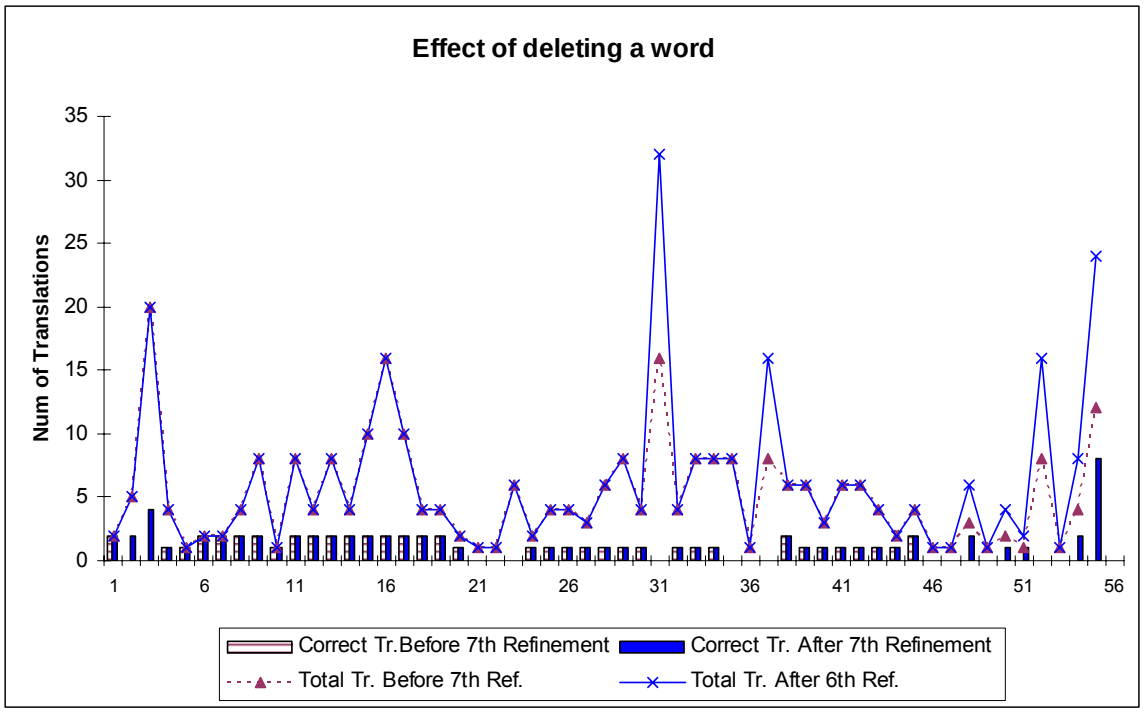


Figure 5.13. Plot showing the effect of having processed all refinement triggered by CSet 7.

5.4.2 Elicitation Corpus

5.4.2.1 ORACLE Experiment

The main goal of our oracle experiments is to evaluate *n*-best list quality. Using a human oracle to select the best alternative translation, we can obtain an upper-bound on performance for all our MT systems. We ran the original, baseline system and the refined MT systems that result from processing Cset 7 (Figure 5.3) and CSet 14 (Figure 5.1) on the EC validation set, generating a *n*-best list for each SL sentence in the EC set and for each one of the three systems. The results reported in Table 5.7 were calculated for the best candidate translation for each SL sentence, as selected by a bilingual speaker, simulating an ideal decoder.

	Correct Tr	Avg. total # Tr	CIs Applied	Changes		METEOR	BLEU	NIST
				+	–			
Baseline	151	6.9	0	0	0	0.8808	0.7263	8.5270
CSet: 7	172	9	3	21	11	0.9098	0.8046	8.7133
CSet: 14	183	4.5	4	32	0	0.9276	0.8635	9.0239

Table 5.7. Human ORACLE scores showing improvements after automatic refinement on 200 EC sentences.

According to all the automatic metrics, both refined systems are better than the baseline system, as shown in the last three columns of Table 5.7. In fact, they correlate well with actual improvements to the MT output, as indicated by the number of correct translations produced by each system (Column 1).

Column 2 shows the average number of total translations per sentence generated by each system. Note that the additional refinements present in the larger Correction Set (CSet 14) result in a decrease of the final translation candidate list size, going from an average of 9 translations per source language sentence to an average of just 4.5, less than the baseline system (6.9). Once again, this clearly indicates that the increase in recall is not at the expense of a decrease in precision and that refinements in the large CSet manage to tighten the grammar while also adding generation capabilities to it, also for the EC test set.

Column 3 (CIs Applied) shows the total number of Correction Instances that applied, namely those that had an effect on this particular validation set. Only a subset of CIs in CSet 7 and CSet 14 actually had any effect on the EC validation set. A more detailed analysis of the impact of the refinements that did apply is given below. To see the impact of all different types of refinements, see the previous section.

Columns 5 shows the changes that lead to correct translations (+), whereas Column 6 (–) shows the number of changes that did not lead to an improvement.

In this ORACLE experiment, automatic score improvements are correlated with an increase in the number of correct translations produced by the systems (Column 1), but changes in ambiguity (Column 2) have no impact, since the human oracle (bilingual speaker) does not get thrown off by additional incorrect translations.

5.4.2.2 Qualitative Analysis

Column 3 in Table 5.7 indicates the number of Correction Instances applied to each system's output. From the set of all CIs processed by the Automatic Rule Refiner, only four of them applied to this particular validation set (Figure 5.14). Specifically, the subj-verb agreement constraint (1), the addition of an animate object marker *a* to the VP rule (8), the addition of a new form of *fell* (*se cayeron*) (11) and another form for *the* (*al*), which covers the addition of the animate object marker when the object is masculine and singular (14). In fact, the 11 changes not leading to improvements shown in Column 5 for CSet 7 resulted from adding an *a* in front of the masculine singular definite article (*el*), thus producing the incorrect Spanish sequence "*a el*". This is typically dealt with in a post processing step, so it is not considered a serious error at this stage.

1. I sleep – *Yo duermen – Yo duermo
CONSTRAIN: Add person and number agreement constraints (Subj-Verb).

8. You saw the woman – *Viste la mujer – Viste a la mujer
BIFURCATE: Add *a* to the appr. Grammar rule (animate object marker).

11. John and Mary fell – *Juan y María cayeron – se cayeron.
LEX. BIFURCATE: Add reflexive form to the lexicon.

14. I love the boy – *Yo amo el niño – Yo amo al niño
LEX. BIFURCATE: Add lexical entry ([the→al]).

Figure 5.14. Correction Instances from CSet 14 that had an impact on the EC validation set.

In this EC validation set, most sentences are affected by refinement 1. Even though this refinement cannot improve the number of correct translation produced by the systems, it does manage to largely decrease the number of total translations produced by them.

Overall, there are 14 sentences that can be improved by refinement 8 (VB "a" NP), whereas 16 can be improved by refinement 14 (*al*). 31 sentences contain *fell*, but only 7 have a third person plural animate subject, and thus can be improved by the "se cayeron" refinement. All in all, from the 200 sentences in the EC validation set, 37 sentences can be improved by the last three refinements. This represents an 18.5% of the validation set. In this experiment, only 32 sentences from these 37 were actually improved by the Rule Refiner. The remaining 5 did not get a full parse, and thus the transfer engine only output the first of all possible partial translations, which did not result from any refinement.

Table 5.8 contains two examples of translations produced by each one of the MT systems evaluated above.

Source Language	The hunter killed an animal	She is killing the deer
Baseline	*El cazador mató un animal	*Ella está matando el ciervo
CSet: 7	El cazador mató a un animal	*Ella está matando a el ciervo
CSet: 14	El cazador mató a un animal	Ella está matando al ciervo

Table 5. 8. Examples of best ORACLE translation produced by each one of the MT systems for the source language sentences displayed in the first row.

5.4.3 Unseen Data: BTEC Test Set

For experiments on unseen test data, we report results for both the baseline system and our best performing refined MT system, namely the one that results from processing the 14 Correction Instances in Figure 5.1. In all previous experiments, this system achieves the highest translation accuracy as indicated by all measures, while controlling ambiguity.

5.4.3.1 No Decoder

The most basic experiment consists of loading the lexicon and the grammar into the Xfer system and selecting the translation first output by the system (Table 5.9). Even though this situation is not very realistic, we use it to set a lower-bound on performance. This situation corresponds to the case when no human can select the best translation from the list of all translations generated by a transfer system, in the absence of a language model or other automatic way of re-ranking the candidate list.

System	METEOR	BLEU	NIST
Baseline	0.5666	0.2745	5.88
Refined	0.5676	0.2559	5.62

Table 5.9. Automatic metric scores for a purely Rule-Based MT System (1st translation output).

In such a setting, different automatic metrics do not agree on the translation accuracy of both systems. On one hand, METEOR, which has been shown to correlate well with human judgments (Lavie & Agarwal, 2007; Snover et al., 2006) seems to indicate that the refined system outperforms the baseline system (as measured by the latest version v0.5.1 with "exact" match). On the other hand, both BLEU and NIST scores are higher for the baseline system (mteval-v11b.pl).

These results reflect poor re-ranking and not *n*-best list quality, since human inspection revealed that the refined grammar is able to augment the *n*-best list with correct translations that the baseline system was not able to generate. We take these results as our lower-bound on performance. In the next section, we describe an oracle experiment to measure *n*-best list quality of both systems.

5.4.3.2 ORACLE Experiment

For the BTEC test set, we approximated a human oracle by calculating automatic metric scores for METEOR, for BLEU and for NIST. Given 100-best lists for each source language sentence,

the best translation hypothesis for each automatic metric was automatically selected (i.e. the one with the highest score).

System	METEOR	BLEU	NIST
Baseline	0.6863	0.4068	7.42
Refined	0.6954	0.4215	7.51

Table 5.10. Automatic metric oracle scores based on a 100-best list

These scores reflect the fact that automatic refinements are able to feed the *n*-best list with better translations, as judged by independent human reference translations. Even with a small set of independent user corrections, the refined system shows potential improved translation quality as indicated by higher scores for all three automatic evaluation metrics in Table 5.10.[33]

Moreover, oracle scores provide us the margin that we can gain when improving on the re-ranking of the *n*-best list produced by the Xfer engine.

5.4.3.3 Decoder: Initial Ranking

As expected, when the Xfer system is run in combination with a language model as well as the Fragmentation penalty (Appendix A), automatic metric scores for the 1-best hypothesis are significantly higher than when just using the first translation output by the Xfer system alone.

Also as expected, these results are lower than the oracle scores for both the baseline and the refined system. However, the important thing to notice from these results is that, like in the oracle case, the refined system consistently outperforms the baseline MT system for all three automatic metrics, as shown in Table 5.11.

System	METEOR	BLEU	NIST
Baseline	0.6176	0.3425	6.53
Refined	0.6222	0.3513	6.56

Table 5.11. Automatic metric scores for 1-best decoder hypothesis.

The difference between the baseline and the refined system in terms of 1-best scores is slightly smaller than the difference between oracle scores, which means that the decoder can not fully leverage the improvements made in the grammar. This is also to be expected, since the decoder fails to select the best translation in most cases.

[33] Differences in METEOR scores are statistically significant, with a very small p value (4.42×10^{-6}). See Appendix C for more details.

Even though the language model for the BTEC data is rather small (123,416 sentences), using a larger LM trained on the Europarl training data (Koehn, 2001 (release 1), 2003 (release 2); Koehn, 2005) did not improve these results.

5.4.3.3.1 Error Analysis

Half of the Correction Instances processed by the ARR involved just the CONSTRAIN operation and resulted in a tighter grammar, which effectively reduced the number of alternative translations produced by the system, yielding an increase in precision on the EC test set. However, on the BTEC test set, the refined system produces 30.4% more translations on average per source language sentence.[34] This is due to the BIFURCATE operations that were applied during refinement.

After manual inspection, most of the differences between the baseline and refined systems were due to three of the 14 user corrections processed by the Automatic Rule Refiner, namely 7, 8 and 12 in Figure 5.1. The fact that the 5 CIs processed by the ARR reduced the size of the final candidate translation list, might have also aided the decoder's task in the refined system. Nevertheless, error analysis also suggests that a strong decoder is to some extent redundant with CONSTRAIN operations, since it favors correct target language sequences over incorrect ones.

Source: is this seat taken ?
Baseline: está este asiento *cogidas* ?
Refined: está este asiento *tomado* ?
Source: please call the police .
Baseline: por favor llame la policía .
Refined: por favor llame *a* la policía .
Source: i would like to put my valuables in the safe deposit box .
Baseline: me gustaría *que* poner mis objetos de valor en la caja fuerte .
Refined: me gustaría poner mis objetos de valor en la caja fuerte .

Table 5.12. MT output examples from the BTEC test set before and after automatic refinements applied to the grammar and lexicon.

[34] On average, the baseline MT system produced 21.7 translations per SL sentence, whereas the refined MT System produced 28.3 translations per sentence.

In 56 cases the additional generation capabilities of the refined system successfully produce a better translation than the baseline system. And 37 of these improvements were picked by the statistical decoder's ranking. Table 5.12 shows examples of the three most common types of fixes result of automatic refinements that were also picked by the decoder.

5.4.3.4 Decoder with Optimized Weights

The large gap between the ORACLE scores and the initial decoder weights suggests that there is still improvement to be gained from better re-ranking of the *n*-best list produced by the Xfer engine. This is the reason we turned to Minimum Error Rate (MER) training (Och, 2003). In MER training, the *n*-best list generated by the translation system is used to find feature weights for the different decoder components. Optimization of decoder weights is done following a hill-climbing process, thereby re-ranking the *n*-best list, which will improve the match between the 1-best translation and the reference translations given. The optimization can use any metric as objective function. Typically, systems are tuned towards high BLEU or high NIST scores, more recently also towards METEOR or TER (Snover et al., 2006).

We used a Minimum Error Training (MER) training module (Venugopal, 2005), originally developed for a statistical MT system, to run MER training on the *n*-best lists generated by the Xfer system for the BTEC test set. We used the BLEU mteval metric as the objective function.

As shown in Table 5.13, when setting optimal weights in the Xfer engine for the language model (LM) and fragmentation penalty scores, both the Baseline and the Refined system obtain statistically significantly higher scores, not only according to BLEU, but also to the METEOR and NIST automatic evaluation metrics (Section 5.4.4). For more detail about the MER training procedure we followed, see Appendix B.

System	METEOR	BLEU	NIST
Baseline	0.6184	0.3609	6.68
Refined	0.6231	0.3780	6.79

Table 5.13. Automatic metric scores for 1-best decoder hypothesis, after LM and Fragmentation weights have been optimized.

Source: where is the boarding gate ?
NO: dónde está *el embarque puerta* ?
WO: dónde está la puerta embarque ?
Src: where is the bus stop for city hall ?
NO: dónde está *el autobús parada* para ayuntamiento ?
WO: dónde está la parada autobús para ayuntamiento ?
Src: i would like a twin room with a bath please .
NO: me gustaría habitación una cama doble con un baño por favor .
WO: me gustaría una habitación cama doble con un baño por favor .
Src: i would like to buy some duty-free items .
NO: me gustaría comprar algunos *duty-free productos* .
WO: me gustaría comprar algunos artículos duty-free .
Src: does he speak japanese ?
NO: él *hablar a* japonés ?
WO: habla japonés ?
Src: it is just round the corner .
NO: *lo* es simplemente a la vuelta de la esquina .
WO: es simplemente a la vuelta de la esquina .
Src: do you sell duty-free items ?
NO: *te* venden artículos duty-free ?
WO: vendéis artículos duty-free ?

Table 5.14. 1-best translations from the BTEC test set output by the Refined MT system before and after MER training.

Table 5.14 shows a few examples from the BTEC corpus with 1-best translations output by the Refined MT system before (No Optimization) and after (With Optimization) MER training, given LM and fragmentation penalty scores. From these examples, it can be observed that re-ranking improves after optimizing the LM and fragmentation weights.

Analyzing the translation results, one important shortcoming became obvious. Currently the translation lexicon only covers about 88% of the words that appear in the reference translations. This severely limits what kind of BLEU score we can achieve. When we generated the phrasal lexicon from the BTEC training data, we deliberately chose to only include few alternatives, mainly to limit the manual labor when adding POS and constraint, but also to control

the amount of noise in the lexicon. Further augmenting the lexicon, especially with translations present in the two sets of reference translations, is expected to increase these results.

5.4.3.5 Automatically Augmented Lexicon

In order to further investigate the reference coverage limitations, we took all the BTEC training data and automatically generated word-to-word and phrasal alignments similarly to what we described in Section 5.2.1.1. Since there were too many entries to manually add POS information, we used the Spanish POS-tagger from the Universitat Politècninca de Catalunya (http://www.lsi.upc.es/~nlp/freeling) (Atserias et al., 2006) to annotate both the Spanish and English sides. Freeling uses the PAROLE tag set, which we automatically mapped onto the POS tag set used by our grammar rules. No feature constraints were added.

This automatically generated lexicon was added to the previous lexicon (manually crafted plus semi-automatically generated); the number of lexical entries went from 1,732 to 25,0921.

In order to test our hypothesis, we run our best performing Refined MT system with the automatically augmented lexicon, which yield a further increase of 2 BLEU points (0.3908).

5.4.4 Significance tests

The results for the unseen data set, the BTEC test set, show consistent improvements across all automatic evaluation metrics. However to address the question of whether such improvements are statistically significant or not, further tests need to be done.

According to the standard paired two-tailed t-Test (Gosset, 1908), both the ORACLE and the decoder METEOR scores with optimized weights are statistically significant, with a p value of 4.42×10^{-6} and 0.0051 respectively. P values of less than 0.01 are interpreted as being significant. Appendix C shows the results of the paired two-tailed t-Tests for the BTEC METEOR scores.

Work using bootstrapping techniques has been done by Zhang and Vogel (2004) to determine what improvements to BLEU and NIST scores are indicative of having a better MT system. We used their method to calculate confidence intervals and statistical significance for BLEU and NIST scores, and found that results reported in ORACLE and optimal weighted decoder clearly indicate that the Refined system is better than the Baseline system both according to BLEU and NIST scores. In particular, for the results in Table 5.13, the Refined system is better between [0.84, 2.35] BLEU points, 95% of times, with a median difference of 1.54. For NIST scores, the difference between the Baseline and the Refined systems is 0.1079, in terms of the median, between [0.0570, 0.1633].

5.5 Discussion

The results presented in this chapter illustrate how translation rules can be automatically refined to increase overall translation recall and precision. Experiments with an English-Spanish Transfer-Based MT system show that automatic refinements generalize well beyond the specific sentences corrected by users to previously unseen data. The added generation capabilities of the refined MT system led to improved translation accuracy as measured by different automatic evaluation metrics.

Figure 5.15 to 5.17 show a summary of the results on unseen test data (BTEC) described at length in Section 5.4.3.

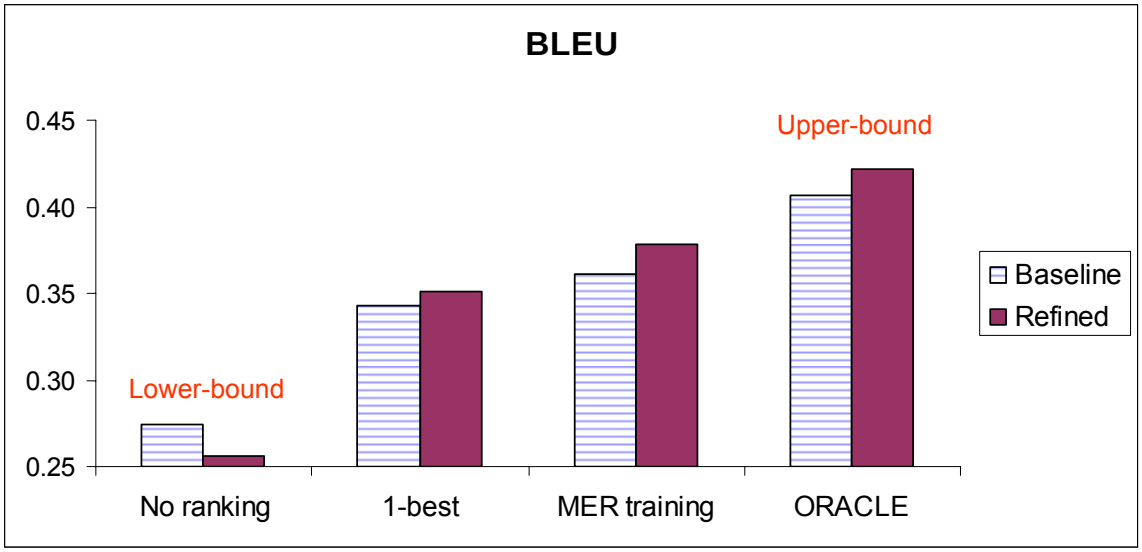


Figure 5.15. Summary of results for the BLEU automatic evaluation metric.

The goal of any MT system is to allow the right amount of ambiguity namely the one that naturally occurs in language and that the system seeks to preserve. However, the right balance between too general and too specific is hard to strike. In the presence of a decoder, which can compensate for the excess of translation hypotheses produced by the lexicon and grammar rules combination, one could argue that underspecified grammar rules are preferred to overly specific rules, since the former type allows for greater generalization. Similarly, in such a context, the CONSTRAIN operation might become superfluous in the presence of a strong decoder.

However, in general, language models are rather weak in finding structural information such as agreement. While MT researchers are trying to modify the LM to incorporate some

structure in order to make it more discriminative between good and bad translations, which should help the decoder pick the best translation hypothesis, our approach contributes to that same end from a different perspective. In particular, the Automatic Rule Refiner can get rid of incorrect translations by reducing grammar ambiguity (adding agreement constraints to underspecified rules in the grammar), which effectively helps the decoder's task, since the *n*-best list is now less noisy and the language model is not distracted by incorrect translations.

Our experiments suggest that striking a good balance between overly general rules and overly specific rules and controlling unnecessary ambiguity can also help the decoder pick the best translation, and is thus a legitimate goal.

These results represent a lower-bound on recall, since the MT system used for these experiments did not include a morphology module. If the system had a Spanish generation morphology module, refinements would generalize to all forms of a specific word, and thus the impact on unseen data could be much larger.

At a larger scale, and even though the Automatic Rule Refiner takes several steps to filter out noise early in the process (Section 4.5.1.1), automatic refinements are still entirely based on non-expert corrections, which ultimately means that there is no guarantee that automatic refinements will always increase translation accuracy.

In this respect, having shown that a decoder can pick up grammar and lexicon improvements generated by automatic refinements, we have closed the feedback loop. This allows such a system to validate refinements that lead to measurable improvements on translation accuracy, and discard those refinements that results in degraded performance (4.5.2.1.1).

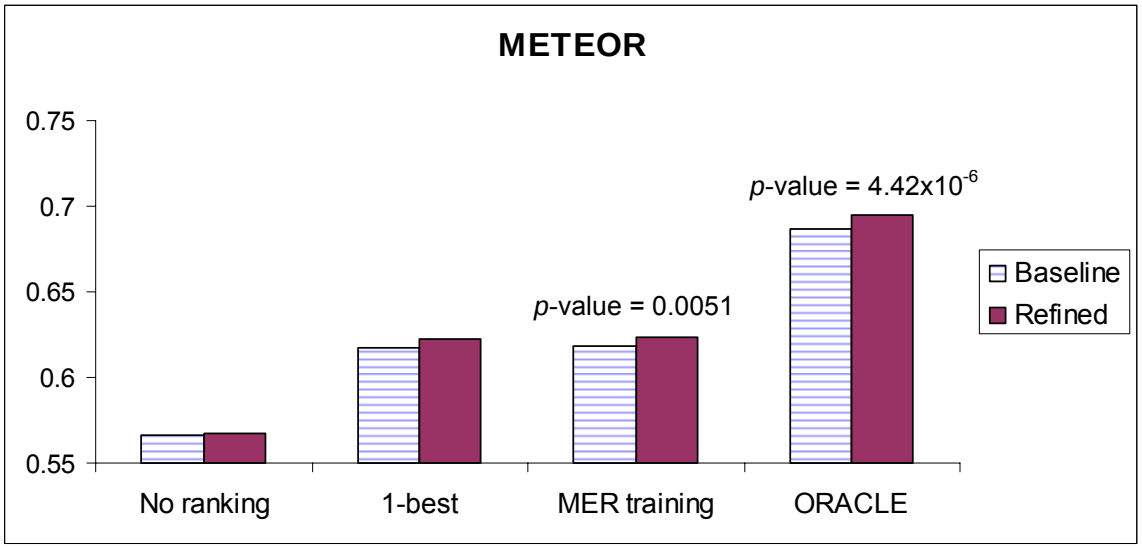


Figure 5.16. Summary of results for the METEOR automatic evaluation metric.

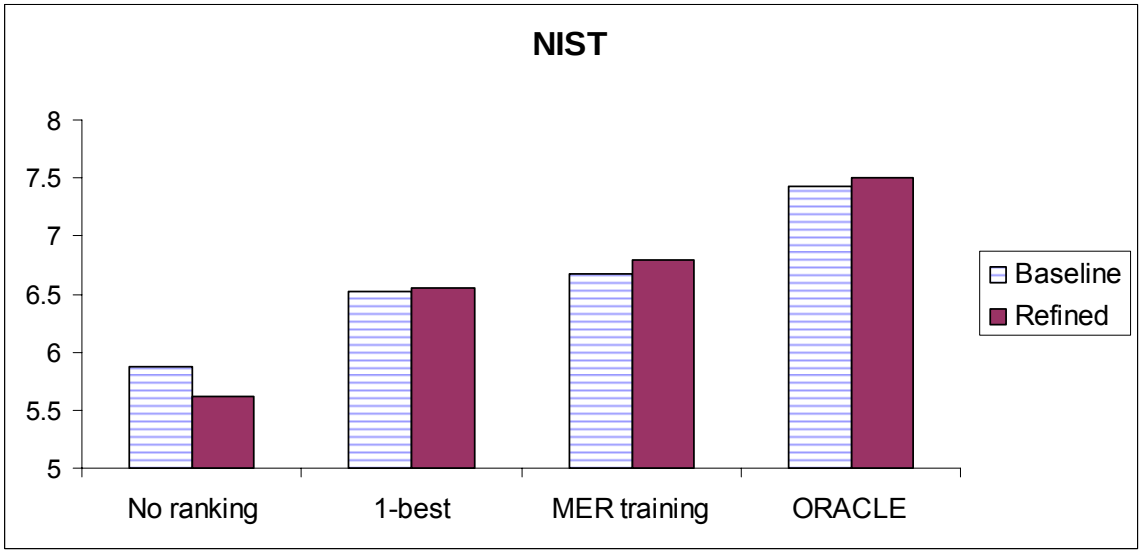


Figure 5.17. Summary of results for the NIST automatic evaluation metric.

Chapter 6

Porting to a different language pair: Mapudungun-Spanish

6.1 Motivation

Implementation and evaluation of our rule refinement methods were done for English-Spanish Transfer-Based Machine Translation (Chapters 4 and 5). The next logical question is how well do these methods port to a different language pair? More specifically, how well does it port to a language pair involving a resource-poor language, preferably a non-Indoeuropean language?

The purpose of this chapter is to show the extent to which our automatic rule refinement approach is language independent. We had two candidate Machine Translation systems available: a fairly well-developed Mapudungun-Spanish MT system and an initial prototype Quechua-Spanish MT system. The first system was developed over two years by a team of computational linguists; grammar development took one year and was done by a linguist with expertise on both Mapudungun and Spanish. Grammar development for the second system took two months, and remains incomplete.

Therefore, automatically improving the Mapudungun-Spanish grammar, which is much more mature, with 139 rules, is the greater challenge. In contrast, the English-Spanish grammar has 40 rules and the Quechua-Spanish prototype grammar has 25 rules. Moreover, the Mapudungun-Spanish grammar was written by a third party (Roberto Aranovich), whereas the Quechua-Spanish grammar, like the one for the English-Spanish system, was written by the author of this thesis.

Since the goal of this experiment is to test the automatic refinement mechanisms on a system as different to the English-Spanish system as possible, we decided in favor of the Mapudungun-Spanish system, for which the author of the thesis had no control over the transfer given to the Automatic Rule Refiner.

6.2 Mapudungun and the Mapuche

Mapudungun[35] is a language isolate spoken in central Chile and west central Argentina by the Mapuche people or "people of the land" (*che* means 'people'). It is also known as Mapudungu, Araucanian (Araucano) (the name given to the Mapuche people by the Spanish) and Mapuche.

Its speakers number 440,000, with 400,000 in the Central Valley of Chile (see map in Figure 6.1) and 40,000 in the Argentinean region of Patagonia. Some 200,000 people use the language regularly. The majority of the Mapuche are bilingual. Spanish is the official language, of primary use in the school and public life, and is associated with progress (INDEC, 2005).

When the Spanish arrived in Chile, they found four groups of Mapuche: the Picunche (from *pikum* 'north' and *che* 'people'), the Huilliche (*huilli/willi* 'south'), the Pewenche (*pewen* 'pine seeds'), who inhabited the slopes and valleys of the Andean Cordillera and the central group, the Mapuche. The Mapuche who live in Argentina are often referred to as Argentinian Mapuche.

Even though there is no demonstrable genealogical or genetic relationship with other living languages, Mapudungun has been classified by some authorities as being related to the Penutian languages of North America. Others group it among the Andean languages (Greenberg, 1987; Key, 1978), and yet others postulate it is related to Mayan (Hamp, 1971). It has some lexical influence from Quechua and Spanish.

[35] M*apu* means 'land, earth', and *dungu* means 'idea, matter, language, word'; *-n* is a verbal noun marker.

Figure 6.2. Map of Chile (on the left) and map of the 9th region (on top). The city of Temuco is the geographical and socio-economic center of Mapuche territory.

6.3 AVENUE Project

The Chilean Ministry of Education has created programs to ensure that each child's cultural and linguistic needs are met in school. In collaboration with the AVENUE team at the Language Technologies Institute (Carnegie Mellon University), the Ministry of Education provided support for the collection of data and other tasks related to building a lexicon and an MT system for Mapudungun.

Native speakers of Mapudungun at the Universidad de la Frontera in Temuco (Chile) were in charge of the data collection, gathering a Mapudungun-Spanish parallel corpus of historical texts and newspaper text and a large parallel corpus consisting of 170 hours of transcribed speech in Mapudungun, which was then translated into Spanish.

In addition, frequency-ordered word lists were created from the corpus. A spelling checker was developed based on the stems and suffix groups in the word list (Monson et al. 2004). The spelling checker uses one boundary between a stem and a list of suffixes. A more sophisticated morphological analyzer was also developed, which identifies all suffixes attached to a stem.

These resources have been used to create the lexicon for our Rule-Based MT system. The grammar was manually develop to cover the AVENUE Elicitation Corpus (Probst et al., 2001) and was tested on unseen data from the family, health and greetings domains.

For a detailed description of the AVENUE project as well as the resources developed for Mapudungun and Quechua, see Font Llitjós *et al*. (2005a).

6.4 Challenges of Translating from Mapudungun

The formalization of Mapudungun poses several challenges, all of which affect the process of Machine Translation. Unlike English and Spanish, historically Mapudungun has not been written; the language of the Mapuche is been preserved though oral tradition. Mapudungun was first written down by missionaries, and the orthographic systems they used were adaptations from European languages, and varied from author to author. As a result, the many written documents that exist today do not all use the same orthographic conventions, nor do they fully agree on an alphabet.

The first challenge our team faced was to agree on an orthography and an alphabet to be used for the project and at the same time could be accepted and used by the Mapuche community.

In contrast with English (isolating and analytic) and Spanish (fusional and analytic), Mapudungun is a highly agglutinative[36] and polysynthetic[37] language. Mapudungun makes extensive use of suffixation, compounding and reduplication, and its verbal morphology is complex. A typical Mapudungun verb is inflected by means of a rather strictly ordered sequence of suffixes. These suffixes occupy slots in the morphological structure of verbs. Smeets (1989) has identified 36 slots in Mapudungun verbs, while according to Zuñiga (2000), the number of slots is 22. A typical complex verb form occurring in our corpus of spoken Mapudungun consists of five to six agglutinated morphemes.

A verb in Mapudungun can express the entire content of a simple clause in English or Spanish. To give an example, the Mapudungun verb *pefiymu* (*pe-see fi*-EXTERNAL DIRECT OBJECT MARKER *y*-INDICATIVE *m*-2nd PERSON *u*-DUAL), translates into English "*you saw him/her/it/them*". In this example the subject pronoun *you* refers to exactly two people. The reason we know this is that, in addition to singular (one person) and plural (which is used to refer to more than two people), Mapudungun marks pronouns and verbs with dual case (*u*-), which is used to refer to precisely two items (in this case people).

In terms of translating between Mapudungun and Spanish, the most important divergences are morphosyntactic in nature. By morphosyntactic divergence we mean a systematic divergence in the degree of synthesis (Comrie, 1989) of different grammatical structures, due to the properties of each language in terms of morphological typology. An example of morphosyntactic divergence is head-switching, which takes place when the head of an expression in one language is not translated as a head but as a modifier or complement in the TL. Roberto Aranovich (2007) focuses on describing such divergences between Mapudungun and Spanish and how they are handled in the Mapudungun-Spanish MT system used for our experiments.

---

[36] In linguistics, agglutination is the morphological process of adding affixes to the base (stem) of a word. Languages that use agglutination widely are called agglutinative languages. Examples of other agglutinative languages are Basque, Finnish and Turkish. These languages are often contrasted with fusional languages and isolating languages. However, both fusional and isolating languages may use agglutination in the most-often-used constructs, and use agglutination heavily in certain contexts, such as word derivation. This is the case in English, which is an isolating language, but has an agglutinated plural marker *-(e)s* and derived words such as *shame·less·ness*.

[37] Polysynthetic languages have words composed of many morphemes, often with strict rules of lexical composition. Examples of other polysynthetic languages are Chukchi and Mohawk.

For more comprehensive descriptions of the Mapudungun grammar, the interested reader can see Smeets (1989) and Arnold (1996) (in English), and Catrileo (1988) and Salas (1992) (in Spanish).

6.5 Mapudungun-Spanish MT System

The input to the Rule-Based Machine Translation system is a Mapudungun sentence, phrase or word, which is processed as illustrated in Figure 6.3 to generate Spanish output.

Unlike the English-Spanish MT system described in Chapter 5 and used for all of our previous experiments, the Mapudungun-Spanish system has two additional morphology components. Thus, the Mapudungun-Spanish MT system consists of three components: a Mapudungun morphological analyzer, a transfer system, and a Spanish morphological generator (Figure 6.2). This adds two layers of complexity to the translation and automatic refinement processes.

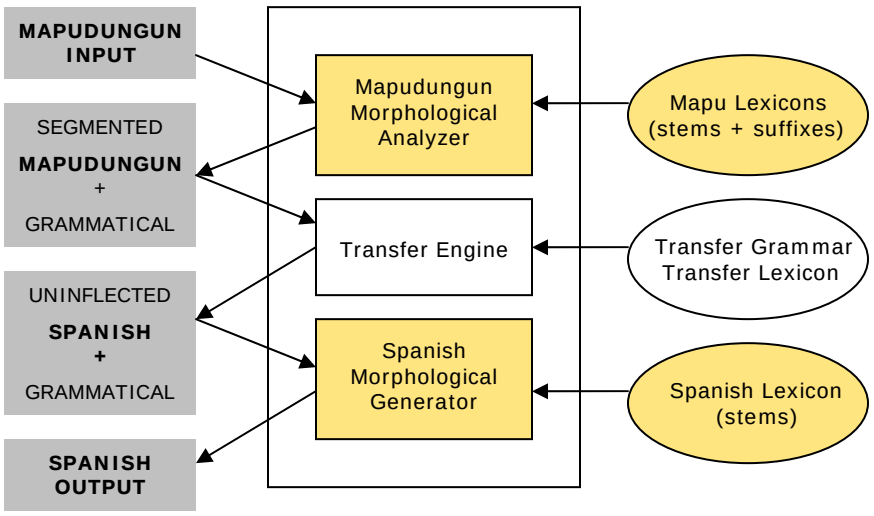


Figure 6.3. Mapudungun-Spanish MT system architecture diagram. The two morphology components were not present in the English-Spanish MT system (Figure 5.4).

The Mapudungun morphological analyzer makes use of two separate Mapudungun lexicons, one containing a list of stems specified with parts-of-speech, and a second one containing a list of suffixes, each one specified with grammatical features. The input to the morphological analyzer is a Mapudungun expression and its output is a morphologically segmented expression plus a specification of the grammatical features of each morpheme, which constitutes the input for the transfer system.

Like in the English to Spanish system described in Chapter 5, the Transfer system for Mapudungun-Spanish makes use of a transfer grammar and a transfer lexicon, which contain syntactic and lexical rules in order to map Mapudungun expressions into Spanish expressions. The output of the transfer system is a Spanish expression composed of uninflected words plus grammatical features, which constitutes the input for the Spanish morphological generator.

The transfer lexicons were constructed from the spoken language corpus and the Elicitation Corpus. All the unique words were extracted from the corpora, and were ordered by frequency. This word frequency list was segmented and used as a guide for translation dictionary development. There were two dictionary development efforts. One effort was led by the Chilean team to create an online translation dictionary with examples of usage (1,926 entries). The morphemes were labeled by native speakers who are not linguists. The other dictionary development effort was conducted by the LTI team, originally derived from the first one, to create a translation lexicon for the MT systems, which included the translations as well as some grammatical features required for the correct application of the transfer rules (such as number and person). The transfer lexicon used in our experiments contained 2,124 lexical entries. Figure 6.3 illustrates examples of lexical entries.

```
{N,8} N::N |: [ruka] → ["casa"]
        ((X1::Y1)
        ((x0 person) = 3)
        ((x0 type) = common)
        ((y0 gender) = fem))

{V,5} V::V |: ["af"] → ["terminar"]
        ((X1::Y1)
        ((x0 lexicalaspect) = episodic)
        ((x0 subcat) = intr)
        ((y0 reflex) = +))

{V,30} V::ADJ |: ["atregnge"] → ["frío"]
        ((X1::Y1)
        ((x0 lexicalaspect) = stative)
        ((x0 morph) = deadj)
        ((x0 subcat) = intr)
        ((y0 AUX form) = ser))

{V,39} V::N |: ["chadinge"] → ["sal"]
        ((X1::Y1)
        ((x0 lexicalaspect) = stative)
        ((x0 morph) = denom)
        ((x0 subcat) = intr)
        ((y0 AUX form) = ser))
```

Figure 6.4. Examples of Mapudungun-Spanish lexical entries.

The transfer grammar for Mapudungun was developed by Roberto Aranovich, a linguist with knowledge of both Spanish and Mapudungun. With 139 rules, it covers the basic grammatical constructions (simple sentences with intransitive and transitive verbs, nominal phrases with determiners and modifiers, verbal phrases with different temporal and aspectual values, passive voice, inverse marking, etc.). Figure 6.4 shows examples of grammar rules.

The Spanish morphological generator (Atserias et al., 2006) makes use of a Spanish lexicon of inflected words.

For more information on the Mapudungun-Spanish MT system, see Font Llitjós et al. (2005a) and for details of the transfer grammar, see Aranovich (2007).

```
{NBar,1}      ;; pl N with collective PU
        NBar::NBar : [PART N] → [N]
        ((X2::Y1)
        ((X2 morph) = *UNDEFINED*)
        ((X1 number) =c pl)
        ((X0 person) = 3)
        ((X0 number) = (X1 number))
        ((X0 human) = (X2 human))
        ((X0 commitative) = (X2 commitative))
        ((X0 type) = (X2 type))
        ((Y0 person) = (X0 person))
        ((Y0 number) = (X0 number))
        ((Y1 number) = (Y0 number))
         ((Y1 person) = (Y0 person))
        ((Y0 gender) = (Y1 gender)))

{NP,10}   ;;NP with oblique case marker (meu)
NP::PREPP : [NBar NSuff] → ["en" DET NBar]
        ((X1::Y3)
        ((X2 commitative) = *UNDEFINED*)
        ((X2 number) = *UNDEFINED*)
        ((X0 person) = (X1 person))
        ((X0 number) = (X1 number))
        ((Y3 number) = (X0 number))
        ((Y2 number) = (Y3 number))
        ((Y2 gender) =c (Y3 gender))
        ((Y2 form) = el))
```

```
{VBar,1}      ;; "plain" VBar
        VBar::VBar :  [V TenseGroup] → [V]
        ((X1::Y1)
        ((X1 lexicalaspect) =c (X2 myVLexAsp))
        ((X2 progressive) = *UNDEFINED*)
        ((X2 object person) = *UNDEFINED*)
        ((X2 object number) = *UNDEFINED*)
        ((X2 voice) = *UNDEFINED*)
        ((X1 morph) = *UNDEFINED*)
        ((X2 nonf-type) = *UNDEFINED*)
        ((X2 mood) = (*NOT* imper))
        ((X0 person) = (X2 person))
        ((X0 number) = (X2 number))
        ((X0 tense) = (X2 tense))
        ((X0 mood) = (X2 mood))
        ((X0 reportative) = (X2 reportative))
        ((X0 negation) = (X2 negation))
        ((X0 subcat) = (X1 subcat))
        ((Y0 person) = (X0 person))
        ((Y0 number) = (X0 number))
        ((Y0 tense) = (X0 tense))
        ((Y0 mood) = (X0 mood))
        ((Y1 person) = (Y0 person))
        ((Y1 number) = (Y0 number))
        ((Y1 mood) = (Y0 mood))
        ((Y1 tense) = (Y0 tense))
        ((Y1 reflex) = *UNDEFINED*))
```

Figure 6.5. Examples of Mapudungun-Spanish grammar rules.

6.6 Generality of Automatic Rule Refinement Approach

For the purpose of testing language independence of our rule refinement approach, we decided to test one refinement that affects the transfer lexicon and one that affects the grammar.

Despite the differences between Spanish and Mapudungun (Section 6.4), the same general refinement mechanisms developed for English-Spanish (Chapters 4) applied directly to the Mapudungun-Spanish MT system.

Porting the Automatic Rule Refiner (ARR) to such a different language pair did involve a couple of minor adjustments, however, which we describe in the following section. Next, we introduce the two Correction Instances that were processed by the ARR and describe the refinements yielded by such corrections. Finally, we conclude with a small evaluation on a subset of the AVENUE Elicitation Corpus.

6.6.1 Representation Issues

Because of Mapudungun orthography and the original design of the Mapudungun-Spanish MT systems, the parts of the Mapudungun-Spanish MT system that interact with the Automatic Rule Refiner (input string, grammar and lexicon files, file with transfer commands, etc.) were converted to UTF-8 encoding.

Two additional morphology layers used to process Mapudungun input and generate Spanish output, added significant complexity to the low-level representation and implementation of the ARR. Unlike the English-Spanish Lexicon, which contains fully-inflected forms ([eats] → [come]), the Mapudungun-Spanish Lexicon contains citation forms ([yafütu] → [comer] for '*to eat*', which together with morphemes for 3rd person singular will generate the fully inflected form *come*).[38] In terms of automatic rule refinement, this means that when bilingual speakers add or modify a Spanish word via the Translation Correction Tool (Chapter 3), the word first needs to be processed using the Spanish morphology module, so that the citation form is added to the lexicon, instead of the fully-inflected word.

In the TCTool, Mapudungun morphemes are represented as separate units aligned to the corresponding Spanish word, so that multiple units (e.g. verb stem plus verbal suffixes) are typically aligned to just one word in Spanish. This design decision of depicting Mapudungun stems and suffixes as separate units, as opposed to as a single unit like it is done for Spanish, was based on the fact that the underlying transfer Lexicon and Grammar manipulates them separately.

Figure 6.5 provides an example for the sentence "*the tree protects the animal*" and its TCTool representation. The Mapudungun verb *ñukumtukefi* is depicted as the stem *ñukumtu-* (*to protect*) and the two suffixes *-ke-* (HABITUAL) *-fi* (3rd PERSON, SINGULAR, INDICATIVE, OBJ is 3rd PERSON).

It is important to note that unlike Spanish, morpheme boundaries in Mapudungun tend to be regular, and thus morphemes can be directly added to the stem (citation form) to produce the correct word from. This facilitates the use of a morphematic representation on the Mapudungun side.

On the other hand, one could choose to represent Mapudungun words as unsegmented units. This would have the advantage of resembling more how Mapuche might write the sentence,

[38] Note that this is not an issue specific to Mapudungun, but rather to the fact that a Spanish generation morphology module was used for this system. If the English-Spanish system had also included a generation morphology module, the same issues would have appeared.

but the disadvantage that the bilingual speakers need to analyze and segment the word themselves before they can determine whether the Spanish translation is correct.

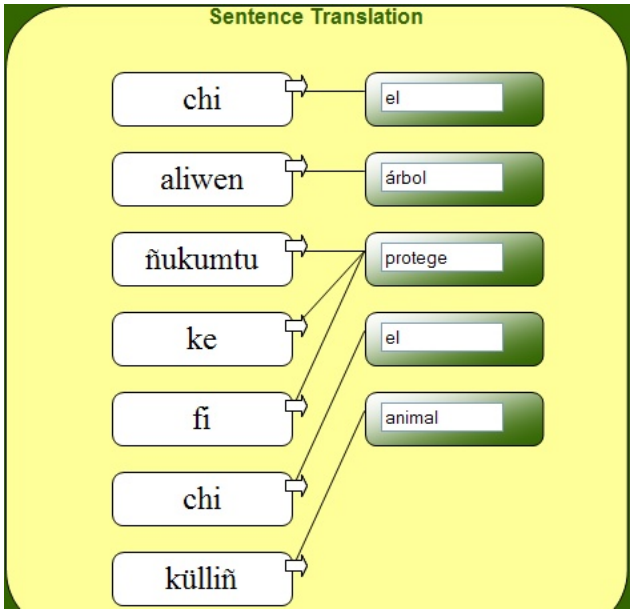


Figure 6.6. TCTool snapshot for a Mapudungun-Spanish example meaning "*the tree protects the animal*". Note that stem and suffixes are represented as separate units in Mapudungun and the Spanish sentence has not yet been corrected.

In order to determine representational preference of TCTool users, one would have to perform a set of user studies testing competing representations and how they relate to error correction accuracy as well as interface usability. A preliminary user study was done for the Quechua-Spanish system, where Quechua stems and morphemes were also represented as separate units, and we found that such representation when correcting MT output posed no obstacle to bilingual speakers (Section 3.5; (Font Llitjós et al., 2005a)).

6.6.2 Correction Instances

After manual inspection of MT output generated by the Mapudungun-Spanish system during an independent evaluation, we selected two Mapudungun sentences that produced incorrect translations to train the Automatic Rule Refiner. Figure 6.6 shows the two sentence pairs with their Correction Instances.

The correction of the error in the first sentence (which corresponds to the one displayed in Figure 6.5) is triggered by the fact that animate direct objects in Spanish require a marker (*a*). In this case, the incorrect MT output is corrected to "*el árbol protege al animal*" by the bilingual

speaker. This error also occurred in the English-Spanish MT output and was remediated by the ARR. The Mapudungun-Spanish MT system correctly handles human directs objects (los cazadores-*hunters*, Juan, la mujer-*woman*, etc.), adding the animate marker in front of them, however it does not do the same for animate, non-human direct objects (animal, culebra-*snake* guanaco, etc.). Correcting this error involves extending the Lexicon. This is described in the next section.

"Pu" is a plural marker in Mapudungun, and thus the subject of the second sentence is missing a masculine, plural determiner. The bilingual informant added *los* (definite determiner) in front of *niños* ("*los niños jugaron un juego*").[39] This error correction is new to the ARR, even though English-Spanish systems are also prone to exhibiting such errors, since English bare noun phrases (NPs) are often translated into definite NPs in Spanish. Note that *los* is not aligned to anything on the Mapudungun side, which results in a grammar refinement, for there is nothing that can be done at the lexical level.

1. **SL**: chi aliwen ñukumtu ke fi chi külliñ
   TL: el árbol protege el animal (*English: the tree protects the animal*)
   Alignements: ((1,1),(2,2),(3,3),(4,3),(5,3),(6,4),(7,5))
   Action 1: edit (W_4=el → W_4'=al)
   CTL: el árbol protege al animal
   Alignements: ((1,1),(2,2),(3,3),(4,3),(5,3),(6,4),(7,5))

2. **SL**: pu püchükeche awkantu y kiñe awkantun
   TL: niños jugaron un juego (*English: children played a game*)
   Alignements: ((1,1),(2,1)),(3,2),(4,2),(5,3),(6,4))
   Action 1: add (W_1=los)
   CTL: los niños jugaron un juego (*English: the children played a game*)
   Alignements: ((1,2),(2,2)),(3,3),(4,3),(5,4),(6,5))

Figure 6.7. Correction Instances processed by the ARR in order to extend both the Mapudungun-Spanish transfer Lexicon (1) and Grammar (2).

In the next section, we describe how the ARR processes these two Correction Instances and what are the resulting expansions of the Lexicon and the Grammar.

[39] If the bilingual speaker had added *unos* (indefinite determiner) instead of *los* in front of *niños*, this would have resulted in a different automatic refinement, namely the creation of a new rule ([**"unos"** NBar]). Our approach mitigates user variability by eliciting the same correction from multiple informants and taking a majority vote approach (Section 4.6.1.2).

6.6.3 Automatic Refinements

6.6.3.1 Lexicon

The processing of the first Correction Instance shown in Figure 6.6 results in a BIFURCATE operation of the lexical entry for the definite determiner *chi* (the-sg.masc) (DET,11), which creates a copy of the original rule (DET, 27). The Spanish translation *el* in the copy is substituted with the translation proposed by the bilingual speaker *al* (Figure 6.7).

This lexical refinement correctly generalizes to all other animate direct objects that are definite, masculine and singular (el guanaco, el gato-*cat*, el perro-*dog*, etc.), following the principle of minimal (safest) generalization mentioned in Section 4.6.4. Other determiners do not involve a conflation of the preposition '*a*' with the determiner (*a la*, *a los*, *a las*, *a un*, *a unas*, etc.) and so they can only be corrected after processing a different sentence pair, similarly to example 8 in Figure 5.1.

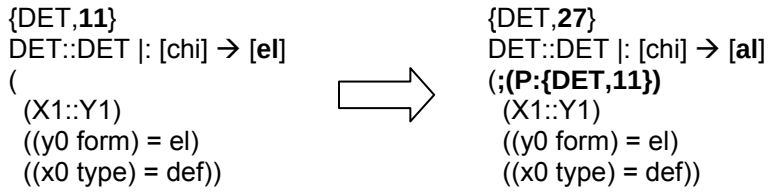


Figure 6.8. Refined lexical entry result of a BIFURCATE operation of lexical entry DET, 11 ([chi]→[el]) that substitutes *el* with the translation proposed by the informant (el→ al).

If the bilingual speaker had identified a clue word (*animal*), the ARR would postulate a new binary feature in order to discriminate between DET,11 and DET,27 and would label the lexical entry of the clue word ([külliñ]→[animal]) with that feature attribute and the same value as the lexical entry for [chi]→[al]. However that would result into a poor generalization of the refinement, since only direct objects with *animal* as the head would be translated correctly, but all other ones would not.

6.6.3.2 Grammar

Processing the second sentence in Figure 6.6 above results in a BIFURCATE operation of the NP grammar rule that applied to translate the subject of the sentence (NP,5) that adds the terminal *los* to the right hand side of the copy (NP,20), as illustrated in Figure 6.8.

This grammar refinement correctly generalizes to all definite NPs that are plural masculine. Given other corrected examples (*las* NP, *el* NP, *la* NP), the ARR would initially add

specific grammar rules for all of the different number and gender combinations. In order to make the grammar more compact, it would be straightforward to merge such overly specific rules into a single one during a post-processing step (DET NP). This is left as future work, and would entail an upward movement in a seeded version space to replace value constraints with a more general relational constraint (see Section 7.1.2).

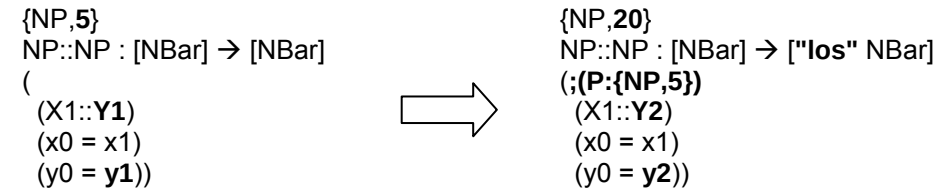


Figure 6.9. Refined grammar rule result of a BIFURCATE operation of original rule NP,5 ([NBar]→[NBar]) that adds *los* to the first position of the right hand side of the rule.

6.7 Evaluation

In order to quantify the effect of these two refinements on a subset of the AVENUE Elicitation Corpus (EC), we selected two relevant partitions of the EC made by Roberto Aranovich. The first one contained transitive verbs (82 sentences), so that we can look at what happens to animate direct objects, and the second one, definite NPs (48 sentences). Since both subsets contained phrases relevant to both refinements, we merged them into a single test set (130 sentences). Tables 6.1 and 6.2 show the total and unique number of relevant phrases in the test set (4th row) and the number of them that were corrected by the ARR (last row).[40] The total number of definite direct objects and subject NPs are also provided to give an idea of what fraction of them were targeted by the automatic refinements.

These numbers show that from 14 sentences where the first refinement (el→al) could have had an effect, 13 were translated correctly (Table 6.1), and that from the 15 sentences where the second refinement (+los) could have had an effect, 9 were correctly translated (Table 6.2). For the remaining 7 sentences, the MT system did not produce a correct translation (1+6) because

[40] Since there is no automatic way to extract the exact number of subject NPs and direct objects automatically, this was manually done, and thus these numbers represent lower bounds. At least these many subjects/objects were found to be affected, but there could be more in the corpus.

they were not fully parsed, and thus the transfer engine only output the first partial translation, which did not contain the correction.

	Total (tokens)	Unique (types)
Sentences from EC test sets	130	130
Definite Direct Objects (DDO)	73	34
Animate (non-human) DDO (ADDO)	28	9
ADDO + masc + singular (ADDOms)	**14**	**4**
ADDOms correctly translated (AL)	**13**	**3**

Table 6.1. Total and unique number of sentences and definite direct objects of several characteristics that are relevant to the automatic refinement described in 6.6.3.1.

	Total (tokens)	Unique (types)
Sentences from EC test sets	130	130
Subject Noun Phrases (S-NP)	58	27
Definite S-NPs (DS-NP)	55	25
Definite S-NPs + plural (DS-NPp)	17	9
DS-NPp + masc (DS-NPpm)	**15**	**8**
DS-NPpm correctly translated (+LOS)	**9**	**7**

Table 6.2. Total and unique number of sentences and subject NPs of several characteristics that are relevant to the automatic refinement described in 6.6.3.2.

Note that there is a significant difference between the total number of relevant phrases (tokens) and the number of relevant phrases that are unique (types). This is due to the repetitive nature of the Elicitation Corpus, since its goal is to elicit the underlying grammatical features of the target language, and so the same subject and object appear combined with multiple verbs and with different forms of the same verb.

The fraction of sentences with relevant definite direct objects (animate, masculine and singular) represents 18% of all definite direct objects that appear in this test set, and goes down to 9% when looking at unique definite direct objects. On the other hand, the fraction of sentences with relevant subject NPs (definite, masculine and plural) represents 15% of all the subject NPs, and goes up to 26% when looking at unique subject NPs.

6.8 Conclusion

In this chapter we have shown that the same mechanisms to automatically extend and refine an English to Spanish MT system are valid when applied to a very different language pair, in this case Mapudungun-Spanish. The principles of automatic rule refinement, blame assignment and principles of specificity/generality do not change. The fact that Spanish and Mapudungun are typologically very different allows us to argue that our automatic refinement approach is language independent to a large extent. However, both language pairs evaluated have Spanish as target language, and thus the set of MT errors that can be corrected by the ARR overlap.

Given other very different language pairs, we do anticipate the need to address representational issues as well as to adapt parts of the low-level implementation. In our experience, encoding issues and the presence or absence of morphology modules are the main causes of adaptation when porting our ARR approach to a new language pair.

Chapter 7

Conclusions and Future Work

This thesis presents a largely automated approach to generalize post-editing efforts in order to improve Transfer-Based MT systems. This is an alternative to traditional Automated Post-Editing approaches, which modify MT output but not the core MT technology, where the MT systems do not learn from their past errors. The Automatic Rule Refiner described in our work extends and improves the lexicon and the grammar to account for exceptions not originally encoded in the translation rules, to handle alternatives not envisioned by the grammar written (e.g., pre-nominal adjectives in Spanish), and to make overly general rules more specific to reduce ambiguity.

Experiments on English to Spanish and Mapudungun-Spanish Transfer-Based MT systems show that automatic refinements to the translation grammar and lexicon increase MT output quality. The Rule Refinement process is not commutative in general; it depends on the order in which refinement operations are applied. In off-line processing, the ARR can rank correction instances in such a way as to maximize translation accuracy.

The general contributions of this thesis are: (1) the development of a mechanism to extend and improve underlying Transfer-Based MT Systems without requiring expert intervention, (2) the improvement of MT quality on unseen data as measured by automatic evaluation metrics, and (3) the application of this approach to two very different language pairs: English-Spanish and Mapudungun-Spanish. The major specific contributions are:

- A system and language independent online GUI to display Machine Translation output and solicit pinpoint fixes from non-expert bilingual users (Translation Correction Tool). This constitutes a new method for generating manually corrected and aligned bilingual

corpora. Furthermore, the corrected data constitutes ideal reference translations since it provides relevant information to address specific errors made by MT systems. This data has been recently shown to be useful to train a phrase-based MT system that significantly improves the performance of a baseline Rule-Based MT system (Simard et al., 2007a).

- An English-Spanish user study with the online Translation Correction Tool shows that users can detect errors with high precision (90%). This makes non-expert bilingual speakers a reliable source of MT output corrections.

- The demonstration of effectiveness of our MT error typology for automatic rule refinement and the validation of the appropriate amount of error information that can be reliably provided by non-expert users, abstracting away from linguistic and technical translation issues (clue words).

- A mapping between error corrections and automatic rule refinement operations. Based on concrete MT output fixes provided by non-experts and implicit error information gathered by the Translation Correction Tool (correction actions, alignment information, and word order), as well as the current state of the grammar and the lexicon, the Automatic Rule Refiner can determine what set of refinement operations can be safely applied automatically.

- A language independent mechanism to extend and improve translation grammars and lexicons, with an expandable set of rule refinement operations and schemata. The Automatic Rule Refiner involves blame assignment, triggering feature detection and rule modification, and can be generalized to all Transfer-Based MT systems with a constraint-unification mechanism and a feature specification language for the grammar and the lexicon. Recycling non-expert post-editing feedback into MT systems alleviates the need to hire experts to improve and extend existing MT systems. This is a step towards overcoming the bottleneck for efficient use of MT technology, as it represents a large cut in cost. The immense distributive power of such an online approach could also represent considerable time savings.

- A mechanism to evaluate the effect of automatic rule refinements on test data. Automatic evaluation metrics, such as precision, recall, BLEU, METEOR, and NIST, can be used to

discriminate between raw MT output and MT output improved by automatic rule refinements, and they can be used to guide automatic refinements on a regression test set.

- A correction instance management system, able to keep different candidate rule refinements for later confirmation or rejection. Correction Instances are stored in a Collection and can be discarded or saved for later user interaction, if automatic rule refinements degrade system performance on a test set. Both original and refined rules are stored in a rule hierarchy, which allows the ARR to revert the grammar and lexicon to their state before problematic refinements applied.

- Improvement of MT output quality on unseen data, as measured by standard automatic evaluation metrics. Experiments with an English-Spanish Transfer-Based MT system show that an automatically refined MT system statistically significantly outperforms the baseline MT system given a decoder that ranks the *n*-best list produced by the system.

- Confirmation of significant language independence of our automatic rule refinement approach. Initial experiments on a radically different language pair (Mapudungun-Spanish) show that automatic refinements can extend both the lexicon and the grammar to translate new sentences.

7.1 Directions for Future Research

There are several research directions in which the work presented in this thesis can be extended. We conclude by discussing some of these directions for future research.

7.1.1. Learning from Bilingual Post-editing Data

In this dissertation, we focused on rule adaptation as guided by a predetermined set of heuristics. However, when the underlying translation system is not accessible, an interesting research direction is to automatically learn a finite state transducer or a synchronous grammar that, given incorrect (corrupted) translations, produces correct translations. This could be done using Machine Learning techniques to process the data extracted with the Translation Correction Tool or the online Translation Game mentioned in Section 7.1.5, namely negative examples (incorrect MT output) paired up with positive examples (corrected translation).

Researchers have recently shown that a Statistical MT system can be effectively used to translate raw MT output from a Rule-Based system into better MT output (Simard et al., 2007a);

they used post-edited MT output to train their system. Their work shows the promise of this new research area of training computational systems to learn a function from bad MT output into good MT output.

7.1.2. Seeded Version Space

Given our goal of only making safe generalizations of grammar refinements, the Automatic Rule Refiner currently selects the most specific (most conservative) generalization. To go beyond minimal generalization, one would need a mechanism to explore the space of all refinements (hypotheses) consistent with a specific correction.

In Section 4.6.4, we discuss the trade-off between making refinements that are most general, typically at the POS level, and those that are most specific, i.e., at the word level. These are but two of the hypotheses of a potentially much larger version space, also called the seeds of a seeded version space (SVS). The seeded version space described here is inspired by the one proposed by Mitchell (1982), who introduced the version space concept; *seeded* means that the version space is only constructed around the current hypotheses (Carbonell, personal communication, July 2007) and exhibits polynomial complexity instead of being NP-hard, as is the case for full version spaces. In our case, transfer rules proposed by the ARR constitute the most specific seed at the bottom of the hypothesis space, and the rest of transfer rules are organized in a partial ordering, where rules at the top of the version space are strictly more general than rules lower in the SVS.

Figure 7.1 illustrates the seeded version space between an actual refinement to the Mapudungun-Spanish grammar (Figure 6.8), which constitutes the most specific hypothesis of the version space, and the most general hypothesis of this seeded version space, namely one generalizing all the way to the POS-level. Moving upward the SVS always represents a generalization step from the previous hypothesis. Thus, for the example there are several ways to generalize from the most specific seed NP,20 (DET_{def} vs $DET_{non\text{-}def}$, DET_{masc} vs DET_{pl}, etc.). Figure 7.1 illustrates two of the possible generalizations from the most specific hypothesis, i.e. replacing the definite masculine plural determiner “los” with all definite masculine determiners (left generalization) and replacing it with all definite plural determiners (right generalization).

One way to explore these seeded version spaces containing automatically refined transfer rules is using Active learning.

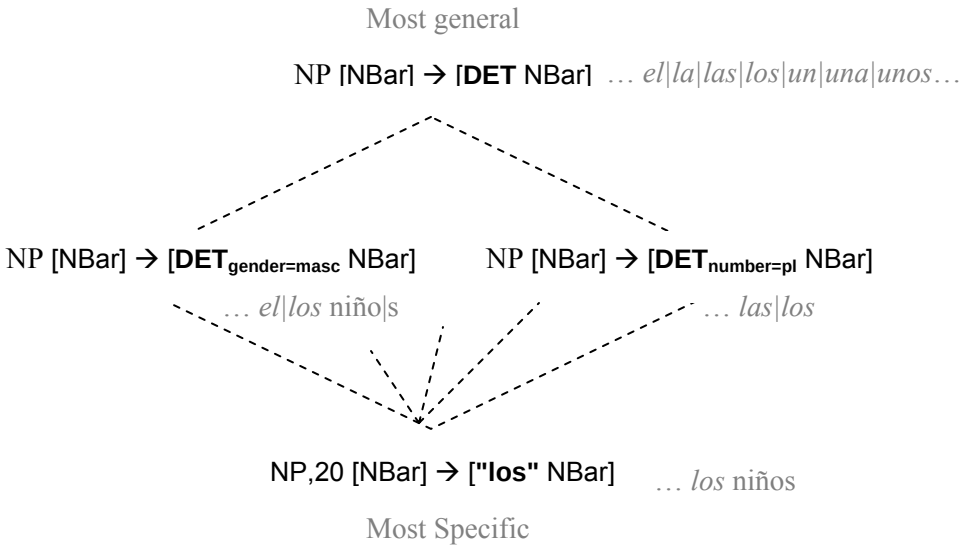


Figure 7.1. Example of Seeded Version Space (SVS), where the seeds are the most specific transfer rule output a refined MT system (Figure 6.8.) and the most general rule given the automatic refinement (+"los").

7.1.3. Active Learning

After initial user interaction, the Automatic Rule Refiner operates off-line and makes conservative generalizations from correction instances. An interactive mode of operation would allow the ARR to prompt users with new sentences to evaluate at run-time, in order to obtain any additional information required to validate the appropriate level of generalization of automatic refinements. Active Learning methods can be used to minimize the number of examples a human annotator must label (Cohn et al., 1994) usually by processing examples in order of usefulness. Lewis and Catlett (1994) used uncertainty as a measure of usefulness. More recently, Callison-Burch (2003) has proposed Active Learning to reduce the cost of creating a corpus of labeled training examples for Statistical MT.

For the task of automatic rule refinement, Active Learning effectively reduces the seeded version space of automatic refinements that needs to be explored by the system (see previous section), for instance by doing binary search over generalization hypotheses. First, given a SL sentence, a TL sentence has to be generated by the current hypothesis (transfer rule) and if the bilingual informant validates the TL sentence as being correct, the search algorithm moves up in the version space. If the TL sentence generated by the transfer rule hypothesized is not correct, the algorithm moves downwards or explores sibling generalizations. Active learning can be used

to both exploit an existing corpus of SL-TL pairs and to make further user interaction more efficient, by asking bilingual informants to evaluate just the example sentences that will be most informative in guiding and validating ARR hypotheses. Generating examples that validate hypothesized rules, however, is a non-trivial task.

TL sentences generated by hypotheses that are one step away from each other in the seeded version space are what we call minimal pairs (Probst et al., 2001). By contrasting minimal pairs via Active Learning, the system can quickly learn in which direction it needs to move in the current SVS. For example, given the following user feedback: "**en tu* → *en ti*" ("*in you*"), the most specific refinement would to be add a constraint that affects the second person singular pronoun (*tu*/*ti*), but it would be interesting to further explore if this correction is required for other personal pronouns as well. In other words, does *en* affect just *tu*, changing it into *ti*? Does it affect all the personal pronouns or a subset of them?[41] In this case, an interactive ARR could present informants with the following minimal pairs to explore the hypothesis space: **"***She believes in me – *Ella cree en yo***",** "*She believes in him – Ella cree en él*", "*She believes in us – Ella cree en nosotros*", "*She believes in you – Ella cree en vosotros*" and "*She believes in them – Ella cree en ellos*". Given the first minimal pair, bilingual informants will change the incorrect MT output "**en yo"* (*in I*) to "*en mi*" (*in me*). This would allow the ARR to validate the hypothesis that the triggering feature is case (δ(tu,ti)={case} and δ(yo,mi)={case}), and that the refinement needs to be at the POS level, since not only the 2nd person singular pronoun is affected.

Minimal pair differences occur at the feature level, which can be calculated with the delta function as described in Section (4.5.2.2). More formally, if EC is the Elicitation corpus, EC_c the part which is correctly translated, then for the translation pair under consideration (x=(s,t)), select $x_c=(s,t_c)$ belonging to EC_c so that they are minimal pairs, namely so that the feature vectors $(\vec{x},\overrightarrow{x_c})$ differ minimally, ideally in just one feature. The current version of the Elicitation Corpus contains feature vectors for all the sentences (Alvarez et al., 2005; Levin et al., 2006). If a minimal pair already exists in the Elicitation Corpus, the ARR can use these feature vectors to select the relevant minimal pair. If no minimal pair exists in the corpus, in addition to an Active Learning problem, this also becomes a language generation problem. The ARR would need to create a feature vector from the lexical entries and grammar rules that translated the originally

[41] The answer is that only the 1st and 2nd person singular pronouns have a different surface form for the oblique case (*mi/ti*), and the rest of pronouns do not change (*en él/ella//nosotros/vosotros/ellos/ellas*).

produced TL sentence, and change one feature at a time and generate the corresponding source language sentence.

In some cases, however, what triggers a specific correction is the syntactic context, which cannot be expressed in terms of minimal pairs strictly speaking, as the relevant TL sentence to allow automatic discovery of the triggering context might have as little as one word in common with the original translation. This is the case of the overgeneralization example given in Section 3.1 (error type 4.2), "*John read the book → *A Juan leyó el libro → Juan leyó el libro*", which corresponds to deleting a word without an alignment. To discover that *a* is only added to NPs in object positions ("*Le doy el libro [a Juan] - I give [Juan] the book*"), but not to subject positions ("*[Juan] lee el libro - [Juan] reads the book*"), the ARR would need to find an example where *Juan* appears in the object position and contains as many words from the original TL sentence as possible. Sentences like "*Ellos invitaron [a Juan]*" ("*They invited [Juan]*") and "*Anna aplaudió [a Juan]*" ("*Anna applauded [Juan]*") might be fine candidates for a linguist to discover that what is causing *a* to appear in front of an NP is its grammatical role. However, such sentences have too few words in common with the original TL sentence, "*[A Juan] leyó el libr*o", for the minimal pair approach to allow the ARR to hypothesize the right cause of the MT error.

In sum, for cases where major structural changes are not the cause of the error, evaluation of minimal error pairs, either generated by transfer rules in a seeded version space using Active Learning techniques or already present in an existing data set, can determine what feature(s) triggered the correction, what is the rule refinement operation suitable to fix a specific error and whether refinements need to happen at the part-of-speech level (most general), at the word level (most specific) or somewhere in-between.

7.1.4. Porting to other MT Systems

The automatic rule refinement approach presented in this thesis is designed for the AVENUE transfer rule formalism (Section 4.2), and therefore the ARR implementation is not formalism-independent. The Rule Refinement Framework, however, can be adapted to other formalisms, and the rule refinement operations and schemata described in Section 4.3 can be generalized to extend other kinds of grammars and lexicons. For example, given a rule formalism based on the Generalized Phrase Structure Grammar (GPSG) (Gazdar et al., 1985), the ARR system could produce a constituency parse tree for the TL-side and convert immediate dominance (ID) rules and linear precedence (LP) statements as well as feature information into Context-Free translation rules, which include dominance and linear precedence information, as well as feature constraints. Alternatively, one could adapt the rule refinement operations to operate at the different levels of

representation postulated by the GPSG formalism (e.g., word order errors would most likely lead to bifurcating an LP statement; agreement constraints would have to be added as feature co-occurrence restrictions, and so on).

Finally, it would be extremely interesting to explore the portability of our Automatic Rule Refinement system to state-of-the-art Rule-based MT systems as well as to syntactic-based Statistical MT systems. Automatically refining very large, complex grammars might prove to have a different set of properties and implications, and undesired rule interactions are sure to emerge (Harvey & Tambe, 1993). Yet research advances in refining such systems is guaranteed to have a major impact on the MT community and in the world of global e-commerce.

7.1.5. Online Translation Game: MT Data annotation

One of the most novel and interesting aspects of this dissertation work is the idea of capitalizing on user time and availability in order to obtain annotated data through the Web, in this case corrected MT output, which can then be used to improve existing systems. Companies and government institutions have typically hired a large number of translators and post-editors to produce high-quality translations at prodigious aggregate costs, producing only limited volumes of translated data. In collaboration with Luis von Ahn, we have designed an online Translation Game that will allow the collection of large amounts of post-edited MT output at a very low cost.

The goal of our game is to harness the large quantities of hours that people spend playing on-line games, and channel it to the worthy cause of producing better quality translations from raw MT output. Estimates run to multi-billions of human-hours spent playing just computer solitaire around the world in one year. What if even a small fraction of this time and energy could be channeled into useful work while providing entertainment? Unlike computer processors, however, humans require an incentive in order to become part of a collective, collaborative computational initiative. We propose a new breed of online games as a means to encourage participation in the process. Similar games to the one we propose to use for MT data annotation are already in place and have been successfully collecting annotated data to improve current search engine performance. An example of such a game is the ESP game, which is a popular online game that collects accurate keyword labels for images on the Web. The ESP Game has collected over 39 million image labels (von Ahn & Dabbish, 2004).

The Translation Game is a two-player *Asymmetric Verification Game*, where players are bilingual to some extent, and where their language skills complement each other in the following way: when correcting MT output from language A into language B, Player 1 has a native or strong knowledge of language B and some knowledge of language A, whereas Player 2's

strongest language is A and weakest language is B. The general strategy of the game is to have Player 1 correct MT output in language B (B'), given A; whereas Player 2 needs to reconstruct A from B', thus validating Player 1's corrections. To make Player 2's task more fun, the number of word slots for A are shown, and she needs to guess the words as if she was playing the popular name of hang man (Gomme, 1973), but in addition B' is given as a hint. During this process, Player 1 can further refine B' and point to specific words in B' to help Player 2.

Given any MT system output, our Translation Game can elicit manual corrections from bilingual internet users around the world. If such a game is made available through a major web portal, this will result in a unique and extremely valuable collection of annotated data, which can be used to improve not only Rule-Based MT systems, but also Statistical MT systems.

Such an online game can also be very appealing to second language learners, who can test their skills by trying to correct MT output, and have the system score their performance and encourage them to try more sentences. The idea of turning work tasks into games is often applied in educational domains for children's and even adults' learning activities. Researchers have noted, as we do, that it is important not to simply slap a game-like interface onto work activities, but to integrate the activities required for learning into the game itself (Laurel, 1986). There must be a tight interconnection between the game interaction and the work to be accomplished. This has been a difficult design challenge in the area of educational games, and should be addressed by any future research in this direction.

This new line of research will benefit both the developers and the consumers of MT technology and, most importantly, it will be a collaborative and multicultural effort of both sectors of society in service of science.

Bibliography

Alfred V. Aho & J. D. Ullman (1969). Translations on a context free grammar. *Proceedings of the first annual ACM symposium on Theory of computing.* Marina del Rey, California, United States: ACM Press. 93 - 112

Allen, J. (1995). *Natural Language Understanding.* The Benjamin Cummings Publishing Company.

Allen, J. (2003). Post-editing. *Computers and Translation: A Translators Guide.* Amsterdam: John Benjamins. 297-317.

Allen, J. & C. Hogan (2000). Toward the Development of a Postediting Module for Raw Machine Translation Output: A Controlled Language Perspective. *Third International Controlled Language Applications Workshop (CLAW-00).* Seattle, WA. 62-71.

Alvarez, A., L. Levin, R. Frederking, J. Good & E. Peterson (2005). Semi-Automated Elicitation Corpus Generation. *Proceedings of MT Summit X.* Phuket, Thailand.

Aranovich, R. (2007). Handling of Morphosyntactic Divergences in the Mapudungun/Spanish AVENUE Transfer Grammar and Lexicon (unpublished). Pittsburgh, PA: University of Pittsburgh.

Arnold, J. (1996). The Inverse System in Mapudungun and Other Languages. *Revista de Lingüística Teórica y Aplicada* 34.

Atserias, J., B. Casas, E. Comelles, M. González, L. Padró & M. Padró (2006). FreeLing 1.3: Syntactic and semantic services in an open-source NLP library. *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC-06).* Genoa, Italy.

Attardi, G. & M. Ciaramita (2007). Tree Revision Learning for Dependency Parsing. *Human Language Technology conference - North American chapter of the Association for Computational Linguistics annual meeting (HLT-NAACL-07).*

Ayan, N. F. (2005). *Combining Linguistic and Machine Learning Techniques for Word Alignment Improvement*. Ph.D. Thesis. University of Maryland, College Park.

Ayan, N. F. & B. J. Dorr (2002). Generating a Parsing Lexicon from Lexical-Conceptual Structure. *Third International Conference on Language Resources and Evaluation (LREC-02)*. Las Palmas, Canary Islands, Spain.

Banerjee, S. & A. Lavie (2005). METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. *Proceedings of Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization at the 43th Annual Meeting of the Association of Computational Linguistics (ACL-05)*. Ann Arbor, MI: ACL.

Barker, K., S. Delisle & S. Szpakowicz (1998). Test-driving TANKA: Evaluating a semi-automatic system of text analysis for knowledge acquisition. *Proceedings of the Twelfth Canadian Conference on Artificial Intelligence (AI-98)*. Vancouver, Canada.

Belz, A. (2002). Learning Grammars for Noun Phrase Extraction by Partition Search. *Third International Conference on Language Resources and Evaluation (LREC-02)*. Las Palmas, Canary Islands, Spain.

Blythe, J. (2005). Task Learning by Instruction in Tailor. *Proceedings of Conference on Intelligent User Interfaces (IUI-05)*. San Diego, CA.

Blythe, J. & Y. Gil (2004). Incremental formalization of document annotations through ontology-based paraphrasing. *Proceedings of the 13th international conference on World Wide Web*. New York, NY, USA: ACM Press. 455 - 461

Blythe, J., J. Kim, S. Ramachandran & Y. Gil (2001). An Integrated Environment for Knowledge Acquisition. *In Proceedings of the 2001 International Conference on Intelligent User Interfaces (IUI-01)*. Santa Fe, New Mexico.

Bredenkamp, A., J. Klein & B. Crysmann (1999). Annotation of Error Types for a German News Corpus. *ATALA sur les Corpus Annotés pour la Syntaxe Treebanks, June 18-19*. Paris, France.

Brewster, C., F. Ciravegna & Y. Wilks (2001). Knowledge Acquisition for Knowledge Management: Position Paper. *Proceedings of the the 17th International Conference on Artificial Intelligence (IJCAI-01), Workshop on Ontology Learning*. Seattle, WA.

Brill, E. (1993). Automatic Grammar Induction and Parsing Free Text: A Transformation-Based Approach. *Proceedings of ACL-93*. 259-265.

Brown, P. F., J. Cocke, S. A. D. Pietra, V. J. D. Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer & P. S. Roossin (1990). A Statistical Approach to Machine Translation. *Computational Linguistics* 6(2): 79-85.

Brown, R. D. (1996). Example-Based Machine Translation in the Pangloss System. *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96).* Copenhagen, Denmark.

Brown, R. D. (1999). Adding Linguistic Knowledge to a Lexical Example-Based Translation System. *In Proceedings of the Eighth International Conference on Theoretical and Methodological Issues in Machine Translation (TMI-99).* Chester, UK. 22-32.

Brown, R. D. (2000). Automated Generalization of Translation Examples. *Proceedings of the Eighteenth International Conference on Computational Linguistics (COLING-2000).* Saarbrücken, Germany. 125-131.

Brown, R. D. (2001). Transfer-Rule Induction for Example-Based Translation. *In Proceedings of the MT Summit VIII Workshop on Example-Based Machine Translation.* Santiago de Compostela, Spain. 1-11.

Callison-Burch, C. (2003). *Active Learning for Statistical Machine Translation*. Ph.D. Thesis Proposal. University of Edinburgh.

Callison-Burch, C., C. Bannard & J. Schroeder (2004). Improving statistical translation through editing. *European Association for Machine Translation (EAMT-04) Workshop.* Malta.

Carl, M. (2000). Combining Invertible Example-Based Machine Translation with Translation Memory Technology. *Envisioning Machine Translation in the Information Future: 4th Conference of the Association for Machine Translation in the Americas (AMTA-2000).* Cuernavaca, Mexico: Springer Berlin, Heidelberg. 301-303.

Carl, M. & S. Hansen (1999). Linking Translation Memories with Example-Based Machine Translation. *MT-Summit VII.*

Catrileo, M. (1988). *Mapudunguyu: Curso de Lengua Mapuche.* Universidad Austral de Chile.

Chklovski, T. (2005). Designing Interfaces for Guided Collection of Knowledge about Everyday Objects from Volunteers. *Proceedings of Conference on Intelligent User Interfaces (IUI-05) (short paper).* San Diego, CA.

Chklovski, T., V. Ratnakar & Y. Gil (2005). User Interfaces with Semi-Formal Representations: a Study of Designing Argumentation Structures. *In Proceedings of Conference on Intelligent User Interfaces (IUI-05).* San Diego, CA.

Clark, S., J. R. Curran & M. Osborne (2003). Bootstrapping POS taggers using Unlabelled Data. *Conference on Computational Natural Language Learning (CoNLL-03)*

Cohn, D., L. Atlas & R. Ladner (1994). Improving generalization with active learning. *Machine Learning* 15(2): 201-221.

Comrie, B. (1989). *Language universals and linguistic typology.* Chicago: University of Chicago.

Correa, N. (2003). A Fine-grained Evaluation Framework for Machine Translation System Development. *MT Summit IX.* New Orleans, Louisiana, US.

Corston-Oliver, S. & M. Gamon (2003). Combining decision trees and transformation-based learning to correct transferred linguistic representations. *Proceedings of MT Summit*

Crysmann, B. (1997). Fehlerannotation. *Technical report.* DFKI GmbH.

Delisle, S., K. Barker, J.-F. Delannoy, S. Matwin & S. Szpakowicz (1994). From Text to Horn Clauses: Combining Linguistic Analysis and Machine Learning. *Canadian AI-94.*

Delisle, S. & S. Szpakowicz (1997). Extraction of Predicate-Argument Structures from Texts. *Recent Advances in Natural Language Processing (RANLP).*

Dillinger, M. & A. Lommel (2004). Implementing machine translation (LISA Best Practice Guides).

Doddington, G. (2002). Automatic Evaluation of MT Quality using N-gram Co-occurrence Statistics. *Proceedings of Human Language Technology Conference.* San Diego, CA. 128-132.

Dugast, L., J. Senellart & P. Koehn (2007). Statistical Post-Edition on SYSTRAN Rule-Based Translation System. *In Proceedings of the Second Workshop On Statistical Machine Translation (WMT), at the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07).* Prague, Czech Republic.

Dupplaw, D. P., D. Brunson, A.-J. E. Vine, C. P. Please, S. M. Lewis, A. M. Dean, A. J. Keane & M. J. Tindall (2004). A Web-Based Knowledge Elicitation System (GISEL) for Planning and Assessing Group Screening Experiments for Product Development. *Journal of Computing and Information Science in Engineering* 4(3): 218-225.

Elliott, D., A. Hartley & E. Atwell (2004). A Fluency Error Categorization Scheme to Guide Automated Machine Translation Evaluation. *AMTA: Machine Translation: From Real Users to Research.* Springer Berlin / Heidelberg. 64-73.

Elming, J. (2006). Transformation-based correction of rule-based MT. *11th Annual Conference of the European Association for Machine Translation (EAMT-06).*

EUROPA (27/04/2006). Translation in the Commission: where do we stand two years after Enlargement? *EUROPA Rapid Press Releases.* *http://europa.eu/rapid/pressReleasesAction.do?reference=MEMO/06/173&format=HTML&aged=0&language=EN&guiLanguage=en (07/10/07).*

Flanagan, M. (1994). Error Classification for MT Evaluation. *Proceedings of the Association of Machine Translation of the Americas (AMTA-94).* 65-72.

Font Llitjós, A. (2006). Can the Internet help improve Machine Translation. *Human Language Technology conference - North American chapter of the Association for Computational Linguistics annual meeting (HLT-NAACL-06), Doctoral Consortium.* New York, USA.

Font Llitjós, A., R. Aranovich & L. Levin (2005a). Building Machine translation systems for indigenous languages. *Second Conference on the Indigenous Languages of Latin America (CILLA II).* Texas, USA.

Font Llitjós, A. & J. Carbonell (2004). The Translation Correction Tool: English-Spanish user studies. *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC-04).*

Font Llitjós, A., J. Carbonell & A. Lavie (2005b). A Framework for Interactive and Automatic Refinement of Transfer-based Machine Translation. *10th Annual Conference of the European Association for Machine Translation (EAMT-05).*

Font Llitjós, A., J. Carbonell & A. Lavie (2007). Improving a Transfer-Based MT System with Automatic Refinements. *To appear in MT Summit XI.*

Font Llitjós, A. & W. A. Ridmann (2007). The Inner Works of an Automatic Rule Refiner for Machine Translation. *METIS-II Workshop.* Leuven, Belgium.

Font Llitjós, A. & S. Vogel (2007). A Walk on the Other Side: Adding Statistical Components to a Transfer-Based Translation System. . *HLT-NAAC-07, Syntax and Structure in Statistical Translation (SSST)Workshop.* Rochester, NY.

Gavaldà, M. (2000). *Growing Semantic Grammars*. Ph.D. Thesis. Carnegie Mellon University.

Gazdar, G., E. Klein, G. Pullum & I. Sag (1985). *Generalized Phrase Structure Grammar.* London, Oxford, UK and Cambridge, MA: Blackwell and Harvard University Press.

George, C. & N. Japkowicz (2005). Automatic correction of French to English relative pronoun translations using natural language processing and machine learning techniques. *Computational Linguistics In the North East.* Ottawa, Canada.

Gil, Y. & T. Chklovski (2007). Enhancing Interaction with To-Do Lists Using Artificial Assistants. *In Proceedings of AAAI Spring Symposium on Interaction Challenges for Intelligent Assistants.* Stanford, CA.

Gomme, A. B. (1973). *Traditional Games of England, Scotland and Ireland.* Dover Publications Inc. .

Gosset, W. S. S. (1908). The Probable Error of a Mean. *Biometrika*.

Greenberg, J. H. (1987). *Language in the Americas.* Stanford: Stanford University Press.

Hamp, E. P. (1971). On Mayan-Araucanian Comparative Phonology. *International Journal of American Linguistics (IJAL)* 37(3): 156-159.

Harvey, W. A. J. & M. Tambe (1993). Experiments in Knowledge Refinement for Large Rule-Based Systems. Pittsburgh, PA, USA: Computer Science Department, Carnegie Mellon University.

Hodász, G., T. Grőbler & B. Kis (2004). Translation Memory as a Robust Example-based Translation System. *the Ninth European Association for Machine Translation Workshop(EAMT-04).* Malta.

Hutchins, J. (2001). Towards a new vision for MT (Introductory speech). *MT Summit VIII*.

Hutchins, J. W. & H. L. Somers (1992). *An Introduction to Machine Translation.* London: Academic Press.

Imamura, K., E. Sumita & Y. Matsumoto (2003). Feedback cleaning of Machine Translation Rules Using Automatic Evaluation. *41st Annual Meeting of the Association for Computational Lingusitics (ACL-03).* 447-454.

INDEC (2005). Encuesta Complementaria de Pueblos Indígenas (ECPI), 2004-2005 - Primeros resultados provisionales. Buenos Aires, Argentina: Instituto Nacional de Estadística y Censos.

ISLE (2000). The International Standards for Language Engineering(ISLE) Classification of Machine Translation Evaluations. Marina del Rey, CA: USC Information Sciences Institute.

ISLE (2004). FEMTI: A Framework for the Evaluation of Machine Translation in ISLE.

Jiménez, M. (2002). Using Decision Trees To Predict Human Nouns In Spanish Parsed Text. *Third International Conference on Language Resources and Evaluation (LREC-02).* Las Palmas, Canary Islands, Spain.

Key, M.-R. (1978). Lingüística comparativa araucana. *VICUS. Amsterdam* 2: 45-55.

Kim, J. (2005). From Reflection to Interaction: Use of Memory in InteractiveKnowledge Acquisition. *Proceedings of the XXVII Annual Conference of the Cognitive Science Society (CogSci-2005).* Stresa, Italy.

Kim, J. & Y. Gil (2003). Proactive Acquisition from Tutoring and Learning Principles. *In Proceedings of the AI in Education (AIEd-2003).* Sydney, Australia.

Knight, K. & I. Chander (1994). Automated postediting of documents. *Proceedings of the twelfth national conference on Artificial intelligence (AI-94).* Seattle, WA.

Koehn, P. (2001 (release 1), 2003 (release 2)). Europarl: A Multilingual Corpus for Evaluation of Machine Translation.

Koehn, P. (2004). Pharaoh: a beam search decoder for statistical machine translation. *In 6th Conference of the Association for Machine Translation in the Americas (AMTA-04).* Springer.

Koehn, P. (2005). Europarl: A Parallel Corpus for Statistical Machine Translation. *MT Summit X.* Phuket, Thailand.

Koehn, P., H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin & E. Herbst (2007). Moses: Open Source Toolkit for Statistical Machine Translation. *Annual Meeting of the Association for Computational Linguistics (ACL-07), demonstration session.* Prague, Czech Republic.

Krings, H. (2001). *Repairing Texts. Empirical Investigations of Machine Translation Post-Editing Processes.* Kent, OH: Kent State University Press.

Laurel, B. (1986). *Towards the Design of a Computer-Based Interactive Fantasy System*. The Ohio State University.

Lavie, A. & A. Agarwal (2007). METEOR: An Automatic Metric for MT Evaluation with High Levels of Correlation with Human Judgments. *Proceedings of Workshop on Statistical Machine Translation (WMT) at the 45th Annual Meeting of the Association of Computational Linguistics (ACL-07).* Prague, Czech Republic.

Lavie, A., S. Vogel, E. Peterson, K. Probst, A. Font-Llitjós, R. Reynolds, J. Carbonell & R. Cohen (2003). Experiments with a Hindi-to-English Transfer-based MT System under a Miserly Data Scenario. *ACM Transactions on Asian Language Information Processing (TALIP)* 2(2).

Lehman, J. F. (1989). *Adaptive Parsing: Self-Extending Natural Language Interfaces*. Ph.D. Thesis. Carnegie Mellon University.

Levin, L., A. Alvarez, J. Good & R. Frederking (2006). Automatic Learning of Grammatical Encoding. In Grimshaw, J., Maling, J., Manning, C., Simpson, J. & Zaenen, A. (Eds.), *To appear in Architectures, Rules, and Preferences: A Festschrift for Joan Bresnan.* CSLI Publications. In Press.

Lewis, D. & J. Catlett (1994). Heterogeneous uncertainty sampling for supervised learning. *Proceedings of the Eleventh International Conference on Machine Learning (ICML-94).* 148-156.

Lin, Y.-C., T.-H. Chiang & K.-Y. Su (1994). Automatic Model Refinement - with an application to tagging. *15th International Conference on Computational Linguistics (COLING-94).* 148-152.

Lingtech (1996). Post-editor Survey 1: Analyse og sammenfatning af sprogrevisorsvar. Lingtechs spørgeskemaundersøgelse.

Lingtech (1997). Post-editor Survey 2: Evaluering af oversættelser af tekstkorpusser K; edb og kontormaskiner. Baseret på input fra Lingtechs sprogrevisorer og datalingvister.

Loffler-Laurian, A.-M. (1996). *La traduccion automatique.* Paris: Presses Universitaires du Septentrion.

Lommel, A. (2007). Chinese Translation Errors Impact Users. *LISA The GLobalization Insider. http://www.lisa.org/globalizationinsider/2007/03/chinese_transla.html (07/01/07).*

Lopez, A., M. Nossal, R. Hwa & P. Resnik (2002). Word-Level Alignment For Multilingual Resource Acquisition. *Third International Conference on Language Resources and Evaluation (LREC-02).* Las Palmas, Canary Islands, Spain.

Maegaard, B. & V. Hansen (1995). PaTrans: Machine Translation of Patent Texts, from Research to Practical Applications. *In Engineering Proceedings of the Second Language Convention* 1-8.

Manning, C. D. & H. Schütze (1999). *Foundations of Statistical Natural Language Processing.* Cambridge, MA and London, England: The MIT Press.

Marrafa, P. & A. Ribeiro (2001). Quantitative Evaluation of Machine Translation Systems: Sentence Level. *MT Summit VIII.*

McClure, S. & M. Flanagan (2003). Overcoming the MT Quality Impasse. *SYSTRAN Documentation. SYSTRAN and the Reinvention of MT.*

McCord, M. (1989). Design of LMT: A Prolog-Based Machine Translation System. *Computational Linguistics* 15(1).

McCord, M. & A. Bernth (1998). Design of LMT: The LMT Transformational System. *Proceedings of Association for Machine Translation in the Americas (AMTA).*

McShane, M., S. Nirenburg, J. Cowie & R. Zacharski (2002). Embedding Knowledge Elicitation and MT Systems within a Single Architecture *Machine Translation* 17(4): 271-305.

Mellebeek, B., K. Owczarzak, J. V. Genabith & AndyWay (2006). Multi-Engine Machine Translation by Recursive Sentence Decomposition. *Association of Machine Translation in the Americas (AMTA-06).* Boston, MA.

Menezes, A. & S. D. Richardson (2001). A best-first alignment algorithm for automatic extraction of transfer mappings from bilingual corpora. *Workshop on Example-Based Machine Translation, in MT Summit VIII.* 35-42.

Meng, H. H. & K. C. Siu (2002). Semiautomatic Acquisition of Semantic Structures for Understanding Domain-Specific Natural Language Queries. *IEEE Transactions on Knowledge and Data Engineering.*

Mitchell, T. (1982). Generalization as search. *Artificial Intelligence* 18: 202-226.

Nakamura, S., K. Markov, H. Nakaiwa, G. Kikui, H. Kawai, T. Jitsuhiro, J. Zhang, H. Yamamoto, a. E. Sumita & S. Yamamoto (2006). The ATR multilingual speech-to-speech translation system. *IEEE Transactions on Audio, Speech, and Language Processing 14 (2).* 365–376.

Naruedomkul, K. (2001). *Generate and repair machine translation*. Ph.D. Thesis. Mahidol University, Thailand.

Niessen, S., F. J. Och, G. Leusch & H. Ney (2000). An Evaluation Tool for Machine Translation: Fast Evaluation for MT Research. *LREC.* 39-45.

Nirenburg, S. & V. Raskin (1998). Universal Grammar and Lexis for Quick Ramp-Up of MT. *COLING-ACL-98.*

Nishida, F. & S. Takamatsu (1990). Automated procedures for the improvement of a machine translation system by feedback from postediting. *Machine Translation* Volume 5(Number 3): 223-246.

Och, F. J. (2003). Minimum error rate training in statistical machine translation. *In Proceedings of the 41st Annual Meeting of the Association of Computational Linguistics (ACL-03).* Sapporo, Japan.

Och, F. J. & H. Ney (2003). A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics* 29(1): 19-51.

Olive, J. (2005). Global Autonomous Language Exploitation (GALE). DARPA/IPTO Proposer Information Pamphlet.

Papineni, K., S. Roukos, T. Ward & W.-J. Zhu (2001). BLEU: A Method for Automatic Evaluation of Machine Translation. *IBM Research Report.* IBM.

Peterson, E. (2002). *Adapting a Transfer Engine for Rapid Machine Translation Development*. Master's Thesis. Carnegie Mellon University.

Phaholphinyo, S., T. Modhiran, N. Kritsuthikul & T. Supnithi (2005). A Practical of Memory-based Approach for Improving Accuracy of MT. *MT Summit X.*

Planas, E. (2000). Extending Translation Memories. *The Fifth European Association for Machine Translation Workshop (EAMT-2000).*

Povlsen, C., N. Underwood, B. Music & A. Neville (1998). Evaluating Text-type Suitability for Machine Translation a Case Study on an English-Danish MT System. *In Proceedings of the First International Conference on Language Resources and Evaluation (LREC-98).* Granada, Spain.

Probst, K. (2005). *Automatically Induced Syntactic Transfer Rules for Machine Translation under a Very Limited Data Scenario*. Ph.D. Thesis. Carnegie Mellon University.

Probst, K., R. Brown, J. Carbonell, A. Lavie, L. Levin & E. Peterson (2001). Design and Implementation of Controlled Elicitation for Machine Translation of Low-density Languages. *Proceedings of the MT2010 workshop at MT Summit 2001.*

SAE (2001). SAE J2450: Translation Quality Metric. In Engineers, S. o. A. (Ed.), Warrendale, USA.

Salas, A. (1992). Lingüística Mapuche. Guía Bibliográfica. *Revista Andina* Año 10 (no. 2.).

Schäler, R. (2001). Beyond Translation Memories. *Workshop on Example-Based Machine Translation, MT Summit VIII.* Santiago de Compostela, Spain.

SDL, I. & TRADOS (2005). SDL Translation Memory - Solution Brief.

Secară, A. (2005). Translation Evaluation- a State of the Art Survey. *eCoLoRe/MeLLANGE Workshop Proceedings.* 39-44.

Seneff, S., C. Wang & J. Lee (2006). Combining Linguistic and Statistical Methods for Bi-directional English Chinese Translation in the Flight Domain. *Association of Machine Translation in the Americas (AMTA-06).* Boston, Massachusetts, United States.

Shäfer, F. (2003). MT post-editing: How to shed light on the "unknown task". Experiences made at SAP. *8th International Workshop of the European Association for Machine Translation (EAMT-03).*

Shimizu, T., Y. Ashikari, E. Sumita, H. Kashioka & S. Nakamura (2006). Development of client-server speech translation system on a multi-lingual speech communication platform. *IWSLT.* Kyoto, Japan.

Simard, M., C. Goutte & P. Isabelle (2007a). Statistical Phrase-based Post-editing. *Human Language Technology conference - North American chapter of the Association for Computational Linguistics annual meeting (HLT-NAACL-07).*

Simard, M., J. Senellart, P. Isabelle, R. Kuhn, J. Stephan & N. Ueffing (2007b). Rule-based Translation With Statistical Phrase-based Post-editing. *In Proceedings of the Second Workshop On Statistical Machine Translation (WMT), at the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07).* Prague, Czech Republic.

Smeets, I. (1989). *A Mapuche Grammar*. Leiden University.

Smith, N. & J. Eisner (2004). Annealing techniques for unsupervised statistical language learning. *In Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL-04).* Barcelona, Spain.

Snover, M., B. Dorr, R. Schwartz, L. Micciulla & J. Makhoul (2006). A Study of Translation Edit Rate with Targeted Human Annotation. *7th biennial conference of the Association for Machine Translation in the Americas (AMTA-06).* Boston, USA. 223-231.

Somers, H. (2001). EBMT seen as Case-based Reasoning *Workshop on Example-Based Machine Translation, MT Summit VIII.* Santiago de Compostela, Spain.

SRI (2006). The SRI Language Modeling Toolkit (SRILM). http://www.speech.sri.com/projects/srilm (07/01/07).

Su, K.-Y., J.-S. Chang & Y.-L. Hsu (1995). A corpus-based statistics-oriented two-way design for parameterized MT systems: Rationale, Architecture and Training issues. *6th Theoretical and Methodological Issues in Machine Translation (TMI-95).* 334-353.

Takezawa, T., E. Sumita, F. Sugaya, H. Yamamoto & S. Yamamoto (2002). Toward a Broad-Coverage Bilingual Corpus for Speech Translation of Travel Conversations in the Real World. *In Proceedings of 3rd International Conference on Language Resources and Evaluation (LREC-07).* Las Palmas, Spain.

Tomita, M. (1990). The Generalized LR Parser/Compiler V8-4: A Software Package for Practical NL Projects. *Proceedings of the 13th Conference on Computational linguistics (COLING-90).* Helsinki, Finland: Association for Computational Linguistics. 59 - 63

Tomita, M. & K. Knight (1987). Pseudo-Unification and Full-Unification. *LTI Technical Report.* Pittsburgh, PA, USA: Language Technologies Institute. Carnegie Mellon University.

Turian, J., B. Wellington & I. D. Melamed (2006). Scalable Discriminative Learning for Natural Language Parsing and Translation. *20th Annual Conference on Neural Information Processing Systems (NIPS-06).* Vancouver, Canada.

van Riisborgen, C. J. (1979). *Information retrieval* London: Butterworths.

Veale, T. & A. Way (1997). Gaijin: A Bootstrapping Approach to Example-Based Machine Translation. *Recent Advances in Natural Language International Conference.* 239-244.

Venugopal, A. (2005). MER Training Toolkit. http://www.cs.cmu.edu/~ashishv/mer.html (01/13/07).

Vilar, D., E. Matusov, S. Hasan, R. Zens & H. Ney (2005). Statistical Machine Translation of European Parliamentary Speeches *In Proceedings of MT Summit X.* Phuket, Thailand: Asia-Pacific Association for Machine Translation (AAMT-05). 259–266.

Vilar, D., J. Xu, L. F. D'Haro & H. Ney (2006). Error Analysis of Statistical Machine Translation Output. *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC-06).*

Vogel, S. (2005). PESA: Phrase Pair Extraction as Sentence Splitting. *MT Summit X* Phuket,Thailand.

von Ahn, L. & L. Dabbish (2004). Labeling images with a computer game. *Proceedings of the International Conference on Human Factors in Computing Systems (CHI-04).* Vienna, Austria: ACM Press. 319 - 326.

White, J. S., T. O'Connell & F. O'Mara (1994). The ARPA MT Evaluation Methodologies: Evaluation, Lessons, and Future Approaches. *Proceedings of 6th conference of the Association for Machine Translation in the Americas(AMTA-94).* 193-205.

Xi, C. & R. Hwa (2005). A Backoff Model for Bootstrapping Resources for Non-English Languages. *Conference on Empirical Methods in Natural Language Processing (EMNLP-05).*

Yamada, S., H. Nakaiwa, K. Ogura & S. Ikehara (1995). A Method of Automatically Adapting a MT System to Different Domains. *Sixth International Conference on Theoretical and Methodological Issues in Machine Translation (TMI-95).* 303-310.

Zhang, Y. & S. Vogel (2004). Measuring Confidence Intervals for the Machine Translation Evaluation Metrics. *Proceedings of the 10th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI-04).* Baltimore, MD.

Zhang, Y. & S. Vogel (2006). Suffix Array and its Ap-plications in Empirical Natural Language Processing. *LTI Technical Report.* Pittsburgh PA: Language Technologies Institute, Carnegie Mellon University.

Zhu, J. & H. Wang (2006). The effect of translation quality in MT-based cross-language information retrieval. *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL (CoLing-ACL-06).* 593 - 600.

Zúñiga, F. (2000). *Mapudungun.* Munich: Lincom Europa.

Appendix A: AVENUE MT System

Architecture

Figure A.1 shows a simplified overview of the AVENUE MT system. It includes its five main components: (1) Elicitation, (2) Morphology learning, (3) Rule learning, (4) Run-time system and (5) Automatic Rule Refinement. The last component is the focus of this dissertation.

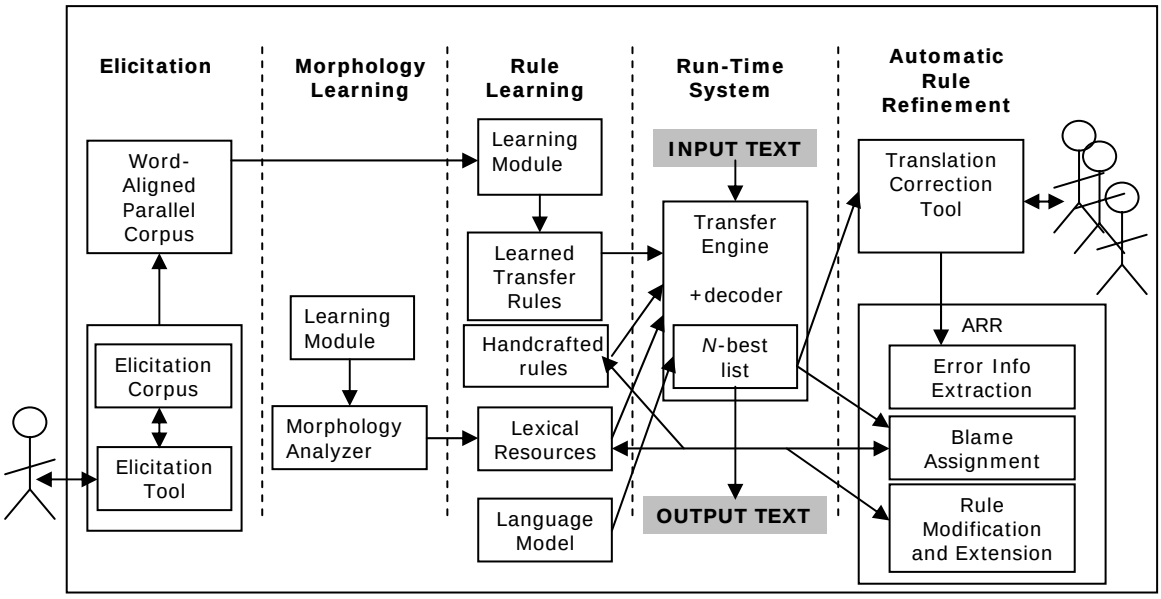


Figure A.1. AVENUE MT System Architecture Diagram.

Transfer (Xfer) Engine

The Transfer Engine, or Xfer engine for short, combines the translation grammar and lexicon in order to produce all the possible translations of a source language sentence into a target language. The AVENUE transfer rule formalism is described in detail in Section 4.3. The output of the Xfer

system is a list of translation alternatives (*n*-best list). The alternatives arise from syntactic ambiguity, lexical ambiguity, multiple synonymous choices for lexical items in the dictionary, and multiple competing hypotheses from the grammar.

The Xfer engine incorporates the three main processes involved in transfer-based MT: parsing of the SL input, transfer of the parsed constituents of the SL to their corresponding structured constituents on the TL side, and generation of the TL output. All three of these processes are performed based on the translation grammar. In the first stage, parsing is performed based solely on the source side of the translation rules. The implemented parsing algorithm is for the most part a standard bottom-up Chart Parser, such as described in Allen (1995). A chart is populated with all constituent structures that were created in the course of parsing the SL input with the source-side portion of the transfer grammar. Transfer and generation are performed in an integrated second stage. A dual TL chart is constructed by applying transfer and generation operations on each and every constituent entry in the SL parse chart. The transfer rules associated with each entry in the SL chart are used in order to determine the corresponding constituent structure on the TL side. At the word level, lexical transfer rules are accessed in order to seed the individual lexical choices for the TL word-level entries in the TL chart. Finally, the set of generated TL output strings that corresponds to the collection of all TL chart entries is collected into a TL lattice, which constitutes the final list of translation candidates, and which can be passed on for decoding (choosing the correct path through the lattice of translation alternatives).

A more detailed description of the Xfer engine can be found in Lavie et al. (2003) and Peterson (2002). The Xfer version used throughout our experiments is version 2, with the exception of the last experiment in Chapter 5 with the automatically augmented lexicon (Section 5.4.3.5). For this experiment, we used version 3 for speed issues. Since version 3 is a different implementation of the core parsing algorithm, a few grammar constraints needed to be converted in order for the new version of the Xfer engine to correctly apply them.

Decoder

The Xfer engine generates multiple translations. These result from multiple translations for source words, as encoded in the lexicon, but also from different parses or partial parses. The traditional approach to selecting the best translation for any given source language sentence was to generate all the possible translations and either just output the first one, or have a human with knowledge of the two languages select the best one from the list of all alternatives.

Nowadays, the existence of statistical components has largely simplified this selection task and, for systems with a very large number of alternatives, such components have also allowed for higher translation accuracy. The job of the decoder is to assign a quality score to all the translation alternatives. Translations are ranked according to their overall scores, i.e. a weighted linear combination of the individual scores. Based on these scores, the 1-best translation is output by the system. The following sections describe components that contribute to the overall decoder score.

Version 2 of the Xfer engine has a decoder integrated into it, which in addition to aiding final ranking of the *n*-best list, it also intervenes during pruning to help make decisions that are supported by the language model.

Fragmentation penalty

In Version 1 of the Xfer engine, the only score used to rank translation alternatives was a fragmentation penalty. This measures if a completely spanning parse could be found or if the translation is glued together from partial parses. The more partial parses necessary to span the entire sentence the less likely is it that we got a good translation. Also, shorter derivations are typically preferred over longer ones. The fragmentation penalty reflects these intuitions, since it is essentially the number of different chunks (rules or lexical entries not embedded in any rule) that span the whole translation.

N-Gram Language Model

The fragmentation feature is rather weak, it does not distinguish between words which are more likely to be seen in the target language and words which are less likely to be used. To generate sentences which are not only grammatically correct, but also use words and word sequences that are more natural and more common, an *n*-gram language model can be used. Instead of just applying it to re-rank the translations, the language model (LM) has been integrated into the decoder. This has the advantage that in the case of pruning the language model score can be used to avoid pruning good hypotheses.

For our BTEC experiments, we built a suffix array language model using the SALM toolkit (Zhang & Vogel, 2006). The language model was built with the 123,416 Spanish sentences from the BTEC training data (Table 5.1).

With or without a decoder, refinements yielding higher recall will increase final translation accuracy. When the decoder selects the *best* translation alternative for each source language sentence, we expect refinements that tighten the grammar (yielding higher precision) to

also have a positive impact on final translation accuracy. The reason is that those refinements reduce grammar ambiguity and thus reduce the size of the final list of candidate translations, eliminating incorrect translation from it. This aids the decoder's task to push better translations to higher ranks.

Length Model

To adjust for the length of the generated translations the number of words is added as a very simple feature. Each word is therefore generating a constant cost or, with a negative scaling factor, a constant bonus. The effect of this feature is to balance globally the length of the translations.

Appendix B: MER Training

Translations are ranked by their total cost by the decoder (Appendix A), which is a weighted linear combination of the individual costs. When adding more features to the translation system a careful balancing of the individual contributions can make a significant difference. However, with each feature added manually tuning the system becomes less and less practical. This means that automatic optimization becomes necessary.

Different optimization techniques are available, like the Simplex algorithm or the special Minimum Error Training as described in (Och, 2003). In this Minimum Error Rate (MER) training the *n*-best list generated by the translation system is used to find feature weights, thereby re-ranking the *n*-best list, which will improve the match between the 1-best translation and the set of reference translations. Decoder parameter optimization is a hill-climbing process and can use any metric as objective function. Typically, systems are tuned towards high BLEU or high NIST scores, more recently also towards METEOR or TER (Snover et al., 2006).

We used a MER training module (Venugopal, 2005), originally developed for a SMT system, to run MER training on the *n*-best lists generated by the Xfer system. The implementation allows for optimization towards different MT metrics, especially to optimize towards BLEU and the NIST mteval metric. In our experiments, we choose to optimize towards BLEU. Since we did not have any held out test data for parameter optimization, we used the BTEC test set described in Section 5.1.3.

N-Best List Rescoring

The information used in the Xfer translation system to rank alternative translations is limited. Essentially, there are three different components that can be used by the decoder, the *n*-gram LM, which is the most important component, a simple sentence length model, and the fragmentation score, which measures if a completely spanning parse could be found or if the translation is glued

together from partial parses (Appendix A). Given an *n*-best list of translations for each source sentence, we can apply additional models to re-rank this *n*-best list, hopefully pushing more good translations into the first rank. We studied the effect of adding different features to the *n*-best lists: rule probability features and language model rule (type) features.

Rule Probabilities

The Xfer MT system can display the parse tree that generated the translation. This allows rescoring the translations with rule probabilities. However, there is no annotated corpus from which the rule probabilities could be estimated. As an approximation to such a training corpus, we decided to run the Xfer system over the BTEC training data and to generate n-best lists with translations and parse trees. Overall, about 6 million parse trees where generated. Using this data to estimate rule probabilities is definitely not ideal, as the translations of the training data are far from perfect, especially as not all the vocabulary has so far been added to the Xfer lexicon. By averaging over all the *n*-best translations a reasonable smoothing is to be expected.

We used this information in three ways. We estimated conditional probabilities for any rule *r* given its rule type *R*, i.e. the distribution over different VP rules or NP rules. For each derivation *D* the overall probability was then calculated as:

$$P(D) = \prod p(r \mid R)$$

As an alternative, we also built *n*-gram language models, one on the rule level and on the rule type level:

$$P(D) = \prod p(r \mid r_{-n} ... r_{-1})$$

$$P(D) = \prod p(R \mid R_{-n} ... R_{-1})$$

Overall, 1,685 different rules and 19 rule types were seen in the training data. For the two Language Models, we again used the suffix array LM to allow for arbitrary long histories (Zhang & Vogel, 2006).

In Table B.1 we can see the effect of adding these LMs as additional features to the refined system with initial decoder weights (Table 5.11) and running the MER training.

	BLEU
Refined System (initial decoder weights)	0.3513
+ Conditional Probabilities	0.3728
+ Rule LM	0.3717
+ Rule Type LM	0.3736

Table B.1. BLEU scores when rescoring the *n*-best list with different rule probability features (as well as the *n*-gram LM).

MER Training Results

In Table B.2, we summarize some of the results from different *n*-best list rescoring experiments. Using only the Xfer engine, without language model, gives a very low score, as the selection is based only on the fragmentation count.

	Rule-Based System + Statistical Components	1-best
Rule Based	Xfer	0.2559
+ Decoder	Xfer + LM + Frag	0.3513
Optimizing weights with MER training	Rule Type LM	0.3180
	Conditional Rule Probabilities (Cond. Rule Prob.)	0.2593
	LM + Rule Type LM	0.3736
	LM + Frag/Len + Rule Type LM	0.3737
	LM + Rule Type LM + Rule LM	0.3744
	LM + Frag + Rule Type LM + Cond. Rule Prob.	0.3743
	LM + Len + Rule Type LM + Cond. Rule Prob.	0.3745
	LM + Rule Type LM + Rule LM + Cond. Rule Prob.	0.3741
	LM + Frag + Len + Rule Type LM + Cond. Rule Prob.	**0.3746**
	LM + Frag + Len + Rule Type + Rule LM + Rule Prob.	0.3741

Table B.2. BLEU scores for the Refined MT System as the weights for the different statistical components are optimized with MER Training.

Adding the *n*-gram language model gives a huge improvement. Adding additional features leads to more then 2 BLEU points improvement. The combination of components that gets a higher BLEU score (second to last row in Table B.2, in bold) suggests that, in addition to the components currently used by the Xfer system decoder, the information added by the Rule Type LM and the Conditional Rule Probabilities can be effectively used to improve the re-ranking of *n*-best lists. However, there is not much difference when using different feature combinations. It seems that the rather small size of the *n*-best list (n=100) is a limiting factor.

For similar results adding lexical probabilities, see Font Llitjós and Vogel (2007).

When setting the optimal weights in the Xfer engine for the basic n-gram LM and Fragmentation penalty scores (second row in Table B.2), both the Baseline and the Refined

system get higher scores, not only according to BLEU, which was used as the objective function, but also to the METEOR and NIST automatic evaluation metrics (Table B.3).

System	METEOR	BLEU	NIST
Baseline	0.6184	0.3609	6.68
Refined	0.6231	0.3780	6.79

Table B.3. Automatic metric scores for 1-best decoder hypothesis, after LM and Fragmentation weights have been optimized.

Note that the BLEU score achieved by the Refined system is actually higher than the one obtained during MER for these same components. We suspect this is mostly due to the fact that the Xfer decoder divides the basic *n*-gram LM score by the number of target words before adding it to the overall score.

Table 5.14 shows a few examples from the BTEC corpus with 1best translations output by the Refined MT system before (No Optimization) and after (With Optimization) MER training, given LM and Fragmentation penalty scores. From these examples, it can be observed that re-ranking improves after optimizing the LM and Fragmentation weights.

Discussion

Adding a statistical language model is crucial in selecting good translations from the *n*-best lists generated by the Xfer engine. To facilitate the experiments with the Xfer system, especially when adding more and more features, a Minimum Error Rate training component becomes relevant. We believe that this will definitely boost the development of the Xfer engine.

In our experiments, finding optimal weights of existing decoder components has enabled us to obtain statistically significant results. Having shown that the results reported in Chapter 7 are statistically significant allows us to make a much stronger claim about the superiority of the Refined MT system over the Baseline system.

Using rule probabilities has shown to be a promising extension to the current Xfer system. While this information would ideally be used in the parsing and transfer steps of the translation system, our initial experiments were targeted at using this in an *n*-best list rescoring setup. As the rule probabilities were estimated from noisy training data these models are far from optimal. These models could be clearly improved by selecting the oracle best translations from the *n*-best list generated on the training data. This will reduce the noise in the training stage. Ultimately, the rule probabilities should be applied not as an *n*-best list rescoring step, but directly in the decoder.

Appendix C: Paired Two-Tailed t-Test Results

These are the paired two-tailed t-test results on the BTEC blind test set (Section 5.4.3):

ORACLE

t-Test: Paired Two Sample for Means

Meteor (before, after)	0.6863	0.6954
	Variable 1	*Variable 2*
Mean	0.663478	0.674276
Variance	0.060179	0.059502
Observations	506	506
Pearson Correlation	0.977132	
Hypothesized Mean Difference	0	
df	505	
t Stat	-4.64118	
P(T<=t) one-tail	2.21E-06	
t Critical one-tail	1.647877	
P(T<=t) two-tail	**4.42E-06**	
t Critical two-tail	1.964673	

DECODER (initial weights)

t-Test: Paired Two Sample for Means

Meteor: (before, after)	0.617605	0.622235
	Variable 1	*Variable 2*
Mean	0.58664	0.588606
Variance	0.065972	0.068138
Observations	506	506
Pearson Correlation	0.974504	
Hypothesized Mean Difference	0	
df	505	
t Stat	-0.75463	
P(T<=t) one-tail	0.225412	
t Critical one-tail	1.647877	
P(T<=t) two-tail	**0.450824**	
t Critical two-tail	1.964673	

DECODER with optimal weights

t-Test: Paired Two Sample for Means

Meteor: (before, after)	0.618436	0.623142
	Variable 1	*Variable 2*
Mean	0.596566	0.6022
Variance	0.067387	0.069134
Observations	506	506
Pearson Correlation	0.985214	
Hypothesized Mean Difference	0	
df	505	
t Stat	-2.81299	
P(T<=t) one-tail	0.00255	
t Critical one-tail	1.647877	
P(T<=t) two-tail	**0.0051**	
t Critical two-tail	1.964673	