Example-Based Machine Translation:

An Adaptation-Guided Retrieval Approach

Bróna Collins

A thesis submitted for the degree of

Doctor of Philosophy in Computer Science

University of Dublin, Trinity College

Department of Computer Science

Spetember 18, 1998

Declaration

I declare that the work described in this thesis has not been submitted for a degree at any other university, and that the work is entirely my own.

Signature

_____

Bróna Collins,

September 18th, 1998.

Permission to lend and/or copy

I agree that the library in Trinity College Dublin may lend or copy this thesis upon request.

Signature

_____

Bróna Collins,

September 18th, 1998.

For Desmond and Valerie Collins — for love, strength and vision.

Acknowledgements

This thesis owes so much to the great people who helped me learn and who made the experience all the more enjoyable. Above all, I am grateful to Dr. Pádraig Cunningham for his excellent supervision, encouragement, eye for detail, and not least, his spirited company during the past four years.

I wish to thank the current and previous members of the Artificial Intelligence Group in Trinity College Dublin who offered their valuable opinions on my ideas before they even came near to reaching print. In particular, my heartfelt thanks go to Dr. Tony Veale, Dr. Mark Keane, Dr. Carl Vogel and Dr. Fintan Costello.

I am grateful to the Hitachi Dublin Laboratory who provided material support for my research, and the staff of the Computer Science Department who have supported me in numerous ways throughout the past four years. Particular thanks go to Prof. Byrne, Head of Department. The humour, good-nature and good taste in music of my colleague, Arthur Hughes, has kept me sane during the last stages of writing. Dr. Steven Collins is being thanked in advance for bringing the manuscript to the printers.

I owe thanks to many people in the field who have been supportive and inspirational, especially Orlagh Neary at Corel Corporation, Michael Carl at IAI, Dr. Christer Samuelsson at XEROX, Dr. Martin Portman at CRE.

My dearest friends must be thanked for their immense loyalty, love and fun during this adventurous four year period of research and teaching. I would not have enjoyed it without you — Jens, Lisa, Mary, Monica, Clíona, Trás, Kristen, Dan, and Killian.

Abstract

EXAMPLE-BASED MACHINE TRANSLATION

Bróna Collins

Supervisor: Pádraig Cunningham

Translation can be viewed as a problem-solving process where a source language text is transformed into its target language equivalent. A machine translation system, solving the problem from first-principles, requires more knowledge than has ever been successfully encoded in any system. An alternative approach is to reuse past translation experience encoded in a set of exemplars, or cases. A case which is similar to the input problem will be retrieved and a solution produced by adapting its target language component. This thesis advances the state of the art in example-based machine translation by proposing techniques for predicting the adaptation requirements of a retrieval episode. An Adaptation-Guided Retrieval policy increases the efficiency of the retriever, which will now search for adaptable cases, and relieves the knowledge-acquisition bottleneck of the adaptation component. A flexible case-storage scheme also allows all knowledge required for adaptation to be deduced from the case-base itself.

The first part of the thesis contrasts such a CBR-motivated approach with current EBMT systems which are either data-intensive or knowledge-intensive. A new EBMT scheme is proposed in which the cases encode knowledge about their own reusability, determined by cross-linguistic mappings. The information allows cases to be generalised carefully, to the degree that is necessitated by the data. Linguistic and translational divergences — the obstacles to reusability — are investigated in the domain of software-manual translation, and on this basis, a suitable case representation scheme is proposed.

The second and third parts of the thesis describe the on-line and off-line processes of an EBMT system in which the case-base is the only knowledge source. Cases are deduced from texts automatically, and at run-time, the matching and retrieval tasks exploit the adaptability information in the cases in order to maximise coverage without compromising on accuracy. The multi-tiered case representation scheme allows adaptation at the sub-sentential and word levels, when necessary. The general performance of the system is shown to degrade gracefully and to improve as the case-base size increases.

Contents

List of Figures

List of Tables

Part I

Background

Chapter 1

Thesis Overview

1.1 Introduction

> *The existence of deep-seated formal universals [..] implies that all languages are cut to the same pattern, but does not imply that there is any point to point correspondence between particular languages. It does not, for instance imply that there must be some reasonable procedure for translating languages.*[Cho65]

When one is suddenly faced with a text in a remote foreign language, the page might as well be blank. We cannot begin to process it, because there are no familiar points of reference. Our innate ability to process language doesn't stretch as far as deducing the meaning of arbitrary collections of symbols of one language on the basis of another. Nevertheless, given a few starting points, such as the meaning of the fifty most frequently occurring words, one could begin to deduce the gist of shorter sentences containing those words. Despite Chomsky's pessimism on the matter, the fastest growing area in empirical MT research has centred around the very idea of identifying points of correspondence in parallel texts. Statistical procedures, let loose on vast amounts of text, churn out models of equivalences between languages whose parameters become more finely tuned to the patterns of both languages, the more exposure to such texts they get. Such techniques have created the raw material for a new brand of Machine Translation - the example-based machine

translation (EBMT) approach. EBMT is the mimicking of previous translation experience which can be expressed as a set of equivalences between different languages. Each translation experience is stored as a trace in memory which can be used to guide later processing.

Simultaneously, a new trend in Artificial Intelligence research has focused on equipping machines with the means to acquire and reuse problem solving experience for tasks in general. This demands an ability to detect similar problems and also a means of adapting the solution of a previous case to suit a new situation. The Case-Based Reasoning (CBR) paradigm is thus the perfect vehicle for EBMT. Emphasis is placed on the representation and indexing of experience in memory which facilitates the selection of a relevant experience. Often the representations are highly abstract or complex denoting air-traffic control decisions, cutting plans for machine tools or routing plans for emergency vehicles. CBR extends the notion of having a translation model to having a set of instances of that model and being able to *adapt* them to new situations.

Now the interest in bringing re-use technology back into MT has been motivated in the commercial sector. Multinational companies and localisation firms have discovered the benefits of translators' tools such as bilingual lexicons, concordancers, terminology databases, and translation memories. New interfaces are being built for old MT systems, and translators, far from feeling threatened by MT, are now eager to exploit the huge quantity of old data for the slavish tasks of translating updated versions of documentation in bulk and consistency checking across multiple versions of similar texts.

Formally, in order for a pair of texts to be reusable, a measure of **translation equivalence** must hold between them. Translation equivalence is a relation which holds between expressions with the same meaning. In some controlled translation scenarios, the sentence is a reliable basic unit of equivalence but even with this assumption, EBMT faces the same extraordinary problems of complex language transfer as every other MT architecture. Below the sentence level, equivalence is a complicated relation to express and symbolic representations can only approximate the point-to-point correspondences. Part of this thesis is devoted to describing how such equivalences between languages can be stored in cases on the basis of information automatically extracted from the data.

To translate a text from scratch from one language to another requires knowledge about the kinds of structures that exist in language and how they can interact. Hence, an EBMT system must reuse cases carefully, for adaptation operations may cause these structures to clash or cancel each other, or render the sentence ungrammatical, or meaningless, or too long. In traditional MT, grammar formalisms and fixed sets of rules describe what combinations are allowable in the domain of expressions. However, the whole idea behind EBMT is *not* to have domain rules but to rely on experience alone, and so the validity of the reused and adapted solution may be at stake.

This dilemma may be solved by identifying pieces of text which, with respect to a new problem, do not require complex adaptation. This is not to say that these pieces do not stand in a complex equivalence relationship, they may well do, but with respect to a given input problem the necessary adaptation may not upset the original transfer relation. With a vast quantity of sentences available for EBMT, many of which will be very similar, a policy of favouring the particular example which is adaptable in the "correct" positions is an improvement on algorithms which merely seek a similar example. This policy is called **Adaptation-Guided Retrieval**. This thesis describes the implementation of such a policy and the learning of the data structures which support it. Most EBMT systems choose adaptable fragments by trial and error, no system reported uses a policy of adaptation-guided retrieval to restrict the search space in the first place. A large part of this thesis describes how an EBMT system can be configured to retrieve cases carefully.

1.1.1 Thesis Structure

In Part I of this thesis, EBMT is examined in context of theoretical and practical approaches to MT to highlight the merits of a CBR approach to translation reuse. Part II presents a novel solution to the case-creation knowledge bottleneck using a data-oriented parser and linker. Part III describes Adaptation-Guided Retrieval (AGR) showing how cautious reuse of translations results in very effective exploitation of the examples present in the case base.

Part I: Motivation and Application

Chapter 2 provides the background against which to introduce a novel approach to EBMT. In introducing a CBR philosophy to EBMT, a methodology is proposed wherein the only knowledge for translation is stored in cases alone, and yet these cases are structured enough to provide some generalisation of patterns in and across languages. The actual patterns which can be expected to arise in "real" data, for sample texts from the localisation domain, are investigated in Chapter 3.

Part II: Automatic Creation of an EBMT Case Base.

Chapter 4 presents the data structures and memory organisation of the ReVerb system, which supports incremental updating of knowledge at three levels of description - word, chunk and sentence. The information flow between such frames in the memory gives rise to a host of knowledge sources for EBMT which reflect the tendencies present in the training corpus. Chapter 5 describes the data-oriented tools which allow the creation of more cases. The symbiosis between case storage and creation is shown to result in a gradual, steady improvement in performance as more cases are created.

Part III: Adaptation Guided Retrieval for full and partial case matches.

In the final part of the thesis, a detailed mechanism for the assessment of adaptability of cases on a structural basis is presented. This includes both full and partially matching cases. A templatisation scheme is proposed whereby cases are generalised on the basis of their individual patterns of equivalence. A novel thresholding filter ensures that a user-determined adaptability score can be imposed *before* retrieval to ensure a certain level of adaptability of the cases retrieved. Candidates are thus chosen on the basis of adaptability and similarity rather than similarity alone. A means of assessing the adaptation *after* candidate selection (as in other EBMT systems) is also presented as an additional indicator of translation reliability. The AGR methodology is tested and evaluated on real data in the final sections.

1.1.2 Contributions of the Thesis

To summarise, the contributions of this work have been the following:

- A Generalisation strategy for examples according to system-determined adaptability
- A knowledge-free means of creating new examples from raw data for Example Based Machine Translation
- Procedures for retrieving cases on the basis of similarity and adaptability combined

The main innovations in this thesis have been evaluated in each of the relevant chapters, on "real" data. It is demonstrated that assessing the flexibility of examples in this way avoids the problems of knowledge-intensive approaches and extends the functionality of translation memories in a dramatic manner. The coverage and generalisation power of the system is demonstrably higher than any other EBMT system working with the same number of examples. In the current information explosion at the very end of this millenium, the reader may foresee many further extensions of the ideas presented here, which will take EBMT right into the next century.

Chapter 2

Background

2.1 Introduction

If language transfer were a trivial problem, the world would have a host of high-quality MT systems whose output quality matched that of the best mono-lingual analysers around. However this is not so. The problem is that the structural rules and the expressivity of lexical items of languages do not map isomorphically at any level of description. And within each language there are many exceptions to those default rules that may be identified at all. This has prompted researchers to climb up the "classic" pyramid of abstraction shown here in Figure 2.1[1] in order to find a suitable representation where information could be transformed from SL to TL, divorced from the confusions and language specificity of string mutations at the bottom of the pyramid. The units of transfer at the top are more primitive elements of language -packets of concepts, which are held as being universal across languages. The *knowledge-acquisition bottleneck* which plagued such approaches caused researchers to climb down the pyramid in recent years. Now however, more sophisticated string-to-string models of translation have been created due to enhancements in technology and the availability of on-line corpora. These in turn have allowed researchers to build more abstract models of translation *processing*, inspired by ideas from Artificial Intelligence, where stored translations

[1] This was first proposed by researchers working on the GETA project.

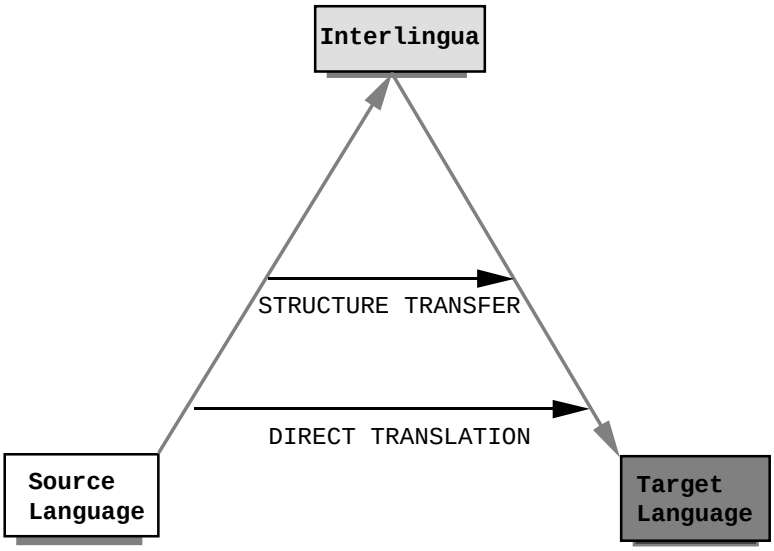


Figure 2.1: *The Pyramid of Transfer in MT.*

are abstracted and reused. So researchers are once again climbing up the pyramid but this time armed with the means of automatically determining regularities within languages. The following sections guide the reader through these more-or-less chronological developments, starting with the well-documented "traditional" MT classification.

Traditionally MT approaches have been divided into three categories (see [HS92] [Kin87] for a detailed survey), characterised by the amount of linguistic abstraction which takes place before transfer of structures. These are:

- The Direct (First Generation) Approach
- The Transfer (Second Generation) Approach
- The Interlingua (Second Generation) Approach

2.1.1 The Direct Approach

The simplest, First Generation (henceforth G1) of MT systems used word-for-word and phrase-to-phrase translations with little or no capability for rearranging syntactic constructions or lexical selection restrictions, i.e. a *direct* approach. Because the analysis stage is so information-deprived,

there is a heavy burden on the bilingual lexicons to guide the translation, and often the structural changes are simply *ad-hoc* or default actions. The first public demonstration of the MT system, which resulted as a colloboration between Georgetown University and IBM, showed a system that could translate a carefully selected 49 Russian sentences into English using a very restricted vocabulary of 250 words and a mere 6 grammar rules. The earlier incarnations of the SYSTRAN system can also be classified as being direct, depending heavily on large bilingual dictionaries and lacking a clear separation between analysis and generation. The USAF Russian English Systran system [Bos86] has been in use since 1970 producing almost 100,000 pages of text per annum with an error rate of less than 5

2.1.2 The Transfer Approach

The Transfer Approach is inherent in most Second Generation (henceforth G2) MT systems, so called because of their step-up in linguistic sophistication from the First Generation of MT engines. In general these systems produce a data structure representing the syntax and semantics of the SL text and transform it into a new one which represents the same semantics but the TL syntax. Sub-tasks such as analysis, transfer and generation are performed in separate modules, and within these, linguistic descriptions are distinct from the algorithms which use them. Also, on the linguistic side, the G2 architecture is "stratificational" in the sense that morphology, syntax, semantics and perhaps non-linguistic knowledge are described separately. In 1977, the successful implementation of the MÉTÉO system which translated Canadian public weather forecasts, confirmed that MT can work extremely well when applied to simple natural sublanguages. There are unfortunately few situations of this kind of success on the translation market. The EUROTRA [ABD+86] and ARIANE (GETA) systems [VB85], [Boi89] systems exemplify the modular, stratified transfer-based approach to MT while others like METAL [HS92] allow some interaction between analysis and transfer. It is a common observation (e.g. in [Tsu89]) that those systems which have been based on existing linguistic theories designed for monolingual analysis are linguistically elegant but disappointing in terms of performance on real data, possibly because monolingual grammars are not suitable for the

task of transferring meaning across languages.

2.1.3 Interlingua

Another G2 approach was to abstract the transfer representation even further in order to capture the meaning of the SL in a structure which is divorced from the surface-level considerations of that language. This *interlingua*, a high-level representation, would act as a mediator between the two languages at a semantic level, and so concepts, rather than syntactic representations would be transferred. The TL would then be generated from a conceptual representation. This approach is exemplified in UNITRAN [Dor93] which uses lexical-conceptual structures [Jac83] and in ROSETTA [LOS89], which uses Montagovian Grammar as an interlingua and adheres to principles of isomorphicity and compositionality. The DLT project [MS89] is unique in using the language, Esperanto, to mediate between source and target. Transfer between highly different language pairs, for example English and Sign language [VC96], clearly warrant a completely abstracted meaning representation. However, researchers have found it extremely difficult to identify a set of primitives for such an all-encompassing meaning representation language, and the coverage and robustness of such systems tend to be rather poor.

2.1.4 Hybrid Approaches

The problem with "traditional" approaches, i.e. the direct approach (G1), and transfer and interlingua systems (G2) approaches, is that natural language expertise has to be manually encoded into their data structures and algorithms, whether as special cases in FORTRAN (as in MÉTÉO), or as a full representation of the conceptual content of the utterance [Dor93] [SA77]. This has been at the expense of coverage and robustness. The practical systems which have enjoyed commercial success are mostly non-theoretical in their syntactic approach and rely heavily on lexical pragmatics rather than any well-defined theory of grammar. This caused one researcher to declare in 1993:

> *"..no existing theory is very good at accounting for real-life data, and many of the better existing systems are based on rather silly theories".* [Isa93].

Attempts were made to improve coverage and accuracy without abandoning completely the notion of a well-defined theory of translation, for example the DLT system incorporated an example-based bilingual knowledge bank, or BKB, [SV90] to complement its otherwise linguistically motivated lexical transfer component. Another hybrid approach, the KANT system [CT87], integrated knowledge into the translation process such that the transfer of syntactic structures was mediated by a separate level of domain-knowledge representation, and translation of terms was performed by statistical means. However in other camps, there was a complete overhaul of the "rationalist" rule-based paradigm, in favour of completely empirical approaches.

2.1.5 Statistical Models of Transfer Functions

In the last two decades, the amount of computing power available to researchers, and the huge increase of linguistic resources on the Web has meant that empirical approaches have enjoyed a dramatic renaissance. The question to be asked is whether computers can deduce the language models from the vast data available which rationalist approaches failed to do. MT in particular had a new raw material —the growing quantity of parallel texts in multiple languages. At the word-to-word level, there have been several proposed algorithms for deducing translation lexicons from text automatically. These are often non-probabilistic or *greedy* algorithms [Fun95] [Mel96b] which operate on a pre-defined similarity function. Statistical *model re-estimation* algorithms based on co-occurrence of word tokens have also been used [BDPDPM92] [Mel98], though to a lesser degree due to the computational complexity of estimating the many parameters involved. In [BCDP+88] the idea was introduced that the word correlations between languages with similar word order could be exploited when estimating translation-model parameters, and subsequent research [DMM93] (also working on a French-English corpus) focused on improving the parameters. Nevertheless, the "word-order correlation bias" inherent in these approaches makes them language-dependent. Statistical translation models which do not take word order into account, i.e. "bag-to-bag" models, are described in [Mel98] who also discusses an algorithm for automatic discovery of non-compositional compounds in parallel texts. In the later version of the IBM research [BDPd+93], the processes

were modularised into analysis, statistical transfer and synthesis. Statistical bilingual parsing is another attempt to avoid using linguistic rules, and some of these approaches [Wu95] have reported success even for languages with vastly different character sets. A more linguistics inspired approach, although still within the empirical paradigm, is to try to infer a grammar and symbolic transfer functions from an aligned bilingual corpus of examples, as described for example in [Juo95] who optimises his model parameters using *simulated annealing*. This approach requires pre-parsing of the texts and a dictionary. Typical performance on French-English test data (simple sentences with one verb) resulted in 36% fully correct translations. A similar, though computationally expensive neural network approach [KG94] performs better (98%) but only on the very simple grammatical constructions it had learned.

2.1.6 MT — The AI Perspective

The problem of translation has occasionally been viewed from an AI perspective, in particular from a Case-Based Reasoning (CBR) or Memory Based Reasoning (MBR) perspective. Indeed the earliest CBR systems were designed to deal with problems of natural language understanding, for example IPP (Integrated Partial Parser) [Leb83] and CYRUS (Computerised Yale Retrieval and Updating System) [Kol83]. From an AI stance, MT is just another problem to solve, and perhaps analogical problem-solving techniques can be applied in this domain. Here we motivate a CBR approach to MT in a broader sense than is usual in the EBMT literature — to demonstrate that recent CBR techniques can be exploited for EBMT, and also that EBMT can mean more than simply another transfer architecture in disguise, as suggested by some researchers [Isa93].

2.2 Case-Based Reasoning

2.2.1 Introduction

Case Based Reasoning (CBR)is founded on an appealing idea: expertise comprises experience. One of the aims of AI research has always been to emulate human problem solving abilities in a com-

putational process [Min63] [Tur50]. In CBR the aim is to know how to modify an old solution to fit a new situation. CBR is a general technique for reasoning from experience. From a theoretical psychological point of view, CBR explores the possibility of encapsulating episodic rather than static knowledge in oder to deal with new situations, and this aspect has been explored by Tulving [Tul72] and by Schank and Abelson [SA77] among others. Notwithstanding these psychological motivations, CBR was in fact largely proposed as a solution to the problems of searching for solutions in a first-principles approach. In general, CBR approaches tend to avoid the need for complicated domain models by relying solely on descriptive cases and rules which can modify their solutions when required, see [Kol93], [AP94] or [RS89] for more details. Indeed, the main thrust of CBR research since the early nineties has focused on enginnering issues. The following sections highlight the contrast between first-principles (FP), case-based reasoning (CBR), and memory-based reasoning(MBR) as problem solving strategies. Interestingly, these three problem solving techniques are analagously reflected in the three general categories of MT — G2, EBMT and MBMT.

2.2.2 First Principles

In first principles problem solving, a solution is generated from a set of specification features. This may involve an elaborate search of the possible solution components derivable from the specification or incorporating knowledge sources which are expensive to compute. In MT for example, G2 approaches rely on the application of consecutive grammar rules where at each level, any number of routes may be chosen which can only be constrained by a set of requirements specified by the domain model (e.g. the grammar formalism) and the rules which have already been applied. Thus, the specification is gradually transformed into a solution. Previous solutions have no bearing on the problem solving process so given the same sentence twice in succession, a G2 MT system will take just as long to process it on the second run. Moreover, if particular problems crop up frequently this will not make them any more solvable.

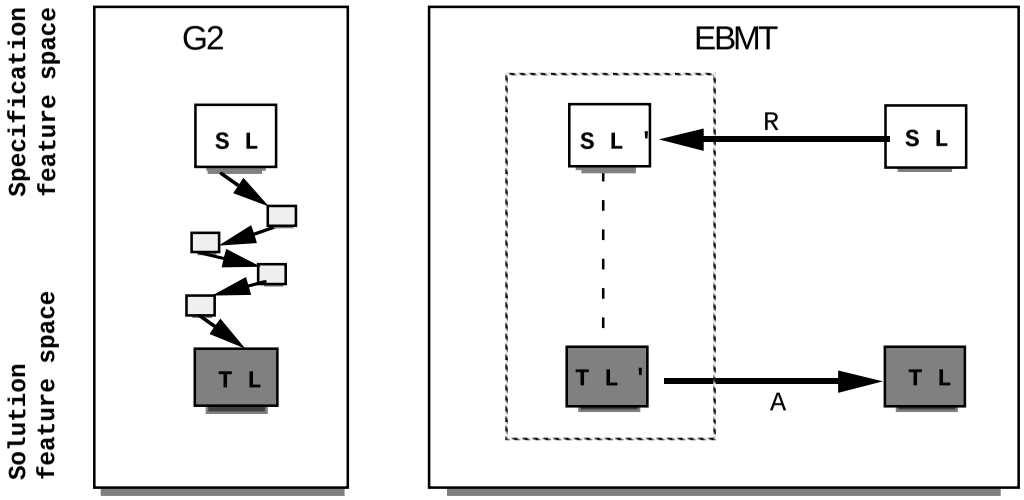


Figure 2.2: *First-Principles vs. EBMT*
First-Principles problem-solving in the G2 model of MT versus the Case-Based Reasoning model of EBMT

2.2.3 The Standard Model of CBR

CBR tackles problem-solving in a manner which has been standardised into a series of steps. Althoff [Alt95] summarised the steps as the memorable four R's: Retrieve, Reuse, Revise, and Retain. Reuse and Revision are commonly collapsed into the term 'Adaptation' [Cun98] which involves changing parts of a case to suit a new, and slightly different problem. The Retrieval-Adaptation-Retention scheme assumes a memory model for representing, indexing and organising past cases and a process model for retrieving and adapting them and assimilating new ones. This engineering overhead often means that CBR approaches only incorporate a subset of these stages, for example Retrieval-only systems exist which simply produce a previous solution in the hope that it is sufficiently similar to solve the problem at hand. Other systems attempt to relieve the cost of reuse, by ensuring that only sufficiently similar or adaptable cases are retrieved.

Exploring the stages in more detail, the most crucial stage of CBR is often regarded as being the first, Retrieval, i.e. seraching for a *best* case in the memory structure whose problem specification is the most analogous to the new problem. This involves searching for a match in the specification space. It is assumed that the solution retrieved will be close to the desired solution in the sense

that a few minor adaptations performed in the solution space will render it into the correct form. This is depicted in Figure 2.2. What is significant from a computational point of view is that there is no laborious transformation between the specification space and the solution space. The work is instead done in the retrieval and adaptation stages. The easier these retrieval and adaptation stages prove to be, the more persuasive the argument for adopting a CBR approach becomes.

2.2.4 Adaptation Rules

In most CBR systems, adaptation is guided by fixed case adaptation rules. Researchers working on the problem [SK94] [Kas94] [Lea94] have shown that it is difficult to determine such definite rules. A trade-off exists between specificity and generalisation. Specific rules are easy to apply and perform reliably but they do not cover many cases, whereas more generalised rules, while spanning a broader range of problems, often do not provide specific task and domain specific guidance. In creating specific rules, the developer is often required to perform detailed analysis of the task and domain to determine which rules are required. On the brighter side however, there are domains in which lightweight, specific adaptation rules can be derived by comparison of the problem and the retrieved case (our approch to EBMT is one), and recent research has begun to focus on the issue of automatic learning of more generalised rules, see [SK94] [LKW95] [HK96].

2.2.5 Adaptation-Guided Retrieval

The CBR processes of Retrieval and Adaptation need not be separate entities. In many systems they are however — the retrieval of a case is done on a similarity basis of the problem specification alone, without any regard for the necessary changes to the solution, possibly close to the way in which humans make analogies. The basic idea behind Adaptation-Guided Retrieval, first introduced in [SK93] (later appearing as [SK94]), is that the retrieval process should be directly influenced by a metric which estimates the adaptation-cost of applying an old case solution to a new problem. Of course, if the engineering overhead in estimating the adaptability of the case is too high then the overall saving in retrieval expense will be minimal. To take an example where the adaptability

of a retrieved solution is a wiser metric to employ than similarity when retrieving a past solution, consider the following informal example. If a novice CBR-robot wanted to buy food in McDonald's he would search through an index of cases embodying human experience and possibly retrieve the RESTAURANT-CASE in order to guide him. The procedure for eating in McDonald's may seem close enough to eating in a restaurant on the face of it (buying food). Unbeknownst to him, had he chosen the case for BUYING-A-STAMP in a post-office, the action sequence constituting the solution would be more similar and hence require less adaptation (queue, order at counter, pay at counter, leave) than the RESTAURANT-CASE did. Re-enacting an old solution for a new problem involves adaptation, which humans are good at. Computers struggle with adaptation — it requires knowledge of how a modification of some aspect will affect an already complete solution. Adaptation-Guided Retrieval (AGR) attempts to redress this problem by allowing the retriever and adaptor to share adaptation knowledge. The retriever uses this knowledge to assess how adaptable a case is *before* deciding whether or not to reuse it, and the adaptor carries out the modifications stipulated in the knowledge.

Adaptation-Guided Retrieval is highly relevant for Example Based Machine Translation in the sense that many previouly translated pieces of text may be close to the new problem translation judging by surface-similarity of the source texts, but yet only a subsection of these might be reusable. This could be due to divergences (see Chapter 3) between the structures of the SL and TL in the case being reused, or the fact that either 'side' of the case omits some information present in the other side. In such scenarios, a retrieval metric which also determines adaptability of the SL-to-TL solution, should save on adaptation costs, as well as reducing the retrieval search space.

2.3 EBMT

EBMT is the application of Case-Based Reasoning to MT. It is an attempt to avoid the problems of assuming a compositional transfer from source to target, translating instead by analogy. EBMT systems store a huge set of translation examples which provide coverage in context for the input.

The store of translation examples is maintained in the form of of SL TL pairs, usually aligned at the sentence level. An input sentence is matched against this repository in order to find a *similar* match. The identification of similarity depends on some measure of distance of meaning; in current proposals this is based on the classification of lexical items in semantic hierarchies [SOI+93] [SN90], or on the distribution of key elements such as function words [Juo95] words, or similarity of part-of-speech sequences[MJS94], or a combination of these [CC96] [CHP94] [Car97]. Although the idea of electronically storing past translations in bilingual format in oder to reuse their contents was discussed as far back as 1978 by Peter Arthern [Art78] and slightly later, by Melby [Mel81], the term Example-Based Translation is accredited to Nagao who introduced the notion of analogical translation in [Nag84]. He stressed the notion of detecting similarity:

> *The most important function ... is to find out the similarity of the given input sentence and an example sentence, which can be a guide for the translation of the input sentence.*
>
> [Nag84]

Nagao also suggested how the *adaptability* of a example could be checked: "the replaceability of the corresponding words is tested by tracing the thesaurus relations". If the thesaurus similarity was high enough then the example was accepted for the translation of that particular substring, if not, then another example was searched for. A simple, and oft quoted example of this procedure was demonstrated by Sumita et al. in [SOI+93]. Here, a system is described which translated only phrases of the form "Noun **no** Noun", where *no* is the general partitive ad-position in Japanese. In most contexts, it is similar to the English preposition **of** see (1) below:

(1) "N1 **no** N2" → "N2 **of** N1".

But in others, the more natural English translation would be something else, as the following mis-translations show:

(2) a. The conference **of** Tokyo

b. The holiday **of** a week

c. Hotels **of** three.

Sumita incorporated a commercial thesaurus of everyday Japanese and calculated semantic distance of nouns in the corpus which appeared either side of the *no* particle on the basis of their distance apart in this hierarchically organised thesaurus. The hierarchy was searched bottom up from the nodes where the relevant words occurred until a *most specific common abstraction* was found. The resulting score was multiplied by weightings based on the probability of the translation, for example in the case of *no*, it was translated as **in** for all its occurrences and so received a maximum score of 1. This method was an improvement on traditional approaches to selecting the correct translation from a set of synonyms of a word in context. However, it still required a suitable parser, generator and thesaurus for similarity judement.

2.3.1 Case Decomposition in EBMT

Similarity judgement is not the only issue in EBMT however. Particularly in systems which strive to produce translations for a whole string, coverage is the major issue. There cannot be an exact example for every sentence so in the absence of compositional rules, examples must be able to provide some means of matching the generative power of natural language. The requirement for case-decomposition is that the problem be covered by sub-cases whose solution parts are recomposable. In EBMT, any number of linguistic contortions (see Chapter 3) may have occurred between the source and target strings (head switching, ellipsis, conflation, promotion, demotion, etc.) so this forces case decomposition to take cross-linguistic mapping into account at some level. In the absence of an accurate model for sub-sentential translational equivalence, the creation of such examples has ironically required rule-based linguistic processing.

Attempts to store translation examples in linguistically motivated structures are described in [SN90] and [SV90]. Sato and Nagao proposed that off-line linking of dependency trees (see Figure 2.3) was the answer to the coverage problem while not compromising on accuracy of the solution.

In order to determine the combinatorial possibilities, they created a database of source and target language dependency trees for each example translation pair, as depicted by the simple example in Figure 2.3. Correspondence points or "links" determined the substitutable regions of the dependency tree by other subtrees in the data base, which provided the generative power required for coverage of a new input, for example:

(3) "He buys a book on international politics."

The source string (3) is decomposed into a "source matching expression" *SME* which indexes the example base for translation units using hashing techniques. For example, decomposition of the input means replacement of the subtree e3 with e13 (see Figure 2.3). The head verb matches here. The isomorphic Target Word Dependency tree is a matching expression called the *TWD*. Transfer is merely the replacement of each source ID with its corresponding ID as indicated by the links in each example. The actions in source and target for a simple substitution are given in (4) below:

(4)

```
(e1 (r, e3, (e13)))
(j1 (r, j5, (j15)))
```

This assumes that the portion of the tree `j1` to `j4` is unaffected by the substitution. To choose among several translation candidates the heuristics of preferring large translation units over smaller ones and choosing a unit from a similar syntactic environment were used. Similarity of environments relied on the use of a hand-coded thesaurus defining semantic similarity between words.

```
sim([book,n][notebook,n],0.8).
sim([buy,n][read,n],0.5).
sim([hon,n][nouto,n],0.8).
sim([kau,v][yomu,v],0.5).
```

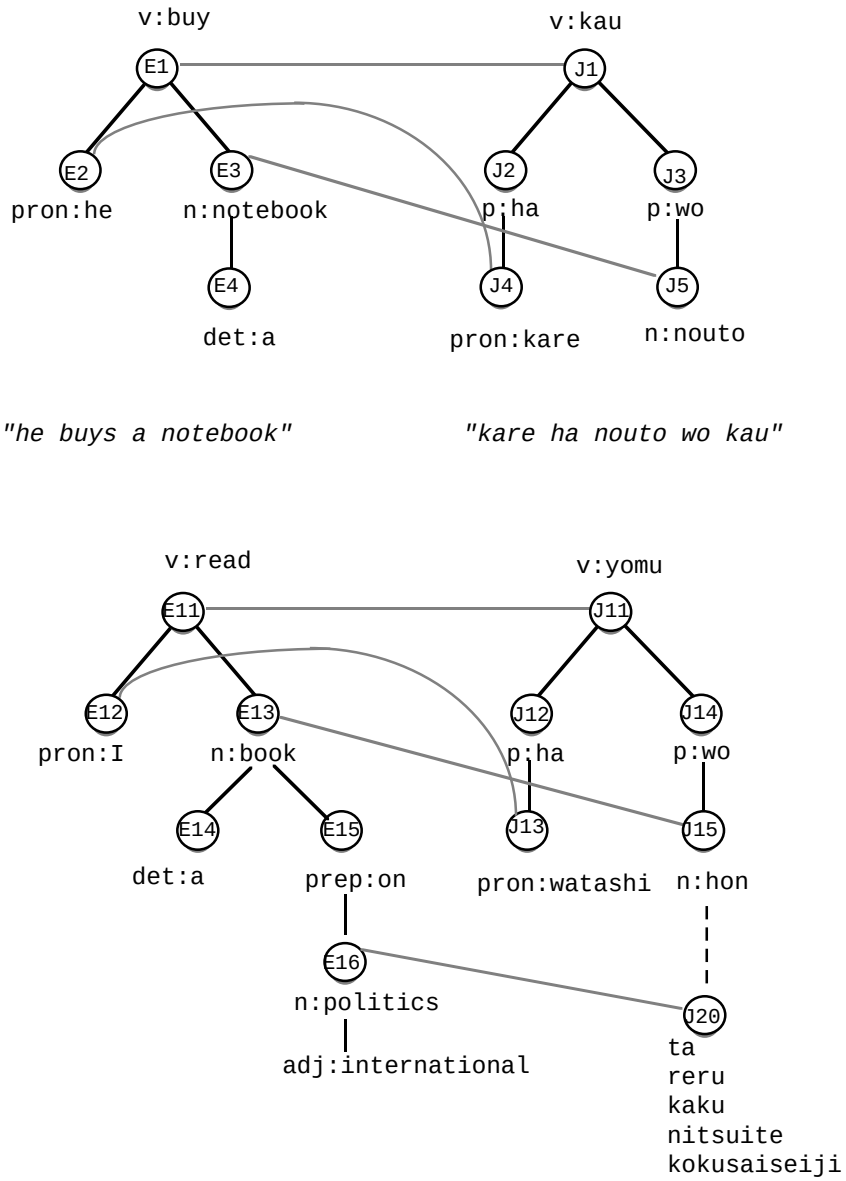


Figure 2.3: *Representing Examples in Dependency Trees*
Taken from [SN90]. The input sentence is "he buys a book on international politics". This can be covered by substituting `e3` above with `e13` below, and performing the reciprocal action in the target language dependency tree, i.e. substituting `j5` with `j15`

Sato and Nagao scored the replacement within a space they call a *restricted environment* preferring substitutions where the substituter subtree comes from a similar background to the substitutee in the input expression *in both languages*. Similarity of environment was approximated by defining a context around the relevant subtree pairs (substituter and substitutee) including the mother node and any sisters and their daughters. This was done for both source and target matching expressions scores and the minimum score is given as the overall translation score. This is interesting as it is judging similarity of matching contexts within the two languages, i.e. they did not assume that similar matches on the source language side will produce good matches on the target side. The overall advantage of this approach is the high-quality of solutions it seems to offer. This is at the price of maintaining a thesaurus for defining synonymy however, and the computation necessary for creating and linking dependency trees of the source and target languages.

Another system based on the off-line alignment of translation fragments or "templates" was that of [KKM92]. At the off-line template learning stage, sentences were parsed and the resulting chart structures were coupled at the word level using a dictionary, and at the chunk level, using constituent boundary information present in the charts of both languages. The coupling was restricted to content words and the following restriction applied:

> *A phrase X in one language sentence S is not coupled to any phrase in the other language T if T does not include a phrase which includes counterparts for all the words inside X. [KKM92]*

Each coupled pair of phrases was a candidate for replacement with a variable. A sentence with variables in replaceable positions was known as a template. Unlike in Sato and Nagao's approach, the templates were flat structures; any substitutions that occurred could not take syntactic similarity (i.e. positioning in a tree) into account. A series of templates could arise from one sentence pair and these fragmentory templates could be embedded in the result of translation by another template. Thus generalisation and fragmentation of templates served to increase coverage while taking care not to chop templates at implausible positions in either the TL or the SL. In [Car97] it is argued that flat

representation is preferable "because the length shape and type of chunks that can be compositionally translated may vary from language pair to language pair". In this CBR-inspired proposal, an input string is abstracted by allowing horizontal decomposition — representing a string in terms of its lexical and grammatical features (POS, lemmas, agreement information), and vertical composition — chunking of the sentence into compositional chunks. Chunks of the SL are compositional if the case base contains examples of similar chunks which map to a fully formed TL phrase. It is not discussed how the examples are entered in the case base and it would appear that an example of every non-decompositional compound is required in the case base in order for correct decomposition to take place. Assuming that the retriever does decompose and cover the problem, the sequence of translated chunks is then passed to the CAT2 MT-system [Str95] which performs adaptation in terms of chunk-ordering and the agreement requirements of the words. It is therefore hard to say whether their sentence decomposition scheme is conducive to avoiding adaptation costs, as the adaptation is performed by a powerful MT system.

2.3.2 Memory-Based Reasoning

Memory-based reasoning(MBR) places memory at the base of all intelligence, and in a sense embodies a highly empiricist stance with regard to problem solving. It is basically CBR with less emphasis on the "R", as it tries to solve problems using the cases alone without relying on a well-defined domain model for validity checking of the solution features [SW86] [CS93] [CSV95]. The techniques used are variations on the classic k-nearest neighbour classifier algorithm. The instances of a task are stored in a table as patterns of feature-value pairs representing the problem specification, along with the associated "correct" solution. When a new pattern is processed the k-nearest neighbours of the pattern are retrieved from memory using some similarity metric. The problem solution is then determined by extrapolation from the k nearest neighbours, that is the output is chosen which has the highest relative frequency among the nearest neighbours. Because adaptation is rarely attempted in MBR, it requires that near matches be retrieved and hence that the case base be large. In certain circumstances, MBR has proven useful for restricted domains and NLP tasks, for

example census-classification [SW86], lexical acquisition [Dae95] word pronunciation [DZBG96] and information-retrieval tasks. With high-power parallel processors, it is also a feasible approach to MT, as demonstrated in [KH91] and [SOI+93].

2.3.3 Memory-Based MT

MBMT involves the direct reuse of previous translations rather than the use of language models extracted from corpora, or otherwise. Since 1990, there have been several approaches to MBMT which try to resolve the bottleneck of rule-based parsing for the case-creation and matching stages by using simpler matching techniques (e.g. the *longest common substring* algorithm, similar to the unix `diff` algorithm). For example, the EBMT component of the multi-engine PANGLOSS system [Nir95] used this approach. To illustrate, `(SLCS)` in (4) below indicates the problem source string and where `CC` denotes the longest matching substring in the corpus which matches it, `TLCS`, the corresponding sentence and `TCC`, the fragment of the `TLCS` which is deemed to be the equivalent sub-part. The following basic problem is highlighted. Even though a word, **z**, may not correspond to anything in the source side of the example `SLCS`, it could quite plausibly be an inherent part of the translation of `CC`, the substring which best matches the input.

(4) SLCS = (a b c d e f g h i)

CC = (a b c d)

TLCS = (**z** a' b' c' d' m n o p)

TCC = (**z** a' b' c' d')

The longest substring for `TLCS` is calculated, and the "best" candidate, chosen from all cases will be used for the translation of the input. This problem was recognised and in [Bro97] it is described how the later PANEBMT engine does not attempt to find an optimal partitioning of the input but instead seeks the corresponding TL chunk in an indexed corpus of every word sequence greater than 2 which appears in the input source string. This involves alignment at runtime of many fragments and so the test functions for determining the TL chunk are kept simple (number of matching and

non-matching words, length differences, etc.) The final selection of the "correct" cover of the input is left for a separate statistical language model. Brown comments on the advantage from a TL point of view:

> *An advantage of this approach is that it avoids discarding possible chunks merely because they are not part of the "optimal" cover for the input, instead selecting the input coverage by how well the translations fit together to form a complete translation.[Bro97]*

The earlier system of [MJS94] used a similar non-linguistic approach to covering the input at run-time in that the fragments were not produced by a parse scheme but by a simple dynamic programming algorithm, which took into account fragment length, word and POS matching. A limited number of gaps were allowed in the matching procedure and the resulting TL fragment recombinations were assessed for plausibility on the basis of corpus statistics and a process of "back-translation".

Cranias et al. [CHP94] used minimal linguistic information to identify function words, lemmas and POS features of words in a sentence and encode this into a vector which is then compared against the corpus also using dynamic programming. To reduce the search space at run-time, the examples were clustered into fragments beforehand and each cluster had a representative fragment called a "cluster centre". A succession of sub-parts of the sentence vector were compared against cluster centres using the same matching criteria as in other MBMT approaches (matching words, POS). The chosen cluster centres were associated with SL sentence fragments. The SL-TL alignment was based on the assumption that words retain their grammatical category under translation. In choosing the TL fragments which correspond to these, Cranias et al. admitted they had a problem:

> *If the sentences in the translation archive have been segmented, the problem is that now we do not know what the "translatable" units of the input sentence are (since we do not know its target language equivalent) We only have potential translation units markers.[CHP94]*

They claimed that by specifying that the threshold for a match of the input with a segment be high enough

> *we can be sure that the part of the input sentence that contributed to this good match will also be translatable an we can therefore segment this part*

This embodied an assumption that sections of parallel text which are each self-composed and translatable as a unit would combine smoothly in the TL solution. It is an extension of Kaji's idea of a replaceable element in a template, or Sato and Nagao's subtree substitutions. However, whereas the former approaches built a translation from one or two basic templates, filling in substitutions where necessary, the MBMT approaches aims to cover an input sentence using a series of n-grams. Without a basic template, no kind of similarity checking can guarantee grammaticality of the TL. In MBMT, there is no structural representation of the TL to adapt. Thus, adaptation in MBMT invariably relies on the existence of monolingual TL models to remedy the solution, and most systems are used in tandem with standard MT engines.

2.3.4 Summary

CBR systems re-use packets of experience when solving problems rather than expensive reasoning from scratch using domain principles, i.e. first-principles. In the standard CBR model, a new problem is solved by retrieving and adapting the solution of a suitable case, and EBMT is a reflection of this approach. Standard G1 and G2 approaches to MT can be seen as a CBR process with an atomic case-base where the whole translation task is performed by the adaptor. These systems cannot make use of larger chunks which allow items to be translated in a similar context and hence require less adaptation. In full EBMT, large chunks can be reused but it is their ability to be decomposed and recombined which will determine the success (coverage) of the system. Some form of case-decomposition is required such that parts of a case can be substituted with fragments from different contexts. In general, EBMT and MBMT models strive to avoid adaptation in the first place either by attaining good coverage (MBMT) or using sophisticated retrieval engines (EBMT). One EBMT method is to translate each fragment and assess its plausibility in its new environment

using thesauri and other outside knowledge sources. MBMT exploits large amounts of text and combines fragments according to likely patterns in the data. To date, no reliable technique between the *knowledge-rich* (in the sense of having structural knowledge) and *knowledge-poor* extremes has been reported which can assess adaptability for EBMT. Moreover, no EBMT system directly uses adaptability assessment information to help prune the search space at retrieval time.

2.4 A CBR Model of MT

The proposed filler of the gap in the EBMT spectrum takes the form of an adaptation-guided CBR approach to machine translation. The examples are structured enough to assess and represent correspondence between sentences and hence, adaptability, offline (à la Sato) but their structures are not so hierarchical as to require rule-based parsing and complex disambiguation. A range of retrieval modes reflect the various types of matching possible. For instance, variabilisation (à la Kaji) is a safer means of providing coverage in most situations. Should that fail, decomposition can also be done in careful stages, taking the potential adaptability of the template into account all the while. Retrieval, being a run-time procedure which should be efficient, adheres to memory-based principles of letting the data decide the indices (à la Cranias) yet this does not mean that the data that decided the indices is necessarily the best source of the solution TL. The data-determined indices indicate typical patterns in the SL data and impose them on the input string. However an adaptation guided retrieval policy uses these indices to identify among all cases those which will be adaptable on the basis of their SL′→TL′ equivalence maps.

Part II will describe the creation of such an example-base for EBMT and Part III, the run-time exploitation of cases for translation. As the merits of a methodology cannot be assessed entirely in isolation from its intended domain of application, the remainder of Part I of this thesis, Chapter 3, discusses the properties of texts in a practical domain for EBMT — that of technical documentation.

Chapter 3

An EBMT Domain

3.1 Abstract

This short chapter exposes the kinds of transfer problems that may exist in a selected text domain identified as being useful for reuse technologies — software documentation. The desiderata for a suitable case representation on the basis of this data are explored.

3.2 Introduction

There are many potential roles for EBMT in the translation process, as described in the literature. It can be integrated into existing MT architectures to fine-tune lexical rules [SV90], or to translate non-compositional compounds [Car97], or to improve coverage in general in a system which integrates multiple knowledge sources [Nir95]. In limited domains, such as avalanche warning reports in the Alps, it can even be a stand-alone generation system [MJS94]. However most of today's tedious, repetitive translations tasks involve texts that are not entirely *sublanguages* and which may have a high readership. A representative text *genre* is that of manuals —technical manuals, software documentation, and instruction sets. This chapter investigates the potential for fully-automatic EBMT on texts from this domain.

3.3 Software Manuals - The Localisation Domain

The reuse of previous translations in the localisation domain is nothing new. An important part of any localisation manager's pre-production analysis is to investigate the *leverage* of existing Translation Memory systems, that is the degree to which the texts recalled are actually useful to translators who use them. The development and exploitation of support tools for the translator has advanced in leaps and bounds since the mid-nineties. At present, the four most widely used translator's workstations originate from Europe: Transit 2.7 from Star, Workbench 2.0 from TRADOS, Optimizer from Eurolang and IBM's translation manager. Despite their ever increasing popularity, they do little or no decomposition of the problem. They store complete phrases and sentences in a database, and use distance-based metrics for comparing string similarities, similar to those of MBMT approaches (Section 2.4.3) in determining a good match. The buzz-words in TM circles are "fuzzy matching" and "fuzzy logic". From a CBR-theoretical viewpoint, a major shortcoming of these systems is the fact that they cannot capture certain "near-misses" successfully. There is also the "utility problem" [SK95] whereby the case-base grows linearly in size while coverage remains very limited, due to lack of generalisation. The following is an investigation into the feasibility of a more knowledge-based approach to reusing texts in this domain. First, the general obstacles to compositional transfer in language translation are examined in Section 3.3.1. and the assumption of parallel text is examined in Section 3.3.2. Then the degree to which these, and other non-linguistic factors arise in a chosen sample text is discussed in Section 3.4. In Section 3.5, the consequences such divergences have on EBMT is discussed and Section 3.6 motivates a particular case-representation scheme for an EBMT approach to machine translation, on the basis of the data investigated.

3.3.1 Cross-linguistic Divergences

Distinctions between the SL and the TL can be divided into two categories - *translation divergences*, in which the same information is conveyed in the source and target texts, but the structures of the sentences are different, and *translation mismatches*, in which the information that is conveyed is

Thematic	changes in argument structure	E: I like the car G: Mir gefaellt den Wagen E': To-me pleases the car
"	"	E: He lacks something G: Ihm fehlt etwas E': To-him lacks something
Promotional	head switching	G: Er ist zufaellig krank E: He happens to be sick E' He is coincidentally sick
Demotional	head switching	G: Er liest gern E: He likes reading E': He reads likingly
Structural	changes in argument structure	E: He aims the gun at him G: Er zielt auf ihn mit dem Gewehr E': He aims at him with the gun
Conflational	1-to-N lexical gaps	G: Schwimmbad E: Swimming pool E' Swimmingpool
	"	E: See again G: Wiedersehen E': (to) againsee
Categorial	category changes	E: Postwar(adj) G: Nach dem Krieg(pp) G' After the war(pp)
Lexical	N-to-1 lexical gaps	E: Give a cough G: Husten E' cough

Figure 3.1: *Divergences and Mismatches between languages.*
[*Taken from [Dor93] On the left side, Dorr's classification is denoted while Lindop and Tsujii's [LT91] classification appears in the second column (which is the preferred one in describing the data in this chapter). The table is modified from the original, in that it only shows the German-English comparisons, with additional examples, where there were none in the original version.*]

different. Translation divergences are widely discussed in MT literature. General cross-linguistic divergences are discussed in [LT91] and [Dor93]. Language-pair specific divergences are described for English-German in [SSW96] and for Italian-French in [Pod92], among others. Translation mismatches tend to be ignored in the MT literature. In EBMT, however, any differences between SL and TL, whether linguistically motivated or not, must be accounted for, and these are also included in the assessment below. The two classifications of translations divergences are summarised in Figure 3.1 above, taken from [Dor93].

3.3.2 Parallel Corpora

EBMT being a corpus based approach to MT fundamentally requires a parallel-aligned corpus, that is, a text together with its translation. Some examples of parallel bilingual corpora in the public domain are the parliamentary proceedings of the Canadian (the "Hansards") and the Hong Kong parliaments, which have been favourites of researchers since the early nineties. Having located a parallel corpus the two texts must then be aligned into corresponding segments, if they are to be useful for EBMT or TM systems. As other multilingual corpora became available, it quickly became obvious that texts like the Hansards were exceptionally clean translations. As pointed out in [GC93], "Real texts are noisy". Earlier alignment methods are likely to wander off track when faced with deviations from the standard "linear" progression of translation, as for instance when parts of the source text do not make their way into the translation (omissions), or end up in a different order (inversions). This might equally be the result of a stylistic choice made by the translator, or an unintended omission. Thus, sentence-level alignment is a non-trivial task and calls for smarter methods, see [FM97], [Som98], and [GC93].

One solution to the sentence alignment problem is to build the database of examples manually. Another is to use a parallel corpus where it is known that strict authoring guidelines have been adhered to by the translators. Many localisation companies, or companies who engage in in-house localisation, impose such guidelines in order to facilitate the creation of future translation memories or example bases. One fundamental requirement is that sentences be translated in a one-to-one fash-

ion, even if readability should suffer slightly. One can go even further and impose restrictions on the vocabulary used, as for example in the ScaniaSwedish Controlled Language for Truck maintenance experiment [SH97] where the vocabulary size was reduced. The resulting decrease in complexity enables more sentences to coincide, and for EBMT this would mean more examples which either coincide and mutually reionforce each other, or conflict with each other, i.e. examples with higher discriminatory power. The bilingual corpus used in the system described here, was not a controlled language in the stricter sense. Nevertheless, the translators did manage to translate over 99

3.4 The English-German Corpus

The parallel bilingual corpus used in this system was the manual for CorelDRAW V.6., a popular drawing package, localised into several languages on an on-going basis by Corel Corporation. The English and German versions of the texts were used. In order to assess the validity of using this corpus as the basis for creating a case-base for EBMT, it was decided to investigate a subsection of the data. Thus, a representative sample of 200 sentences was taken from the bilingual corpus. This text was made up of the Help files, and so contained vocabulary specific to many of the functionalities explained further in the manual. The sentences were translated almost exclusively in a one-to-one fashion, and the average sentence length was approximately 20 words. The amount of divergence was therefore the main factor that needed to be investigated. Most sentences in the sample exhibited at least one translation divergence type, and examples are given per divergence type in the following:

CHANGES IN ARGUMENT STRUCTURE

(1) X lets you..

Mit X können Sie..

With X can you..

(2) There's no limit to X.

X ist night eingeschrankt

X is not limited

(3) Use X when you want to..

X dient folgenden Zwecken.

X performs the following goals

LEXICAL GAPS

(4) ..as long as ..

..vorausgesagt

(5) You can also **weld** single objects with ***intersecting lines***.

Sie können einzelne objekte auch durch ***Schnittpunkte*** **miteinander verschmelzen**.

You can individual objects also through Intersection-points together weld

CATEGORY CHANGES

(6) X is **unavailable**.

X steht **nicht zur Verfügung**.

X is not for availability.

(7) **Closer inspection shows**..

Wenn Sie dies **genauer ansehen**..

When you this closer inspect..

VOICE: active → passive

(8) Ungroup **breaks up** one level of grouping at a time

Die jeweils letzte Gruppierungsebene **wird aufgelöst**

The respective last Grouping-level is broken-up

3.4.1 Translation Mismatches

There are other extra-linguistic factors which affect the SL↔TL mapping. These relate to the translator's style, dislike of the SL construction, or absent-mindedness. Another problem is anaphora, especially when the objects of reference appear in other sentences. Anaphoric effects are very much kept under control in this corpus. They are not, however, completely eliminated. Some sample translation mismatches are shown below:

PARAPHRASES

(9) this means...

Mit anderen Wortern..

With other words

OMISSIONS

(10)**With Inside selected**, the Offset value will take precedence over

the Steps value.

Bei Konturen die nach Innen verlangen, hat der eingestellten

Abstand Vorrang vor den Schritten.

For Contours that to Inside tend, has the set Offset precedence over

the Steps

(11)**Samples of the selected style** appear in the sample field

Der gewälte Stil wird im Feld Beispiel angezeigt.

The selected Style is in the field Sample shown.

ANAPHORA

(12)..**the blend** will reform automatically to incorporate your changes

..wirkt sich **dies** automatisch auf alle anderen Elements der

Überblendungsgruppe aus

..effects itself this automatically on all elements of the Blending-group

3.5 Discussion

It is clear from the data in Figure 3.2 that the likelihood of at least one divergence type occurring in a sentence of this domain is very high. In fact, only 8.5% of the sentences in the sample contained no divergences at all. The divergences shown have been specific to language-pair and domain of application, but the divergence types are language universal. The three predominant divergence types found were lexical gapping, category changes and voice differences. Lexical gapping can cause 1-to-n mapping at the word level within corresponding constituents as in (4) above, or between constituents, as in (5). What this means for adaptation is that when replacing an item in the SL with another it might very well happen that the replacer word does not exhibit the same 1 to n mapping. For example, a given sentence (13):

(13) You can also delete single objects with intersecting lines.

is quite likely to retrieve:

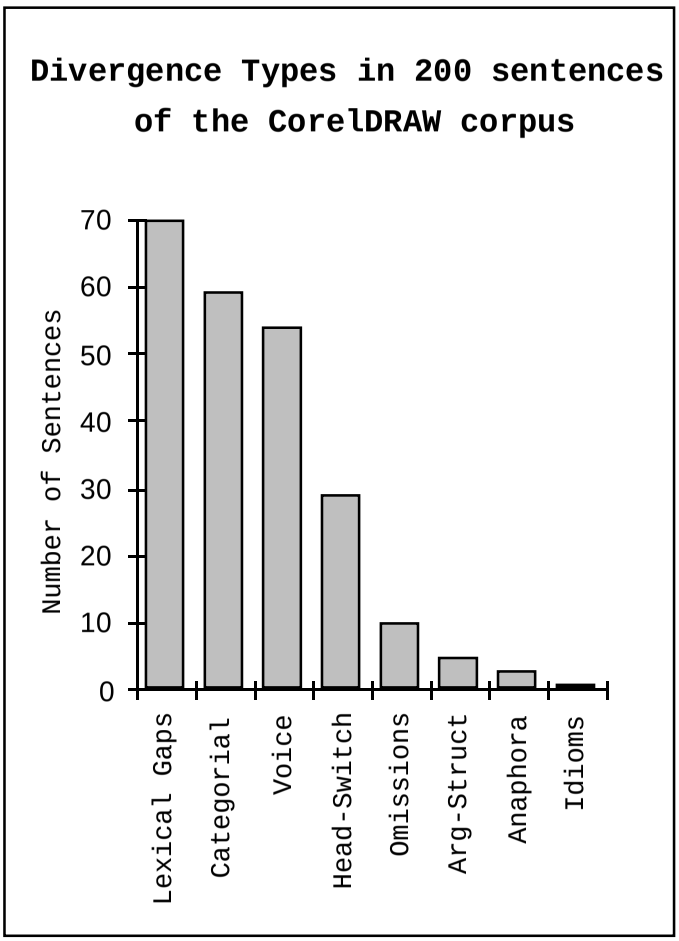


Figure 3.2: *Divergence-types in a 200 sentence sample from the CorelDRAW corpus.*

(5) You can also **weld** single objects with intersecting lines.

Sie können einzelne objekte auch durch Schnittpunkte **miteinander verschmelzen**.

You can individual objects also through Intersection-points together weld.

Unfortunately however, the word which requires replacement is one which undergoes the divergence (delete → **weld** → **miteinander verschmelzen**) and the resulting translation must adapt the solution by deleting two words and replacing them with one. It would be advisable to store this template in such a way as to disallow the variabilisation of the sentence in the position for **weld** (and **intersecting lines** for that matter). This of course assumes an ability to detect the divergence in the first place. The second-most common divergence type was category changes as typified in (6) and (7). In terms of their reuse, they are extremely dangerous and non-decomposable! Even if the input sentence is semantically similar, as in (8), the cross-linguistic divide renders the TL into a sentence which reflects a different meaning (7)′:

(7) **Closer inspection shows**..

Wenn Sie dies **genauer ansehen**..

When you this closer inspect..

(8) **Closer inspection indicates**..

(7)′ Wenn Sie dies **genauer andeuten**..

When you this closer indicate (= *When you mark this more clearly*)

Some of the divergences involve a change in syntactic function – passivisation (8), argument-structure ((1), (2), and (3)) which, in both English and German, affects the agreement restrictions between constituents. In German, case-marking is also affected and in English, strict constituent ordering

must be maintained (subjects always precede objects). From the point of view of coverage, much structural repetition was evident. For example the phrase "For more information see X" occurred several times, as did "X allows you to Y". Sometimes X and Y were quite long which suggests that a TM or MBMT system might miss them. However at a syntactic functional level, the similarities would be captured.

3.6 Conclusion

There is no one representation above the word-to-word level which successfully accounts simultaneously for all the matching possibilities between SL and SL′ and all the divergences which may occur in isolation or interactively. Yet most divergences taken in isolation are describable in terms of syntactic functional changes and often, if an input string is very similar to an example then only one or two adaptations will be required. Even translation mismatches can be expressed as syntactic functional non-maps. It is therefore desirable to represent the syntactic functional mappings across the source and target languages as an approximation of its underlying complex linguistic structure. This allows for generalisations in the more stable positions (replace the whole syntactic function with a variable) to improve coverage. A word level correspondence is necessary to maintain if the divergence is only describable at a sub-constituent level. We conclude on this basis that, for EBMT, the desiderata for a suitable representation of bilingual data are the following:

- Allow for similarity assessment of SL and SL′ above the word-level
- Detect SL′↔TL′ correspondances at the syntactic-functional level, for adaptation of structurally divergent sentence-pairs
- Allow SL′↔TL′ correspondances at a word level for adaptation of lexical changes

In the next part of the thesis, a flexible case-representation scheme is described which was designed specifically to deal with the issues involved in translating such "real" data via a careful generalisation strategy. The language independent scheme represents bilingual data at three levels: the sentence

level, syntactic functional level, and the word level. We will subsequently argue that this granularity of representation is coarse enough to capture many linguistic generalisations and fine enough to allow links to be established between non isomorphic SL and TL structures along which adaptation information can be propagated.

Part II

Case Creation and Storage

Chapter 4

A CBR Approach to EBMT

4.1 Abstract

Our system achieves its generalisation templates by investigating which surface level links exist between SL′ and TL′, as determined by the set of previous inter-sentential links contained in the case base. Not only are links between words and word chunks recorded, but also the non-links. That is, the template which describes an SL′ sentence in the case-base, will crucially contain information on the degree to which it linked to the TL′. Non-mapped words are retained in their surface form whereas well-mapped words or groups of words are abstracted by variable-replacement, with the syntactic function and linear ordering of the original words acting as the only index.

4.2 Introduction

In this chapter we present our general architecture for EBMT and its main data structures. Our approach to EBMT sits firmly in the realm of CBR where well-defined symbolic retrieval and adaptation procedures are the order of the day. Moreover, it is an adaptation-guided system in that the retrieval and storage of cases takes their potential adaptability into account. There are three principal characteristics of our system which make it different from other EBMT systems. These

are namely:

- Cases based on off-line sub-sentential links (Section 4.4)
- Step-wise Generalisation instead of Fragmentation (Chapter 5)
- Adaptation Guided Retrieval (Chapter 6)

Most reuse approaches to MT use a sentence-aligned corpus as the primary source of knowledge. However string-to-string alignments are too coarse-grained to allow for flexible and reliable matching of the input problem and example-problem specifications. Even the "purest" of EBMT systems either attempt some form of word alignment between SL and TL′ examples to account for differences that arise on comparison with the input specification, or they use an outside bilingual dictionary. The preferred approach for most MBMT (Chapter 2) researchers is to perform alignment at run time [Bro97] [BF95] [MJS94] when the matching occurs. This makes sense if the corpus is huge and most sentences will never be reused. However, in keeping with a more case-based reasoning approach, we argue for the off-line linkage of the case base for three reasons. Firstly, this represents a valuable means of restricting the search for adaptable cases (see Chapter 6). Secondly, once sentences are matched, the information remains in the case base and does not have to be computed every successive time the matcher retrieves the template. In some domains, particularly localisation domains, certain sentences recur frequently. Thirdly, the subsentential links approximate a domain model for translation, which is used for assessing and performing adaptability, assessing similarity and linking. Section 4.5 demonstrates how a flexible contextualised dictionary function can be defined on this data-set, while the linker and retriever are described in Chapters 5 and 6 respectively. The added flexibility of cases counterbalances their inevitably smaller number in comparison to MBMT.

The second characteristic is one of performance, namely ReVerb's stance on the generalisation versus fragmentation question. While case decomposition at run time is the standard means of overcoming the coverage problem as deployed in [SN90] [MJS94] and [Bro97], the resulting solution is unpredictable as fragments originating in many different contexts meet up unharmoniously in a

new string; a stumbling block become known in EBMT circles as the *boundary friction* problem [MJS94]. The second means of overcoming coverage deficiencies is to create generalised templates [KKM92] [CC96] [CC97] [Car97] which match a variety of strings by noting some features in common which stem from a representation more abstract the string itself. A typical heuristic is to replace *terms* or nouns, which arguably behave predictably across languages, with a variable thus allowing the example to match with any string, at that position. We present a novel case representation scheme which supports a step-wise generalisation of templates for matching. Instead of simply making certain linguistic types (e.g nouns) in the sentence invisible by variabilisation, we associate each part with a score indicating the strength of the link to the target. By means of thresholding, this score will determine whether a "chunk" will be replaced by a variable for the matcher or not on the grounds of concrete linking evidence. Those chunks that are less adaptable according to the linker will be forced to match with the input at the string level. We show how the system architecture supports such a scheme in this chapter.

The remainder of this chapter proceeds as follows. In Section 4.3 we provide a view of the basic architecture of the ReVerb system. Section 4.4 describes the case representation scheme in terms of the levels of linguistic information stored in the translation examples themselves, and their overall organisation in memory. Section 4.5 shows how a bilingual dictionary can be defined on this data and the way in which the lookup procedure can be parameterised for different tasks (e.g. for adaptation or for helping alignment). The template generalisation scheme is presented in Section 4.6 and the chapter conclusions appear in Section 4.7.

4.3 System Overview

ReVerb's functionality can be described in terms of its two main tasks — which are offline and online (or runtime), as depicted in Figure 4.1. The offline task is to learn reusable cases from sentence-aligned bilingual corpora. This can be sub-divided into a bootstrapping stage and a fully automatic case-learning stage which includes case-based parsing and linking (Chapter 5). At runtime, ReVerb

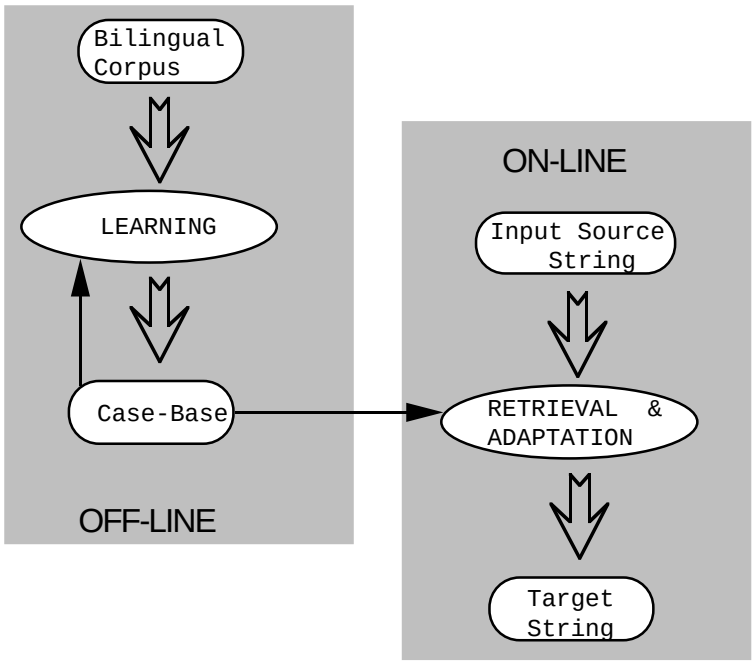


Figure 4.1: *Overview of the* ReVerb *system architecture.*

performs adaptation-guided retrieval on the cases learned in the learning stage and adapts the selected candidate translation example(s) to produce a target solution. Both runtime procedures are described in Chapter 6. This chapter is devoted to presenting the actual structure of the knowledge that cases embody and their organisation in memory.

4.4 Case Representation in ReVerb

Cases are the sole repository of information for all the functions in the ReVerb EBMT cycle. A retrieved case's SL′↔TL′ mapping description is mirrored during translation (Chapter 6). The SL′ components of all cases are used for parsing the new input sentence and for providing chunk-boundary information during linking (Chpater 5). For adaptation and linking purposes, a bi-lingual dictionary is required. This dictionary function (see Section 4.5) can be deduced from the SL′↔TL′ knowledge spread across all the cases. In this section, a general description of case content is given first, and then the motivation for certain aspects of this representation is given by referring to the

various processes which depend on the case-base. To support the flexible reuse of translation cases for translating from SL to TL[1] a case must represent the following knowledge:

- SL′ problem specification -The SL′ "side" of a case
- TL′ target specification - the TL′ "side" of a case
- links between SL′ and TL′

ReVerb uses a frame-based representation for its cases (Figure 4.2). Each case represents an SL′ → TL′ sentence pairing and acts as a pointer to all its component chunks which contain the linguistic properties of the substring pairs. Each "chunk" is a syntactic function somewhere between the pre-terminal level and the top nodes of a traditional parse tree. A chunk is annotated with the sub-strings' syntactic-functions in the SL′ and TL′ sentences respectively. Cross-linguistic divergences and general non-correspondence in translation mean that links are given an approximated score based on quantitative dictionary correlation and qualitative comparison of their respective syntactic functionalities.

This looks deceptively like the chunks are being considered to be compositional elements of the SL′ and TL′ sentences, but as will soon be demonstrated in Chapter 5, compositionality is only assumed when chunks receive a good "adaptability" score. Furthermore, this compositionality, which is determined with respect to a new input problem (see Chapter 6), restricts how the target specification is pieced together at run time in ReVerb. That is, ReVerb adheres to the notion that recombining the chunks after adaptation can only guarantee a valid solution TL if the policy of adaptation guided templatisation and retrieval have been followed.

The Case Representation Frame

At the topmost level, a case representation frame specifies what the source and target languages are, for example, German, Irish, etc. This is depicted in the `CASE-1` representation frame in Figure

[1]The labels SL and TL are actually arbitrary and used for clarity. The case representation scheme is unbiased towards direction. A given case SL′ ↔TL′ can equally translate in the TL→SL as well as the SL→TL direction without any modification.

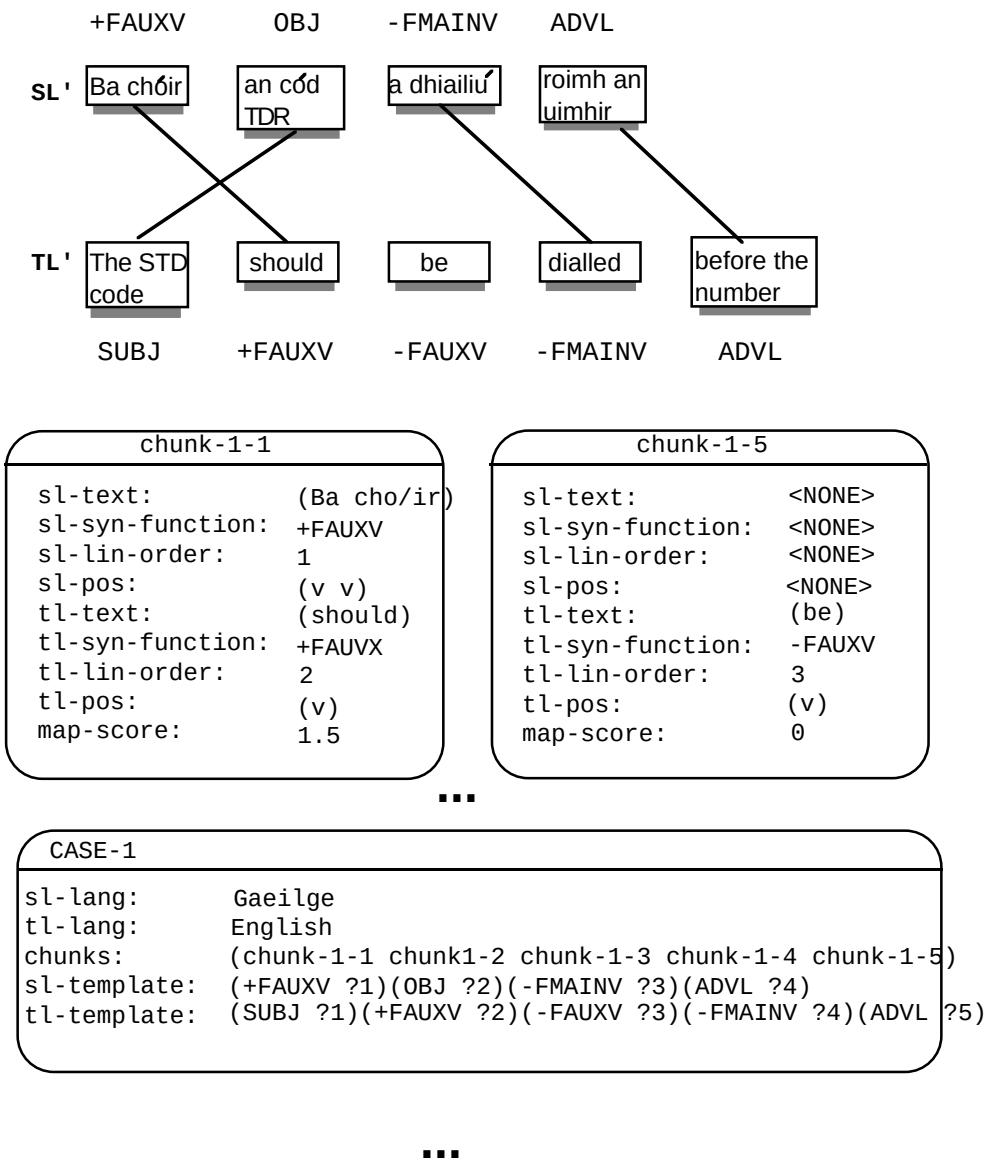


Figure 4.2: *A case-frame and chunk-frames in* ReVerb.
This corresponds to a linked Irish-English sentence pair from the 'Telecom' Corpus.

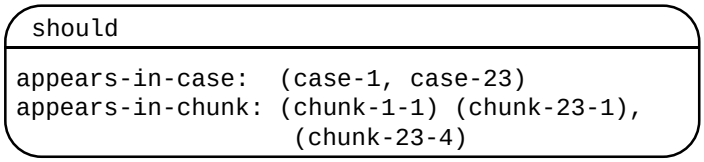


Figure 4.3: *A* ReVerb *word frame*

4.2). The identities of the case's component chunks, and a templatised version of the SL′ and TL′ sentences it represents, based on mapping scores, are also provided.

Chunk Representation Frame

A chunk representation frame provides all the linguistic information required to describe aligned sub-components of SL′ and TL′. If a chunk on either "side" did not align to the opposite language then that side of the chunk will be given the value <none> for all features, and the adaptability score is set to zero. Figure 4.2 shows a sample sentence alignment in an Irish-English corpus along with the chunk frames which describe it.

The frame-based language in which the cases are represented supports (multiple) inheritance among frames. Chunks inherit global information, such as the identity of SL′ and TL′, from their mother case. The local slots are chunk-specific and denote the linear order, syntactic functionality, POS information and the actual string of the SL′ and-or TL′ chunk(s). There is very little typing, or restrictions of any kind, on the values save to say that `linear-order` can only be a positive natural number, and `map-score`, a non-negative integer. The frames can thus be used to store any form of bitext mappings not necessarily linguistic in nature. It is assumed for this application however that the `syn-function` slots receives a single value which is a valid syntactic function (see Appendix C.1.1 for the full range) and that the `pos` values are a subset of the POS set described in Appendix C.1.2. Appendix B shows a full example of a case. The `adaptability` value denotes the strength of the link between SL′ and TL′ units. This is zero if either side is absent.

Word frames

Each individual word type receives a separate WORD object. This is a frame indexing a list of the cases and chunks which contain an occurrence of the word type. The WORD frame structure is shown in Figure 4.3. The ID of a WORD frame is the word token itself. The `appears-in-case` and `appears-in-chunk` slots contain at least one CASE-ID and CHUNK-ID respectively (otherwise they wouldn't have been created). As previously mentioned, entering a chunk containing an unknown word will trigger the creation of a WORD frame. So, for every word in the system there is a record of where it appears in the data and from this, one can ascertain its neighbouring words, the syntactic function of the chunk which contains it, and its left and right context extended to the sentence extremities if required. The only information that is not accessible via the WORD object is the identity of the previous or following sentence.[2]

4.4.1 Demonology

A demon[3] is a program that is not invoked explicitly, but lies dormant waiting for some condition(s) to occur. Demons are often used at the operating systems level, e.g. for device drivers. Certain demons hang off frame slots waiting to perform some action as soon as a slot's value is modified. For example, when a chunk's `source-text` or `target-text` slot is set, a WORD demon is activated which creates a new object for that word type, or if the word type is already present in the system, this will be updated by the demon as shown below. In general, demons are responsible for preserving the consistency of the expanding case base as it grows during the learning phase.

[2]Case-id's are treated here as being arbitrary, though case-naming according to their ordering in the original text could be imposed if required, in order to provide extra-sentential context.

[3]The term *demon or* daemon was introduced to computing by people working on the CTSS (Compatible Time-Sharing System) project which was an early (1963) experiment in the design of interactive time-sharing operating systems, ancestral to Multics, Unix, and ITS. The name ITS (Incompatible Time-sharing System) was a hack on CTSS, meant both as a joke and to express some basic differences in philosophy about the way I/O services should be presented to user programs.

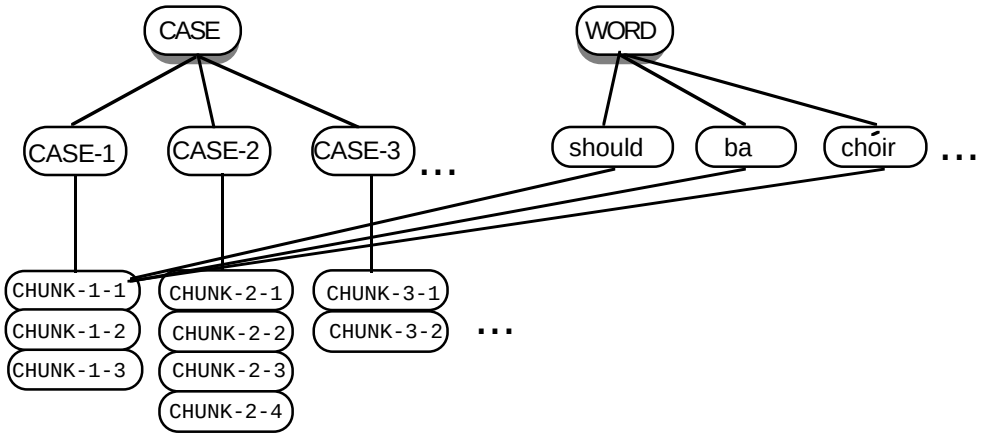


Figure 4.4: *A view of* ReVerb*'s memory organisation*
Links are created by demons between cases and their chunks, and between word-frames and cases (chunks). Top-level objects `WORD` *and* `CASE` *keep track of all words and cases currently in the system.*

4.4.2 Overall Memory Organisation

Demons ensure that the entire case base is updated each time a new case is entered. This in effect only means updating WORD objects with pointers to the new context of their word types, that is the chunk and hence the case which contains one or more occurrences of it. A higher level CASE-object frame maintains a list of case-ids of all cases in the system so this is updated also. A small section of the memory is shown in Figure 4.4.

4.5 The System Dictionary

This scheme for organising memory supports the creation of a dynamic model of word to word translation equivalence, in other words, a *dictionary*. Every new case addition causes WORD objects to be updated with links to a new context of use. On the monolingual side, the system is receiving more information on the context of the particular word, in terms of chunk regions and sentence regions. This will be useful for parsing (Chapter 5) On the bilingual side, the word is associated with a new opposite language chunk and sentence. Therefore, the higher the frequency of the word, the more reliable is its list of proposed translation candidates.

The algorithm for proposing the translation candidate for a particular word SLw, is shown below.

We assume that SL is the source language.

`dict(SLw,SL-ID,TL-ID,n,SPEC-LIST,CB)`

Inputs:

`SLw`: *The source word from SL*

`SL-ID`: *The identity of the SL, e.g. English*

`TL-ID`: *The identity of the TL, e.g. German*

`n` : *Desired length of translation candidates*

`CB` : *The case base*

`SPEC-LIST` : *Extra restrictions*

Outputs:

`TL` : *list of length* $\leq n$

`procedure dict`

begin

```
1.  CHUNK-LIST ← get-chunks(SLw)
2.  until empty(CHUNK-LIST)
3.  do    CURRENT-CHUNK ← first(CHUNK-LIST)
4.        CURRENT-TL-WORDS ← get-tl-text(Current-chunk)
5.        TL-WORD-LIST ← CURRENT-TL-WORDS
6.  enduntil
7.  TL ← sort(TL-WORD-LIST)
8.  return(first-n(TL)
```

If the `SPEC-LIST`, is empty, `TL-WORD-LIST` will contain all words contained in TL′ sides of chunks which are connected to an SL′ chunk containing that word. That is, there is no *one-to-one* assumption about words within linked chunks. The *sort* function sorts the `TL-WORD-LIST` according to relative word frequency, moving the most frequent towards the head of the list. Some

examples of unrestricted word lookups are shown below.[4]

a) `dict('the', EN, GE, 4, []) = (der die das des)`

b) `dict('das', GE, EN, 2, []) = (the this)`

c) `dict('the', EN, IR, 2, []) = (an na)`

Explicit probability scores for each correspondence using the heuristic that the probability of a particular word, s, being the translation equivalent of t is proportional to the ranking of s in the dictionary's output, as shown in (4.1) below. Here P does not mean conditional probability in the strict sense, for the sum of the probabilities of all events conditional on a given event should be 1 which is not the case here[5]

$$P(s \mid t) = \frac{1}{rank(s, Dict_{TL}(t))}; P(t \mid s) = \frac{1}{rank(t, Dict_{SL}(s))} \tag{4.1}$$

$$Score(s,t) = \frac{P(s \mid t) + P(t \mid s)}{2} \tag{4.2}$$

In the example above, the word occurrences are restricted to the chunk level. Thus for a) above the probabilities would be:

$$P(der \mid the) = \frac{1}{1}$$

$$P(das \mid the) = \frac{1}{3}$$

[4]Here a) and b) are based on a German-English aligned corpus of size 750 sentences. c) is based on an Irish-English corpus of size 120 sentences.

[5]There are other possibilities for calculating the probability, which could use probability scoring in this strict way, but the metric given above was the one which proved most useful in experiments on linking. One suggestion (by Michael Carl) is to use the following:

for all i $< rank_{max}(dict(b))$

$$P(a_i \mid b) = \frac{1}{2^{rank(i,dict(b))}}$$

for all i $= rank_{max}(dict(b))$

$$P(a_i \mid b) = \frac{1}{2^{rank(i,dict(b))-1}}$$

$$P(des \mid the) = \frac{1}{4}$$

$$P(the \mid der) = \frac{1}{1}$$

The formula for calculating the probability that a word pair stands in a translation equivalence relation is calculated by summing the probability in each direction (4.2) and dividing by 2. This joint probability, denoted *Score*, is always less than or equal to 1. If we wish to restrict the dictionary *lookup* we do so by adding information to the `CONTEXT-LIST` list. For example, if we wish to retrieve correspondences for SL′ from TL′ but wish to restrict the words retrieved to being of a certain part of speech then we put the symbol for that part of speech in the `CONTEXT-LIST` list. The algorithm mirrors that of **dict** without restrictions except for the addition of an extra line:

```
5.  CURRENT-TL-WORDS = check-spec(CURRENT-CHUNK,TL,CONTEXT-LIST)
```

This has the effect of only putting those TL words into the candidate translation list which satisfy some requirement. This is a crucial functionality of the dictionary as will become obvious in Chapter 6 when adaptation is discussed. Adaptation requires highly contextualised lookups. In fact, myriad restrictions are possible on the dictionary function but here a sample few of the parameters which proved most useful for particular system tasks are described.

Chunk-level Dictionary for Adaptation

For chunk-level replacement, (see Section 6.3.4) words are preferably chosen to substitute TL′ elements which appear in a similar context, and are linked to the same SL′ word(s). Every chunk containing the SL chunk in a certain syntactic function `CONTEXT-SL` must also be linked to an TL chunk of a certain function `CONTEXT-TL` in order to be an optimal adaptation.

adapt `(SL-CHUNK,SL-ID,TL-ID,n,CONTEXT-LIST)`

forall *s* **in** `SL-CHUNK`

```
      CHUNK-LIST ← find(s,SL,TL,n,CONTEXT-LIST)
      COMMON-LIST ← CHUNK-LIST ∩ COMMON-LIST
find-most-freq(get-tl-text(COMMON-LIST))
```

Here is an example:

```
adapt(``the mouse'',EN,GE,(EN:SUBJ,GE:ADVL)) =
find(``the'',EN,GE,4,(EN:SUBJ,GE:ADVL))
∩ find(``mouse'',EN,GE,4,(EN:SUBJ,GE:ADVL))

find-most-freq(get-tl-text (CHUNK-3-2,CHUNK-5-6,CHUNK-24-9)...)

=find-most-freq((auf der maus)(der maus)(von der maus)
                          (auf der maus) (mit der maus) ...)
= (auf der maus)
```

As their names suggest, the *find-most-freq* procedure gathers all chunks containing the SL words specified, in the SL↔TL context specified, and *get-tl-text* gathers the text strings associated with these chunks; the chunks themselves are indexed on the basis of all SL words in the adapter chunk, i.e. "the" and "mouse".

4.5.1 Non-linked words

Sometimes a WORD object points to chunks containing no translation equivalents for the word. This omission may be due to a translation error, a mis-alignment, or a genuine ϵ-production[6]in the cross-linguistic transformation. If the latter, then the symbol <NONE> is likely to be the most frequently

[6] A common *epsilon production* is the disappearance of the SUBJECT under passivisation: "This command saves your drawing" → "Die Zeichnung wird gespeichert" (The Drawing is saved).

occurring equivalent for certain words. For example, the desired translation for the reflexive particle "sich" in a German to English mapping is a blank.

1. *Die Objekte bewegen* SICH *am oberen Rand*

2. *The objects move (*THEMSELVES*) to the uppermost edge*

`dict('sich', GE, EN, 3, []) → (<NONE>, it, them)`

4.6 Template Creation

At matching time an input problem specification and an example problem specification will match if all their chunks are similar, or, the chunks have been generalised in those positions where the input happens to differ. Therefore generalisation should occur in those positions where problem-example differences can be felicitously transported to the problem solution where a reciprocal adaptation is performed in the corresponding position. REVERB's heuristic in determining such "safe" adaptation sites is based on the following two metrics:

- translation equivalence probability between the words on the SL′ and TL′ side of the chunk.
- functional equivalence on either side of a chunk.

The intuition is that if two groups of words perform the same function in their respective sentences and if all their meaning is self-contained in the lexical items then *adapting the set of words in one language will have the same effect in the other.*

The following sections are devoted to explaining the processes involved in measuring these equivalences. The equivalence of syntactic function is assessed simply by looking at the relevant slots in the chunk's frame representation. This is a Boolean measure of similarity — either identical or not so. The second equivalence, a cross-linguistic measure, is determined by the strength of word-to-word translation equivalence and some fanout heuristics as described in section 5.4. The combined scores produce a base adaptability measure for each chunk of a case, which is stored as a template

and which will ultimately determine whether the case is reusable for the specific input problem at retrieval time.

4.6.1 A Careful Generalisation Strategy

If the domain of the case base is a sublanguage (e.g. weather bulletins, avalanche warning reports), and the input is from the same genre, then one would expect reasonable coverage even when the example templates retain some degree of specificity. In certain restricted domains, the probability of certain "shapes" of sentences reoccurring is high enough to allow the luxury of "careful" generalisation strategies based on the intuition described above. As the current domain of application is the translation of technical documentation and thus relatively well-behaved, the emphasis here is too on careful generalisation. In ReVerb 'generalisation' is a restricted operation whereby surface lexical details are made invisible. Here, all linguistic information is bunched together on an equal basis in the chunks. No modularity between any levels of linguistic description is assumed. Generalisation in both the input and the examples is simply the wiping out of the chunks' surface details (i.e. the words), whose specificity might otherwise prevent a match.

4.6.2 Coverage versus Accuracy

The more variables a SL template contains, the more input sentences it will be able to match. At retrieval time (run-time), a *threshold of adaptability* is set, above which only chunks of a certain mappability score may be variablised. This blanks out word information, and therefore any implicit linguistic constraints that their presence ensured (e.g. number, case and gender agreement). The justification lies in the idea that well-linked chunks are usually parts of a sentence that are replaceable by chunks of similart syntactic functionality. Setting a threshold of adaptability will determine the range of cases that the system is allowed to retrieve for a given input. If the case base is large, the threshold may be set quite high. If a certain input construction is mirrored frequently in the corpus the threshold can be raised even higher, ensuring with some certainty that only adaptable templates will be retrieved. This is good news for translators of highly repetitive text.

In REVERB, the granularity of a case is potentially at two levels -at the syntactic functional description which the parser decided upon during linking (for the examples) and during run-time (for the input), and at the word level (via WORD objects). The criterion in AGR is to retrieve examples which are adaptable with respect to the knowledge available to the system and we wish to demonstrate how our policy of careful template generalisation supports this. Firstly it is safer to adapt entire chunks than it is to poke around inside the chunks adapting single words. The most reliable adaptation is that of replacing an entire chunk with another from the same context. The adaptability score is biased towards a reliable "clean" chunk mapping at the lexical level not just similar syntactic functionality. For example the verb **to remove** in English could be translated as **das Entfernen** (*the removal*) in German via nominalisation:

This command enables you **to remove** the last object.
Dieser Befehl ermöglicht **das Entfernen** des letzten Objekts.
This Command allows the removal of-the last Object

```
dict('the', EN, GE, 4, []) = (der die das des)
```

This pairing recieves a high score by virtue of there not being any other words in the SL′ or TL′ (Chapter 5) which have the same lexical characteristics as the verb **remove**. Even though **the** corresponds to two chunks in the TL′, one either side of the word **Entfernen**, there is very little possibility that there will be any confusion in the linking pattern, as the **the** in the SL maps so well to the **des** in the last chunk of the TL′. Thus the *LexScore* (Chapter 5) is high. If the syntactic functions of the pair are different then when it comes to adapting the chunk in the SL′ at run rime, this difference is noted and a suitable adaptation SL′ TL′ pair is chosen which reflects a similar divergence in syntactic function (i.e. a pair with a Finite Main Verb in SL′ and Subject in TL′). On the other hand, if the lexical information in the chunk of one language gets interspersed among various chunks in the TL′, then the chunk link decided for that group of words is given a low

score. The overall map-score which appears in the `map-score` value position of the CHUNK frame is calculated on the basis of its lexical score (Chapter 5) and the similarity of the syntactic functions (Boolean value), as shown below:

$$MapScore = SynFunScore + LexScore \tag{4.3}$$

$$SunFunScore = \begin{cases} 1 & \text{if synfun(SL) = synfun(TL),} \\ 0 & \text{otherwise.} \end{cases} \tag{4.4}$$

If however the chunk link is poor, that is, several words are missing from the mirror chunk, it could mean that some transformation has scattered the words across different chunks, making an entire substring unadaptable, starting at the point where the first word of the chunk containing the non-mapping words occurs and ending at the last word of the chunk containing the rightmost non-mapping word(s). Indeed a well-mapping chunk may occur in the middle of this dangerous area.

(The last object) (you) (select) is placed on the upper layer.
(Das zuletzt markierte Objekt) wird auf der obersten Ebene positioniert.
(The last selected object) is on the uppermost layer positioned.

Here **you** is missing in the TL′. Surrounding **you** in the SL′, are words which map to the first chunk in the TL′ only. So the TL′'s words are scattered among two chunks in the SL′, and this would get a poor score as a result. The score is calculated by dividing the dictionary determined lexical score (Section 5.3.1) by the cardinality of the larger matching chunk. In the example above, the singleton chunks **you** and **select** get a score of zero. and **The last object** receives a score of its *LexScore* divided by 4.

4.7 Conclusion

The memory organsiation presented here allows for the flow of information between WORDS, CHUNKS and `cases`, thus providing the basis for dictionary creation and the structured data necessary to deduce links between new sentence pairs. The dictionary is highly customisable to each task in the EBMT process — linking, adaptability assessment and adaptation itself. The careful generalisation strategy, introduced here as a concept, is solely based on the linking data which is calculated in the algorithms which are the subject of the next chapter. The test of the validity of the careful generalisation scheme which supports an adaptation guided retrieval policy, is evaluated in terms of accuracy and coverage in Chapter 6.

Chapter 5

Case Creation and Learning

5.1 Abstract

Cases are created via a process of parsing both sides of a bilingual corpus and linking the corresponding chunks between the two halves. As with most empirical approaches to multilingual NLP, the first step in linking is to detect sets of corresponding word tokens in the two halves of a bitext. This is followed by a novel technique of superimposing chunk-boundary information on this word-mapping in order to determine one-to-one chunk mapping patterns desired for case creation. These two tasks are performed by the ReVerb Linker. Chunk-boundary information is supplied by ReVerb's data-oriented Parser which uses only the cases as its knowledge source. This parsing algorithm is described. Both the ReVerb Linker and ReVerb Parser are testably effective, even when the training corpus is less than 1,000 cases.

5.2 Introduction

Bitexts are texts which express the same meaning in two different languages. They provide the raw material for many corpus-based and EBMT systems. In a direct fashion, translation models can be derived by statistically comparing tokens across texts [BCDP+88] [Mel96a]. Also, tools such as

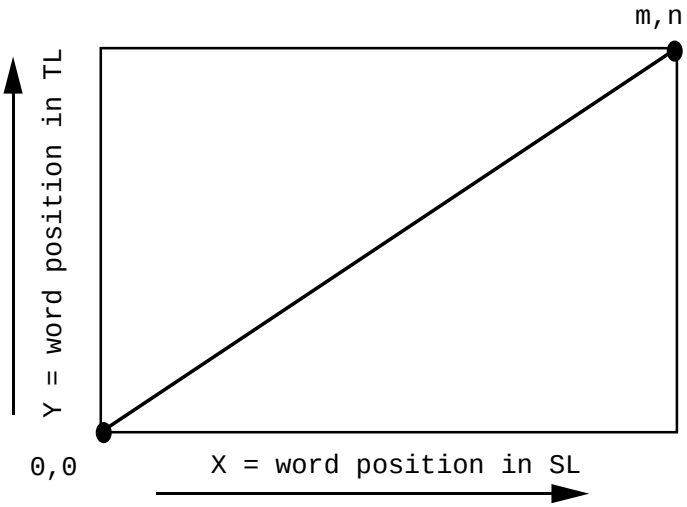


Figure 5.1: *A bitext space*

translation memories, concordancers and lexicons which help the (human or machine) translator can be defined on the parallel data. Bitext correspondence can be expressed at various levels of granularity to provide a picture of how two languages match up in terms of the relative positioning of their corresponding elements. For two languages with similar alphabets, the word level is a useful first step in determining translational equivalence. Every bitext defines a rectangular space as shown in Figure 5.1. The lower left corner is the origin, representing the beginning of both texts, and the upper right corner represents the terminus of both. The X and Y axes are the lengths of the two texts in terms of the units under comparison (e.g. character, word, sentence). Most of the previous work in bitext mapping has focused on sentential alignment as this is the starting point of statistical model derivation.

The main diagonal from origin to finish has a certain angle. For some closely related language pairs, the points of correspondence between sections of text will create a diagonal line with an angle close to that of the main diagonal, indicating similar relative positioning of tokens. Some approaches [Mel96a] exploit this quality of bitexts to limit the search space or to determine omissions. It is rare for two languages to have the same tokens in the same order so there is usually some deviation in the angle of the bitext map.

5.3 The Linker

The ReVerb Linker assumes that each sentence in the bitext has been aligned on a one to one basis[1]. Hence, linking proceeds at the word level. ReVerb builds a chain of correspondence points for each sentence in two stages. In the first stage, generation, all points of correspondence are plotted in the bitext space. Correspondence is detected by a matching predicate for words which uses the system dictionary. This decides if two words are mutual translations. If the words match, their respective positions in the SL and TL sentences are plotted as an *(i,j)* co-ordinate. A heuristic of map-uniqueness is used to filter out some of the noise. There are no word order assumptions[2]; a word match any distance from the main diagonal in the rectangular bitext map has the same *apriori* probability.

The second stage of linking sees a coarser grained chunk-level map being superimposed on the word-level map. As the sentence pair has already been parsed by the system, each word token will occur with the boundary of some chunk. the combined word-links in a given chunk space determine the overall one-to-one chunk linking pattern. The ReVerb Linker is a general pattern-matching algorithm which in this application happens to be operating on words, and chunks of words. The dictionary (Chapter 4) is the sole information source for the matching predicate upon which the Linker depends. This dictionary is automatically updated each time a new case enters the system after Linking. This cyclic, symbiotic process allows the dictionary and the Linker to become more accurate as the case-base increases.

5.3.1 Linker Step One: Point Selection

The system dictionary is likely to have more than one translation equivalent for a given word, i, say. Let $Dict_{SL}(i)$[3] be the word set derived from looking up i. If one of the words in $Dict_{SL}(i)$ is

[1] This assumption is not unrealistically optimistic in software localisation domains, where translators are often instructed to adhere, as far as stylistically possible, to the sentential structure of the original text. See Section 3.3.2 for more details on this assumption.

[2] Additional heuristics exploiting English and German's relatively similar word-order were tested in initial experiments, but these were ultimately abandoned due to their unattractive language dependency.

[3] Here, the mnemonics $dict_{SL}$ and $dict_{TL}$ are used for the functions `dict(x,source,target,n,[])` and `dict(x,target,source,n,[])` respectively. See Section 4.5 for more details.

identical to a word in the TL, then a tentative link is made between i and its assumed counterpart j, say. At this word level, the Linker does not assume a one-to-one mapping however and it will try a list of n possible translations of i against the TL, creating links where applicable. Each link is given a score based on the ranking of the corresponding TL word in the dictionary's list (see Figure 5.2). The Linker then proceeds to word $i+1$ in the SL, and repeats the process until the end of the SL string is reached. The whole process is repeated in the opposite direction. Ultimately, a good lexical link between the word pair i and j occurs if $i \in dict_{SL}(j)$ and $j \in dict_{SL}(i)$. The point selection algorithm proceeds as follows:

dotimes i in SL,

dotimes j in TL,

if $match(i,j)$ **or** $match(j,i)$

then $plot(Score(i,j))$

The matching predicate $match(i,j)$ is defined as follows:

$$match(i,j) = \begin{cases} True & \text{if } Score(i,j) > 0, \\ False & \text{otherwise.} \end{cases} \tag{5.1}$$

The linking is fully bi-directional so even if the link is not obvious in one of the directions, a single link in the opposite direction will still capture the word pairing:

$$Score(i,j) = \frac{P(i \mid j) + P(j \mid i)}{2} \tag{5.2}$$

$$P(i \mid j) = \frac{1}{rank(i, Dict_{TL}(j))} P(j \mid i) = \frac{1}{rank(j, Dict_{SL}(i))} \tag{5.3}$$

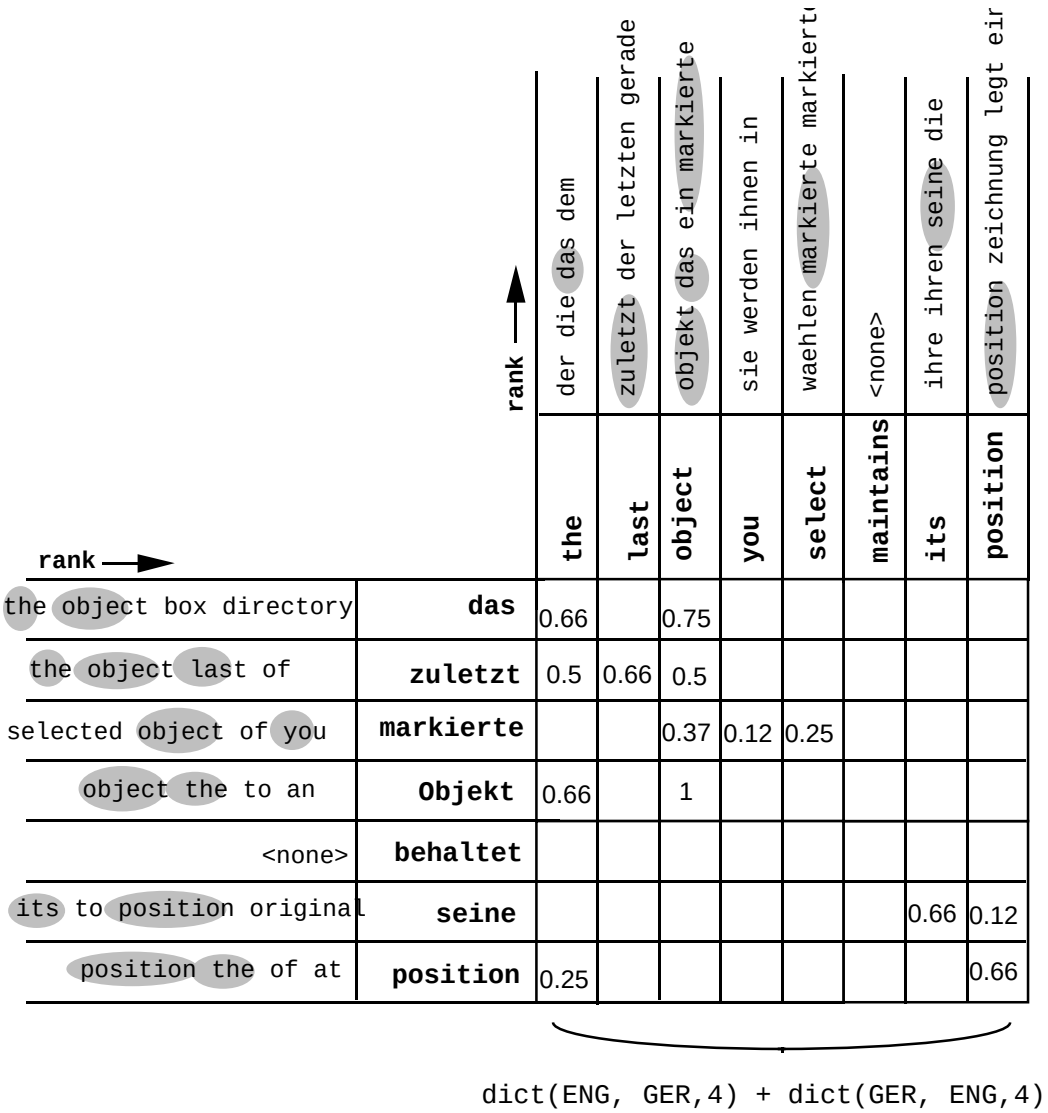


Figure 5.2: *Linking words from SL and TL in the bitext space.*
Linking is performed on the basis of dictionary scores (where n=4 in this example) in both directions.

Step Two: Chunk-based Alignment

Although individual word tokens rarely map in one-to-one fashion between SL and TL, their higher level constituent forms (syntactic functions) often do. The procedure at the chunk level is analogous to that for word alignment except that now the output required is a bijective mapping between the chunks of the two languages. The sentences are parsed by the ReVerb Parser, and the resulting chunks can be of variable size (see Figure 5.3). The linking pattern may contain crossings also, for example where a chunk at the beginning of SL corresponds to a chunk at the end of TL (or vice versa). Each chunk region $I_r \times J_s$ delimits a subset of word pairs from SL $\times$ TL and the lexical-scores for each word-pair contained in this space is summated and normalised by the number of words in the longest chunk, as shown in (5.4).

$$ChunkLex(I_r, J_s) = \frac{\sum_{i \in I_r} \sum_{j \in J_s} Score(i,j)}{max(size(I_r, J_s))} \tag{5.4}$$

The overall score of a chunk link also depends on its uniqueness. For each chunk pair (I_r, J_s), let $fanout(I_r)$ be the number of SL→TL matches for chunk I_r, and $fanout(J_s)$ be the number of TL→SL matches for chunk J_s. The overall score for the chunk link (I_r, J_s) is given in (5.5) below.

$$ChunkScore(I_r, J_s) = \frac{ChunkLex(I_r, J_s)}{fanout(I_r) + fanout(J_s)} \tag{5.5}$$

If two chunks in one language map to one in the other the highest scoring chunk wins out. The linking algorithm chooses the highest scoring chunk-pair for each I in SL and each J in TL. This "winner takes all" chunk linking algorithm is as follows:

until empty(MATRIX)

dotimes I_r in SL

dotimes J_s in TL

WIN-SL ← get-highest-link(I_r)

WIN-TL ← get-highest-link(J_s)

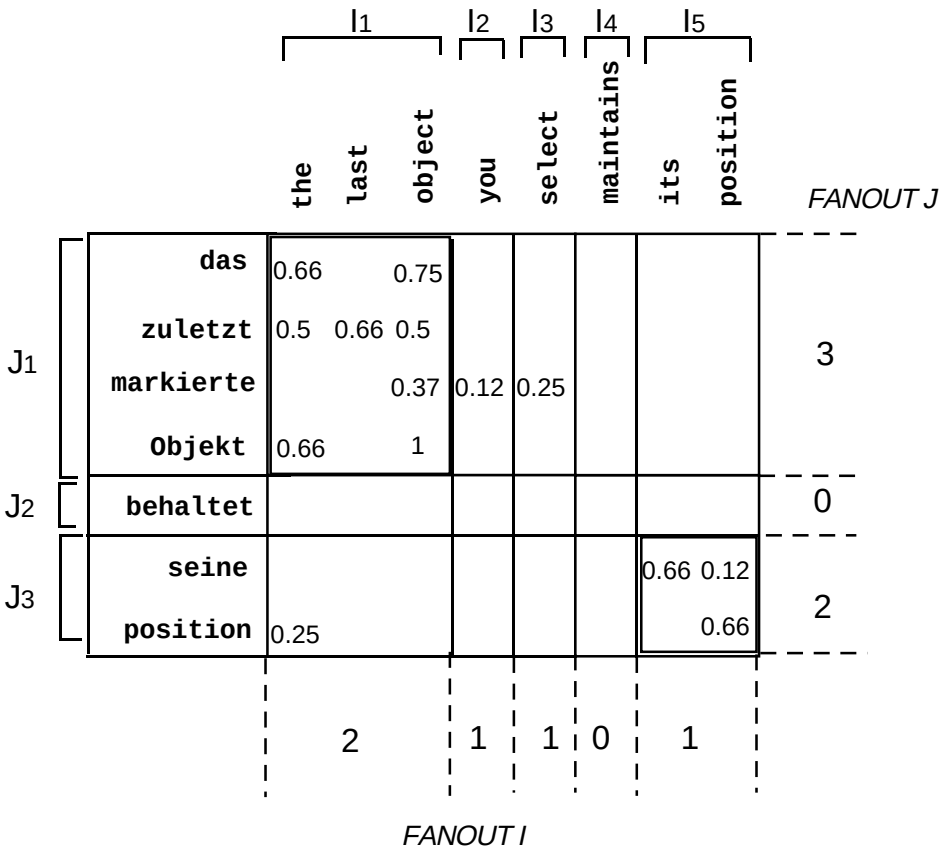


Figure 5.3: *Linking parsed chunks of the SL and TL in the bitext space. This is performed taking into account the word-linking scores of all words contained in a given chunk and the fanout of the chunk with respect to the opposite language.*

if WIN-SL = WIN-TL

WINNER ← WIN-SL

WMATRIX ← plot(WINNER, WMATRIX)

MATRIX ← delete-all(WINNER,MATRIX)

Any rival chunks are deleted from the bitext map, that is, in subsequent calculations, no points with I_r as an x co-ordinate or J_s as a y co-ordinate will be regarded as being potential candidates for mapping. This may allow loosers to win out on the next iteration. The algorithm terminates when all points have been eliminated from the start matrix, MATRIX. The winner matrix, W-MATRIX contains all the winning chunk pairs in a one-to-one mapping. The potential one-to-oneness of the initial MATRIX, depends on a number of factors: the extent of linguistic or stylistic divergence and translation noise, the quality of the data upon which the dictionary is defined, and the parameter n of the dictionary matching predicate. For any one of these reasons, some chunks in either language may not be mapped. In the sense that ReVerb cannot distinguish between the source of the non-map, that is, translational divergences and mismatches (Chapter 1), they are all treated the same and receive a score of zero. The output of the aligner is therefore a set of chunk pairings in the $SL \times TL$ space. This is the raw input for the case-building component.

5.3.2 Previous work on subsentential alignment

In general, sentences are an easy starting point for alignment because they usually appear in the same order in a translation (but see Section 3.3.2), and early bitext mapping concentrated on sentential alignment, see [KR93] or [GC93] or [BLM91]. Bitext mapping algorithms at the word-level usually assume that the texts have been pre-aligned at the sentence level. However some approaches do not even assume this much. Melamed [Mel98] reports on a portable algorithm for mapping bitext correspondence called SIMR which does not use sentence boundary marking as a clue to alignments. His matching predicate works on raw text and detects common *cognates* -words that appear with the same surface or phonological form in both languages due to a similar etymology, borrowing,

etc. A matching predicate based on characters is reported in [MEO95] who use the longest common subsequence ratio as a similarity metric and [KG97] describe how to find phonetic cognates even in languages with different alphabets.

5.4 Case-Based Parsing

The Parser has two functions in the ReVerb system. It provides the Linker with the necessary chunk boundary information, as just described. It also parses the input problem sentence at run-time (Chapter 6). The data-oriented algorithm is described below.

5.4.1 Introduction

The case base contains structural knowledge about both languages because each side of a translation pair in the case-base represents the result of monolingual parsing to a flat-dependency level. The SL side of each case is information that can be reused in order to produce a most likely parse of the input SL sentence. Even if the system is faced with a new word, the Parser gives a good pattern-matching guess as to where it should stand in relation to its neighbours, on the basis of all previous instances of that word token in the data. The fact that it is parsing to a flat-dependency structure means that the procedure can be reduced to a simple "include in this chunk or not" -type decision for each word, because at this shallow level of description, there are only three levels to be concerned with -the case-level (i.e. the entire sentence), the chunk-level, and the word-level.

5.4.2 Activating cases via word objects

ReVerb maintains a WORD object for every word type in the case-base. Demons (see Chapter 3) update this object by recording the case and chunk which contains every new occurrence of the same word token. It is worthwhile to note here the difference between parsing and translating. In parsing, the task is simply to cover the sentence with the most probable sequence of syntactic function chunks for those words in that particular order. Although cases are being re-used in parsing, the TL

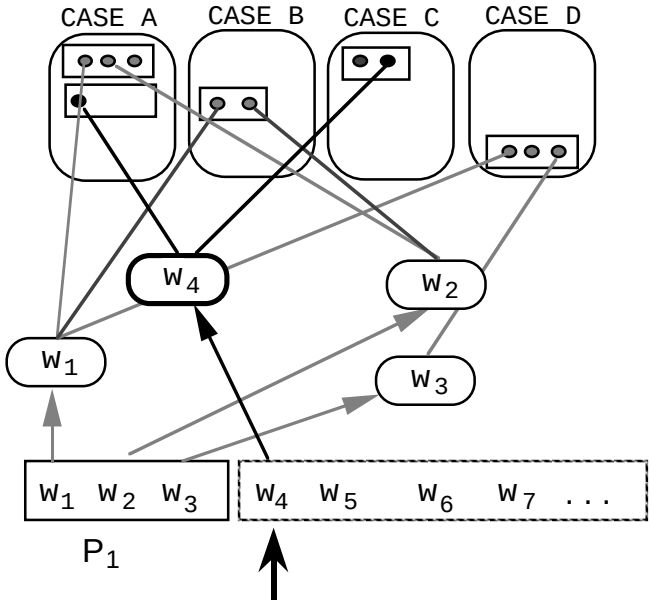


Figure 5.4: *Parsing in* ReVerb
Determining chunk boundaries in the input string.

information is ignored. Translation on the other hand requires not only coverage of the input sentence but also that the target language chunks hanging off the covering SL example chunks will actually combine smoothly, which may involve additions, deletions and reordering of words. So parsing is reduced to the task of producing the most likely series of chunks which cover the input string. The input string then assumes this chunking pattern and all syntactic information annotated onto each covering chunk (syntactic function and POS info). The algorithm is deterministic and proceeds in bottom-up fashion. On encountering the next word, the parser activates the corresponding WORD object. This in turn beckons all chunks which are linked to it. In Figure 5.4, a typical snapshot of a parsing session is depicted. The current word is w_4, and the previous words $w_1 ... w_3$ have collectively activated cases A, B, C and D but only CASE-A has managed to cover the string entirely up to w_4 so it is the natural choice to pursue.

5.4.3 Chopping and Glueing Chunks

Optimally, some case will exist that contains all the input words, because then it is quite likely that the input words will be in a similar grammatical relationship. The parser activates cases whose chunks contain words which coincide with the string so far and as it moves from left to right across the string, the list of chunks which are still valid will trail off. Once a word is encountered for which the set of chunks containing both it and the previous sub-string is empty, then case fragmentation is necessary, as depicted in Figure 5.5. Case fragmentation in this application simply involves choosing a new chunk, which may or may not entail a new case (one could continue using a case by skipping over a non-matching chunk), from which to derive information concerning the current input word. However, before the old chunk is forgotten, the parser decides whether or not the current parse chunk should be extended to include the current word or not. This will depend on whether the case chunk had nearly been completely "used-up" to cover the string to this point, or only partially. Preferably, the next chunk will originate from a case which already covered part of the input string. The following algorithm describes the parser's decision process when a word in the input string is different from the word in the appropriate position of the chunk being followed:

1. Match the input word w_i against the next word

   in the currently activated chunk CH_j from the case-base

   if success include w_i in current parse-chunk, P_k

   move rightwards in CH_j

   move rightwards in P_k

   else if CH_j is used-up, Store P_k

   Remember $CASE_{CH_j}$ which contains CH_j, proceed to 2

   else use statistics to decide whether to abandon or chop P_k

2. Index new set of chunks for w_{i+1}, giving preference to CH_{j+i}.

   Proceed to Step 1.

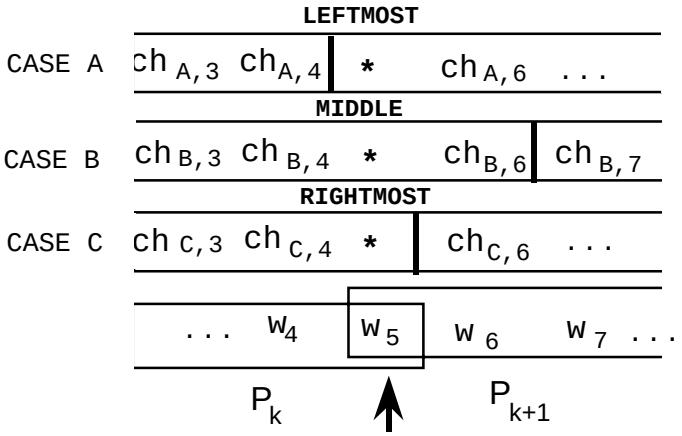


Figure 5.5: *Three possible chunk positions for the non-matching word* w_5

5.4.4 Statistical positioning of words

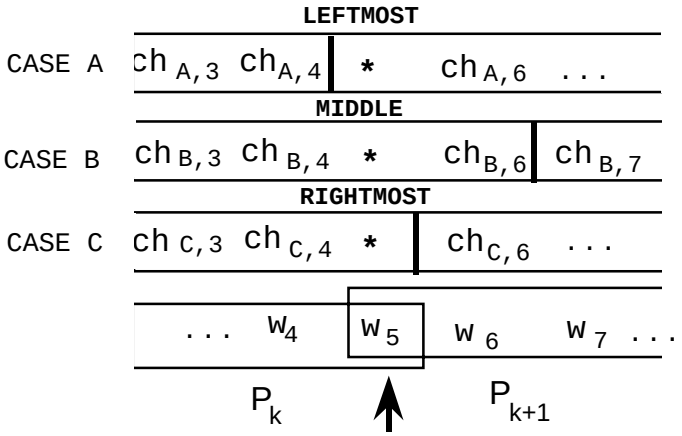
The notion of modifying chunks to fit the data in the input string may seem ad-hoc but statistics can be used to increase the likelihood of a reasonable chopping point of a chunk. ReVerb's data representation scheme allows the calculation of every word's statistically most-probable positioning within a chunk, based on all its occurrences [4]in the data. The data in Table 5.1 indicates the typical positioning in their respective chunks of the 20 most frequent words in a case-base of size 800. For example, determining the *right-peripherality* of a word involves activating all case-chunks which contain the word of the input chunk, and counting the times it appears in the rightmost position in each chunk. This is expressed as a fraction of the total appearances of the word in the case-base. If the pre-defined threshold is reached, the chunk is deemed right-peripheral. A high right-peripheral followed by a high left-peripheral probability score indicates a reliable cutting point. In English, determiners and infinitive markers words are very typically left-peripheral, and common nouns and infinitival verbs are right peripheral. Words which have closely scoring left and

[4]This calculation does not take polysemy into account, so all similar word tokens, e.g. *points*(verb) and *points*(noun) are presumed to be occurrences of the same word type.

right counts are either typical *middles* (e.g. adjectives) or one-word constituents, called *singletons* (e.g. verbs, and co-ordinators like "if"). Note that the chunks reflect the word-groupings which appear in the case base, which are determined by linking patterns between source and target, and are not absolute mono-linguistically derived constituents. In other words, the positioning is data-dependent; a word considered to be *left-peripheral* in one data-set and one language-pair may be typically *right-peripheral* in another domain, and/or for another target language.

5.4.5 Overcoming Boundary Friction

With the aid of the statistical positioning data, the parser now has a better chance of making the correct decision for any word which deviates from the example and there are two dimensions along which to decide how to "glue" chunks from different cases together - the position of the word in the chunk which covered the previous substring, and the typical placement of such a word in this data set. Figure 5.6 summarises this decision process. The three columns indicate the departure point in the case chunk, i.e., where the case- and input strings begin to deviate, whether in a chunk-initial position, in the middle or at the end of the case chunk. The four rows distinguish actions taken when w_i is statistically a *singleton, left-peripheral, middle,* or *right-peripheral*, (see Table 5.1) respectively. If a word deviates at chunk-initial position, it means that either a more-rightmost chunk from the same case must be used, or a chunk from an entirely new case. Regardless of whether the input word is statistically leftmost or not, the parser trusts the structure it followed from the previous case, and does not try to extend the previous parse chunk. Hence the uniform decision in the first column to seek a completely new chunk. The situation is different, however, when the parser has committed at least one word to a certain chunk, and then discovers that a word doesn't match. It can decide either to ignore the fact that the words are different and include the deviant word into the current parse chunk anyway *include*, or it can abandon the current chunk *restart* in the search for a new chunk which can cover the string from the current point onwards. The *include* decision is taken when the word is typically a middle, and the decision to chop the previous chunk is taken when the word is typically rightmost.



Statistics	*Actions for each Position of w_i in Example Chunk*		
	Leftmost	Middle	Rightmost
Singleton	activate w_i	chop, activate w_i	chop, activate w_i
Leftmost	activate w_i	chop, activate w_i	chop, activate w_i
Middle	activate w_i	include w_i	include w_i, chop
Rightmost	activate w_i	include w_i, chop	include w_i
Unknown	activate w_{i+1}	include w_i	include w_i, chop

Figure 5.6: *Deciding how to chunk the input sentence.*
The diagram at the top indicates the three potential sites of deviation between the input word w_5 and a case from the case-base. In the table below, general word inclusion decisions are based on the deviation site (X axis) and the corpus-based statistical positioning (Y axis).

5.4.6 New words

In the case of new words, the parser has no access to statistical data and will make the include or chop decision purely on the basis of its position in the chunk. In practice, this method could result in over-long chunks for words which have low frequencies, say one or two occurrences, which might easily have occurred atypically in non-right-peripheral positions but which in this context really should mark a chunk boundary. To address this problem, the parser has an additional check for uncharacteristically long chunks. It calculates the average lengths of all case-chunks which feature the words in the chunk so far. If the chunk length exceeds this average value the chunk is cut at the position of the word, leaving the new word as a singleton.

Word	*Freq*	*L*	*R*	*L%*	*R%*	*Verdict*
THE	996	574	0	58%	0%	*leftmost*
TO	369	346	6	94%	1%	*leftmost*
IN	298	119	8	40%	3%	*middle*
YOU	285	274	281	96%	99%	*singleton*
A	226	154	0	68%	0	*leftmost*
OF	199	44	14	22%	7%	*middle*
AND	185	168	169	91%	91%	*singleton*
OBJECT	129	5	116	4%	90%	*rightmost*
FILE	123	14	80	11%	65%	*rightmost*
OR	116	107	107	92%	92%	*singleton*
ON	109	91	8	83%	7%	*leftmost*
YOUR	103	41	10	40%	10%	*middle*
WITH	102	94	9	92%	8%	*leftmost*
FOR	97	88	5	91%	5%	*leftmost*
OBJECTS	89	28	73	31%	82%	*rightmost*
IS	84	83	80	99%	95%	*singleton*
IF	84	82	84	98%	100%	*singleton*
BOX	83	5	77	6%	93%	*rightmost*
DRAWING	83	6	72	7%	87%	*rightmost*
PRINT	80	40	34	50%	42%	*leftmost*
COMMAND	75	11	48	15%	4%	*middle*

Table 5.1: *Statistics to help glue-together cases.*
The positioning information depicts typical positions of English words calculated on the basis of 750 cases. The 20-most frequent words are shown.

5.4.7 An example

In Figure 5.7, the parser pursued `CASE-A` until it came to the word *file* in the input string which fails to match *centre* from the case. It has already committed to two words of this chunk, namely *in* and *the*, and thus uses statistical data to determine the next move. The data shows *file* to be right-peripheral, so according to the decision process outline above, the decision is to include *file* into the chunk. The parser then moves on to the next word *with* and abandons the current case chunk, in favour for any chunks that begin with the word *with*. It will prefer any chunks of this form that `CASE-A` has to offer first, and then search the list of cases which coincided with the previous substring albeit not as directly as `CASE-A`. For example, it would choose `CASE-B` to cover *with the other options*, by virtue of the fact that it contained some of the string so far, namely *the dialog box*.

```
INPUT: (The dialog box) (will) (appear) (in the file) (with the other options)
CASE-A: (The dialog box) (will) (appear)(in the centre of the screen)
CASE-B: (The dialog box) (gives) (information) (on how to print) (with the other
options).
```

Figure 5.7: *A sample 2-case coverage of the input string*

5.5 Evaluation

The performance of the Linker and Parser were assessed separately on a training case-base size of 800 cases taken from the *CorelDRAW* v6 manual. These originated in 10 different help files, describing functionalities of that drawing package such as Document Editing, File Management, and particluar drawing functionalities. This case-base was bootstrapped using outside knowledge sources (Section 5.3) for the first 200 cases and the remaining 600 were subsequently acquired via a process of automatic parsing, and post-editing. The test data for the parser consisted of 100 English sentences from the *CorelDRAW* v7 manual which were taken from a different section (Colour Commands) than that of the test data. For the Linker, the same test corpus was used; this time both the English and German texts were parsed beforehand. In a fully-automatic system configuration, the Linker relies heavily on the Parser's ability to determine correct boundaries. The overall system performance depends in turn on the Linker's ability to create re-usable, accurate cases.

5.5.1 Evaluation of The Parser

The algorithm's evaluation was performed by manual assessment of each parse result by a bilingual evaluator. The evaluation here was therefore restricted to 100 cases. For this experiment, the case-base was not updated with each new parsed sentence. This was in order to have a steady measure of the coverage of the training set.

Results

Each false inclusion or exclusion of word(s) was recorded. Also, each wrongly assigned syntactic function was noted, and unknown input. With an average of 9.2 chunks per sentence, this means that approximately one chunk in every sentence is askew, see Table 5.2 above.

ReVerb Parser Performance	
Av. chunk-splitting accuracy:	92.25%
Av. categorisation accuracy:	79.6%
Av. categorisation accuracy (exc. unknowns)	85%

Table 5.2: ReVerb *Parser Performance*

Discussion

Of the chunk categorisation errors the most typical at 20.8% was classifying FMAINVs as ADVLs, for example (to create ADVL) → (to create -FMAINV). This is due to "to" being interpreted as a preposition as in "to the screen" instead of as an infinitive marker. Other common mis-classifications (9.6%) were between subjects and objects, which is not surprising as these are not case marked in English. It can be concluded that parsing (chunk-splitting, and chunk categorisation) can be done on a pattern-matching basis, without applying linguistic knowledge at run-time, but that on average, one chunk in every sentence is wrong, more than likely due to incorrect labeling. The chunk-splitting accuracy is quite high which means that the linking algorithm will not suffer. The chunk-categorisation accuracy is lower however. As the parser is also used at run-time for retrieval, incorrect labeling of the input chunks may decrease the likelihood of finding a similar match.

5.5.2 Evaluation of The Linker

The Linker algorithm's evaluation was performed by manual assessment of each linking result by a bilingual evaluator. The input to the linking algorithm was a set of 100 error-free pre-parsed sentence pairs, which were not contained in the system. The output, a set of new cases, was assessed for chunk-linking accuracy between the SL and TL. This was as much a test of the linker's ability to deal with sparse data as a test of its link-selection algorithm. For the dictionary function, `dict(word,lang1,lang2,n,[])`, the length n of the word list was set to 4.

Results

Linking accuracy was calculated as the fraction of correct links of the total links which the linker created. Coverage was assessed in terms of the number of links divided by the total number of links

ReVerb Linker Performance	
Av. chunk-linking accuracy:	96%
Av. chunk-linking recall:	40.4%
Av. performance	72.9%

Table 5.3: ReVerb *Linker Performance*

which the human evaluator made. These scores are provided in Table 5.3 above.

Discussion

The linker performed extremely well once the data was available in the system dictionary, which would suggest that the link-selection algorithm is suitable for this task. Future experimentation with a larger case-base, and perhaps varying the parameter n, could improve the lower recall score. From a translation point of view, the 100 cases created automatically would be useful for translation as each was found to contain a certain number of matching chunks, albeit not all possible, some of which have a high lexical score (see Chapter 5). The cases which had 100% linking recall tended to be less than 7 words long, and cases with over 25 words (which made up over a quarter of the test corpus) were likely to have a low linking score, leveling out at around 30%. This would suggest breaking longer sentences into clauses if possible.

5.5.3 Bootstrapping

Bootstrapping the linker and parser is a necessary part of the system cycle in ReVerb, as with many other approaches to EBMT (see [VW97], [Bro97]). Some initial input of cases and supervised linking is necessary to create the system dictionary. The quality of off-line case creation gradually improves as the size and quality of the case-base increases. An incomplete dictionary will cause ReVerb to miss some potential links and a smaller case base results in less accurate chunk patterns, but there is no point at which the entire engine suddenly ceases to function. There is no definite point at which optimum reliability can be said to be reached but a workable level of dictionary accuracy was evident for the CorelDRAW corpus after linking 200 cases under supervision. One can take advantage of this gradual behaviour by building the knowledge source (the case base) incrementally

word	ENGCG-tag	ReVerb-tag	ReVerb chunk
he	@SUBJ	@SUBJ	(he @SUBJ)
is	@+FMAINV	@+FMAINV	(is @+FMAINV)
in	@ADVL	@ADVL	(in the car @ADVL)
the	@DN>	premodifier	
car	@<P	postmodifier	

Table 5.4: *Post-processing of chunk-inclusion decisions made by Lingsoft's* ENGCG *Output from this constraint grammar was reformatted in order to bootstrap* REVERB *by "flattening" any dependency structures, e.g. post- and pre-modifiers were placed in the same chunk as their head, and their modification-direction information subsequently lost.*

and using it for creating new cases even when the case-base size is small. In particular, by adding the post-edited cases back into the case-base after automatic parsing and linking, the system can be bootstrapped from a relatively modest case-base size.

English

In the case of English, 200 sentences were parsed first by the ENGCG grammar [Vou95] [VJ95] [KVHA95] using heuristics, to a syntactic functional description and then manually corrected. ENGCG proved roughly 90% accurate for this corpus, with the majority of the errors arising due to the presence of headings and domain-specific command names, for example "Copy Attributes From Command", which it treated as being several constituents rather than a single noun-phrase. The resulting lists of annotated words were then gathered into chunks by distinguishing between pre-modifiers, post-modifiers and non-modifiers. To highlight the post-processing performed on the ENGCG output, Table 5.4 shows a typical sentence.

German

There was no equivalent German parser available from Lingsoft at the time of training, however there was a lemmatiser GERTWOL, which output morphological information for the German input. The 800 sentences were passed through this module and subsequently parsed using the author's context-free grammar. The results were checked manually to ensure that the chunking was accurate.

5.5.4 Conclusion

The off-line procedures have been described which allow the ReVerb system to create its own cases. Bootstrapping provides the system with chunking information and cross-linguistic linking patterns. From this data, which may be any number of examples, ReVerb guesses new monolingual chunking patterns and links between languages. The linked chunks are scored on the basis of their common lexical similarity and syntactic functional similarity, and this information is stored in memory as a case-frame. The case-building procedures improve as more cases are created and it has been shown that reasonable performance can be expected even when the number of cases on which to base parsing and linking decisions is quite small. The next chapter describes an innovative approach to reusing these cases carefully during run-time, thus completing the ReVerb system description.

Part III

Adaptation-Guided Retrieval

Chapter 6

Retrieval and Adaptation

6.1 Introduction

Retrieval is the most important stage in any case-based reasoner and much effort has been devoted to this aspect alone. The general procedure is as follows. A system seeks to be reminded of a previous episode, call it SL′ →TL′ of problem solving, on the basis of salient features of the new problem task at hand, SL. This so-called "reminding" is a process of searching the retrieval space (Chapter 3) for a suitable case by promoting SL features which are deemed relevant to the problem solving process. These relevant features or *indices* may describe very low-level features of a problem, or they may be highly abstract. The reminding may even be performed on multiple levels of abstraction in an effort not to miss out on good matches. An observation, which is quickly becoming a standardly recognised issue, is that similarity between input and retrieved problem specifications does not necessarily guarantee that the solution TL can be constructed with ease. We have found this to be particularly true in the area of EBMT because the problem specifications are in one language and the solutions in another! The SL→ SL′ comparisons may not produce clear TL→ TL′ adaptation requirements when the description language in which to represent these analogies is not rich enough. This means that some cases which may seem similar to the problem at the SL′ side may not have a TL′ component which is also easy to adapt. The solution is to choose an adaptation-guided

approach, as motivated in Section 6.2.

When it comes to the actual adaptation itself, some CBR application domains are so well defined in terms of states, actions and goals that it is possible to store most of the knowledge required to perform even complex adaptation in an explicit model of the domain. It has been demonstrated by [HK96] that such adaptation rules can even be learned from the case base itself. For EBMT however there are simply too many domain rules to enumerate. Of course, the question of adaptation only arises if the system is expected to produce a final solution. If the EBMT engine is merely a component of a larger multi-engine MT system [Bro97], [Car97] then a standard MT engine can take the result of SL $\rightarrow$SL$'$ similarity-based retrieval and apply the necessary MT rules to produce a solution. However, in fully-automatic EBMT, the solution is heavily dependent on the retrieval of valid cases from which a quality solution can be derived by simple substitutions without having to acquire an explicit model of the domain. The only "model" of the domain should be an implicit model of translation spread across the entire case-base of previous translations. In our system we show that by limiting the amount of complex adaptation required, chunk-based substitutions provide good solutions without resorting to extra knowledge sources for adaptation

The major features of our run-time algorithm for fully-automatic EBMT are the following:

i. A pattern-matching similarity metric which prunes the case-base

ii. Adaptability ranking of retrieved cases to suggest best candidates

iii. Simple adaptation and further ranking of candidates

As the case base is not partitioned in any way beforehand, the run-time similarity metric (i) must be efficient at ciphering out the "good" cases from the entire case-base. To combat the needle in the haystack effect, we describe in Section 6.4 how features are promoted in an efficient manner. For ranking (ii), the trick is in finding the most specific example specification which matches the input problem specification especially in all its non-compositional positions. Section 6.3 describes the two types of adaptation knowledge required by the system. The first is *Adaptation-Safety knowledge* which predicts whether the necessary changes to a case are likely to render the solution

grammatical or not. This knowledge is a combination of the chunk-link strengths of the case and the similarity of the SL and SL′ strings. We describe the calculation of this information for full-case reuse and partial-case reuse in Sections 6.3.1 and 6.3.2 respectively. High-scoring cases will be preferred over low-scoring ones at retrieval time. The second form of adaptation knowledge is what we call *Adaptation-Availability knowledge*. For efficiency, this is only calculated after candidates have been chosen and it indicates the level of success the system had when adapting chunks with the only souce of information available to it the cases themselves, i.e. the system-dictionary. This gives the user a measure of confidence in the output translation which is desirable if the user is not proficient in the target language. In Section 6.4, the filtering stages of retrieval are described which test for Adaptation-Safety knowledge. Section 6.5 describes the actual adaptation of candidates and the subsequent finer-grained Adaptation-Availability assessment. This is where translation happens. Some examples of translation in ReVerb are presented in Section 6.6. Then, by way of evaluation, Section 6.7 presents a set of experiments translation. Finally, section 6.8 discusses these results.

6.2 Adaptation Guided Retrieval

To appreciate the difference between retrieval for similarity and adaptation-guided retrieval consider the following example. Figure 6.1 depicts typical matches that are computed during the retrieval stage between an input sentence and two cases from the case base. On similarity grounds alone, Case-A wins hands down because only one of its chunks does not match. The opposite is true for Case-B, which would seem to be the poorer candidate for translation. However if we look to the interlingual mapping patterns for both cases, Case-B now fares much better -it has good correspondances between all chunks indicating that a compositional transfer took place and hence that it should be relatively "safe" to adapt chunks of the solution where necessary. Case-A, the victim of linguistic divergences or lack of linking data, is less adaptable. The effect of modifying the chunk containing the verb "specify" is not determinable.

In general, cases will prove adaptable either if they have very similar surface-level descriptions to

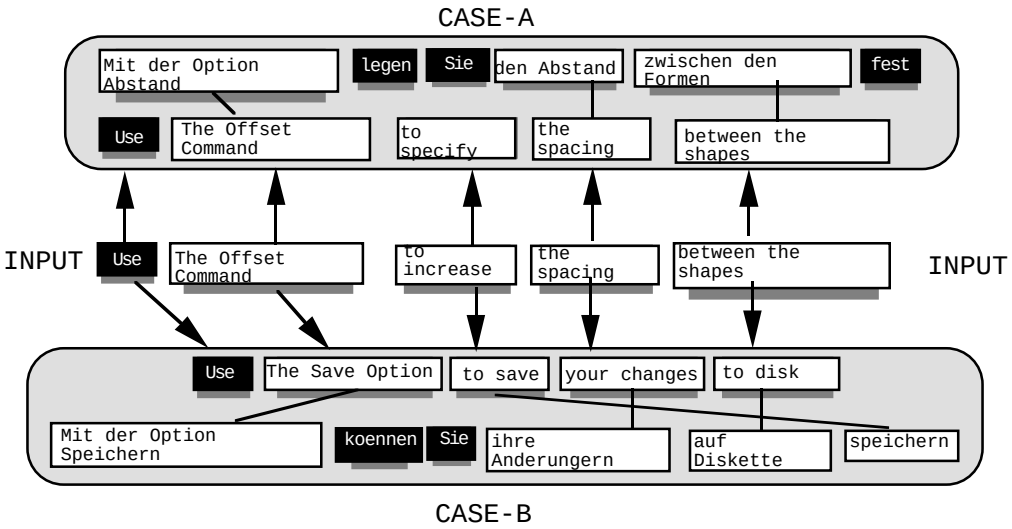


Figure 6.1: *Adaptability versus Similarity in retrieval.*

Case-A is more Similar to the Input but Case-B is more adaptable because all its chunks that differ from the input, are adaptable. In Case A, it is only necessary to adapt "to specify" but this has not linked well to the TL, and hence it is dangerous to reuse Case A.

the input specification thus requiring little or no adaptation, or if the case represents a very compositional translation, that is, lots of variabilised positions which allow for string differences, or both (the measures are not necessarily diametrically opposed). The close string-match scenario produces translations which mirror the "free translations" present in the corpus. The highly compositional case is more structure-preserving and is relying on the SL$\rightarrow$ TL mapping being describable as a compositional mapping such that adapted sub-parts of the solution can be re-assembled without boundary friction or violation of underlying linguistic relations.

6.3 Adaptation-Safety Knowledge (Links)

Adaptation-safety knowledge quantifies the risk involved in choosing a particular case (SL$'$ $\leftrightarrow$TL$'$) given that the SL $\rightarrow$ SL$'$ differences must be percolated across these SL$'$ $\rightarrow$TL$'$ links. It is related to the compositionality of the solution. The chunk linking scores are the key to compositionality and hence, adaptability. The template storage mechanism described in Chapter 4.6 already enforces non-compositional components to show their surface details on a template, so in a full-case matching

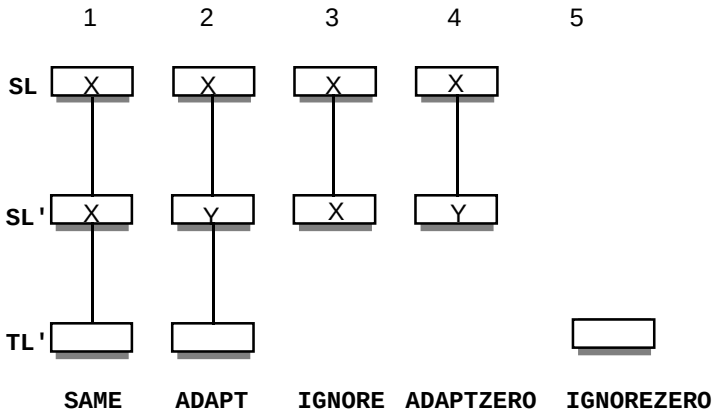


Figure 6.2: *The five possible scenarios in the SL → SL′ → TL′ interface for full-case matching. Operations Ignore and AdaptZero will only arise if the Adaptability Threshold is set to 0. At a threshold of above 0, the input will not unify with zero-mapping SL′ chunks.*

scenario, simple unification ensures that the example specification will not unify with the input string in these positions unless the surface details are exactly matching or are deemed similar enough to allow the equivalent target solution to remain un-adapted. If a case only matches partially then extra information has to be calculated as described in Section 6.3.2 below.

6.3.1 Full-Case Adaptation Safety Knowledge

When a problem is fully covered by a case then the target solution is created by substitutions alone. The only possible structure-changing operations are `Adapt` or `AdaptZero` as shown in Figure 6.2.

To ensure a basic level of "safety", a threshold is set before the retriever starts to work, such that only those chunks above this level will have their words variabilised thus allowing a new problem to unify. Figure 6.3 shows the attempted unification of the input string with the example at different levels of threshold-determined adaptability. This simple thresholding scheme ensures that, at given levels of adaptability, certain templates will not unify with the input specification. This in effect means that, in order to replicate the problem solution, the adapter will not be asked to percolate changes to TL via unreliable links. Raising the level of adaptability required increases the specificity of the problem specification and is analogous to increasing the level of similarity except that this time the similarity is more than skin deep — it looks beneath the surface at the compositionality of

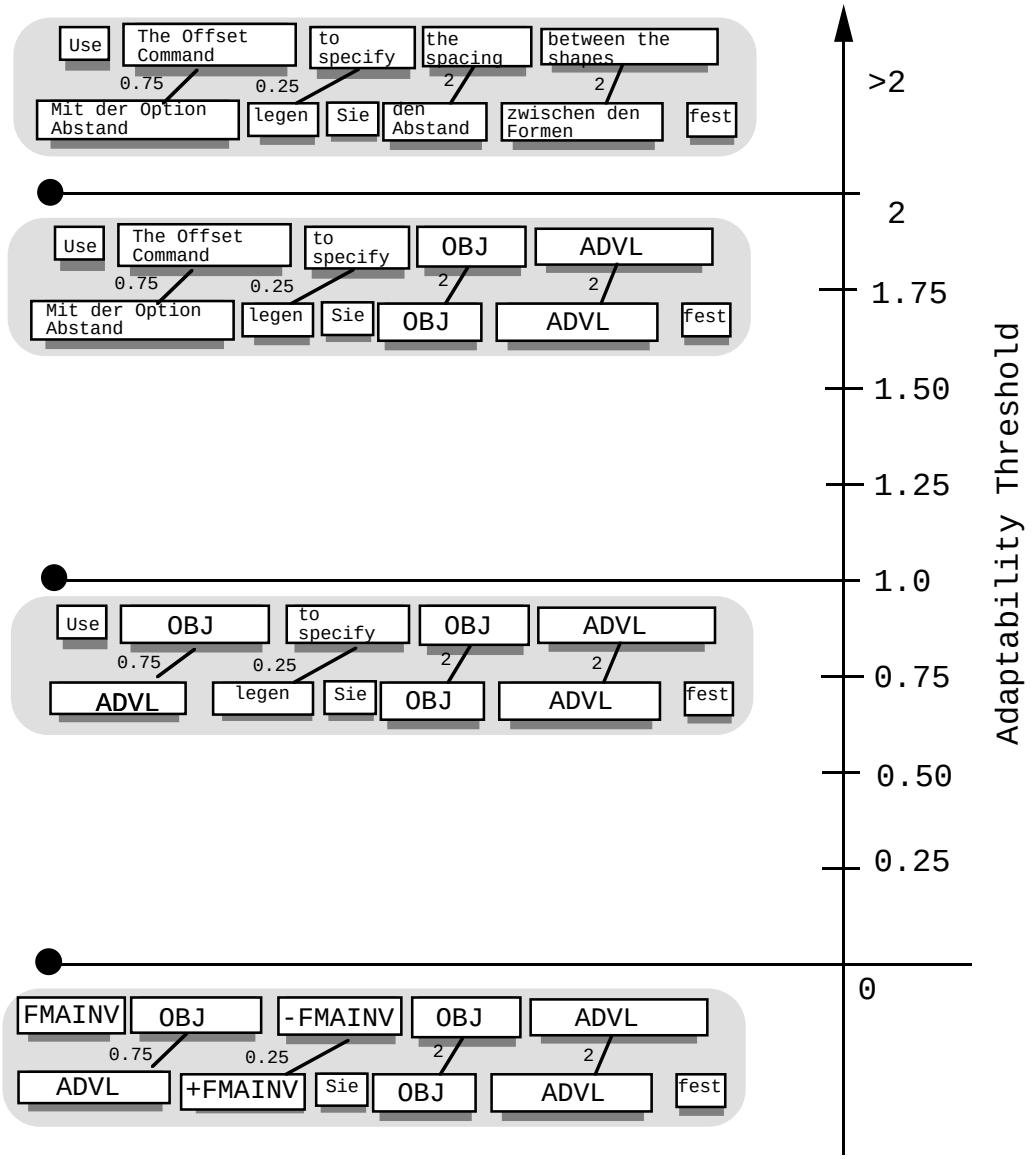


Figure 6.3: *Abstraction of a Case at different Adaptability Thresholds.*

If the threshold is lower than the link-score of a particular chunk, then it can be variabilised.

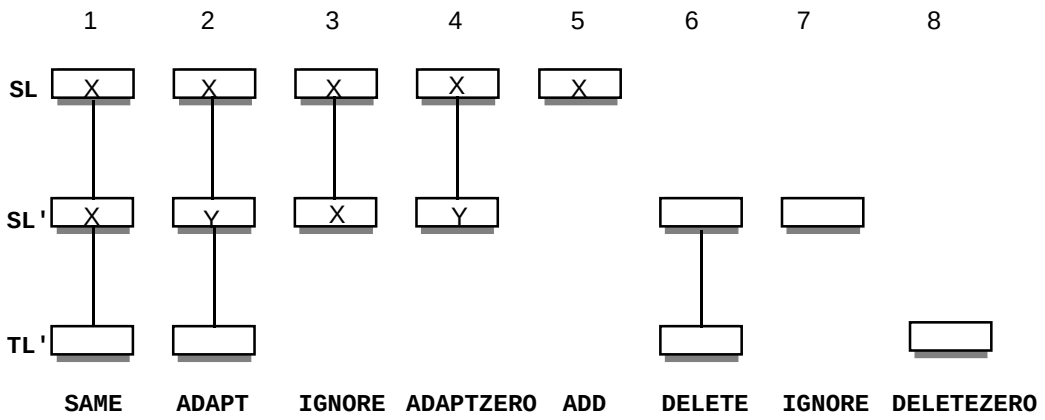


Figure 6.4: *The 8 possible matching scenarios in the SL→ SL′ → TL′ interface when partial case matching is permitted.*

the links to the target specification.

6.3.2 Partial-Case Adaptation Safety Knowledge

If a single, unifying case does not exist then the process of translation becomes an even more extreme form of "adaptation" than is required when the two problem specifications actually unify. Now there will be some SL chunks which have no SL′ counterparts let alone adaptable ones! In fact, with the bijective chunk mapping patterns, there are eight possible scenarios as depicted in Figure 6.4.

Situations 1 to 3 (and optionally, 4) can occur in a whole-case scenario, i.e, with unification. Situations 5—7 do not arise in whole-case reuse because they involve some deviation between SL and SL′. Situation 8 can also arise in full-case matching but there it is called `IgnoreZero`. This is because in full-case matching, the TL′ non-link is likely to still be part of the solution, whereas in partial-case matching, the non-link may occur in a portion of the TL′ whose SL′ equivalent does not even cover the input string, SL. In 5, The input chunk has no counterpart in the case and so the chunk must be somehow `Add`ed to the target solution but this time without guidance from the case. In both 6 and 7, the case problem specification contains a chunk which is not in the input. The SL′ chunk in 6 has a counterpart in the TL′ solution which must be `Delete`d. The SL′ chunk in 7 has no obvious counterpart in the solution, and it will simply be ignored, but this is less desirable as it indicates that

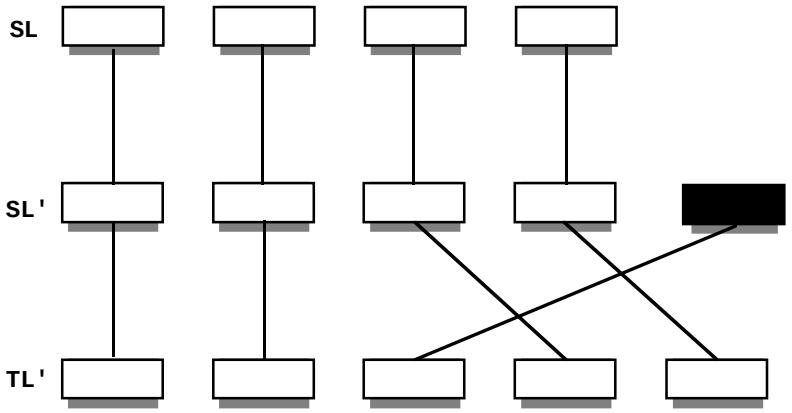


Figure 6.5: *Delete Operation in partial-case matching.*

This is dangerous, due to "crossing". ReVerb penalises such Delete operations highly.

some complex transfer took place which escapes description by the chunk links and that the elements may still be present in the TL′ which link invisibly to this SL chunk. Finally, from the TL′ point of view, there may be non maps, as shown in 8. Deleting these, using `DeleteZero` may upset the target's grammaticality but including them may be including extra information not required by the input which confuses the translation. We have developed meta-adaptation heuristics which delete such chunks if they are obvious content words and leave them there if they are "helper" function words. The distinction is based purely on frequencies of non-mapping words in the dictionary and not on any outside knowledge.

It will now be shown why some of these adaptations are dangerous. In Figure 6.5 below, we wish to delete the extra shaded chunk in SL′ which corresponds to nothing in the input, yet due to crossing, this effects the resultant target's canonical ordering. This problem is dealt with in many systems by imposing a crossing constraint, disallowing this sort of SL→SL′ matching. In the methodologies of [KKM92, BF95, Wu95] for example, the TL′ would be split into three fragments. ReVerb on the other hand, allows it, but at a penalty higher than normal for a `Delete` operation.

Thus the overall assessment of Adaptability is calculated by summing all the Deletions, Additions, AdaptZeros, DeleteZeros, and dividing the total number of good matches (Adapts or Sames) by this

number, as shown in equation (6.1)

$$\frac{Adapt + Same}{Adaptzero + Deletezero + Delete + DeleteCrossing * 2 + Add} \tag{6.1}$$

This is a simple metric which has many adjustable parameters such as the penalty weighting for each operation. In all the following experiments the weightings were kept equal, except for Deletions over Crossings, which were doubly penalised. Ignores are not penalised. The adaptability assessment for an input and case pair usually yields a score between 0.2 and 1.

6.3.3 Chunk-level Adaptation Knowledge (Dictionary)

Adapting chunks is performed in the same manner regardless of whether the solution is whole-case or partial. While REVERB's Adaptation-Safety knowledge is created cheaply via unification of the input template with the source specification template, the second type of knowledge is case-base dependent and more expensive to compute. Therefore this knowledge is used only to assess the best n matches from the adaptation guided retrieval stage. This could also be seen as the final stage in translation before the translation is output to the user. At this stage the adapter, which could equally be called "the translator" is endeavouring to choose an SL-chunk which not only matches on the SL side but which also matches on the TL side. For instance, given the input sentence SL and the SL′ TL′ below, :

SL ..., allowing you to restore an object **to its original location**

SL′ ..., allowing you to restore an object **to its original size**.

TL′ ..., so dass Sie **die Originalgrosse** eines Objektes wiederherstellen kann.

..., so that you the original-size of-an object restore can

It is wiser to replace the pair:

(**to its original size** ADVL) → (**die Originalgrosse** OBJ)

with:

(**to its original location** ADVL) → (**der Originalplatz** OBJ)

than it is to replace it with:

(**to its original location** ADVL)→ (**zu seinem OriginalPlatz** ADVL)

The actual directive to the dictionary is:

```
dict(to its original location, SL:ADVL, TL:OBJ)
```

The resulting TL strings would be as follows, where the first is grammatical and meaningful in German whereas the second is dubious:

ADVL → OBJ

So dass Sie (**den Originalplatz**) eines Objektes wiederherstellen können.

so that you (the original-location) of-an object restore can.

ADVL → ADVL

? So dass Sie (**zu seinem Originalplatz**) eines Objektes wiederherstellen können.

so that you (to its original-location) of-an object restore can.

TL words from chunks of similar syntactic functionality[1] are preferred for replacing adaptation-needy words in the TL case, with a subsequent relaxing of this constraint if no such items are found. The heuristics in Figure 6.6 show the decision process. The parameters which have been relaxed (are not required to match) in the dictionary are highlighted in **boldface** with a corresponding adaptability score on the right hand side:

Each constraint relaxation for the dictionary lookup procedure has an associated penalty as summarised in Figure 6.6. Failing a match under these circumstances, the next preference is to find a chunk containing a similar, albeit unequal, string of words and with the same syntactic functions in SL and TL, as in the chosen case to adapt. The first place to look for such a chunk is in the record

[1] ReVerb is capable of using the surrounding lexical content and syntactic functionality to determine the best chunk to use for substitutions. However our data-set was not large enough in our tests to render this option beneficial.

Dictionary Lookup Constraints	Adaptability Score
dict(I-String, SL-String, SL-SynFun, TL-SynFun)	1
dict(I-String, SL-String, **SL-SynFun**, TL-SynFun)	0.5
dict(I-String, SL-String, SL-SynFun, **TL-SynFun**)	0.5
dict(I-String, SL-String, **SL-SynFun**, **TL-SynFun**)	0.25

Figure 6.6: *Dictionary-based substitutions with various degrees of constraint relaxation.*

The constraints are indicated in bold such that these can differ from the desired values. Each constraint relaxation has an associated Adaptation-Availability score, which decreases as constraints on the dictionary lookup are relaxed.

of parsing, which indicates the chunk originally used to assign syntactic functionality to the input words at that position. The parse trace will indicate the best chunk match for this position in the input string, and it is this which is used as the basis for "keyhole" adaptation.

6.3.4 Chunk-internal Adaptation Knowledge

Keyhole adaptation is adaptation of the internals of a chunk. This is a form of decomposition at a lower level than the chunk-linking description and can have undesirable effects if chunks of an already low adaptability are altered. Often the poor linkage of a chunk is due to *1-to-n* word mapping which in turn is a consequence of lexical gaps (Chapter 3) in either of the languages. For example, the English verb *weld* maps to two separate chunks in German *miteinander* (*together*) and *verschmelzen* (*melt*). The latter chunk receives the link to *weld* on account of it appearing more frequently in its context, whereas the former is left as a stray chunk in TL. Altering **weld** to say **delete** in the adaptation fails to delete the extra dependent modifier *miteinander*.

You can **weld** objects with this option.

You can **delete** objects with this option.

Mit dieser Option können Sie Objekte miteinander **verschmelzen**.

Mit dieser Option können Sie Objekte miteinander **löschen.**

With this option can you objects together weld/ delete

However, often the adaptation of a chunk only has a chunk-internal effect. This means that all the words inside each chunk map only to the chunk to which they are linked. In this domain, chunk internal adaptation is often merely the substitution of a single noun or adjective. If there is more than one word in a chunk then the words have to be linked chunk internally[2]. The system locates the word(s) to be adapted in the TL chunk and leaves the rest untouched thus retaining the original context as much as possible. Word adaptations in the middle of a multi-word chunk are preferable to those at the edges to lessen the effects of boundary friction between chunks. The partial keyhole adaptation of a chunk is shown below:

SL: (the **Printing** process)

SL′: (the **scanner calibration** process)

TL′: (mit der **Scanner Kalibrierung**)

This results in a word level directive of:

adapt(Printing, (Scanner Calibration))

which in turn calls `dict` with the parameters `(Printing,n,(n n))` where now POS information of the SL word and TL words (arguments 2 and 3 respectively) is used to contextualise the dictionary lookup. The final result is the TL solution chunk with the word substitution, for example:

TL: (mit der **Drucken**)

Note that there is a case agreement problem; *der* is accusatively marked when it should be dative. This is a side-effect of adapting inside a chunk. Had the dictionary lookup succeeded on a full-chunk basis as in Section 4.5, then this problem would not arise. Hence, keyhole adaptation

[2]Word-linking specific to this example has already been performed by the linker at the learning stage, but such information is not stored in the system as it rarely gets reused.

receives a lower adaptability score. Finally, if the dictionary did not contain any equivalents for *Printing*, it will inform the user of its gap by printing the following as part of the solution:

(mit der ≪**Printing process**≫).

6.4 Retrieval

An overall view of the retrieval process is shown in Figure 6.7. The system first tries to find a single matching by unification (Section 6.4.1). This involves using the input template as a key for hash-table lookup. The threshold can be set to any level of adaptability. The retriever then passes the templates through an Adaptation-Safety Filter (Section 6.4.2), whose threshold of adaptability is set by the user at run time depending on the amount of data available in the case-base. This ensures that only adaptable cases remain to be refined further if necessary, hence the name adaptation guided retrieval. Should one or more full matches occur at the desired level of adaptability then these are assessed for adaptability and similarity and passed to the adaptor.

If the retriever fails to find a complete match at the desired level of adaptability then the partial-case retrieval mechanism comes into force. This first finds cases which are syntactically similar to the input template as described in Section 6.5.1 and then searches the filtered cases for the best (most adaptable) case according to the global adaptability scoring mechanism outlined in Section 6.3.2.

6.4.1 Full-Case Retrieval

The first step of the retriever is to ascertain whether at least one single case can cover the input problem. If so, the retriever will perform, and score the adaptation of each and present the result to the user, as described in the next section. Before determining the level of thresholding required for a match however, for efficiency reasons the retriever first tests whether the templates will unify at all, i.e. at their most general level of description. The template for matching now only contains

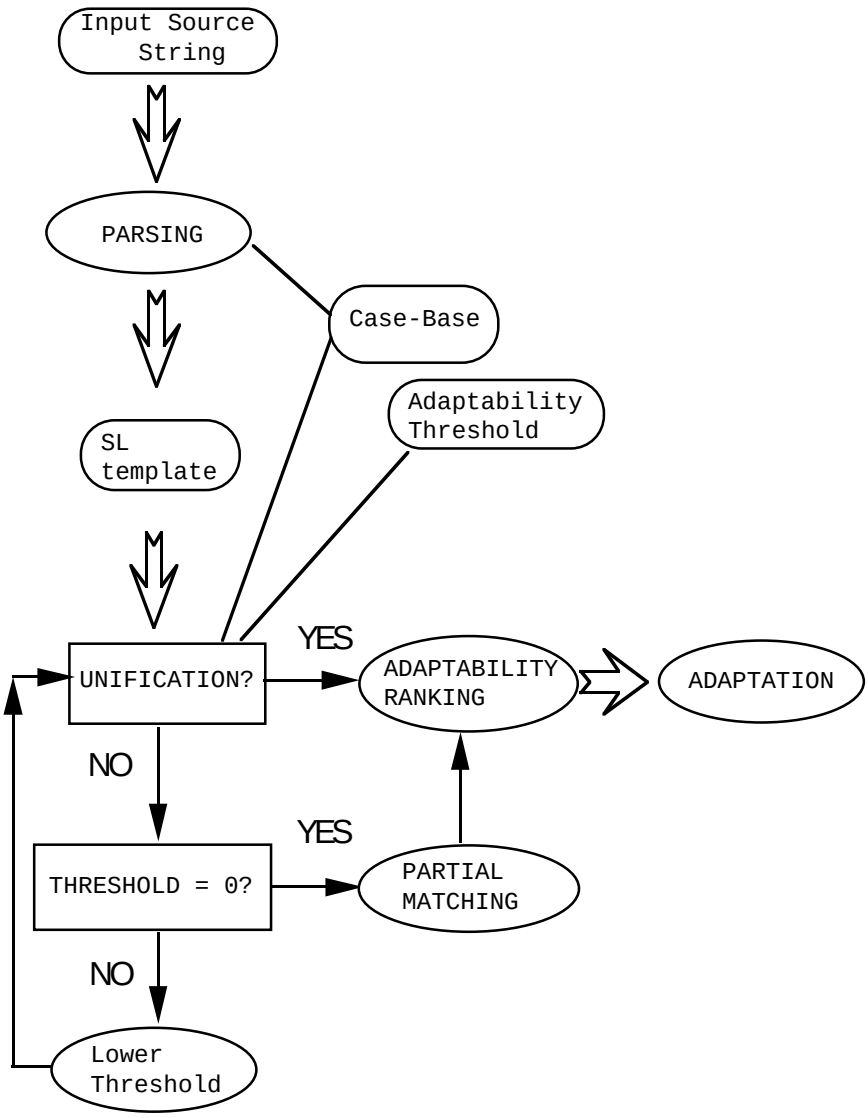


Figure 6.7: *Retrieval Stages in ReVerb.*

syntactic functional information in terms of an ordered flat structure of syntactic-function labels and variable pairs. Cases are automatically stored off-line in a hash table using their most general template as a key. This is (optionally) done at case-storage time by demons. At run time, the retriever now only needs to perform a hash lookup using the input template as a key in order to retrieve a set of structurally matching cases. All cases which share the same basic syntactic structure (same syntactic functions in the same order) will appear under the same key in the hash table. As there is repetition in the sentences which belong to a domain, the hash table grows at a less than linear rate[3]

6.4.2 Full-case Adaptability Assessment

The matching cases are first assessed and ranked for global adaptability according to equation (6.2) below. If a case chunk requires no adaptation because of a complete string match as well as syntactic functional matching then a score of 3 is given to this match. Otherwise the adaptation score is taken to be the chunk link score as determined by the Linker (cf. Section 5.5).

$$\frac{\sum_{i \in case} linkscore_i}{|Chunks|} \tag{6.2}$$

$$linkscore_i = \begin{cases} 3 & if InputChunk_i = CaseChunk_i, \\ ChunkScore & otherwise. \end{cases} \tag{6.3}$$

This means that the attempt to adapt a non-mapping chunk (e.g. "use" in Section 6.3.1 above, will be given a score of zero, as this is the *linkscore* of any non-linked chunk. This will decrease greatly the Adaptability Score of the case. Adaptation scores will range between 0 and 2, as this is the range of *linkscore* values for a given chunk (see Section 4.6). Chunks with link scores higher than 1 have similar syntactic functions in their respective languages, and the closer to 2 this figure is, the more similar the lexical content of the linked chunks, as determined by the system dictionary.

[3] In a case-base of 750 cases taken from the CorelDRAW corpus, approximately 20% were stored under the same SL template (key) as at least one other case where the SL was English.

This score is pushed right up to 3 if no adaptation is required, i.e. if there is a complete SL $\rightarrow$ SL′ match

6.4.3 Full-Case Translation Assessment

To include the case-base dependent measure of adaptation availability, each solution is ranked according to equation (6.4) below. This predicts how reliable the actual substitution is.

$$\frac{\sum_{i \in Adaptee} TranslationAvailability_i + KeyholeAdaptation_i}{|AdapteeChunks|} \tag{6.4}$$

The higher the score, the more reliable ReVerb considers the translation to be according to the knowledge it has available to it, that is, the corpus alone.

6.5 Partial-Case Reuse

6.5.1 Feature Promotion

The Retriever uses the Input template as the basis for feature promotion. Each cell of the template activates all chunks in the case-base which match it in syntactic functionality and linear-order. No case will match exactly for otherwise it would have unified but it is quite possible that many near matches exist. This pattern matching scheme is flexible enough to allow gaps of arbitrary length but crossovers[4] will be penalised. The best n-matches are retrieved and assessed for adaptability as explained below.

6.5.2 Adaptability Assessment

$$\frac{\sum_{i \in case} linkscore_i}{Adaptzero + Deletezero + Delete + Add} \tag{6.5}$$

[4]If English were not the source language, crossovers may not be detrimental to the accuracy of the retrieval for in many languages word order is not the primary encoder of meaning. Hence this constraint can easily be relaxed by allowing near-matches in linear-ordering to be valid.

$$linkscore_i = \begin{cases} 3 & if InputChunk_i = CaseChunk_i, \\ ChunkScore & otherwise. \end{cases} \tag{6.6}$$

6.5.3 Translation Assessment

The translation of partial case solutions are judged in the same way as full cases and the assessment of (6.2) is repreated here as (6.7) for the sake of clarity.

$$\frac{\sum_{i \in Adaptee} TranslationAvailability_i + KeyholeAdaptation_i}{|AdapteeChunks|} \tag{6.7}$$

6.6 Examples

A diagrammatic view of a translation retrieval episode is provided here for a) a full case solution as in Figure 6.8 and b), a partial-case solution as in Figure 6.9. In Figure 6.8, the SL $\rightarrow$ SL$'$ full-case match is strong; only two chunks differ — the first, denoting "choose", and the third, "to increase". However, the first chunk of the SL$'$, "Use", which requires adaptation, has no obvious counterpart in the TL$'$. It has instead been translated as "Mit .." (*with..*) which is contained within a chunk which links to the second SL$'$ chunk, "The Offset Command". The `AdaptZero` operation would translate "Use" nevertheless, and join it to the beginning of the TL solution. For the other adaptation, that of "to increase" $\rightarrow$ "to specify", there is a link to the target TL$'$, namely the chunk "legen". However this is a poor link because "legen" is a separable verb and its dependent particle "fest" is stranded at the end of the TL$'$. It will be IgnoreZeroed which means that it will surface in the TL solution exactly as it did in the TL$'$, but penalised. As it is not a correct particle in the translation of "to specify", which is not a separable verb, this will cause an error (extra word) in the TL solution. The resulting TL solution that would be expected in this scenario is (1) below:

(1) (Wählen)(Mit der Option Abstand)(vermehren)(Sie)(den Abstand)(zwischen den Formen)(fest)

(1)$'$ *(Choose) (with the Offset Command)(increase)(you)(the distance)(between the shapes) [particle]*

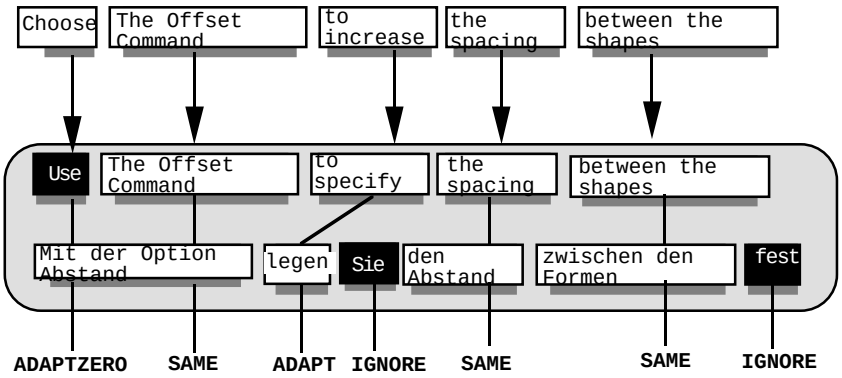


Figure 6.8: *A Full-Case Translation Episode.*

(1)″ *Choose Increase the distance between the shapes with the Offset Command [particle]*

The resulting translation in (1) is equivalent to (1)″ in which there are two extra words, "Choose" and the particle. The adaptor did not realise that in this context, "use" and "choose" could be translated in the same way and hence be ignored. Only a word-based similarity metric would achieve this. Instead, ReVerb penalises such adaptations so heavily that other cases would be favoured before this one. This is a safer policy as "use" and "choose" may not always be translatable in the same way, and it is quite reasonable to expect this template shape to recur frequently in the data of the test corpus. The second example, in Figure 6.9 is actually extremely poor, and it demonstrates the full-range of adaptation operations which may be required to produce a TL. The first half of the SL chunks are covered by the chosen case. The blackened chunks in the SL′ and TL′ have no equivalents in the opposite language according to the Linker. The blackened chunks in the input SL have no covering chunks in the SL′ so these must be `Add`ed to the solution TL eventually. Any non-linking SL′ chunks are to be `AdaptZero`ed. This means the SL chunk is looked up in the dictionary and simply added to the solution where an non-linking chunk can be found. The non-linking TL′ chunks themselves will be `DeleteZero`ed, i.e. deleted. Any non-linking SL′ chunks which are outside of the partial match (e.g. "it" in Figure 6.9) are `IgnoreZero`ed, i.e. Ignored, but penalised, unlike the `Ignore` operation (see Figure 6.4).

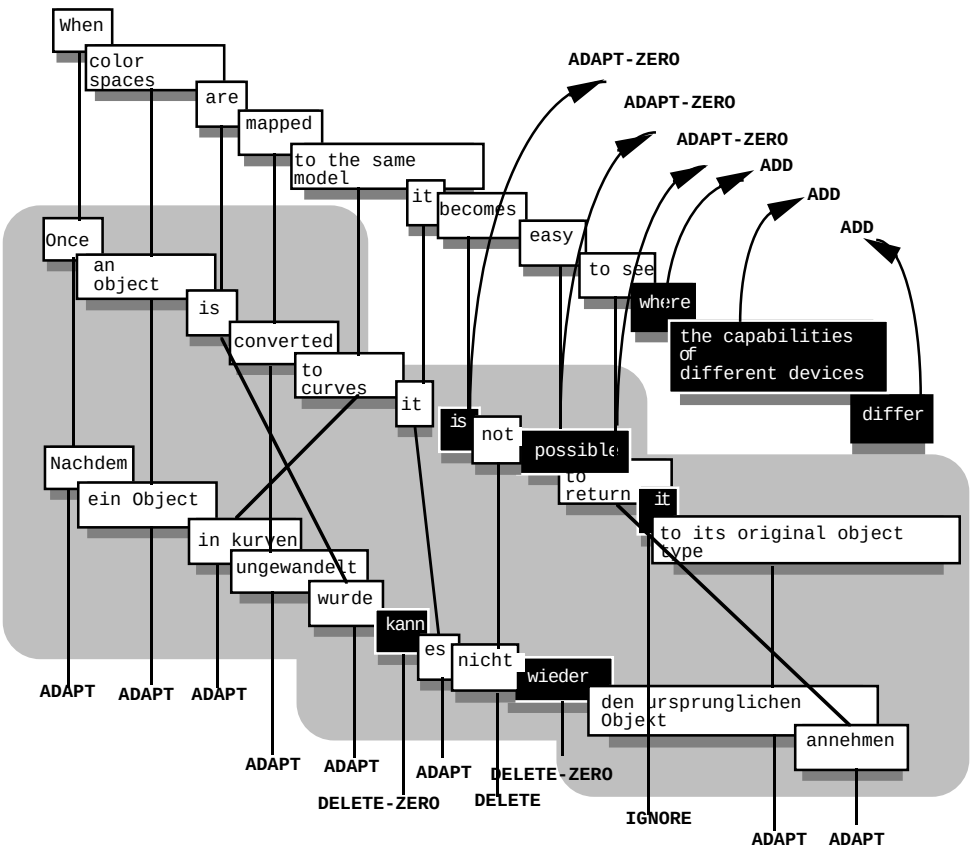


Figure 6.9: *A Partial Case Translation Episode.*

6.7 Evaluation of Adaptation-Guided Retrieval

This section describes the results of both types of translation — full-case and partial — which the system performed on the input data at different levels of Adaptability.

6.7.1 Inputting the test-data

A set of 90 input sentences was taken from the English Corel Draw (v7) manual. For each sentence, the best two cases would be retrieved so that a total of 180 translations were produced. The 90 sentences were chosen at random, but they were all between 4 and 25 words long. The files used were not from the equivalent part of the previous version of the manual, which the system was trained on, and they contained much new terminology. We had access to the equivalent German sentences, which allowed us to do a manual comparison of the German sentences that ReVerb was producing. It was not uncommon for the system to produce reasonable paraphrases of the input sentences which differed quite extensively from the human translations, therefore these could not be used as part of an automatic evaluation measure. Each sentence retrieved the two highest scoring cases on the basis of the Adaptation Safety scoring (Section 6.3). The output of 180 sentences was examined by a bi-lingual evaluator. The evaluator specified for each translation the necessary changes to render it into grammatical form conveying all the information in the input problem string. Errors occurred at both the chunk and word level. The chunk errors noted were the following:

- chunk-order errors
- missing chunks
- extra chunks

All chunk errors were assigned a penalty of 1. These errors result from adapting poorly linking chunks, and all the operations described in section 6.3, namely, `Delete, Add, DeleteZero, AddZero and IgnoreZero`. In addition, the performance of the dictionary was assessed in terms of the grammaticality and information content of the substitutions (whether whole-chunk or "keyhole" as described in Section 6.4. The errors were classified into the following types:

- word-order errors within chunk
- missing word
- extra word(s)
- *n-to-1* or *1-to-n* errors
- incorrect translation
- POS error

Although these errors were sometimes very minor in hindsight, it still takes a translator time and effort to identify and correct each one, hence in the spirit of creating a tool that quantifies how much post-editing work is really necessary, they were all assigned a score of 1. The chunks errors are a function of the cases themselves and the input sentence, and we would expect them to be distributed according to the adaptability score of the cases. The second error group is more arbitrary as it depends on the input sentence words being present in the system in the right context. An unknown word was recorded as being a 'missing word' error.

6.8 Discussion

As predicted in the previous Section, these results show a steady increase in accuracy as the Adaptability Score increases (see Figure 6.10). The overall error rate, including unknown words tended to diminish as more adaptable cases were reused. This is because the highly adaptable cases often received their high scores by virtue of the fact that very little adaptation was required (i.e. high similarity). The retrievals around the half-way mark, that is from an adaptability score of 0.7 to 1.0 had fluctuating numbers of errors, but once the number of exact matches and adaptable chunks (`Same, Adapt`) together exceeded the number of badly adaptable chunks (`AdaptZero, Add, Delete`), i.e. any value over 1, the performance improved dramatically. The coverage of ReVerb degrades gracefully, that is, at no point does it refuse to retrieve a sentence. Many of the input words may not be adaptable and will be returned as English. Thus, the less the systems can adapt

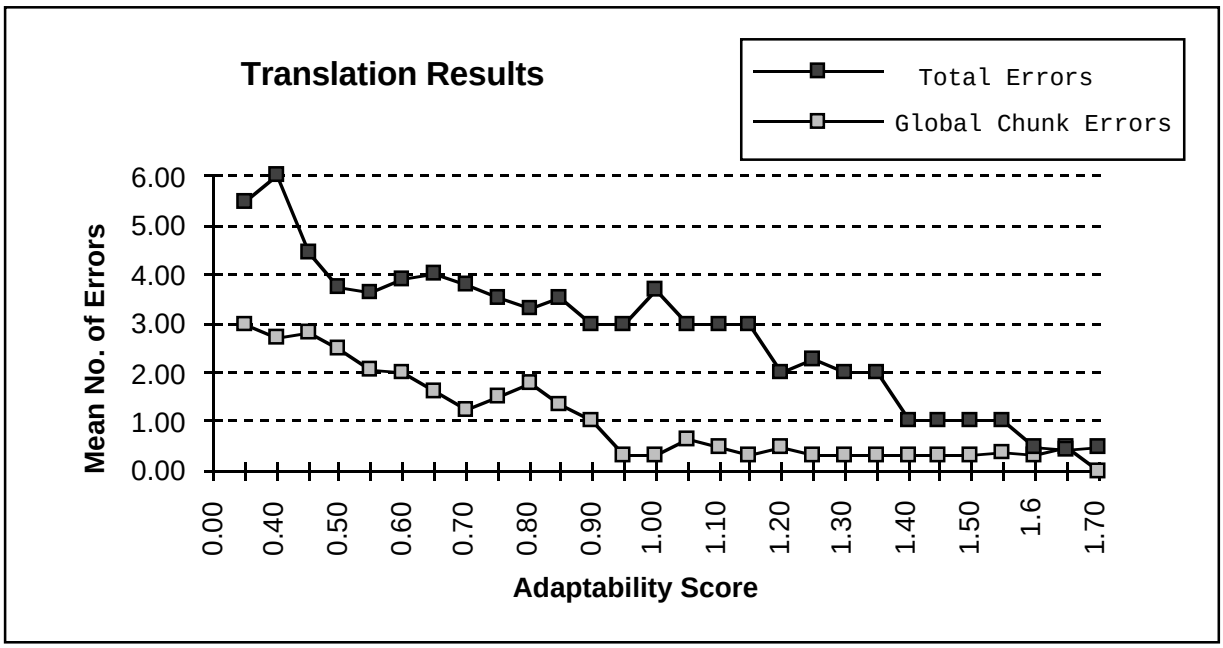


Figure 6.10: *Results of Translation of 180 sentences at Different Levels of Adaptability.*

the sentence, the more Source-language-like the translation will be. The higher scoring retrievals result in sentences which capture the stylistics and idioms of the German expressions available in the case base The results here show that performance improves in relation to the systems adaptability calculations, and hence, that adaptability is a good indicator of the usefulness of a case for EBMT. Bilingual readers can judge for themselves the usefulness of cases retrieved, as solutions with both Adaptability Scores and Translation Scores are provided in Appendix D.

Conclusion

The question for EBMT retrieval is what level of abstraction is necessary to extract structures from the case base whose TL components will reflect the meaning of the entire SL structure in grammatical form while maintaining a reasonable retrieval cost. Indexes should preferably delimit the composable parts of a problem and solution and inhibit the splitting up of a sentence in these areas. A method for exploiting the similarity and adaptability information simultaneously in retrieval has been described. The system's performance embodying this methodology has been shown to degrade gracefully and

in tune with the adaptability assessment performed at retrieval time. Even when the case-base is relatively small (800 cases), the output indicates that this system would be useful to translators in this domain. Also, the guarantee of accuracy it offers at high levels of adaptability mean that, on scaling up the case-base, high-quality automatic translations would be possible for texts of this domain.

Chapter 7

Summary and Outlook

7.1 Introduction

In Part I of this thesis, EBMT was examined in context of theoretical and practical approaches to MT to highlight the merits of a CBR approach to translation reuse. Part II presented a novel solution to the case-creation knowledge bottleneck using a data-oriented parser and linker. Part III described Adaptation-Guided Retrieval (AGR) showing how cautious reuse of translations results in very effective exploitation of the examples present in the case base.

Part I: Motivation and Application

Chapter 2 provided the background against which to introduce a novel approach to EBMT. The spectrum of EBMT was found to contain a "gap". This rift spanned from the knowledge-rich approaches which used EBMT in conjunction with dictionaries and thesauri to retrieve and patch together phrases in context, to the *memory-based* (MBMT) approaches which retrieve many fragments on the basis of surface similarity. CBR is a methodology which relies on cases having structure (unlike MBMT) but where the domain knowledge is only inferred from the cases themselves (as opposed to "thesaurus-based" EBMT). In introducing a CBR philosophy to EBMT, a methodology is proposed wherein the only knowledge for translation is stored in cases alone, and yet these cases are

structured enough to provide some generalisation of patterns in and across languages. The actual patterns which can be expected to arise in "real" data, for example texts from the localisation domain, were investigated in Chapter 3. Here it was concluded by analysis of a representative sample, that a flexible multi-layer representation at a shallow syntactic level of abstraction would capture the range of divergences evident in the data.

Part II: Automatic Creation of an EBMT Case Base.

Chapter 4 presented the data structures and memory organisation of the ReVerb system, which supports incremental updating of knowledge at three levels of description - word, chunk and sentence. The information flow between such frames in the memory give rise to a host of knowledge sources for EBMT which reflect the tendencies present in the training corpus. Chapter 5 showed that an effective an language-independent data-oriented parser and linker for text from a similar corpus could be devised, which in turn allows the creation of more cases. The symbiosis between case storage and creation ensures that there is a gradual, steady improvement in performance as more cases are created.

Part III: Adaptation Guided Retrieval for full and partial case matches.

In the final part of the thesis, a detailed mechanism for the assessment of adaptability of cases on a structural basis was presented. This includes both full and partially matching cases. A templatisation scheme was proposed whereby cases are generalised on the basis of their individual linking patterns. A novel thresholding filter ensures that a user-determined adaptability score can be imposed *before* retrieval to ensure a certain level of adaptability of the cases retrieved. Candidates are thus chosen on the basis of adaptability and similarity rather than similarity alone. A means of assessing the adaptation *after* candidate selection (as in other EBMT systems) was also presented as an extra indicator of translation reliability. The feature promotion and adaptability filtering stages were organised schematically to ensure maximum efficiency at run-time. Base-filtering before adaptability assessment minimised the number of cases to be assessed, for both full and partial

matching. The AGR methodology was tested in the final sections. A correlation was found between Adaptability and Accuracy of the solutions. This demonstrated that the Adaptation-Guided Retrieval methodology is a highly effective means of producing the accurate translations even with a very small training corpus.

Outlook

The solution proposed here is a wide-ranging *rethink* on the reuse of examples in translation. It encompassed the full cycle of translation and in each sub-area, possibilities for future exploration and improvement became evident. The most central of these issues include the following:

- The idea of reusing translations on the basis of the trace between SL′ and TL′ as well as SL′→ SL′ similarity does not in itself exclude memory based approaches to Adaptation-Guided Retrieval. This could be attempted without the limited form of linguistic abstraction inherent in ReVerb, for example, by using a Memory-based classifier algorithm like k-NN.

- As the system is bidirectional, a procedure for automatic evaluation of the translation may be to backtranslate the TL, and compare it against the SL. If this results in a paraphrase of SL, call it SL″, then some means of assessing the similarity of SL and SL″ must be devised.

- The granularity of chunks at the syntactic functional level may not be the only subsentential linking pattern applicable to this system. This may be suitable for Indo-European languages but alternative sub-sentential alignments may prove more beneficial when dealing with less related language pairs (e.g. English, Japanese). At any rate, the system should be tested on a variety of language pairs and domains before any claims about universality of the benefits of AGR in EBMT can be made.

- Information at the sub-word level would increase the generality and accuracy of the dictionary and linker. A word frame would thus refer to a morphological root and point to all its morphological variants in the case base. Other forms of word decomposition (e.g. case marking, gender marking) would be useful particularly for languages with a richer morphology than

English.

- The system has the potential to learn from its *own* translation efforts. by storing the result of translation (SL, TL). This would require some automatic checking of the TL and the *utility problem* would have to be addressed so as to discard cases which no longer provide new information to the system.

Part IV

Appendices

Appendix A

A Sample Case

This shows one case from the training data (*CorelDRAW v.5*). It was created by ReVerb but subsequently checked manually for any alignment errors. The sentence pair it represents is:

(Click) (on the color) (you) (want) (or) (choose) (More) (to display) (another dialog box) (where) (you) (can) (create) (your own colors) (and) (select) (them) (by name)

(Klicken) (Sie) (auf die gewuenschte Farbe) (oder) (auf Mehr) (um) (ein dialogfeld) (zu offnen) (in dem) (Sie) (eigene Farben) (einstellen) (und) (nach Namen) (auswaehlen) (koennen)

(krell-frame case-C15

(super case)

(source-language English)

(target-language German)

(chunk chunk-C15-1 chunk-C15-2

chunk-C15-3 chunk-C15-4 chunk-C15-5

chunk-C15-6 chunk-C15-7 chunk-C15-8

chunk-C15-9 chunk-C15-10 chunk-C15-11

chunk-C15-12 chunk-C15-13 chunk-C15-14

chunk-C15-15 chunk-C15-16 chunk-C15-17

chunk-C15-18 chunk-C15-19 chunk-C15-20))

(krell-frame chunk-C15-1

(super chunk)

(SOURCE-SYN-FUNCTION @ADVL)

(SOURCE-LIN-ORDER 25)

(SOURCE-POS N)

(MAPPING 1/6)

(SOURCE-TEXT BY NAME)

(TARGET-SYN-FUNCTION @ADVL)

(TARGET-TEXT NACH NAMEN)

(TARGET-LIN-ORDER 15)

(TARGET-POS PREP N))

(krell-frame chunk-C15-2

(super chunk)

(SOURCE-SYN-FUNCTION @-FMAINV)

(SOURCE-LIN-ORDER 23)

(SOURCE-POS V)

(MAPPING 3/2)

(SOURCE-TEXT SELECT)

(TARGET-SYN-FUNCTION @-FMAINV)

(TARGET-TEXT AUSWAEHLEN)

(TARGET-LIN-ORDER 16)

(TARGET-POS V))

(krell-frame chunk-C15-3

(super chunk)

(SOURCE-SYN-FUNCTION @CC)

(SOURCE-LIN-ORDER 22)

(SOURCE-POS CC)

(MAPPING 3/2)

(SOURCE-TEXT AND)

(TARGET-SYN-FUNCTION @CC)

(TARGET-TEXT UND)

(TARGET-LIN-ORDER 14)

(TARGET-POS CC))

(krell-frame chunk-C15-4

(super chunk)

(SOURCE-SYN-FUNCTION @OBJ)

(SOURCE-LIN-ORDER 19)

(SOURCE-POS PRON N)

(MAPPING 1/3)

(SOURCE-TEXT YOUR OWN COLORS)

(TARGET-SYN-FUNCTION @OBJ)

(TARGET-TEXT EIGENE FARBEN)

(TARGET-LIN-ORDER 12)

(TARGET-POS ADJ N))

(krell-frame chunk-C15-5

(super chunk)

(SOURCE-SYN-FUNCTION @-FMAINV)

(SOURCE-LIN-ORDER 18)

(SOURCE-POS V)

(MAPPING 9/8)

(SOURCE-TEXT CREATE)

(TARGET-SYN-FUNCTION @-FMAINV)

(TARGET-TEXT ERSTELLEN)

(TARGET-LIN-ORDER 13)

(TARGET-POS V))

(krell-frame chunk-C15-6

(super chunk)

(SOURCE-SYN-FUNCTION @+FAUXV)

(SOURCE-LIN-ORDER 17)

(SOURCE-POS V)

(MAPPING 3/2)

(SOURCE-TEXT CAN)

(TARGET-SYN-FUNCTION @+FAUXV)

(TARGET-TEXT KOENNEN)

(TARGET-LIN-ORDER 17)

(TARGET-POS V AUXMOD))

(krell-frame chunk-C15-7

(super chunk)

(SOURCE-SYN-FUNCTION @SUBJ)

(SOURCE-LIN-ORDER 16)

(SOURCE-POS PRON)

(MAPPING 1/2)

(SOURCE-TEXT YOU)

(TARGET-SYN-FUNCTION @SUBJ)

(TARGET-TEXT SIE)

(TARGET-LIN-ORDER 11)

(TARGET-POS PRON))

(krell-frame chunk-C15-8 (super chunk)

(SOURCE-SYN-FUNCTION @ADVL)

(SOURCE-LIN-ORDER 15)

(SOURCE-POS ADV)

(MAPPING 7/16)

(SOURCE-TEXT WHERE)

(TARGET-SYN-FUNCTION @ADVL)

(TARGET-TEXT IN DEM)

(TARGET-LIN-ORDER 10)

(TARGET-POS PREP DET))

(krell-frame chunk-C15-9

(super chunk)

(SOURCE-SYN-FUNCTION @OBJ)

(SOURCE-LIN-ORDER 12)

(SOURCE-POS N N)

(MAPPING 3/8)

(SOURCE-TEXT ANOTHER DIALOG BOX)

(TARGET-SYN-FUNCTION @OBJ)

(TARGET-TEXT EIN DIALOGFELD)

(TARGET-LIN-ORDER 7)

(TARGET-POS DET N))

(krell-frame chunk-C15-10

(super chunk)

(SOURCE-SYN-FUNCTION @-FMAINV)

(SOURCE-LIN-ORDER 10)

(SOURCE-POS V)

(MAPPING 9/16)

(SOURCE-TEXT TO DISPLAY)

(TARGET-SYN-FUNCTION @INFMARK>)

(TARGET-TEXT ZU OEFFNEN)

(TARGET-LIN-ORDER 8)

(TARGET-POS INFMARK V))

(krell-frame chunk-C15-11

(super chunk)

(SOURCE-SYN-FUNCTION @OBJ)

(SOURCE-LIN-ORDER 9)

(SOURCE-POS PRON)

(MAPPING 1/2)

(SOURCE-TEXT MORE)

(TARGET-SYN-FUNCTION @ADVL)

(TARGET-TEXT AUF MEHR)

(TARGET-LIN-ORDER 5)

(TARGET-POS PREP N))

(krell-frame chunk-C15-12

(super chunk)

(SOURCE-SYN-FUNCTION @CC)

(SOURCE-LIN-ORDER 7)

(SOURCE-POS CC)

(MAPPING 3/2)

(SOURCE-TEXT OR)

(TARGET-SYN-FUNCTION @CC)

(TARGET-TEXT ODER)

(TARGET-LIN-ORDER 4)

(TARGET-POS CC))

(krell-frame chunk-C15-13

(super chunk)

(SOURCE-SYN-FUNCTION @ADVL)

(SOURCE-LIN-ORDER 2)

(SOURCE-POS N)

(MAPPING 13/252)

(SOURCE-TEXT ON THE COLOR)

(TARGET-SYN-FUNCTION @ADVL)

(TARGET-TEXT AUF DIE GEWUENSCHTE FARBE)

(TARGET-LIN-ORDER 3)

(TARGET-POS PREP DET ADJ N))

(krell-frame chunk-C15-14

(super chunk)

(SOURCE-SYN-FUNCTION @+FMAINV)

(SOURCE-LIN-ORDER 1)

(SOURCE-POS V)

(MAPPING 3/2)

(SOURCE-TEXT CLICK)

(TARGET-SYN-FUNCTION @+FMAINV)

(TARGET-TEXT KLICKEN)

(TARGET-LIN-ORDER 1)

(TARGET-POS V IMP))

(krell-frame chunk-C15-15

(super chunk)

(SOURCE-SYN-FUNCTION @OBJ)

(SOURCE-LIN-ORDER 24)

(SOURCE-POS PRON)

(MAPPING 0)

(SOURCE-TEXT THEM)

(TARGET-SYN-FUNCTION <NONE>)

(TARGET-TEXT <NONE>)

(TARGET-LIN-ORDER <NONE>)

(TARGET-POS <NONE>))

(krell-frame chunk-C15-16

(super chunk)

(SOURCE-SYN-FUNCTION @+FMAINV)

(SOURCE-LIN-ORDER 8)

(SOURCE-POS V)

(MAPPING 0)

(SOURCE-TEXT CHOOSE)

(TARGET-SYN-FUNCTION <NONE>)

(TARGET-TEXT <NONE>)

(TARGET-LIN-ORDER <NONE>)

(TARGET-POS <NONE>))

(krell-frame chunk-C15-17

(super chunk)

(SOURCE-SYN-FUNCTION @+FMAINV)

(SOURCE-LIN-ORDER 6)

(SOURCE-POS V)

(MAPPING 0)

(HEAD WANT)

(TARGET-SYN-FUNCTION <NONE>)

(TARGET-TEXT <NONE>)

(TARGET-LIN-ORDER <NONE>)

(TARGET-POS <NONE>))

(krell-frame chunk-C15-18

(super chunk)

(SOURCE-SYN-FUNCTION @SUBJ)

(SOURCE-LIN-ORDER 5)

(SOURCE-POS PRON)

(MAPPING 0)

(SOURCE-TEXT YOU)

(TARGET-SYN-FUNCTION <NONE>)

(TARGET-TEXT <NONE>)

(TARGET-LIN-ORDER <NONE>)

(TARGET-POS <NONE>))

(krell-frame chunk-C15-19

(super chunk)

(SOURCE-SYN-FUNCTION <NONE>)

(SOURCE-LIN-ORDER <NONE>)

(SOURCE-POS <NONE>)

(MAPPING 0)

(SOURCE-TEXT <NONE>)

(TARGET-SYN-FUNCTION INFMARK>)

(TARGET-TEXT UM)

(TARGET-LIN-ORDER 6)

(TARGET-POS INFMARK))

(krell-frame chunk-C15-20

(super chunk)

(SOURCE-SYN-FUNCTION <NONE>)

(SOURCE-LIN-ORDER <NONE>)

(SOURCE-POS <NONE>)

(MAPPING 0)

(SOURCE-TEXT <NONE>)

(TARGET-SYN-FUNCTION @SUBJ)

(TARGET-TEXT SIE)

(TARGET-LIN-ORDER 2)

(TARGET-POS PRON))

Appendix B

Case Features

B.1 SYNTACTIC FUNCTION

subject, @SUBJ:

You *can however click any of the patches to select and enter a measurement for it*

object, @OBJ:

You can however click ***any of the patches*** *to select and enter* ***a measurement*** *for it*

finite main verb, @+FMAINV:

If this option ***is*** *not available* ***inform*** *your supplier*

non-finite main verb @-FMAINV:

You can however ***click*** *any of the patches to select and* ***enter*** *a measurement for it*

adverbial @ADVL:

You can ***however*** *click any of the patches to select and enter a measurement* ***for it***

finite auxilliary verb `@+FAUXV`:

*You **can** however click any of the patches to select and enter a measurement for it*

non-finite auxilliary verb, `@-FAUXV`:

Check to see if it has `been` loaded

constituent sub-ordinator, `@CS`:

***If** this option is not available please inform your supplier*

constituent coordinator, `@CC`:

*You can however click any of the patches to select **and** enter a measurement for it*

'stray' noun-phrase, `@NPHR`:

***Note** You can however click any of the patches to select and enter a measurement for it*

complement of a subject, `@PCOMPL-S`:

*If this option is not **available** please inform your supplier*

negative particle, `@NEG`:

You can however click any of the patches to select and enter a measurement for it

infinitive verb phrase, `@INFMARK>`:

You can however click any of the patches `to select` and enter a measurement for it

B.2 PART-OF-SPEECH

noun	`n`	*patches, measurement*
verb	`v`	*click, enter*
auxiliary verb	`fauxv`	*can*
negative particle	`neg`	*not*
preposition	`p`	*of, for, to*
pronoun	`pron`	*you it*
determiner	`det`	*a, the, any*
infinitival marker	`inf`	*to*
coordinator	`cc`	*and*
subordinator	`cs`	*if*
adverbial	`adv`	*however*

Appendix C

Sample Translations at Various Levels of Adaptability

The folling is a selection of translations which ReVerb performed on a test set of 90 cases, given a training set of 800 (see Chapter 6). In each scenario, the Adaptability Threshold is stated, above which the match would not have happened. In the Adaptation Recipe the `ADAPT` operation is formatted as follows:

(1 :`ADAPT` (SL′-CHUNK) TL′-SYNFUN (SL-CHUNK) (TL′-CHUNK))

C.0.1 Full-case matching Threshold 1.6

CASE-91 Found Match of strength 1.666667 with CASE-CO24

`SL` :((GAMMA) (CONTROLS) (THE MONITOR-S BRIGHTNESS))

`SL′`:(CANCEL CLOSES TASK LIST)

TL′:(ABBRECHEN MIT DIESER OPTION WIRD DIE TASK-LISTE GESCHLOSSEN)

Adaptation Recipe

(2 IGNORE: MIT DIESER OPTION)

(3 IGNORE: WIRD)

(1 :ADAPT (CANCEL) @NPHR (GAMMA) (ABBRECHEN))

(5 :ADAPT (CLOSES) @-FMAINV (CONTROLS) (GESCHLOSSEN))

(4 :ADAPT (TASK LIST) @SUBJ (THE MONITOR-S BRIGHTNESS) (DIE TASK-LISTE))

TL: (gamma) (mit dieser option) (wird) ((das ≪monitor-s ≫) helligkeit)) (optionen)

Translation Score = .625

Errors: 2

One missing word "monitor's" and one mis-translated word: "optionen". The chunk-ordering is perfect. This would be a useful translation for a translator.

The threshold for this match was 1.6. This means that all the SL′ chunks have to match the SL and that all SL′ chunks match a TL′ chunk with a score of 1.6 or more. Hence, they are either complete string and synfun-matches(3) or they are synfun matches with some word differences (1...3). In this example, there were no string matches, so all words of the TL′ had to be adapted, except for those that were IGNORE'd. The mistranslated word, "optionen"(*options*) was supposed to be the translation of "controls". This suggests a dictionary look-up as follows (see Section 4.4):

`dict(controls,EN,GE,v)` → `NIL`

`dict(controls,EN,GE,n)` → `Optionen(n)`

The third-person singular verb *controls* should in fact be translated as "kontrolliert". The possessive noun "monitor's" did not occur in the data. As mentioned in Chapter 7, a possible remedy for this

sparse data effect would be to encode morphological variants of word forms.

C.0.2 Full-case matching. Threshold 1.

SL: :((TO DO) (SO) (CLICK) (VIEW COLOR CORRECTION NONE))

SL′:(TO PRINT ODD PAGES ENTER 1TILDE)

TL′:(UNGERADE SEITEN ZU DRUCKEN GEBEN SIE 1 EIN)

CASE-104 Found Match of strength 1.0 with CASE-F200

(1 :IGNORE UM)

(5 :IGNORE SIE)

(7 :IGNORE EIN)

(3 :ADAPT (TO PRINT) @NPHR (TO DO) (ZU DRUCKEN))

(2 :ADAPT (ODD PAGES) @OBJ (SO) (UNGERADE SEITEN))

(4 :ADAPT (ENTER) @+FMAINV (CLICK) (GEBEN))

(6 :ADAPT (1∼) @OBJ (VIEW COLOR CORRECTION NONE) (1∼))

TL: (um) (dass) ((zu tun)) (klicken) (sie) ((ansicht farben ≪correction ≫ keine)) (ein)

Translation Score = .607

Errors 2.

One missing word “correction” The chunk-ordering is perfect, but there is a stray particle “ein” at the end.

The verb which was replaced “enter → geben” is actually a separable verb “eingeben”. The lower match strength is partially due to the fact that “geben” received a low score. This would be a useful translation for a translator.

C.0.3 Partial-case matching. Threshold 0.3

CASE-118 Found Match of strength 0.32118326 with CASE-CO13

SL: ((SEPARATIONS PRINTERS) (PROCESS) (IMAGES) (USING) (THE CMYK COLOR MODEL))

SL′:(MAXIMIZE COMMAND CONTROL MENU EXPANDS THE ACTIVE WINDOW TO FILL THE ENTIRE SCREEN)

TL′: (BEFEHL VOLLBILD SYSTEMMENUE DIENT ZUR VERGROESSERUNG DES AKTIVEN FENSTERS AUF DIE GROESSE DER GESAMTEN BILDSCHIRMFLAECHE)

(2 :IGNORE DIESER BEFEHL)

(3 :IGNORE DIENT)

(1 :ADAPT (MAXIMIZE COMMAND CONTROL MENU) @NPHR (SEPARATIONS PRINTERS) (BEFEHL VOLLBILD SYSTEMMENUE))

(4 :ADAPT (EXPANDS) @ADVL (PROCESS) (ZUR VERGROESSERUNG))

(5 :ADAPT (THE ACTIVE WINDOW) @ADVL (IMAGES) (DES AKTIVEN FENSTERS))

(7 :ADAPT (TO FILL) @ADVL (USING) (AUF DIE GROESSE))

(8 :ADAPT (THE ENTIRE SCREEN) @ADVL (THE CMYK COLOR MODEL) (DER GESAMTEN BILDSCHIRMFLAECHE))

TL: ((auszuege drucker)) (dieser befehl) (dient) (zur bearbeitung)(bildtypen) (verwenden) ((das cmyk farben farbmodellname))

Translation lity 0: useless, 1: perfect = .536

Errors 3.

One *n-to-1* word error: "Separations Printer" would normally be translated as one word in German : "Auszugsdrucker". One *1-to-n* word error: "Images", a generic noun in English, cannot be translated as a generic noun in German ("Bildtypen"). It would be necessary to add a preposition here ("von Bildtypen") (*of images*). Finally, one erroneous chunk inclusion "verwenden" for "using". The substitution ("using"@-FMAINV) involved the following chunk in the example:

(‘‘using’’ @-FMAINV) → (‘‘to fill’’ @-FMAINV) ↔ (‘‘auf die grosse’’ @ADVL)

This is very tenuous link as "auf die grosse" (to the size of) is a highly specific translation of "to fill". It would probably not arise often in the data and hence, the chunk Adaptability score is low. This is reflected in the overall low score of 0.32 above.

C.0.4 Partial-case matching. Threshold 0.5

CASE-175 Found Partial Match of strength 0.53846157 with CASE-F333

SL : (COREL COLOR MANAGER ALLOWS YOU TO CREATE A NEW PRINTER PROFILE OR REVISE AN EXISTING PROFILE)

TL: (THIS ALLOWS THE PRINTER TO READ THE TYPE ONE FONT RATHER=THAN HAVING THE FONTS CONVERTED TO CURVES OR BITMAPS)

TL′:(DADURCH KANN DER DRUCKER DIE TYPE 1-SCHRIFTART LESEN WAS ZU BESSEREN ERGEBNISSEN FUEHRT ALS DAS UMWANDELN VON SCHRIFTARTEN IN KURVEN ODER BITMAPS)

(7 :ADD <NONE> <NONE> (REVISE) <NONE>)

(1 :IGNORE DADURCH)

(2 :IGNORE KANN)

(6 :IGNORE WAS)

(7 :IGNORE ZU BESSEREN ERGEBNISSEN)

(10 :ADAPT (THE FONTS) @ADVL (AN EXISTING PROFILE) (VON SCHRIFTARTEN))

(6 :ADAPTZERO (RATHER=THAN) <NONE> (OR) (<NONE>))

(4 :ADAPT (THE TYPE ONE FONT) @OBJ (A NEW PRINTER PROFILE)

(DIE TYPE 1-SCHRIFTART))

(5 :ADAPT (TO READ) @+FMAINV (TO CREATE) (LESEN))

(3 :ADAPT (THE PRINTER) @SUBJ (YOU) (DER DRUCKER))

(2 <NONE>)

(1 :ADAPTZERO (THIS) <NONE> (COREL COLOR MANAGER) (<NONE>))

TL: ((corel farben farben-manager)) (dadurch) (kann) (sie) ((neue strecke)) ((erstellen)) (oder) (was) (zu besseren ergebnissen) ((≪(revise≫)) ((eine vorhandene zeichnung))

Translation lity 0: useless, 1: perfect = .615

Errors 4.

One agreement error: "kann" should be "koennen" to agree with the polite pronoun "Sie". two extra chunks "was" (*what*) and "zu better ergebnissen" (*to better results*). Both were non-matching chunks in the case, hence they were ignored (Chapter 6). While the other IGNORE operations are prefectly valid (and necessary), this chunk is adding irrelevant information to the TL. It links indirectly to something in the SL′ but the linker couldn't make the connection. In fact, this seems to be a translation mismatch (See Chapter 2) as shown below:

"rather than having to X"

"was zu bessere Erbebnisen fuehrt als das X"

what to better results leads as X = which leads to better results than X

This may be useful to a translator as the additions are only locally upsetting to the translation. The rest of the sentence is fine. If a monolingual German speaker was using ReVerb to get the *gist* of a text however, he or she might be rather confused.

Appendix D

The ReVerb Retriever Code

```
(defun reverb(l &optional manual)
  (setf *position-in-pchunk* 0)
  (setf *sl-list* nil)
  (setf *translation-list* nil)
  (if manual (setf *sl-list* (input-new l))
    (parse l))
  (get-case-name nil *sl-list* nil)
  (krell-demon-status :ON)
  (load "n1.lisp")

  (let ((probe-case (car (krell-children 'case))))
    (streamline-linear-order probe-case)

    (format t "~&ReVerb is now retrieving cases..")
```

```
    (format t "~& ~S" (ebmt-template probe-case))
    (let ((stream (open *out-data* :direction :output :if-ex
ists :append)))
      (find-and-show-matching-template probe-case 0 stream)
      (close stream))))



(defvar *out-data* "ReVerbOutput")
(defvar *template-hash* nil)

(defun ebmt-template(case &key (threshold 0))
  (let* ((chunks         (krell-get-local-values case 'chunk))
 (tchunks  (krell-get-local-values case 'gerchunk))
 (target-chunks (if tchunks (unlist (cons chunks
tchunks)) chunks))
         (num-list nil)
 (t-num-list nil)
 (nums          (dolist (c chunks num-list) (if (krell
-get-local-value c 'lin-order) (push (cons (krell-get-local-
value c 'lin-order) c) num-list))))
 (t-nums          (if tchunks (dolist (c target-chunks
 t-num-list) (if (krell-get-local-value c 'target-lin-order)
 (push (cons (krell-get-local-value c 'target-lin-order) c)
t-num-list)))))

 (size (caar (sort nums #'> :key #'first)))
 (t-size (if t-nums (caar (sort t-nums #'> :key'first))))
```

```
 (source-template nil)
 (target-template nil))

    (when chunks

    (dotimes (position size)

    (dolist (chunk chunks)
      (when (eq (krell-get-local-value chunk 'lin-
order) (+ position 1))

(if (>= (or (krell-get-local-value chunk 'mapping) 0) threshold)

    (push (make-template-variable (krell-get
-local-value chunk 'syn-function) position) source-template)
  (push (list  (krell-get-local-value chunk
'syn-function)(krell-get-local-values chunk 'source-text))
source-template))))))

    (when tchunks
     (dotimes (position t-size)

    (dolist (chunk target-chunks)
      (when (eq (krell-get-local-value chunk 'target
-lin-order) (+ position 1))

(if (>= (or (krell-get-local-value chunk
```

```
'mapping) 0) threshold)

(push (make-template-variable (krell-get
-local-value chunk 'target-syn-function) position) target-tem
plate)
   (push (list  (krell-get-local-value chunk
 'syn-function)(krell-get-local-values chunk 'target-text))
target-template))))))

     (krell-set-values case 'target-template (nreverse target
-template))
     (krell-set-values case 'source-template (nreverse source
-template))))




;; Generate a template variable that codes for syntactic
;; function

(defun make-template-variable(syn pos)
  (list syn
(read-from-string (format nil "?~D" pos))))

(defun create-template-table()
  (setq *template-hash* (make-hash-table :size 1000 :test
'equal)))
```

```
(defun index-for-retrieval(case)
  (streamline-linear-order case)

  (let ((similar-case-list (gethash (source-template case)
*template-hash*)))

    (if similar-case-list
(setf (gethash (source-template case) *template-hash*)
      (push case similar-case-list))
      (setf (gethash (source-template case) *template-hash*)
    (list case)))))

(defun index-all(&optional limit)
  (create-template-table)
  (clrhash *template-hash*)
(dolist (c (nthcdr (- (length (krell-children 'case))
 limit) (krell-children 'case)))
  (index-for-retrieval c)
  (format t " ~S" c)
"finished"))

;;* Simple accessors

(defun source-template(case)
  (reverse (krell-get-local-values case 'source-template)))
```

```
(defun target-template(case)
  (krell-get-local-values case 'target-template))



;;* compare the templates of two cases


(defun compare-templates(case1 case2)
  (unify::unify
   (if (listp case1) case1 (source-template case1))
   (if (listp case2) case2 (source-template case2))))



;;* Build templates for the cases in memory


(defun build-all-templates(&key (threshold 0))
  (let ((count 0))

    (dolist (case (krell-get-values 'case 'children))
       (when (krell-get-local-values case 'source-text)

     (incf count)
     (ebmt-template case :threshold threshold)
     (format t "~&[~S] ~S~&" case (krell-get-local-
values case 'source-template) )))

    (format t "~&~D cases templatized.~&" count)
    "o.k"))
```

```
(defun find-matching-template(probe)
  (let ((found nil))

    (dolist (case (gethash (ebmt-template probe) *template-
hash*))
       (unless (eq probe case)
   (when (krell-get-local-values case 'source-template)
     ;(format t "~&~S" case)
 (let ((match? (compare-templates probe case)))
   (when (listp match?)
 (push (list case (cons (sim-cases
probe case) (carry-across-adaptation probe case)))
       found))))))
    found))



(defun find-and-show-matching-template(probe threshold
&optional outf)
  (let ((found 0))

     (when (gethash (ebmt-template probe) *template-hash*)
    (dolist (case (reverse (gethash (ebmt-template probe)
*template-hash*)))
```

```
       (unless (eq probe case)
   (when (krell-get-local-values case 'source-template)
     (let ((ex-template (ebmt-template case :threshold
 threshold)))
   (format t "~& ~S" ex-template)
   (let ((match? (unify::unify (ebmt-template
 probe :threshold threshold) ex-template)))
     (when (listp match?)

 (incf found)
 (let ((source  (krell-get-local-
values probe 'source-text))
       (example-text  (krell-get-local
-values case 'source-text))
       (score   (sim-cases probe case))
       (adapt   (carry-across-adaptation
probe case)))



   (format t "~&[~S]  Found Match of
 strength ~S with [~S]~&" probe (+ 0.0 score) case)
   (if outf (format outf "~&[~S]  Found Match of strength ~S with [~S]~&" probe (+ 0.0 score)
 case))
   (dolist (a adapt) (format t "~&~S" a))
   (if outf (dolist (a adapt) (format
 outf "~&~S" a)))
```

```
  (format t "~&Probe  :")
  (dolist (word source) (princ " ")
(princ word))
  (terpri)
  (if outf (format outf "~&Probe  :
~S  " source))


  (format t "~&Example: ~S" example-text)
  (if outf (format outf "~&Example: ~S" example-text))


  (if outf (translate-with-adaptation case outf adapt)
    (translate-with-adaptation case outf adapt))))


    ; *********    Partial Matching

    (when (equal found 0)
(format t "~&PARTIAL MATCH")
(let* ((case-list (template-retrieval probe)))
  (dolist (candidate case-list)
    (let* ((case (second candidate))
   (pprobe (third candidate))
   (pcase (fourth candidate))
   (case-left (fifth candidate))
   (probe-left  (sixth candidate))
```

```
    (source  (krell-get-local-values probe
 'source-text))
    (example-text  (krell-get-local-values case
 'source-text))
    (score (car candidate))
    (adapt   (carry-across-adaptation-partial
 probe case pprobe pcase case-left probe-left)))
  (format t "~&[~S]  Found Partial Match
 of strength ~S with [~S]~&" probe (+ 0.0 score) case)
 (if outf (format outf "~&[~S]
Found Partial Match of strength ~S with [~S]~&" probe
(+ 0.0 score) case))
 (dolist (a adapt) (format t "~&~S" a))
 (if outf (dolist (a adapt) (format

 outf "~&~S" a)))

 (format t "~&Probe  :")
 (dolist (word source) (princ " ")
(princ word))
 (terpri)
 (if outf (format outf "~&Probe  : ~S
  " source))

 (format t "~&Example: ~S" example-text)

 (if outf (format outf "~&Example: ~S"
```

```
 example-text))

 (if outf (translate-with-adaptation
 case outf adapt)
   (translate-with-adaptation case
 nil  adapt)))

found))




;;* Find all possible matches in a case-base

(defun find-all-matches(&optional (threshold 0))

  (Let ((fname nil))
    (format t "Put output in which file? [~a]: " *out-data*)

    (let ((given (read-line)))
      (setf fname (if (equal given "") *out-data* given)))

    (let ((threshold nil))
      (format t "~&Specify Threshold (press return for default
 = 0)")

      (let ((given (read-from-string (read-line))))
(setf threshold (if (equal given "") nil given)))
```

```
    (find-all-matches-file fname threshold))))

(defun find-all-matches-file(fname &optional threshold)

  (let ((total 0)
(stream (open fname :direction :output)))

    (format t "~&Find all matches in ~D cases ...~&~&" (length
(krell-children 'case)))
(format stream "~&Find all matches in ~D cases ...~&~&" (length
 (krell-children 'case)))
    (dolist (case (nthcdr 80 (krell-get-values 'case 'children)))
(when (krell-get-local-values case 'source-text)
  (format t "~& ~S" case)
      (incf total (find-and-show-matching-template
case threshold stream))))

  (format t "~&~D matches found.~&" (/ total 2))
  (format stream "~&~D matches found.~&" (/ total 2))

      (close stream)
      "o.k"))



;partial case matching

(defun template-retrieval(probe)
```

```
  (let ((bigstore nil)
(bigstore-vals nil)
(count 0))

    (dolist (cell (reverse (source-template probe)) bigstore)

      (push (one-cell count cell) bigstore)
      (incf count))
    (format t "~&best structure: ~S" (car (check-most-freq
bigstore)))
    (setf bigstore (reverse (delete-nils (check-most-freq
bigstore))))

    (dotimes (i 2)
      (when bigstore
(format t "~&Checking ~S and ~S" probe (caar
bigstore))
(let* ((case (caar bigstore))
       (sim (partial-assess-sim probe case))
       (pcase (first sim))
       (deletezero (krell-get-values case 'gerchunk))
       (pprobe (second sim))
       (add (third sim))
       (delete (fourth sim))
       (ignore (fifth sim))
       (adaptzero (sixth sim))
```

```
       (score (/ (if (listp pprobe) (length pprobe) 1)
 (+
  (if (listp add) (length add) 0.2)
  (if (listp delete) (length delete) 0.2)
  (if (listp ignore) (length ignore) 0.2)
  (if (listp deletezero) (length
deletezero) 0.2)
  (if (listp adaptzero) (length
adaptzero) 0.2)))))

  (push (list score case pprobe pcase delete add)
bigstore-vals))
(setf bigstore (cdr bigstore))))
    (sort bigstore-vals '> :key 'car)))




(defun one-cell(count cell)
  (let ((store nil))
    (maphash #'(lambda (key val) (if (equal (car (nthcdr
count  key)) cell)  (push val store)))  *template-hash*)
     store))

(defun partial-assess-sim(probe case)
  (let ((add-chunk-list nil)
(delete-chunk-list nil)
(common-chunk-list-probe nil)
```

```
(common-chunk-list-case nil)
(adaptzero-chunk-list nil)
(deletezero 0)
(ignore-chunk-list  nil))


    (dolist (chunk-1 (krell-get-values probe 'chunk))
      (when chunk-1
      (let* ((syn-fun (krell-get-local-value chunk-1
'syn-function))
     (lin-ord (krell-get-local-value chunk-1
'lin-order))
     (corrs   (krell-get-values case syn-fun))
     (corr    (if (cdr corrs)
  (dolist (try corrs (first corrs))


    (if (eq lin-ord (krell-get-local
-value try 'lin-order))
(return try)))
(if (eq lin-ord (krell-get-local-
value (first corrs) 'lin-order)) (first corrs)))))



(cond ((or (null corr) (member corr common-chunk-list
-case)) (push chunk-1 add-chunk-list))
      ((equal (krell-get-value corr 'mapping) 0) (progn
(push corr adaptzero-chunk-list)
(push chunk-1 common-chunk-list-probe) (push corr common
```

```
-chunk-list-case)))
      (t (progn (push chunk-1 common-chunk-list-probe)
(push corr common-chunk-list-case)))))))



    (dolist (chunk-2 (krell-get-values case 'chunk))
      (let* ((syn-fun (krell-get-local-value chunk-2 'syn-fun
ction))
     (lin-ord (krell-get-local-value chunk-2 'lin-order))
     (corrs   (krell-get-values probe syn-fun))
     (corr    (if (cdr corrs)
  (dolist (try corrs (first corrs))

                          (if (eq lin-ord (krell-get-local-
value try 'lin-order))
                           (return try)))
     (if (eq lin-ord (krell-get-local-value
(first corrs) 'lin-order)) (first corrs)))))


(cond ((and (null corr) (> (krell-get-value chunk-2
'mapping) 0))
       (push chunk-2 delete-chunk-list))
      ((null corr) (push chunk-2  ignore-chunk-list)))))
      ;(t (push chunk-2 common-chunk-list-case)))))



    (list (or common-chunk-list-case '<nocommon-case>)  (or
```

```
common-chunk-list-probe '<nocommon-chunk>) (or add-chunk-list
'<no-add>) (or delete-chunk-list '<no-delete>)
  (or ignore-chunk-list '<no-ignore>) (or adaptzero
-chunk-list '<noadaptzero>))))



(defun retrieve-on-structure(case  &key (display t) (structu
re nil))
  (let ((source nil))
    (dolist (word (krell-get-values case 'source-text))

        (push word source))

    (when display
      (format t "~&Source String: ")
      (dolist (word (reverse source))
        (format t " ~s" word))
      (terpri))

    (let* ((lexical-filter    (retrieve-on-sentence source))
 ;;:structure t :display display))
   (structural-filter (reward-structure lexical-filt
er case)))

      (display-returned-cases  structural-filter))))
```

Appendix E

The ReVerb Parser Code

```
(defun parse (sentence &optional previous-case previous-chunk)

    ;indexing any cases whose chunks contain this word..
  (let*
      ((word (car sentence))
       (chunk-list (krell-get-values word 'appears-in-chunk))
       (all-cover-cases nil)
       (current-case nil)
       (current-chunk nil))

   (when sentence
     (when (and chunk-list (>= (length chunk-list) 1))
       (progn
 (dolist (chunk chunk-list all-cover-cases)
```

```
   (dolist (cover-case (krell-get-values chunk 'case
-of) all-cover-cases)
     (push cover-case all-cover-cases)))
; Now we have a set of cases which contain a chunk covering
; the current word of input. Now check to see if it is the
; case we used to cover the previous substring



 (cond ((and (member previous-case all-cover-cases
:test 'equal)
     (member previous-chunk chunk-list :test
 'equal))

(progn
       (setf current-case previous-case)
       (setf current-chunk (cadr (member
previous-chunk (krell-get-values current-case 'chunk))))))

; SAME CASE DIFFERENT CHUNK


       ((member previous-case all-cover-cases :test
'equal)
(let ((cover (find-longest-cover sentence
previous-case)))
  (setf current-case (car cover))
  (setf current-chunk (cdr cover))))
```

```
; DIFFERENT CASE DIFERENT CHUNK


       (t (progn (let ((cover (find-longest-cover
sentence)))
   (setf current-case (car cover))
   (setf current-chunk (cdr cover))))))

; Now we have a case and its chunk to follow, push the
; current word onto a parse chunk which'll assume the
; characteristics of the cover-chunk

 (let ((parse-chunk-text (list word)))
   (if (and (> (length (krell-get-values current-
chunk 'source-text)) 1)
    (cdr sentence))


       (if (cdr sentence)
   (continue-case parse-chunk-text (cdr
sentence) current-case current-chunk)
 (write-parse-chunk parse-chunk-text current
-chunk))
     (progn (write-parse-chunk parse-chunk-text
current-chunk)
(parse (cdr sentence) current-case current-
chunk))))))


       (unless  chunk-list (write-parse-chunk word '<none>)
```

```
(parse (cdr sentence) previous-case previous
-chunk))))

   *sl-list*)


(defun continue-case(parse-chunk-text sentence current-case
current-chunk)

; at this stage we are comitted to one chunk and have read
; one word in the current chunk is not exhausted but it may
; not match!

 (when sentence
    (let ((word (car sentence)))
      (if (and (member word  (cdr (krell-get-values current-
chunk 'source-text)) :test 'equal)
       (> (length (krell-get-values current-chunk
'source-text)) (length parse-chunk-text)))
  (progn (push word parse-chunk-text)

 (if (and (cdr sentence) (not (or (equal (kr
ell-get-values current-chunk 'source-text) parse-chunk-text)
 (equal (krell-get-values current-chunk 'source-text)
(reverse parse-chunk-text)))))
     (continue-case parse-chunk-text  (setf
sentence (cdr sentence)) current-case current-chunk)
```

```
   (progn (write-parse-chunk parse-chunk-text
 current-chunk)
  (if (cdr sentence)
      (parse (cdr sentence) current-
case current-chunk)))))
(let ((stat-pos (determine-statistical-pos word)))
 (cond ((or (equal stat-pos 'singleton)(equal stat-
pos 'left-peripheral))
 (progn (write-parse-chunk parse-chunk-text
current-chunk)

(format t "~&cut, new parse with  ~S"
 word)
(parse sentence current-case current-
chunk)))
((or (equal stat-pos 'middle) (equal stat-pos
 'unknown))

 (progn (push word parse-chunk-text)
(if (cdr sentence)
    (continue-case parse-chunk-text
(cdr sentence) current-case current-chunk)
    (write-parse-chunk parse-chunk
-text current-chunk))))



((equal stat-pos 'right-peripheral)
```

```
 (progn (push word parse-chunk-text)
(write-parse-chunk parse-chunk-text
current-chunk)
(parse (cdr sentence) current-case
current-chunk))))))))
 "finito")



(defun find-longest-cover(sentence &optional previous-case)
  (let ((one-word nil)

(cclist NIL)
(ans nil)
(candidate-chunks (krell-get-values (first sentence)
 'appears-in-chunk)))

    (dolist (c candidate-chunks one-word)
      (if (equal (first sentence) (first (krell-get-
values c 'source-text)))
  (push c one-word)))

    (if previous-case
(progn (setf cclist (cons previous-case (car (inters
ection (krell-get-values previous-case 'chunk) candidate-
chunks :test 'equal))))
       (setf ans 'Some)))
```

```
    (unless previous-case
      (when one-word

(let* ((two-words (if (cdr sentence)
    (intersection
     one-word (krell-get-values (sec
ond sentence) 'appears-in-chunk))))
   (three-words (if (third sentence)(intersection
two-words (krell-get-values (third sentence)'appears-in-chunk))))
   (four-words (if (fourth sentence)(intersection
three-words (krell-get-values (fourth sentence)'appears-in-
chunk)))))

  (cond (four-words (progn (setf cclist (cons (krell-
get-value (car four-words) 'case-of) (car four-words))) (setf
 ans '4)))
(three-words (progn (setf cclist (cons (krell
-get-value (car three-words) 'case-of) (car three-words)))
(setf ans '3)))
(two-words (progn (setf cclist (cons (krell-
get-value (car two-words) 'case-of) (car two-words)))
(setf ans '2)))
  (t  (let ((best-single-cover (find-best-single-cover
 one-word)))
(setf cclist (cons (krell-get-value best-
single-cover 'case-of) best-single-cover))
(setf ans '1)))))))
```

```
    (format t "~& ~S words in a row  from ~S [~S] " ans
(krell-get-values (cdr cclist) 'source-text) (car cclist))
    cclist))

; this starts off the search for a new case when you know
; that the previous case cannot be followed anymore..



(defun find-best-single-cover (chunk-list)
  (let ((common-features nil)
(most-common-feature nil)
(good-case nil))

(format t "~&position >>>>>> ~S~&" *position-in-
pchunk*)
    (dolist (c chunk-list most-common-feature)
      (push (length (krell-get-values c 'source-text)) common
-features))
    (setf most-common-feature (check-most-freq common-
features))
    (format t "~S" most-common-feature)

    (unless good-case
   (dolist (c chunk-list)
     (unless good-case
     (cond ((and (equal (length (krell-get-values c
'source-text)) (car most-common-feature))
```

```
 (equal (krell-get-value c 'lin-order
) *position-in-pchunk*))
    (progn (setf good-case c) (format t "~&
pos and length match")))


   ((equal (krell-get-value c 'lin-order)
*position-in-pchunk*)
    (progn (setf good-case c) (format t "~&
just position match")))
   ((equal (length (krell-get-values c
'source-text)) (car most-common-feature))
    (progn (setf good-case c) (format t
"~& just length match")))))))



    (format t "GOOD CASE ~S lin-ord: ~S" good-case (krell-
get-value good-case 'lin-order))
    (if good-case good-case (car chunk-list))))




(Defun determine-statistical-pos(word)
    (when word
      (let ((chunk-list (krell-get-values word 'appears-in
-chunk))
  (chunk-length-vals nil)
  (lpscore 0)
```

```
  (rpscore 0)
  (verdict nil))


      (when chunk-list
;(format t "~&~S" chunk-list)
(dolist (c chunk-list)
  (let ((length-chunk (or (length (krell-get-values
c 'source-text)) 1))
(left-peripheral (equal (length (member word
 (reverse (krell-get-values c 'source-text)) :test 'equal))
1))
(right-peripheral (equal (length (member word
 (krell-get-values c 'source-text) :test 'equal)) 1)))
    (if right-peripheral (progn (incf lpscore)))
    (if left-peripheral (progn (incf rpscore)  ))))
(cond  ((> (- (+ lpscore rpscore) (length
chunk-list)) 10)
       (setf verdict 'singleton))
       ((> lpscore (/ (length chunk-list) 2))
       (setf verdict 'left-peripheral))
      ((> rpscore (/ (length chunk-list) 2))
       (setf verdict 'right-peripheral))


      (t (setf verdict 'middle))))


      (unless chunk-list (setf verdict 'unknown))
```

```
      verdict)))



(defun write-parse-chunk (words current-chunk)
   (let ((parse-chunk-text (or (if (atom words) (list words)
 (reverse words)) '<none>))
 (parse-chunk-syn-function (or (krell-get-value curr
ent-chunk 'syn-function) '<none>))
 (parse-chunk-lin-order (or (incf *position-in-pchunk
*) '<none>))
 (parse-chunk-cat (or (krell-get-value current-chunk
 'cat) '<none>))
 (translation-words (or (krell-get-values current
-chunk 'target-text) '<none>))
 (translation-syn-function (or (krell-get-value curr
ent-chunk 'target-syn-function) '<none>))
 (translation-lin-order (or (krell-get-value current-
chunk 'target-lin-order) '<none>))
 (translation-cat (or (krell-get-value current-chunk
 'target-cat) '<none>)))



    (push  (list (list translation-words parse-chunk-text)
 (list translation-syn-function parse-chunk
-syn-function)
 (list translation-lin-order parse-chunk-lin
-order)
```

```
 (list translation-cat  parse-chunk-cat)
 (krell-get-values current-chunk 'source-text)
 current-chunk) *translation-list*)
     (push (list parse-chunk-text parse-chunk-syn-function
parse-chunk-cat parse-chunk-lin-order) *sl-list*)))
```

Bibliography

[ABD+86] V. Allegranza, P. Bennett, J. Durand, F. van Eynde, L. Humphreys, P. Schmidt, and E. Steiner. Linguistics for machine translation: The eurotra linguistic specifications. Technical report, CEC, 1986.

[AE95] S. P. Abney and W. H. Erhard, editors. *Proceedings of the Seventh Conference of the European Chapter of the Association for Computational Linguistics EACL*, Dublin, Ireland, 1995. ACL.

[Alt95] K. D. Althoff. Knowledge acquisition in the domain of cnc machine centres; the moltke approach. In S. P. Abney and W. H. Erhard, editors, *Proceedings of the Seventh Conference of the European Chapter of the Association for Computational Linguistics EACL*, pages 180–195, Dublin, Ireland, 1995. ACL.

[AP94] A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AICOM*, 7:39–59, 1994.

[Art78] P. Arthern. Machine translation and computerized terminology systems: A translator's viewpoint. In B. Snell, editor, *Translating and the Computer: Proceedings of a Seminar*, pages 77–108, London, 1978. North Holland.

[BCDP+88] P. F. Brown, J. Cocke, S. A. Della-Pietra, , V. J. Della-Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roosin. A statistical approach to machine translation.

In *Proceedings of the 12th International Conference on Computational Linguistics*, Budapest, Hungary, 1988.

[BDPd+93] P. F. Brown, V. J. Della-Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer. Class-based n-gram models of natural language. *Computational Linguistics*, 18, 1993.

[BDPDPM92] P. F. Brown, S. A. Della-Pietra, V. J. Della-Pietra, and R. L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19:263–311, 1992.

[BF95] R. Brown and R. Frederking. Applying statistical english language modelling to symbolic machine translation. In van Eynde [vE95], pages 221 – 239.

[BLM91] P. F. Brown, J. C. Lai, and R. L. Mercer. Aligning sentences in parallel corpora. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, Berkeley, CA, 1991. ACL.

[Boi89] C. Boitet. Geta project. In M. Nagao, editor, *Machine Translation Summit*, pages 54–65. Ohmsha Tokyo, 1989.

[Bos86] D. A. Bostad. Machine translation in the usaf. *Terminologie et Traduction*, 1:68–72, 1986.

[Bro97] R. Brown. Automated dictionary extraction for knowledge free example-based translation. In Nirenburg and Somers [NS97], pages 111–119.

[Car97] M. Carl. Case composition needs adaptation knowledge: a view on ebmt. In W. Daelemans, editor, *ECML-97 MLNet Workshop: Case-Based Learning: Beyond Classification of Feature Vectors*, Prague, 1997.

[CC96] B. Collins and P. Cunningham. Adaptation guided retrieval in ebmt. In Smith and Faltings [SF96], pages 91–104.

[CC97] B. Collins and P. Cunningham. Adaptation guided retrieval: Approaching ebmt with caution. In Nirenburg and Somers [NS97], pages 119–126.

[Cho65] N. Chomsky. *Aspects of the Theory of Syntax*. MIT Press, Cambridge MA, 1965.

[CHP94] L. Cranias, P. Harris, and S. Piperidis. A new approach to matching technique in example-based machine translation. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, Las Cruces, New Mexico, 1994. ACL.

[CS93] S. Cost and S. Salzberg. A weighted nearest neighbour algorithm for learning with symbolic features. *Machine Learning*, 10:57–78, 1993.

[CSV95] P. Cunningham, B. Smyth, and T. Veale. On the limitations of memory based reasoning. In J. P Haton, M. Keane, and M. Manago, editors, *Advances in Case-Based Reasoning*. Springer-Verlag, 1995.

[CT87] J. Carbonell and M. Tomita. Knowledge-based machine translation, the cmu approach. In S. Nirenburg, editor, *Machine Translation: Theoretical and Methodological Issues*, pages 68–89. Cambridge University Press, 1987.

[Cun98] P. Cunningham. Cbr strengths and weaknesses. In A. P del Pobil, J. Mira, and M. Ali, editors, *Proceedings of the 11th International Conference on Inductrial and Engineering Applications of Artificial Intelligence and Expert Systems*, Lecture Notes in Artificial Intelligence 1416, page 3. Springer-Verlag, 1998.

[Dae95] W. Daelemans. Memory based lexical acquisition and processing. In P. Steffens, editor, *Machine Translation and the Lexicon*, pages 3–12. Springer, 1995.

[DMM93] I. Dagan, S. Marcus, and S. Markovitch. Contextual word similarity and estimation from sparse data. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, Columbus, OH, 1993. ACL.

[Dor93] Bonnie J. Dorr. *Machine Translation. A View from the Lexicon*. MIT Press, Cambridge MA, 1993.

[DZBG96] W. Daelemans, J. Zavrel, P. Berck, and S. Gillis. Mbt: A memory-based part of speech tagger-generator. In I. Dagan and E. Ejerhed, editors, *Proceedings of the Fourth Workshop on Very Large Corpora, ACL SIGDAT*, pages 14–27, Copenhagen, 1996.

[FM97] P. Fung and K. McKeown. A technical word and term translation aid using noisy parallel corpora across language groups. *Machine Translation*, 12:53–87, 1997.

[Fun95] P. Fung. A pattern matching method for finding noun and proper noun translations from noisy parallel corpora. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, Boston, MA, 1995. ACL.

[GC93] W. A. Gale and K. W. Church. A program for aligning sentences in bilingual corpora. *Computational Linguistics*, 19:75–103, 1993.

[HK96] K. Hanney and M. Keane. Learning adaptation rules from a case-base. In Smith and Faltings [SF96], pages 179–192.

[Hov96] E. Hovy, editor. *Expanding MT Horizons - Second Conference of the Association for Machine Translation in the Americas - AMTA-96*, Montreal, Canada, 1996.

[HS92] W. J. Hutchins and H. L. Somers. *An Introduction to Machine Translation.* Academic Press, London, 1992.

[Isa93] P. Isabelle. Current research in machine translation: A reply to somers. *Machine Translation*, 7:265–273, 1993.

[Jac83] R. S Jackendoff. *Semantics and Cognition.* MIT Press, Cambridge, 1983.

[Juo95] P. Juola. *Learning to Translate: A Psycholinguistic Approach to the Induction of Grammars and Transfer Functions.* PhD thesis, University of Colorado, Boulder, USA, 1995.

[Kar90] H. Karlgren, editor. *Papers Presented to the 13th International Conference on Computational Linguistics*, volume 3, Helsinki, 1990.

[Kas94] A. Kass. Tweaker: adapting old explanations to new situations. In R. Schank, C. Riesbeck, and A. Kass, editors, *Inside Case Based Reasoning*, pages 263–295. Lawrence Erlbaum Associates, 1994.

[KG94] N. Koncar and G. Guthrie. A natural language translation neural network. In *International Conference of the International Conference on New Methods in Language Porcessing (NeMLaP)*, pages 71–77, Manchester, UK, 1994.

[KG97] K. Knight and J. Graehl. Machine transliteration. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, Madrid, Spain, 1997. ACL.

[KH91] H. Kitano and T. Higuchi. Massively parallel memory-based parsing. In *Proceedings of the thirteenth International Joint Conference on Artificial Intelligence*, pages 918–924, Chamberry, France, 1991. IJCAI.

[Kin87] M. King, editor. *Machine Translation today the state of the art.* Edinburgh Information Technology Series 2. Edinburgh University Press, Edinburgh, 1987.

[KKM92] H. Kaji, Y. Kida, and Y. Morimoto. Learning translation templates from bilingual text. In *Papers Presented to the 14th International Conference on Computational Linguistics*, pages 672–678, Nantes, 1992.

[Kol83] J. Kolodner. Reconstructive memory: A computer model. *Cognitive Science*, 7(4), 1983.

[Kol93] J. Kolodner. *Case-Based Reasoning.* Morgan Kaufmann, San Mateo, 1993.

[KR93] M. Kay and M. Röscheisen. Text-translation alignment. *Computational Linguistics*, 19:121–143, 1993.

[KVHA95] F. Karlsson, A. Voutilainen, J. Heikkila, and A. Anttila, editors. *Constraint Grammar - A language-Independent System for Parsing Unrestricted Text.* Natural Language Processing. Mouton de Gruyter, Berlin New York, 1995.

[Lea94] D. Leake. Towards a computer model of memory search strategy learning. In *Proceedings of the 16th Annual Conference of the Cognitive Science Society*, pages 549–554, Atlanta, 1994.

[Leb83] M. Lebowitz. Memory-based parsing. *Artificial Intelligence*, pages 363–404, 1983.

[LKW95] D. Leake, A. Kinley, and D. Wilson. Learning to improve case adaptation by introspective reasoning and cbr. In M. Veloso and Aamodt A., editors, *Case-based Reasoning Research and Development: Lecture Notes in AI 1010*, pages 229–240. Springer, 1995.

[LOS89] J. Landsbergen, J. Odijk, and A. Schenk. The power of compositional ttranslation. *Literary and Linguistic Computing*, 4:191–199, 1989.

[LT91] J. Lindop and J. Tsujii. Complex transfer in mt: A survey of examples. Technical report, Centre for Computational Linguistics, UMIST, 1991.

[Mel81] A. Melby. A bilingual concordance system and its use in linguistic studies. In *Proceedings of the Eight LACTUS Forum*, pages 541–549, Columbia SC, 1981. Hornbeam Press.

[Mel96a] D. I. Melamed. A geometric approach to mapping bitext correspondance. In *Proceedings of the First Conference on Empirical Methods in Natural Language Processing EMNLP 96*, pages 1–10, Philadelphia, PA, 1996.

[Mel96b] Dan I Melamed. Automatic construction of clean broad-coverage translation lexicons. In Hovy [Hov96], pages 125–134.

[Mel98] D. I. Melamed. *Empirical Methods for Exploiting Parallel Texts.* PhD thesis, University of Pennsylvania, Pennsylvania, 1998.

[MEO95] T. Mc Enery and M. Oakes. Cognate extraction in the crater project: Methods and assessment. In E. Tzoukermann, editor, *Proceedings of the ACL SIGDAT Workshop*, pages 77–86, Dublin, Ireland, 1995. ACL.

[Min63] M. Minsky. Steps towards artificial intelligence. In E. Feigenbaum and J. Feldman, editors, *Computers and Thought*. McGraw-Hill, 1963.

[MJS94] I. McLean, D. Jones, and H. Somers. Experiments in multilingual example-based generation. In A. Monaghan, editor, *Proceedings of the third International Conference on the Cognitive Science of Natural Language Processing*, Dublin, Ireland, 1994. DCU.

[MS89] D Maxwell and K. Schubert. *Metataxis in practice: dependency syntax for multilingual machine translation.* Foris, Dordrecht, 1989.

[Nag84] M. Nagao. A framework of a mechanical translation between japanese by analogy principle. In A. Elithorn and R. Banerji, editors, *Artificial and Human Intelligence*, pages 173–180. North Holland Publications, 1984.

[Nir95] S. Nirenburg. Cmu-cmt-95-145 the pangloss mark iii machine translation system. Technical report, Computing Research Laboratory (NMSU) Center for Machine Translation (CMU), Information Sciences Institute (USC), 1995.

[NS97] Sergei Nirenburg and Harold Somers, editors. *Seventh International Conf. on Theoretical and Methodological Issues in Machine Translation*, Santa Fe, New Mexico, 1997.

[Pod92] J. Podeur. La trasposizione. In *La practica delle traduzione*. Liguoiri Editore, 1992.

[RS89] R. K. Reisbeck and R. C. Schank. *Inside Case-Based Reasoning.* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1989.

[SA77] R. C. Schank and R. P. Abelson. *Scripts, Plans, Goals, and Understanding.* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1977.

[SF96] Ian Smith and Boi Faltings, editors. *Third European Workshop, EWCBR-96, Advances in Case-Based Reasoning*, Lausanne, Switzerland, 1996.

[SH97] A. Sagvall Hein. Language control and machine translation. In Nirenburg and Somers [NS97].

[SK93] B. Smyth and M. Keane. Retrieving adaptable cases. In I. Smith and B. Faltings, editors, *Proceedings of the First European Workshop on Case-Based Reasoning, EWCBR-93, Volume 1*, pages 76–82, Germany, 1993.

[SK94] B. Smyth and M. Keane. Retrieving adaptable cases. In M. Richter, S. Wess, and K. Althof, editors, *Topics on Case-Based Reasoning. Lecture Notes on AI*, pages 209–220. Springer-Verlag, 1994.

[SK95] B. Smyth and M. Keane. Remembering to forget: A competence-preserving deletion policy in case-based systems. In *Proceedings of the fifteenth International Joint Conference on Artificial Intelligence*. IJCAI, 1995.

[SN90] S. Sato and M. Nagao. Towards memory-based translation. In Karlgren [Kar90], pages 247–252.

[SOI+93] E Sumita, K. Oi, H. Iida, T. Higuchi, N. Takahashi, and H. Kitano. Ebmt on massively parallel processors. In *Proceedings of the thirteenth International Joint Conference on Artificial Intelligence*, pages 1283–1288, Chamberry, France, 1993. IJCAI.

[Som98] H. Somers. Further experiments in bilingual text alignment. *International Journal of Corpus Linguistics*, 3:1–36, 1998.

[SSW96] O. Streiter and A. Schmidt-Wigger. Patterns of derivation. In Hovy [Hov96], pages 256–272.

[Str95] O. Streiter. Linguistic reference manual for the cat2 machine translation system. Technical report, IAI, Saarbrücken, 1995.

[SV90] V. Sadler and R. Vendelmans. Pilot implementation of a bilingual knowledge bank. In Karlgren [Kar90].

[SW86] C. W. Stanfill and D. L. Waltz. Toward memory-based reasoning. *Communications of the ACM*, 29(12), 1986.

[Tsu89] J. Tsujii. Machine translation: Current research trends. In I. Bátori, editor, *Computational Linguistics, An International Handbook on Computer Oriented Language Research and Application*. Walter de Gruyter, 1989.

[Tul72] E Tulving. Episodic and semantic memory. In E. Tulving and W. Donaldson, editors, *Organization of Memory*. Academic, 1972.

[Tur50] A. Turing. Computing machinery and intelligence. *Mind*, 59:433–460, 1950.

[VB85] B. Vauquois and C. Boitet. Automated translation at grenoble university. *Comptational Linguistics*, 11:28–36, 1985.

[VC96] T. Veale and B. Collins. Space, metaphor, and schematization in sign: Sign language translation in the zardoz system. In Hovy [Hov96], pages 157–168.

[vE95] Frank van Eynde, editor. *Sixth International Conf. on Theoretical and Methodological Issues in Machine Translation*, Leuven, Belgium, 1995.

[VJ95] A. Voutilainen and T. Jarvinen. Specifying a shallow grammatical representation for parsing purposes. In Abney and Erhard [AE95], pages 210–214.

[Vou95] A. Voutilainen. A syntax-based part-of-speech analyser. In Abney and Erhard [AE95], pages 157–164.

[VW97] T. Veale and A. Way. Gaijin: A bootstrapping approach to example-based machine translation. In R. Mitkov, editor, *International Conference, Recent Advances in Natural Language Processing*, pages 239–244, Tzigov Chark, Bulgaria, 1997.

[Wu95] D. Wu. Grammarless extraction of phrasal translation examples from parallel texts. In van Eynde [vE95], pages 354–372.