Phrase-based Memory-based Machine Translation

Maarten van Gompel
S232231
HAIT Master Thesis series nr. 09-003

Thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Arts in Communication and Information Sciences,
Master Track Human Aspects of Information Technology,
at the Faculty of Humanities
of Tilburg University

Thesis committee:

Prof. Dr. A.P.J van den Bosch
Drs. P.J. Berck
Dr. M.M. van Zaanen

Tilburg University
Faculty of Humanities
Department of Communication and Information Sciences
Tilburg, The Netherlands
20 July 2009

Abstract

This master thesis aims to investigate a phrase-based approach of Memory-based Machine Translation. This is a form of automatic translation powered by lazy-learning classifiers to translate fragments of the input sentence. A parallel corpus serves as the basis for training such a classifier. In the phrase-based approach the principal component of these fragments is a phrase of arbitrary length. This can be contrasted to prior research in the field in which this component was a single word. A key element in the research is a comparison of three methods of phrase extraction. A new decoder has been developed to deal with the characteristics unique to this approach, and re-assemble the translated fragments into one final translation. This research will show that one of the proposed phrase-extraction methods is capable of outperforming previous word-based approaches, even though this gain is limited and the impact of phrases proves to be smaller than anticipated.

Contents

List of Figures

List of Tables

List of Algorithms

Preface

Ever since my early teens, I have had a keen interest in both language and technology. Initially these interests were quite separate. The language aspect started with the playful creation of constructed languages, which as I grew older moved away to make place for more a serious acquisition of several foreign languages. The technology aspect has for me always been a focus on software and the development thereof, starting out in the form of a child's first playful coding attempts in `BASIC`, and moving later into more serious development with an increasing focus on the world wide web. In my late teens, the passion for language and technology converged and culminated into me co-founding UniLang, a popular on-line language-learning community.

Seeking a further synthesis between my interests in language and technology, I several years later enrolled in the master track "Human Aspects of Information Technology", at the University of Tilburg. This thesis is to be the final fulfilment in the completion of this track. With the subject of the thesis being *machine translation*, automatic translation by computers, it is clear that both two interests carried substantial weight in the determination of the subject. I consider machine translation to be one of the most fascinating studies in the field of Natural Language Technology.

This thesis would not have been possible without the assistance of several people: First of all my supervisor Antal van den Bosch, who has been supervising and guiding me throughout my research, in spite of his busy schedule, and on whose previous research this thesis is founded. A second word of gratitude goes to my friend Rogier Kraf, who kindly volunteered to read my thesis and provided valuable comments to improve it. Last but not least, thanks also goes out to my boyfriend, for his indispensable emotional support and keeping me well-fed during the writing stage, and to my parents for having made it possible for me to pursue the academic career that now culminates in this thesis.

Chapter 1

Introduction

"Language is the dress of thought."
– Samuel Johnson

"Translation is at best an echo."
– George Borrow

Language is something that sets us humans apart from other species. The importance of language can be illustrated by the undoubtedly great evolutionary advantage it provided us. We humans are capable of expressing our thoughts in language and communicating our ideas in detail to others. Though it can be rightfully argued that communication is not exclusively verbal, and that almost all animals communicate in one way or another, the sheer richness of human language in either verbal or written form is unprecedented. It would be hard to imagine ourselves without this capacity to put our thoughts into words, and the ability to interpret the words of others.

However we can gain a partial understanding of such an incapacity if we envision ourselves in some distant foreign hotel, being enthusiastically spoken to by a receptionist who chatters in a language we neither understand nor recognise, and who in turn does not understand our language either. Both parties have the same unique human capacity for a rich language in which almost everything can be expressed, but both languages are too distinct to be mutually intelligible. If there are no ways to relate the two languages to each other, communication will have to fall back to more basic forms; pointing, nodding and gesturing.

Language is deeply symbolic. We associate certain symbols with objects in the real world. For example the symbol *"chair"*, in either written or pronounced form, relates to objects in the real world upon which we are meant to sit. In philosophy of language this is called the *"extension"* of the symbol. Symbols can also refer to more abstract concepts, such as for example the symbol *"love"*. A single symbol can be ambiguous, *"capital"* for example can refer to financial assets, or a city. These two concepts are very unrelated, but ambiguity can also be more subtle, *"leg"* does not merely refer to a leg of a human or animal, but can also be applied to chairs. There may also be multiple symbols that can point to similar concepts, instead of *"chair"* we could also say *"seat"*. All of these symbols pertain to a certain language, in the case of the above examples, that language was English. When we look at other languages, this intricate network of symbols and their *extensions*, the concepts to which they point, may be organised differently. In Dutch a human leg is called *"been"*, but this is not the word used for the leg of a chair. The Dutch symbol *"been"* however is ambiguous and may also refer to what is called

"bone" in English, whereas the English word *"leg"* is never used to refer to bones. Moreover, the Dutch word *"been"* has no relation to the orthographically identical English word *"been"*, participle of the verb *"to be"*.

What I mean to invoke with the above example, is an appreciation for the complexity of language and an understanding for the level of ambiguity inherent in the symbols of a language. Moreover, it should be understood that different languages are organised differently, not just having a completely different set of symbols, but also yielding them in different ways.

Despite the substantial differences between languages, languages can generally be related to others through translation. Hiring an interpreter could help clear up the confusion with the foreign receptionist in the Bablonian situation we sketched earlier: the receptionist would speak in his own language to the interpreter, the interpreter has knowledge of both languages and has some kind of mental model which allows him to convey the message of the source in the language of the recipient. This is nicely reflected in the etymology of the verb *"to translate"*, which comes from Latin *"translatus"*, literally meaning something like *"to carry across"*. A translation carries the meaning of a piece of text over to a different language, changing the written or verbal form but carrying the semantics, the essence of the text, across to the new language.

If you have knowledge of two or more languages, it is fairly easy to recognise that translation is not at all a trivial matter. The naive approach of looking up every word in a dictionary, and noting down the corresponding word in the target language, is doomed to lead to unsatisfactory results. I already illustrated the fact that a one-on-one correspondence between words in most cases does not even exist. Translation is an extremely complex endeavour, and it seems highly unlikely that any two translators will produce the exact same translation for any given page of text. Translation can with right be said to be an art, a creative endeavour in which human interpreters are indispensable.

Nevertheless, human interpreters are not always available, their services are costly, time-consuming, and they do not fit in our suitcase when we go on holiday to a foreign country. In a society in which automation plays a pivotal role, it makes sense to attempt to delegate the task of translation to computers where possible. Here we enter the field of *machine translation*, which is the focus of my research.

The idea is to automate translation, translation performed by computers instead of human interpreters. The summit of machine translation might be embodied in the appealing idea of a *"Universal Translator"*, as portrayed in the science-fiction series Star Trek. The universal translator is a fictional piece of technology which is capable of instantly translating from and to almost any alien language. Both parties can speak to each other in a normal fashion, each in their own native tongue, and the translation seems to be without delay. They perceive each other as if the other were speaking in their own language.

Needless to stay, this probably always remains science-fiction. The current state of the art in machine translation is nowhere near that level of advancement, and it is highly unlikely that such a translation device can be achieved in the near future, if at all. Such a universal translator does not deal merely with machine translation, but would also require speech recognition and speech synthesis, both of these are separate tasks that will not be considered in this study.

We can envision machine translation in more realistic applications; such as applications on our phones or PDAs, probably in the form of web services, that allow us to submit a text for automatic translation and which after processing return the translation. There are currently already plenty of examples of such technologies, even though the resulting translations are

not always satisfactory. Notable machine translation systems on-line are for example Google Translate and Systran, which is used by Yahoo's Babelfish.

Machine translation is still an area of active research, in which the end-goal is to improve the quality of automatically produced translations. Ideally this quality would approximate the quality delivered by human interpreters, but there remains an enormous gap between the two. This gap will not be bridged by the study laid out in this thesis, but I do aim to make a modest contribution to further the research on machine translation.

1.1 Research Question

In this thesis, I will be researching a particular new method of machine translation and will try to assess the effectiveness thereof. The name of this approach, also the title of my research, is *"Phrase-Based Memory-Based Machine Translation"*. This rather long title demands some explanation. I will attempt to summarise the focus of my research without going into too much detail at this stage. An in-depth discussion of these will be left to subsequent chapters.

Memory-based machine translation, as described in van den Bosch et al. (2007), is considered a form of Example-Based Machine Translation. A parallel corpus serves as the main knowledge base, which is a huge body of text available in two languages. All sentences in this parallel corpus are paired up with their counterparts in the other language. Between the words of each sentence pair, an alignment is computed; a linkage of words in one language to their translations in the other language. Now we have a basis from which a machine learning system can learn. All sentence pairs are cut into small fragment pairs, consisting of *phrases* of variable length, also allowing for single words. This is the *phrase-based* nature of my approach, which can be contrasted to approaches that are merely word-based. We thus have a mapping of fragments in the source language to fragments in the target language. Subsequently, *machine learning* techniques are used to learn from this data. This process results in a *memory-based model*, and this is what constitutes the *memory-based* character of the proposed approach. This *memory-based model* can then be used to *classify* fragments that were previously unseen: Given a sentence we want to translate, we again cut this into various *phrase-based* fragments, and for each fragment, a translation will be predicted. Last, but certainly not least, all translations of these fragments will be recombined together to form the translation of the given sentence.

I base my research primarily on prior research by Antal van den Bosch, Sander Canisius, et al. on *"Memory-based Machine Translation'* Canisius & van den Bosch (2009). The "phrase-based" character of my research is what distinguishes my study from prior research. Previous approaches of memory-based machine translation were essentially word-based approaches, even though context information played an important role and provided a somewhat phrase-like character. But in essence the approach was one in which words in the source text were translated one by one.

The focus of my research is to develop and test methods for phrase-based memory-based machine translation, thus operating on the phrase-level instead of word level. Note that throughout this thesis the word "phrase" is not synonymous with the word "sentence": A phrase simply consists of one or more words, and as an entity it is associated with a translation.

Does a phrase-based approach improve memory-based machine translation? This is the main research question that will be addressed in this thesis. Main questions that will need to be addressed to eventually answer this are the following:

- *What advantages do we expect from a phrase-based approach as opposed to a word-based approach?*
- *What problems can we expect when working with phrases instead of words?*
- *What constitutes a phrase and how can we extract phrases from sentences?*

We thus ask whether and to what extent a phrase-based approach is an improvement over the word-based approach, and I will identify the problems that arise in this approach.

1.2 Thesis Setup

Having provided a short introduction into the subject under investigation, the second chapter will revisit the issues posited in this introduction and go into the details of the theoretical background thereof. This will lay a necessary foundation that is indispensable in order to understand the remainder of the thesis.

After obtaining the necessary theoretical background, the third chapter will go into the heart of the matter and describe the setup and implementation of my research. I shall explain in detail the workings of phrase-based memory-based machine translation and the software I wrote for this purpose.

Then this system will be put to the test in the fourth chapter, various experiments will be conducted and the results will be thoroughly compared and discussed.

Last but not least, the fifth and final chapter will put forward the conclusions of this research and suggestions for future research.

Chapter 2

Theoretical background

"Whenever a theory appears to you as the only possible one, take this as a sign that you have neither understood the theory nor the problem which it was intended to solve"
– Karl Popper

2.1 Introduction

This chapter will provide the necessary theoretical background that will be needed for a proper understanding of the remainder of the thesis. We will take a look at related work in all the relevant areas, most notably the related work upon which this study is founded. First we will go into a deeper investigation of machine translation as such, and compare some of the existing approaches. Secondly, we will take a closer look at machine learning and a particular method thereof, which is after all something that distinguishes our technique from more mainstream methods of machine translation. After having discussed both machine translation and machine learning, we will see how exactly machine learning has been employed in previous research by Antal van den Bosch et al.

2.2 Machine Translation

Although this is by no means a philosophical dissertation, it might nevertheless be a good idea to introduce this section on Machine Translation with some philosophical considerations regarding the feasibility of machine translation and the inherent difficulties in machine translation and translation in general. Nirenburg (1996) writes about the views of philosopher Yehoshua Bar-Hillel, whom he credits to be known to every student of machine translation and to be amongst the most widely quoted in the field. In the introduction I already elaborated on the complex nature of language and translation, and acknowledged that translation can be rightfully considered a form of art. This leads us to ponder on the feasibility of machine translation. Here Bar-Hillel comes in, who according to Wilks (1979), argued that machine translation was not only practically, but also theoretically, impossible:

"Expert human translators use their background knowledge, mostly subconsciously, in order to resolve syntactical and semantic ambiguities which machines

> *will either have to leave unresolved, or resolve by some 'mechanical' rule which will ever so often result in a wrong translation."* (Bar-Hillel, 1962)

What leads Bar-Hillel to conclude that high-quality machine translation is impossible, is the following illustrative and famous example. Consider the sentence *"The box was in the pen"* in the following context:

> *"Little John was looking for his toy box. Finally he found it. The box was in the pen. John was very happy."* (Bar-Hillel, 1964)

The ambiguity here is that "pen" in this sense is clearly not a writing utensil, but rather a kid's playpen. For a human this interpretation is fairly obvious, but for a machine it may be impossible to to determine, as he himself explains:

> *"Why is it that a machine is powerless to determine the meaning of pen in our sample sentence within the given paragraph? [...] What makes an intelligent human reader grasp this meaning so unhesitatingly is [...] his* knowledge *[of] the relative sizes of pens, in the sense of writing implements, toy boxes, and pens, in the sense of playpens, are such that when someone writes under ordinary circumstances and in something like the given context, "The box was in the pen," he almost certainly refers to a playpen and most certainly not to a writing pen."* (Bar-Hillel, 1964)

I will not go into too much depth about his views. Clearly the argument here is that in order to translate accurately, background knowledge about the world around us is essential. If after all, symbols refer to objects in the real world, or to abstract concepts derived from the real world, it is not insensible to posit such knowledge as a prerequisite for translation.

Wilks (1979) also provides an example of semantic ambiguity with an illustrative example. This example will be used now as illustration to demonstrate some of the difficulties in machine translation. The sentence he uses is *"Time flies like an arrow"*, and the semantic ambiguity arises from the fact that *all* of the first three words could be interpreted as a verb, thus yielding three different readings. Only one reading is sensible to us, the other two readings are rather absurd to all those of us with knowledge of the world around us. In the next example, the verb is emphasised in bold, and a speculative Dutch translation is added each time to show that the various readings can not be produced with the same Dutch sentence, but that each has to be translated with different vocabulary. This reiterates the point made in the introduction about the difference in organisation of the intricate network of symbols and their extensions across different languages.

Consider the following three readings:

- *"Time* ***flies*** *like an arrow"* - Normal reading: time passes quickly, just like an arrow

  "De tijd vliegt als een speer voorbij"

- *"****Time*** *flies like an arrow"* - Reading 2: timing the speed of flies, in a similar fashion as timing an arrow

  "Vliegen klokken als een speer"

- *"Time flies **like** an arrow"* - Reading 3: A particular type of flies, "time-flies", are fond of an arrow

  "Tijds-vliegen houden van een speer"

Based on the reading we take, the Dutch translation is different. For us the first reading is clearly the most sensible, as the other two are absurd, but can a machine come to the same conclusion? One is inclined to agree with Bar-Hillel's argument that background knowledge is essential. We will see later that this is in sharp contrast to the actual approach taken in this study, in which there is no background knowledge, nor even linguistic knowledge of any kind. This does however not invalidate Bar-Hillel's argument.

Dutch-speakers may have observed something interesting in my translation of the expression *"time flies like an arrow"*. The English expression uses the word *"arrow"*, that which springs from a bow, whereas the Dutch uses "spear". If we were to translate the English sentence literally, *"tijd vliegt als een pijl"*, we would end up with a rather odd translation. The translation of such idiomatic expressions is one of the challenges that machine translation faces.

The same goes for a literal word-by-word translation, as we already ascertained in the introduction. In Dutch, the word *"tijd"* (*"time"*) is preceded by the definite article *"de"*, whereas in English *"the time flies like an arrow"* would be incorrect, one never speaks of "the time" when one refers to the concept of time in general. The Dutch sentence also has an extra adverb *"voorbij"* to emphasise the passing motion of flying. The inadequacy of literal word-by-word translation becomes even more apparent when regarding languages that are more distant from each other than Dutch and English, both part of the Germanic subfamily of Indo-European languages. The next example in figure 2.1 illustrates a literal translation from Chinese to English:

我	跟	我的	哥哥	明天	买	大	的	房子
wǒ	gen	wǒ de	gēgē	míngtiān	mǎi	dà	de	fángzi
I	with	my	elder brother	tomorrow	buy	big		house

Figure 2.1: A Chinese to English translation

We notice immediately that the English literal translation is not a very natural one. The natural translation would change the word order, add an article, which do not exist in Chinese, and possibly an auxiliary verb as well. We would then obtain something like *"I will buy a big house tomorrow with my elder brother"*. We can also observe some peculiarities, firstly the Chinese word 的 , which is a so called possessive marker, has no direct equivalent in English. Secondly the word 哥哥, one word in Chinese, translates to two words, a noun and an adjective, as unlike the Chinese the English language has no single word for *"elder brother"*. Thirdly, the reverse is also the case, in Chinese we see two words 我 (*"I"*) and 的 (possessive marker), which together translate to a single possessive adjective; *"my"*. In fact, such a translation of two words to one already resembles a phrase-based translation.

In the above example we have seen three well-known issues in machine translation, which have to be dealt with in any implementation:

1. Spurious words, words that have no translation in the other language – (*null alignments*)
2. Single words in the source language that translate to multiple words in the target language – (*one-to-many alignments*)

3. Multiple words in the source language that translate to a single word in the target language – (*many-to-one alignments*)

Note that of the first issue we have only seen an example in one direction, but the opposite may occur just as well: there may be an extra word in the target sentence for which their is no real corresponding word in the source sentence. Such words are called fertility words. In addition to these three, we can add a fourth that illustrates the nature of phrase-based systems, and provides us with a definition of what constitutes a phrase:

4. Multiple words in the source language that translate to multiple words in the target language – (*many-to-many or phrase alignment*)

The fact that machine translation is a difficult problem, should not dissuade us from attempting to further the research in this field. Though there still is validity to Bar-Hillel's argument and machine translation remains imperfect, there have been made considerable advances since his publication. Most successful machine translation solutions rely on huge bodies of text, on which statistical or machine learning methods are employed. The perspective on machine translation in this thesis will also take such a modern angle, whilst at the same time acknowledging that the system proposed is no match for a human translator.

Now that we have seen some of the philosophical and linguistic implications of machine translation, it is time to look into the actual workings of machine translation systems. I will give a short overview of methods of machine translation that will ultimately lead to the presentation of *Memory-based Machine Translation*, on which this research is founded. For the sections on Classical- and Statistical Machine Translation, the excellent work of Jurafsky & Martin (2009) has been my main source. So to their work I gladly direct the reader who desires to gain a more thorough understanding than can be provided here.

2.2.1 Classical Machine Translation

The strength of current techniques for machine translation based on machine learning and statistics, may be better understood when we contrast them with classical approaches to the problem. Classical Machine Translation is not one particular kind of translation method, but instead groups four main approaches that all have in common that they are generally *rule-based*. The classical methods each have an *analysis* stage, a *transfer* stage according to a set of rules, and a *generation* stage. This is unlike the statistical- and example-based methods that will be discussed in later sections.

These four methods of machine translation and the three aforementioned stages are depicted nicely in the famous Vauquois triangle (figure 2.2), an image of which can be found in almost any introduction to Machine Translation. The triangle shows the four methods which will be very briefly discussed in this section. The closer to the top of the triangle, the more difficult and the more level of analysis is required for that method.

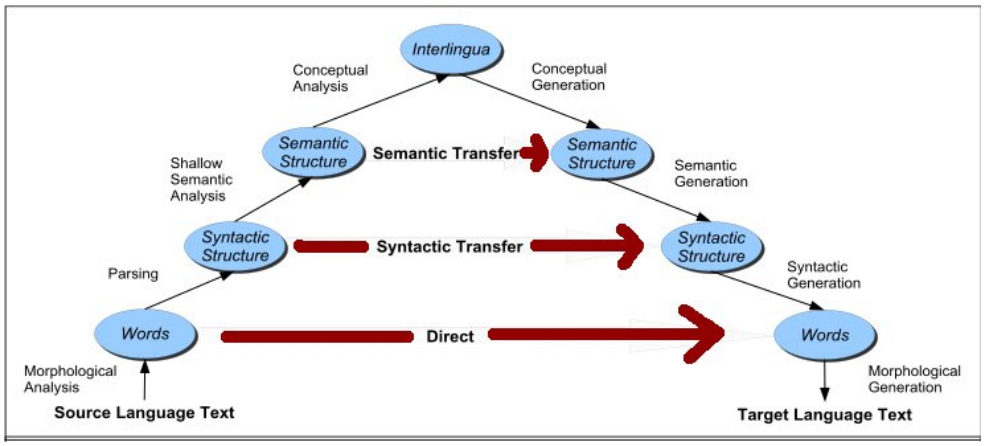


Figure 2.2: The Vauquois triangle (Vauquois, 1968)

The most naive form of translation, which we have already seen, is simply translating every word, one by one, looking up the word in a bilingual lexicon. This is also the basis of the so-called **direct translation** approach, to be found at the very bottom of the Vauquois triangle. We already determined that such a naive approach is inadequate for almost all purposes. But fortunately that's not all there's to it, and *direct translation*, even though being the simplest of approaches, can be more sophisticated than that.

In *direct translation* there is generally first a stage of morphological analysis of the source sentence. After that the word by word translation is performed using a bilingual lexicon. Then there is a set of rules that reorders the resulting translation, and lastly there is a morphological generator, capable of producing the correct morphological variant of a word, according to for example gender, number, case, tense, mood, verbal aspect etc... The power of such a system is largely dependent on the rules that constitute its parts, and these rules have to be constructed, either by humans or possibly semi-automatically. This is however certainly not a trivial task.

If we move up one level we encounter the **syntactic transfer** approach. In this approach, the syntax of the source sentence is analysed, and the resulting syntactic structure is mapped, by rules, to a new syntactic structure in the target language. The actual words are translated in the same fashion as in the *direct translation* approach, through lookup in a lexicon. The advantage of this approach is that richer structures can be translated, if those structures are of courses mapped in the rule-base. A main advantage of such a method of syntactic transfer is that the syntax and corresponding word-order in the target language is constructed in a grammatically sound way.

Semantic transfer, is similar to syntactic transfer, but attempts to analyse the semantic structure of the source sentence, and uses rules to map these to a semantic structure in the target language. This is achieved using for example *semantic roles*, in addition to syntactic analysis.

At the very top of the Vauquois triangle, we find the Interlingua model. The previous three approaches rely extensively on various sets of rules that map words, syntax, or semantic roles from the source language to the target language. This is a disadvantage when we consider settings in which there are not just two, but multiple languages to relate to each other. One would have to reconstruct the rule sets for each language pair, which is quite an undertaking, as we already emphasised. The Interlingua idea is coined as a solution to this; instead of translating from all languages to all others, translation goes from the source languages to one

Interlingua representation, and from that representation to the target languages, thus in the end reducing the amount of languages that have to be related to each other.

The European Union is one of those settings in which many languages have to be translated to many others, a complicated, time-consuming, and costly process. A group of Esperantists, speakers of the artificial language Esperanto, even participated in European Elections with a political party (Eŭropo Demokratio Esperanto) promoting the use of Esperanto as official "interlingua" within the European Parliament and Union as a whole. In machine translation however, an Interlingua can also be a more formal semantic representation, rather than a representation in natural language. The source sentence is subjected to semantic analysis, and the interlingua representation is constructed. Subsequently, there is a stage of semantic generation to the target language, based on the interlingua representation.

This interlingua must know a wide range of concepts to accommodate for all the different nuances that the source- and target- languages can contain. To illustrate this, Jurafsky & Martin (2009) mentions the Chinese word 哥哥 (gēgē), meaning *"elder brother"* as we already saw earlier. If Chinese is one of the source- or target- languages, then a good Interlingua representation should contain an ELDER-BROTHER concept as well, but when translating from English to German or vice versa, which are both languages which do not place emphasis on elder or younger siblings and do not have special vocabulary for those, usage of such a concept in the Interlingua would require spurious disambiguation.

In practise the Interlingua approach, and previously discussed classical approaches, have not proven successful enough so far. The Interlingua approach does not scale well and is best used in a very confined domain. All approaches rely on an extensive human-compiled rule-base, which are cumbersome to construct and which in practise are no match against the multitude of constructions in natural language.

2.2.2 Statistical Machine Translation

Rather than constructing a complex and vast set of rules like classical machine translation does, *statistical machine translation* takes an entirely different approach. Various concepts in Statistical Machine Translation also have a bearing on the approach I develop in this thesis, so a certain understanding of Statistical Machine Translational is essential. Statistical machine translation is a data-driven method, using huge bilingual corpora. It regards translation as a *probabilistic* task, and tries to discover what configuration of which words yields the *most probable* translation of a *given* input sentence. We can formalise this as follows:

$$\text{best-translation } \hat{T} = argmax_T P(T|S)$$

Here, and all throughout this thesis, T stands for a sentence in the target language, and S stands for a sentence in the source language. In the statistical approach, the above is very hard to compute as such. Estimating $P(T|S)$ for a given pair of source- and target sentences would require knowledge of the probability distribution for all possible target sentences, and we do not know all possible target sentences. We need to find a different paradigm that is computationally attainable. In order to do that we will go into the question of what makes a translation a good translation.

There are two main characteristics that any good translation should possess; a translation should be true to the meaning of the original, and a translation should be formulated in fluent,

natural language. These two factors are often referred to as *faithfulness* and *fluency*. Both factors need to be balanced to produce an optimal translation. This dilemma is nicely worded in a somewhat politically-incorrect saying attributed to the Russian poet Yevgeny Yevtushenko: *"Translation is like a woman. If it is beautiful, it is not faithful. If it is faithful, it is most certainly not beautiful."*. Translations that are faithful to the letter are prone to lack in fluency; it is necessary to find a compromise between both faithfulness and fluency to come to an acceptable result, aiming to maximally satisfy both aspects. This is exactly what statistical machine translation aims to do, and this can be formalised in the following equation from Jurafsky & Martin (2009):

$$\text{best-translation } \hat{T} = argmax_T \text{ faithfulness}(T,S) \cdot \text{ fluency}(T)$$

Note that this equation is equal to the earlier one, as they both express the best-translation $\hat{T}$. The solution to our problem of the non-computability of $P(T|S)$, and the key to converting the former equation into the latter, lies in Bayes' theorem. Bayes' theorem is famous in statistics and explains how a conditional probability (A *given* B) relates to its inverse (B *given* A). The theorem is stated and applied as $P(T|S) = \frac{P(S|T)P(T)}{P(S)}$.

The underlying paradigm that makes this transformation comprehensible is the *noisy channel model*, a model that plays an important role all throughout the field of Natural Language Processing, and conceived by Shannon & Weaver (1963). We imagine a noisy channel that somehow distorts the information that goes through it. We imagine that we have our target sentence as source, which passed through a noisy channel and gets distorted in the progress, ending up as our source sentence. The source sentence is thus treated as a distorted version of the target sentence. Now the idea is to work backwards, to predict and search for the original source sentence *given* the target sentence. The situation thus becomes reversed, which has the great benefit of making it computationally tractable and which captures the intuition of the two factors faithfulness and fluency. Since we deal with the same source sentence $P(S)$ throughout the computation, we can drop the denominator from the equation and we obtain:

$$\text{best-translation } \hat{T} = argmax_T P(T|S) = argmax_T P(S|T)P(T) = argmax_T \underset{\text{faithfulness}(T,S)\cdot\text{fluency}(T)}{}$$

Here we remark that $P(S|T)$ corresponds to the faithfulness and $P(T)$ corresponds to the fluency, so we now have something that corresponds nicely to our intuition about what makes a translation a good one. Statistical machine translation systems use two separate models for the *estimation* of the two characteristics, a *translation model* estimates the faithfulness ($P(S|T)$), and a language model estimates the fluency ($P(T)$). The faithfulness here reflects the distorting nature of the noisy channel. A *decoder* then uses these models and searches for the optimal, most probable, solution in the vast search space of possibilities.

Now the question becomes, how do we estimate $P(S|T)$ and $P(T)$? In other words, what constitutes the translation model and what constitutes the language model?

Language Model

Let us start with a review of the language model. The Language Model is not just something we see in statistical machine translation, but also in many other fields of Natural Language

Processing as a whole, such as in speech recognition, and spelling checkers. The Phrase-based Memory-based Translation method proposed in this thesis will also make extensive use of a language model, therefore it is worth providing the necessary theoretical background in certain detail.

The purpose of a language model is to predict the likelihood of a particular sentence. Now this may strike the reader as an odd notion at first, and the reader would be amongst good company in thinking so: The famous American linguist and philosopher Chomsky (1969) said: *"It must be recognised that the notion "probability of a sentence" is an entirely useless one, under any known interpretation of this term"*. Language is so rich that for example this very sentence you are reading right now, one that is perfectly valid, meaningful, and grammatically correct, has until now never been uttered before in this exact way in all of the history of mankind. So it would seem that that the idea of probability is not very meaningful when applied to language.

However, one of the useful implications of a language model is that malformed sentences are in a certain sense "less likely" to be produced in a language than a well-formed sentence. A non-sentence such as *"I window red from van see my"* is a very unlikely series of words in the English language and obtains a lower probability than the well-formed sentence *"I see a red van from my window"*.

Getting a probability score for a sentence as a whole is impossible, as most sentences have never been uttered before, but we can get scores for smaller fragments of the sentence, and take the product thereof. These smaller fragments are n-grams, a sequence of n consecutive words. The relative frequency of these fragments are looked up in a corpus. Thus in order to compute the probability for a given input sentence sentence, we extract all possible n-grams, obtain the score for each of those, and take the product of these. For a sentence S consisting of a series of words $s_1...s_N$, we can formalise a bi-gram model as follows:

$$P(S) = \prod_{i=1}^{N+1} P(s_i|s_i - 1) \tag{2.1}$$

Here we define s_0 to have the special role of holding a starter symbol, and $s_(N+1)$ to hold a finalising symbol. We take for example the sentence ``this is an example'' as input, then wrap it with a starter symbol and a finalising symbol, producing ``<s> this is an example </s>''. We can then obtain five the bigrams: ``<s> this'', ``this is'', ``is an'', ``an example'' and ``example </s>''. For each bigram we obtain the score from a corpus, and we take the product of all these scores. The example was conducted with bi-grams, but this approach can of course be extended to trigrams, quadrigrams, and further. We will see later that the language model used here is in fact a trigram model. Higher n values generally produce more reliable predictions, as more context is considered, but of course the corpus must be substantial enough to support that, because larger n-grams are sparser than shorter ones.

A problem arises when one of the n-grams can not be found in the corpus, as that would result in it obtaining a probability of zero, and thus the outcome for the whole sentence would be zero as well. We have to understand that any corpus is by definition limited and merely an approximation for modelling a language. Bigger corpora produce more reliable results, but due to the nature of language, no corpus can be big enough. Special *smoothing* techniques are therefore often needed to ensure that fragments which are not known in the corpus, still receive a minimal score.

It should be clear by now that a language model is capable of assigning a probability score to

a sentence, a score which corresponds nicely to the factor *fluency*.

Word Alignment

Before we go into an overview of the Translation Model. We first need to re-address the issue of word-alignment and how to obtain it in training data. In the introduction to this chapter, I already provided an example of a word-alignment between a Chinese sentence and its' English translation, so we only need to recapitulate briefly. We saw three types of word-to-word alignment:

1. Spurious words or zero-fertility words, words that have no translation in the other language – (*null alignments*)
2. Single words in the source language that translate to multiple words in the target language – (*one-to-many alignments*)
3. Multiple words in the source language that translate to a single word in the target language – (*many-to-one alignments*)

To build a word-alignment model which we can later use for machine translation, we first assume we have available a large parallel corpus with data in both source and target language, and we assume that this parallel corpus is already tokenised, i.e. punctuation has been separated from the words and end-of-sentence markers have been identified. Our aim is to align the words in all sentences in this corpus to their counterparts. First, we need to align all sentences in this corpus with their translated equivalent, because we will operate on a sentence level. This is a so-called *sentence alignment*, not to be confused with phrase-alignment. Only sentences that have a counterpart in the other language are usable.

Given each sentence pair in this corpus, we can compute word alignments using the *Expectation Maximisation (EM) algorithm* (Dempster et al., 1977). This is a dynamic programming method that computes which words are most often associated with which other words, it is able to compute which connections are more likely, and on the basis of that it links the words. So whereas we started with a parallel corpus without connections, we now end up with a word-aligned corpus.

This word-aligned corpus is a necessary basis for statistical machine translation, as well as for memory-based machine translation and other forms of machine translation.

Translation Model

The translation model is a more complicated matter compared to the language model. Whereas the language model is a monolingual model, it considers just one language, the translation model in statistical machine translation models the relation between the words or phrases of two languages. It predicts the probability of the source sentence *given* a target sentence ($P(S|T)$). Keep well in mind the reversed approach of the noisy channel model. If the source sentence is a likely translation of a given target sentence, this probability is high, if it is unlikely to be a translation, it is low.

There are two main kinds of translation-models, word-based models and phrase-based models. Both either directly or indirectly make use of *statistical word-alignment algorithms* such as the

IBM models first proposed by Brown et al. (1993). Usage of these or similar models is an essential characteristic of statistical machine translation. What the IBM models and similar models essentially do is generate a "story" about how the target sentence T that goes into the noisy channel comes to emerge as the source sentence S. If we translate for example from Spanish (S) to Tagalog (T), then the translation model generates a "story" about how the Tagalog sentence becomes a Spanish sentence, rather than vice versa.

In classical machine translation systems, we also saw a similar story, made up from a set of rules that explicitly describe possible transformations. But the generative story in word-alignment models such as the IBM Models Brown et al. (1993) is statistical in nature. To facilitate understanding, we shall use an example of translating Spanish (S) to Tagalog (T), where we assume a given sentence in Tagalog and we construct a generative story of how it comes to be a Spanish sentence:

1. Choose an arbitrary length for the Spanish sentence.
2. Choose an alignment from Tagalog to Spanish (this may include one-to-many alignments).
3. Choose Spanish words by translating the aligned Tagalog words.

The key part of these models is a particular alignment configuration (A). This relates to the translation model as follows (Jurafsky & Martin, 2009):

$$P(S|T) = \sum_{A} P(S, A|T) \tag{2.2}$$

This, as we expect from a translation model, defines the character of the noisy channel. Having constructed this model of probabilities, the task is to select the actual best alignment $\hat{A} = argmax_A P(S, A|T)$ between sentences S and T. Note well that word-alignment models such as the IBM models thus define a large search space, and it is the task of the *decoder* to efficiently search in this.

The inner workings of the actual algorithms in question are fairly complicated and it would go beyond the scope and relevance of this thesis to provide an adequate enough overview. The memory-based approach that is the focus of this research makes use of an entirely different translation model.

I will however elaborate on the phrase-based model of statistical machine translation, as I will be using part of it later.

Phrase-based translation

A phrase-based approach groups words together into phrases, and translates these phrases as an entity. The phrase-based translation model can be formulated as follows (adapted from Jurafsky & Martin (2009)):

$$P(S|T) = \prod_{i=1}^{N} \phi(s_i|t_i) d(s_i, t_i) \tag{2.3}$$

For each source-target phrase-pair $s_i \in S$ and $t_i \in T$ the translation probability $\phi(s_i|t_i)$ is computed. In addition to the translation probability, there is usually also a distortion probability $d(s_i, t_i)$ which measures the distance between the positions of the word or phrase between the target and source. The distortion probability can of course also be one of the parameters in a merely word-based approach. What it does is determine the extent to which re-ordering of the words or phrases is allowed.

The probabilities $\phi(s_i|t_i)$ can be inferred from a so-called *phrase-translation table*. A phrase-translation table lists the phrases in both languages and assigns a probability. This phrase-translation-table approach is one approach that I too will use later on in memory based machine translation. One of the questions that needs to be answered in order to answer the research question of this thesis, is the question: "what exactly constitutes a phrase and how can we extract phrases?". The definition of a phrase we have already more or less seen in the introduction to machine translation, but it is time to make it a bit more precise. We can define a phrase simply as *any group of consecutive words*, and it is in this sense that this concept is used throughout this thesis. Note that a phrase need not even have to make sense as an actual linguistic unit. What gives a phrase its sense in this thesis, is the fact that it can be paired up with another phrase in the other language.

Of course these phrase-translation tables do not appear out of thin air, but they have to be generated somehow. That is where again the statistical word-alignment models come in. For the word-based translation model, these were used directly, but for the phrase-based translation model, these are only used to extract the phrases for the phrase translation table.

Once we have found a word alignment for a given sentence pair, we can use that to extract phrases. The procedure is as follows: We use the methods discussed in previous sections to create a word alignment from source to target, and an alignment from target to source. This produces for example the two alignments shown in figure 2.3, for a Spanish and English sentence.

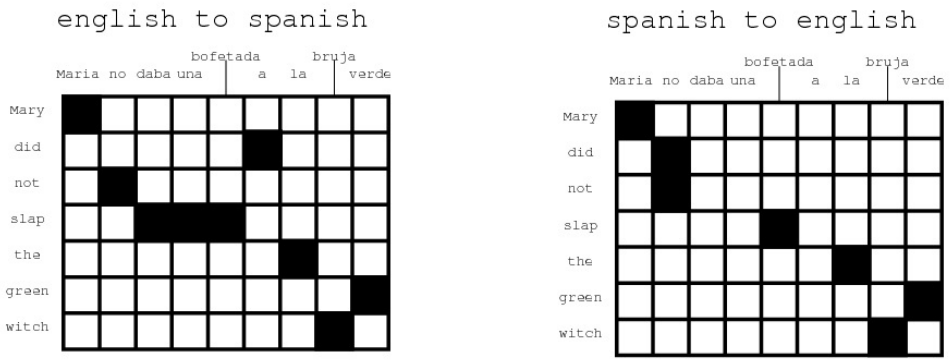


Figure 2.3: Two alignments for a Spanish and English sentence, one in each direction (Koehn, 2004)

The next step is to combine these two alignments into one. This is often done by starting to take the *intersection* of both. We then get the alignments that are present in both, and are thus of high quality. As shown in the following example from (Koehn, 2004):

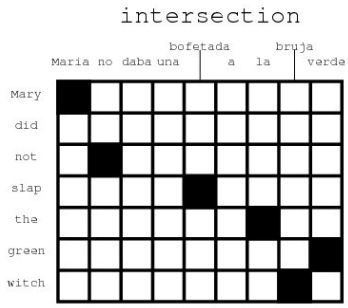


Figure 2.4: The intersection of the two alignments (Koehn, 2004)

This is however often not sufficient, as it contains too few points to extract meaningful phrases. Och & Ney (2003) developed an algorithm that incrementally adds points from the *union* of the two alignments. The full union would produce less accuracy, but the algorithm selects words by first taking the *intersection*, and then incrementally adding certain points of the full union. Once this combining process has completed, we end up with an alignment from which phrases can be extracted, as shown in figures 2.5 and 2.6, depicted below:

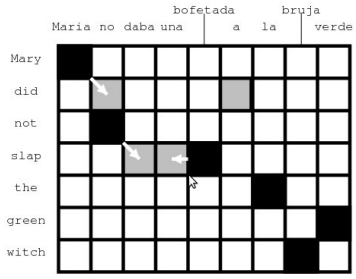


Figure 2.5: Incrementally adding points of the union alignment to produce the final alignment (Och & Ney, 2003; Koehn, 2004)

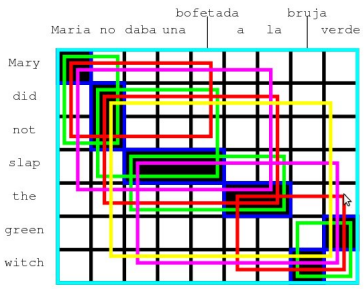


Figure 2.6: Phrases that can be extracted on various levels (Koehn, 2004)

In figure 2.6 we see that phrases can be extracted on various levels of granularity, for example, by following the green squares we can extract the following sentence-pairs for our phrase-translation table: *("Maria no", "Mary did not"), ("no daba una bofetada", "did not slap")* , *("daba una bofetada a la", "slap the"), ("bruja verde","green which")*

Decoder

A brief last word is dedicated to the role of the *decoder*. It is the task of the decoder to find the optimal balance between the language model and the translation model. It takes the source sentence as input, searches through the vast search-space described by the language-model and translation-model, and produces and eventually returns the actual final translation that maximises the product of the language model and the translation model. This is thus in essence a search problem. We will not go into the inner workings of this particular decoder, but instead we will later on see in detail how a decoder works when applied to memory-based machine translation.

2.3 Machine Learning

In the previous section I presented an overview of various machine translation approaches. Before going into a discussion of memory-based machine translation, it is first essential that I introduce *machine learning*, as this forms a vital component of memory-based machine translation. As we already saw in the introduction, the *memory-based* character of the machine translation under investigation here, is a reference to the usage of machine learning.

Machine learning can be defined (after Witten & Frank (2002)) as the (semi)-automatic discovery of common patterns in a substantial amount of data, which results in the emergence of meaningful new information that was previously not apparent. It is a field that can be subdivided into more specific tasks such as classification and clustering.

The common denominator in all of Machine Learning is thus that machines "learn" in a sense from the available data, by detecting commonalities. From this learning emerges new useful knowledge. Of course this notion of "learning" has the pitfall that it can once again lead us to certain philosophical deliberations: What is learning? Can a computer learn? The provided definition, by omitting the word "learn", cleverly attempts to avoid these philosophical issues. Machine learning is after all a very practical discipline with real-life practical applications. The question what exactly learning is, is not an easy one to answer. Learning might be defined as the acquisition of knowledge or a skill. One gains certain *information* or an *ability* that one did not have prior to the learning. In the machine learning sense, patterns are extracted and used, and what emerges from this could be considered a form of knowledge. A clear definition of what constitutes as knowledge, and what is knowable in the first place, is again a difficult philosophical matter.

A prerequisite of any form of learning is that there has to be a *memory*, which crudely put could be seen a kind of storage space for both knowledge and skill. Learning can not take place if knowledge or skill can not be retained at least temporarily. The patterns a machine learning algorithm extracts from the data, are embodied in a *model* that is stored or kept in memory. The model thus is the product of the learning process, and embodies the knowledge or skill which can be used to make certain predictions.

This leads us to *"classification"*, which is a particular kind of machine learning task that is of direct relevance to this research, and which will further clarify the idea of machine learning. Classification, according to Tan et al. (2005), is the task of assigning instances to one of several predefined categories. It predicts to what *class* a certain instance belongs, based on the model it has constructed from from the data.

To make the concept more insightful, I will provide a small general example inspired on an example by Tan et al. (2005). Imagine we want to classify to what class of species certain animals belong. Our instances are thus animals, and the predefined categories are mammal, reptile and bird. The objects (animals) are described according to a number of meaningful *features*. These features are the basis upon which patterns of similarity and dissimilarity can be sought and a model can be constructed. Take a look at the dataset in table 2.1:

Species	body temperature	skin type	lays eggs?	has wings?	legs	class
dog	warm-blooded	hair	no	no	4	mammal
cat	warm-blooded	hair	no	no	4	mammal
crocodile	cold-blooded	scales	yes	no	4	reptile
chicken	warm-blooded	feathers	yes	yes	2	bird
sparrow	warm-blooded	feathers	yes	yes	2	bird

Table 2.1: A sample training-set

Now a classifier can *train* on this dataset and construct a model in memory. This model is subsequently used by a classifier to *classify* previously unseen instances. Take table 2.2 to be a sample test-set:

Species	body temperature	skin type	lays eggs?	has wings?	legs	class
human	warm-blooded	hair	no	no	4	**?**
duck	warm-blooded	feathers	yes	no	2	**?**

Table 2.2: A sample test-set

The classifier will be able to predict the class with a certain degree of accuracy that depends on the nature and relation of the various features, and the amount of instances. Although somewhat simplified, the general principle is that the more objects are available for training, the better results can be expected. We can formalise the notion of a classifier as a function mapping a feature vector onto a class label, where each feature, as well as the class label, is within a certain finite domain of its own that is known beforehand.

Classifiers come in many varieties, there are various different algorithms: *Decision Tree Induction, Rule Induction, Naive Bayes, k-Nearest Neighbours, Neural Networks, Support Vector Machines*, just to name a few. A discussion and comparison of all these methods is beyond the scope of this thesis. The interested reader I glady refer to the works of Tan et al. (2005), Witten & Frank (2002) and Mitchell (1997). We will instead be focussing now on only one particular method.

Memory-based learning

Classifiers can be applied in a wide range of Natural Language Processing tasks, including machine translation. In order to understand Memory-based translation as laid out in the next section, we will first have to discuss its underlying principle: Memory-based learning.

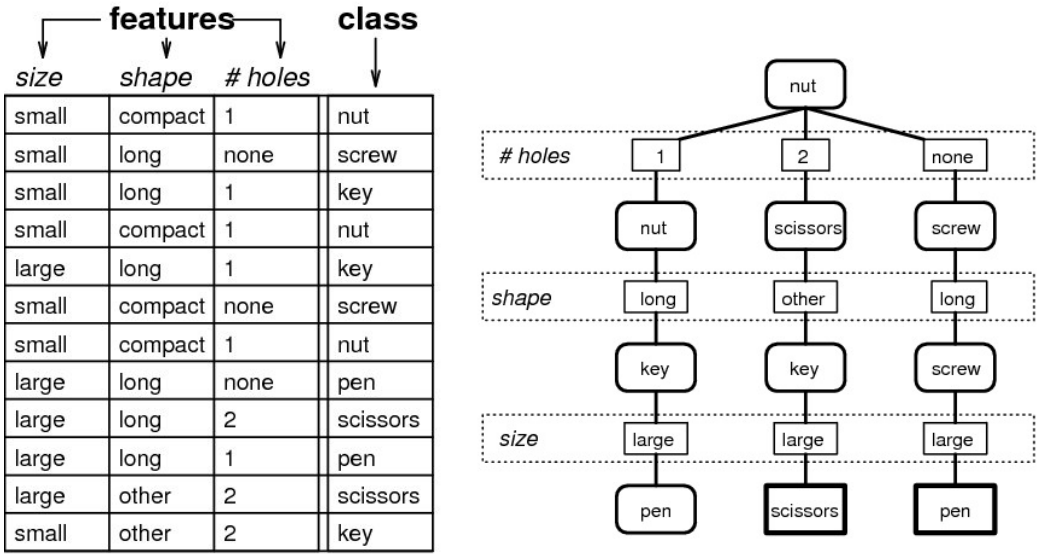
We can contrast two kinds of classification methods, *eager* learning methods versus *lazy* learning methods. Memory-based learning, as described by Daelemans & van den Bosch (2005), is one of the latter. *Eager* methods are defined as methods which seek to create an abstract, generalised, model of the data. In these methods, exceptional, and infrequent cases

are effectively filtered out. But Daelemans & van den Bosch (2005) observe that in Natural Language Processing, exceptional and infrequent cases constitute an important part of the required knowledge, and as such should not be dismisses as noise. Therefore the *lazy* methods takes another approach and leave all data available for processing.

The difference is thus that lazy methods keep a huge model in memory that reflects all of the data, whereas the models of eager methods are smaller and more abstract. Lazy models such as the k-Nearest Neighbour algorithm, work on the basis of computing the similarity of a test instance to the training instances in memory. The k training instances that are found to be most similar to the test instance, determine the class label that will be assigned to the test instance. The likelihood comparison is performed on the basis of the features, for which various metrics exist. One notable advantage of such an approach, is the ability to work with a substantially sizeable domain of different classes. This is the case in many NLP tasks, including Machine Translation.

In this thesis we use two lazy-learner algorithms, or rather a hybrid approach that incorporates these two. The first algorithms is $IB1$, originally proposed by Aha et al. (1991), an implementation of the k-Nearest Neighbours algorithm. The second algorithm is IGTree (Daelemans, van den Bosch, & Weijters, 1997), a decision tree classifier that approximates the functionality of the k-Nearest Neighbours algorithm. The objective of the algorithm is to reduce the computational costs of working with large datasets. This gives it a notable advantage over classic $IB1$, which can be too slow and too memory-intensive for NLP applications.

IGTree compresses the entire instance base into an ordered decision tree structure at training time, and is subsequently able to issue look-ups in this tree structure at test time. To clarify the notion of such a decision tree, take a look at the training instances in figure 2.7 below and the IGTree derived from those. The instances describe a few objects according to several characteristics about their appearance.



features			class
size	*shape*	*# holes*	
small	compact	1	nut
small	long	none	screw
small	long	1	key
small	compact	1	nut
large	long	1	key
small	compact	none	screw
small	compact	1	nut
large	long	none	pen
large	long	2	scissors
large	long	1	pen
large	other	2	scissors
small	other	2	key

Figure 2.7: On the basis of size, shape and number of holes, an instance is to be classified as nut, screw, key, pen or scissors. On the left hand side the training instances, and on the right hand side the *IGTree* produced by training. (Daelemans, van den Bosch, & Weijters, 1997)

The distinctive feature of *IGTree* in comparison to other top-down induced decision tree meth-

ods such as C4.5, is that features in IGTree are tested in a fixed order. This is computed *only once and in advance* for all features. This order is determined using metrics from information theory, such as *information gain* or *gain ratio*. These metrics determine the relative informativeness or disambiguating power of the feature and provide a ranking of all features. The notion of *entropy* plays an important role in these metrics. Entropy is a measure of uncertainty, chaos, or or informativeness. If the expected value of a process is more predictable, then it is less informative and it has a smaller entropy than a more chaotic and more informative process. In information gain, entropy plays an important role. Information gain of a feature can be defined as the reduction in entropy of the classification when knowing the value of that feature. A closely related metric, gain ratio, is what we use throughout this research. It goes beyond the scope of this thesis to fully explain the information theory behind these metrics, so we shall instead return to the application of these metrics in *IGTree*.

The implication of the fixed ordering based on gain ratio is that at each depth in the tree the same feature is tested throughout that level for all nodes. With each node in the tree, a class label is associated that classifies the set of instances following the path up until that point, this class label thus represents the *most probable* outcome for the path in the tree thus far. Only at leaf nodes shown as non-rounded squares in figure 2.7 above, this outcome is exact and non-ambiguous. The further down the tree, the more precise the classification will become. Paths followed during classification of a new instance may finish before reaching an end-node if any of the feature tests along the way fail, in which case a homogeneous class prediction cannot be found, and the most likely class represented at the last visited node is returned. This makes it possible for leaf nodes that have the same class label as their parent to be pruned, achieving extra compression without loss of precision.

It must be stressed again that the idea of *IGTree* is to achieve a compressed model of all training instances in such a way that all classifications can be precisely reconstructed. It attempts to stay closer to actual lazy learning algorithms such as *IB1*. IGTree can thus be contrasted with more *eager* decision tree induction algorithms like *C4.5*, which recompute information gain at each node, test for multiple feature values, and through pruning form a more *abstracted* model from the training data. The idea behind the tree organisation, as explained in (Daelemans, van den Bosch, & Weijters, 1997), is to restrict the search for a given test instance to those training instances that have the same feature value at that feature. This can be done for all features. An instance is thus essentially stored as a *path* in the tree, in which each subsequent child node stores a smaller amount of instances that its parent. Overall this leads to a considerable compression, as multiple instances share similar paths.

Due to this compression IGTree's memory footprint is $O(n)$, far lower than that of IB1, $O(fn)$, where f is the number of features and n the number of instances. Regarding CPU time, IGTree takes $O(flog(v))$ (where v is the average branching factor), IB1 again takes $O(nf)$ here (van den Bosch et al., 2007).

IGTree's performance relies on the differences in information gain. If these are small then IGTree may perform significantly below IB1 (Daelemans et al., 2007). A hybrid approach called TRIBL2 (Daelemans et al., 2007) starts out with IGTree and switches to a IB1 variant when a value mismatch occurs in the tree. It is mainly this hybrid approach that we will use in this research.

The $IGTree$, $IB1$ and $TRIBL2$ algorithms are all implemented in the software TiMBL[1] (Daelemans, Zavrel, van der Sloot, & van den Bosch, 2007). This is the software which is used throughout this thesis, as well as by all of the memory-based machine translation research upon which this study is founded.

[1] Tilburg Memory-based Learner

2.4 Memory-based Machine Translation

Memory-based machine translation can be considered as a form of Example-Based Machine Translation. In characterising Example-Based Machine Translation, Somers (1999) refers to the common use of a corpus or database of already translated examples, and the process of matching new input against instances in this database. This matching is followed by extraction of fragments which are then again recombined to form the final translation.

In his work Somers (1999) also emphasises the *"obvious affinity"* between Example-Based Machine Translation and Machine Learning techniques, and he in fact stresses the fact that example-based machine translation has gone under alternative names by those wanting to bring out some key difference distinguishing their approach from others. In this context he also mentioned the label "memory-based". The clear role that *examples* play in Example-Based Machine translation and memory-based learning, indeed affirms the above-mentioned obvious affinity.

We here define Memory-based machine translation as referring to a machine translation approach using memory-based learning, as laid out in the previous section. The task of mapping one language to the other is delegated to the classification algorithms we discussed. This approach was presented in van den Bosch & Berck (2009), and as such is one of the primary references for my own study. The method can be described as one in which the sentence to be translated is decomposed into smaller fragments, each of which is passed to a classifier for classification. The memory-based classifier is trained on the basis of a parallel corpus in which the sentence pairs have been reduced to smaller fragment pairs. The assigned classes thus correspond to the translation of the fragments. In the final step, these translated fragments are re-assembled to derive the final translation of the input sentence.

This will undoubtedly require further explanation. I will now go into elaborating in more detail the workings of word-based memory-based machine translation as proposed by van den Bosch & Berck (2009). In their method we can distinguish two phases: a *local* phase, in which small fragments are translated using memory-based learning, and a *global* phase, in which these translated fragments are again assembled into one solution.

2.4.1 Local phase - Memory-based learning

For the local phase, we need to have a training set which we can hold into memory and on the basis of which we can translate. First of all we remark that like in all methods of example-based machine translation, we rely on a parallel corpus as our main source. On the basis of this, we compute a word-alignment between all sentence pairs (S, T) in the corpus, as explained in the discussion on statistical machine translation. To make things clearer, the aligned sentence pair in figure 2.8 will serve as an example throughout this section:

Figure 2.8: An alignment between a Dutch and English sentence (van den Bosch & Berck, 2009)

On the basis of this, we can generate training instances. The instances are constituted as follows: The feature vector consists of a word of the source sentence, with one context word on the left hand, one focus words, and one context word on the right hand. The class consists of the aligned word, and also takes left- and right context words. Note at this point that the feature vector in this case consists of three features, and although the output class is composed out of three words as well, it is from the perspective of machine learning a single non-decomposable entity. The instance for the word "stemming" and its aligned equivalent "vote" is illustrated in figure 2.9:

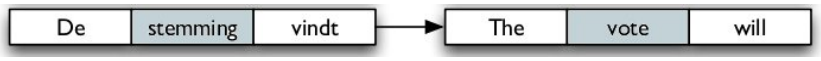


Figure 2.9: A training instance for the word "stemming" and its aligned equivalent "vote" (adapted from van den Bosch & Berck (2009)). The *focus word* in the middle is highlighted and the context words are left white. The left hand side illustrates the feature vector, and the right hand side is the output class.

We can create such instances for all words, except those that are part of a null alignment, by generating trigrams for the entire input sentence, resulting in the instances shown in figure 2.10:

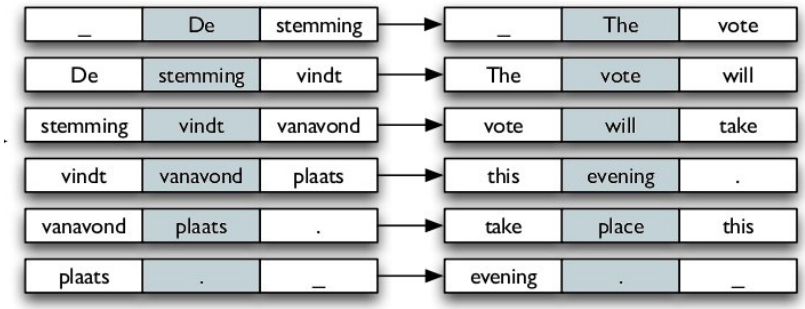


Figure 2.10: Training instances for all trigrams of the example sentence (adapted from van den Bosch & Berck (2009))

Previously unseen sentences that we want to translate are decomposed into trigrams in the same way. Based on the memory-based model of the training data, the classifier can predict a translation for each particular fragment. It thus predicts a class on the basis of one context word on the left hand side, a focus word, and a context word on the right hand side. Moreover, the predicted class itself can be seen to consist also of one context word to the left, a focus word, and one context word to the right. Of course the number of context words in either feature vector or output class, is not restricted to one, nor are they theoretically bound to be symmetric.

Note that this approach is referred to as a *word-based* approach because the *focus part* of the trigram, both in the feature vector as well as in the output class, is always a single word, and only the focus words are by definition translations of each other, the context words need not be aligned to each other. However, due to the role of the context information in both input and output, the word-based character described here is not a word-based character in the sense a naive word-by word translation would be: the usage of trigrams in both input and output already counts as a certain approximation of phrases. However, this is a far more

limited notion of phrases than the one that shall be presented in this thesis.

So also in the word-based approach, words are translated *in context*. This usage of context can be contrasted to *statistical machine translation*. Stroppa, van den Bosch, & Way (2007) explain that the statistical machine translation models express dependencies between source and target phrases, but not between the source phrases themselves, i.e. the context in which those phrases occur is never taken into account during translation. They remark that language models can be used to take into account *target* similarity, but statistical machine translation generally does not take into account *source* similarity, dependency between the source phrases themselves. Memory-based machine translation however, explicitly takes into account source-similarity, by using left and right context words as features. This is one important characteristic that sets it apart from approaches that do not do so, and where the aim is to improve upon them.

Let us now take a somewhat closer look at the way the *IGTree* memory-based learning algorithm actually predicts translations. In the previous section we briefly discussed this lazy-learning algorithm, which is an algorithm that compresses the instance-base into a decision tree structure and functionally approximates k-Nearest Neighbours. Applied to memory-based machine translation, we can trace the workings of IGTree as aptly explained by van den Bosch & Berck (2009):

> *"During translation, IGTree's classification algorithm traverses the decision tree, matching the middle, left, and right words of each new trigram to a path in the tree. Two outcomes are possible: (1) IGTree finds a complete matching path, upon which it returns the most probable output trigram; (2) IGTree fails to match a value along the way, upon which it returns the most probable output trigram given the matching path so far. Instead of the most probable path, it is also possible for IGTree to return the full distribution of possible trigrams at the end of a matching path."*

This again emphasises the fact that instances are represented as a path in the tree, a path which need not be a complete path. van den Bosch & Berck (2009) in this also referred to another important issue: Instead of just predicting one translation, one class, the algorithm can thus return a distribution of possible translations, each associated with a certain probability $P(t_i|s_i)$.

2.4.2 Global phase - Decoding

The next task, having classified all possible trigrams from S, is to again form one coherent sentence T using the predicted target fragments. This is a task for the *decoder*. Decoding is essentially a huge search problem, in which the task is to find the optimal, most probable translation. This translation consists of a composition of the predicted target trigrams. This search problem can often be stated as a task of optimising a certain objective function. Each hypothesis translation is assigned a certain score that is an expression of the quality of solution, and the task is to find the one with the highest score.

The size of the solution space can be illustrated by taking a source sentence S of nine words. If we assume all the words have an alignment, then nine test instances will be generated. In the classification stage these will each receive a probability distribution of class labels. If we were to only consider the best translation from this distribution, or if there were only one a

single output class per trigram, then our only task would be to find the correct order of these predicted classes, which would be a search-space consisting of $9! = 362880$ possible solutions. If we not only consider the best-predicted translation, but also take other options into account, then the search space increases significantly. In a configuration in which the class contains context words, we can exploit the *overlap* in context to aid in tying fragments together. This is exactly what is done by the MBMT decoder proposed in van den Bosch & Berck (2009). Figure 2.11 below shows this for the example sentence we used throughout this section, assume that these are the *predicted* trigrams from the local phase:

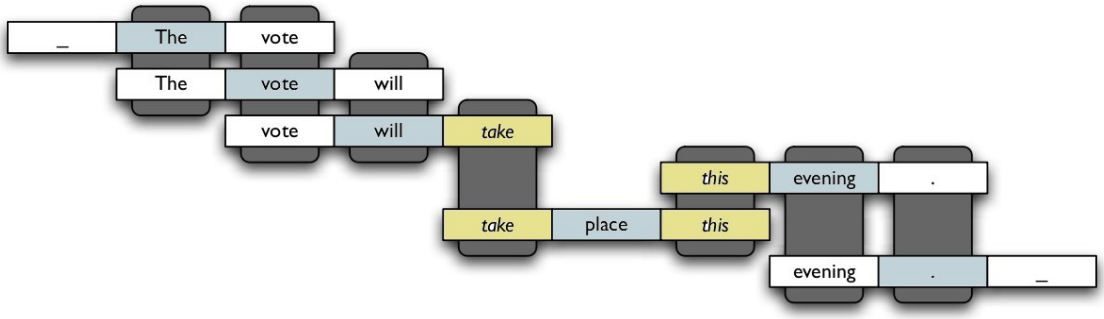


Figure 2.11: Re-assembling the target sentence through the usage of context overlap (adapted from van den Bosch & Berck (2009))

Generally only the focus words are used in the resulting translation, but the notable exception to this is when the right context of one class overlaps with the left context of another, in such a case the context can become a meaningful component of the final product. This is emphasised in the above figure for the words "take", and "this", which play a pivotal role in the overall assembly process. The sentence T that emerges from this can be found by following all the highlighted words, i.e. focus words and context overlap, in proper order, producing the sentence: *"The vote will take place this evening"*.

Note that the example case shown above is an *ideal* case. In practise, memory-based learning will not always predict trigrams that can be perfectly re-assembled. Situations will occur in which a trigram either has *no overlap* at all, or in which it can overlap with multiple trigrams. In the former case, a strategy that can be applied is for example simply generating the focus word of the next fragment. Moreover, we already stated that there can be a distribution of predicted trigrams, rather than a single one. All of this complicates the decode process. A language model, as discussed in section 2.2.2, is therefore used to select the most probable solution in the vast solution space.

2.4.3 Decoding by Constraint Satisfaction Inference

A new approach to decoding in memory-based translation was introduced by Canisius & van den Bosch (2009). Here the task of decoding is viewed as a *constraint satisfaction problem*. This is a search problem in which each state consists of an assignment to certain variables that are required to meet certain constraints. The goal is to find a set of values that satisfies the constraints or optimises a certain objective function; the latter applies to the application illustrated in this section. There thus is a *constraint model*, which specifies which combinations of values are allowed over the various variables, and assigns a weighted to each. This in effect makes it a *"weighted constrained satisfaction approach"*, where the weights correspond to the importance of the constraints (Canisius & van den Bosch, 2009). The candidate solutions to

such a problem are scored according to the sum of the weight of the constraints that they satisfy. In the end, the solution with the highest score is selected as the translation.

When comparing this approach to statistical machine translation systems, the constraint model can be seen as replacing the *translation model*. In addition to this, three other components are used in computing the final score of a translation hypothesis:

1. **Constraint Model** – Optimises translation quality *in context*.
2. **Language Model** – Optimises fluency
3. **Null Model** – A probabilistic model that deals with null-alignments and allows for the deletion of those words from the hypothesis that are often null-alignments.
4. **Length Penalty** – Compensates for the tendency of the language model to favour shorter sentence by adding a length penalty for shorter sentence, or length bonus for longer sentences.

The constraint model, at the heart of this decoding method, introduces one variable for *each trigram* of the source sentence, and each trigram as we know corresponds to a word in the source sentence. The domain of these constraint variables is the set of classes predicted by the classifier for the trigram in question. Note that spurious words in the source sentence did not generate test-instances; they are instead covered by the null model and are translated to a null symbol. This null symbol is part of the domains of all variables, allowing for the deletion of words deemed spurious.

In solving the constraint satisfaction problem, Canisius & van den Bosch (2009) base their approach on a greedy decoding algorithm by Germann (2003). This algorithm first generates an initial hypothesis in which all fragments are selected in the order in which they occur in the source sentence, and each is translated with the translation that came out best in classification. Next, a hill-climbing search applies simple transformations to this initial hypothesis and continues this until no further improvement can be attained. The following operators are implemented to transform hypotheses. The descriptions are adapted from Canisius & van den Bosch (2009).

1. **Change (Substitute)** – Substitute the translation of a source-language word. If the target word is aligned from multiple source words, a new target word is inserted in the translation at the position maximising the translation score; otherwise, the current translation is changed, while its position is left unchanged. Among the translation candidates, the null symbol is also tried, resulting in the *deletion* of a word from the translation hypothesis.
2. **Insert** - Insert a spurious word in the translation. A spurious word can be inserted when there is overlap between the right context of one predicted translation and the left context of the subsequent one. An example of this was seen in figure 2.11, with the words *"take"* and *"this"*. Due to the context size being one, only one spurious word can be inserted.
3. **Erase** – Erase a spurious word
4. **Join** – Join two target-language words, i.e., removing one of the words from the translation and aligning with the remaining word all words previously aligned with the word that was removed. (Canisius & van den Bosch, 2009)

5. **Swap** – Swap two non-overlapping segments of the target sentence. This allows for reordering the fragments in the translation hypothesis.

A constraint satisfaction problem can be visualised as a *constraint graph*, in which the nodes of the graph correspond to variables of the problems and the arcs correspond to constraints Russell & Norvig (2003). A *constraint graph* for the search space of one test phrase and its predicted classes, is shown in figure 2.12, from Canisius & van den Bosch (2009).

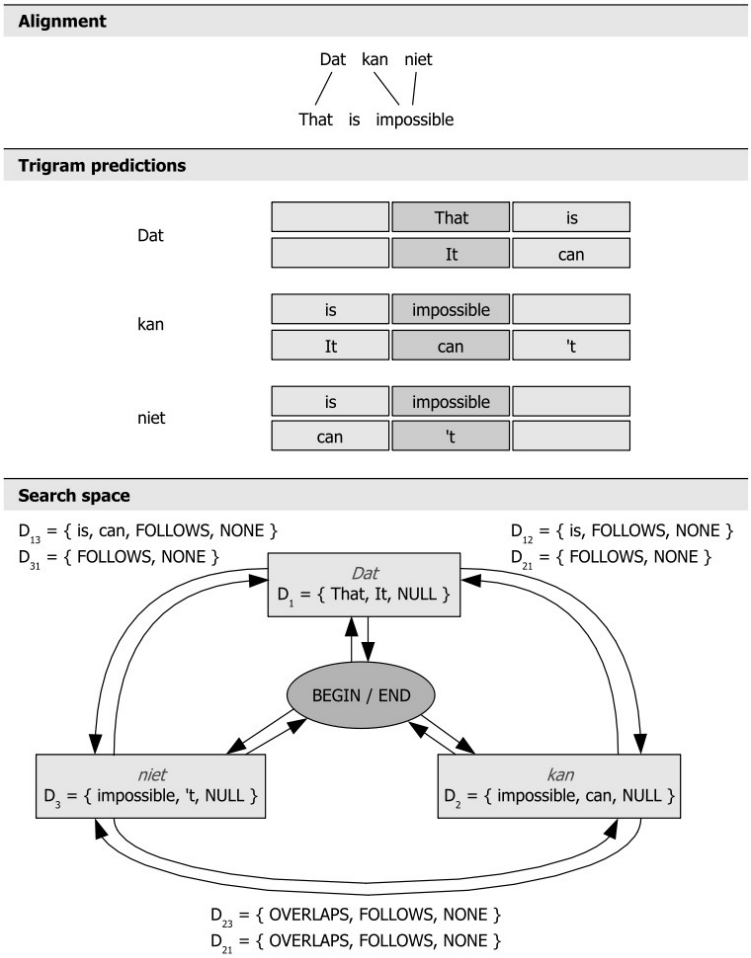


Figure 2.12: Constraint graph for one sentence composed of three test fragments (Canisius & van den Bosch, 2009). The top part shows the correct alignment, the middle part shows the trigram predictions for each of the word, and the bottom part shows the actual constraint graph. Any valid translation is a directed cycle in this graph that starts and ends in the `BEGIN/END` node

Figure 2.12 shows a classification that is ambiguous. For each source fragment, two classes have been predicted, resulting in a larger search space. On the basis of the predicted fragments, the domains of the variables are populated and constraints are added to the inference. Each domain thus has two words, and an additional `NULL` symbol that is part of all domains, allowing for a word to be dropped from the target translation hypothesis.

The edges of the graph each come with an associated domain as well, here the keyword `FOLLOWS` appears if the one pointed to may follow the other, which is always part of the domain as we can reorder every fragment in such a way that it follows another. `NONE` is

always included as well, and indicates no constraint between the two. The words that appear in these domains indicate spurious words that may be *inserted*, i.e. words that are found in the right respective left context of the fragments. The symbol OVERLAPS indicates that the two words overlap.

Chapter 3

Setup & Implementation

Many critics, no defenders,
translators have but two regrets:
when we hit, no one remembers,
when we miss, no one forgets.
– Anonymous

In this chapter I will present my own research. In the previous chapter, I laid out the theoretical background that is essential for a proper understanding of this chapter. We discussed machine translation in general, memory-based learning, and lastly we saw memory-based learning applied to machine translation, leading to *word-based memory-based machine translation*. The focus of my research, as has already been emphasised on previous occasions, is to incorporate phrase-based support in memory-based machine translation and to investigate how such a method relates to a mere word based approach. The actual results of this approach will be discussed and compared in the fourth chapter, in this chapter we will focus only on the theoretical setup and the implementation of phrase-based memory-based machine translation.

3.1 Phrase-based Memory Based Machine Translation

In word-based memory-based machine translation we worked with fixed trigrams consisting of a left context word, focus word, and right context word, both in the feature vector as well as in the corresponding output class. Of course the number of context words in both feature vector and class can be chosen differently, and need not even be symmetric. The actual focus word however always consisted out of one single word. We thus dealt with a configuration that was always fixed.

The idea that underlies this thesis is: what if we take here a phrase instead of a word? Instead of one focus word, we can take a phrase consisting of multiple words. So again it should be noted that a phrase is defined as a sequence of consecutive words, a subset of a sentence. Phrases in the source language would be mapped to phrases in the target language, again taking into consideration the context in which these phrases occur.

To illustrate this, we take first the following *word* alignment between a French source sentence and its English translation:

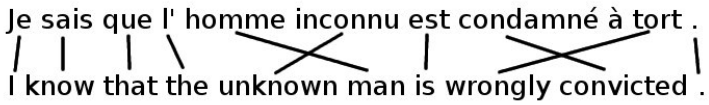


Figure 3.1: A *word* alignment between a French and English sentence

Now suppose we can extract *"l'homme inconnu"* as a phrase in the source language, aligned to the phrase *"the unknown man"* in the target language. The actual means of extracting these paired phrases will be clarified later in this chapter, for now we will just presuppose we can extract this pair. An example for this phrase-based instance for training can then be visualised as in figure 3.2:

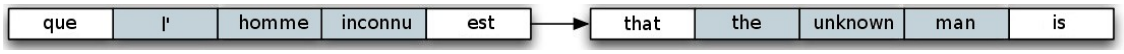


Figure 3.2: Visualisation of a phrase-based training instance in context. From French to English

We thus see that whereas we previously had a single focus word, we now have a phrase consisting of multiple words. If we take a look at yet another phrase-pair we presuppose to be extractable from the example sentence pair (figure 3.3), then we see clearly that the span of the source phrase need not be equal to the span of the target phrase. Moreover, there will be source phrases and target phrases of different lengths. We thus have a far more dynamic approach than in word-based memory-based machine translation:

Figure 3.3: Visualisation of another phrase-based training instance.

3.1.1 Hypothesis

One of the questions posited in the first chapter, was what advantages we can expect from a phrase-based approach as opposed to a word-based approach. After all, the phrase-based method is only worth pursuing if we can expect a gain from it. We already established the inadequacy of a very naive word-by-word translation. Words simply do not map one on one, and expressions differ greatly across languages. The word-based memory-based machine translation approach greatly makes up for this by incorporating context information, both source-side and target-side. But with the focus consisting out of merely one word, this approach still has some disadvantages that might be solved in a phrase-based approach. Let us look at some of the advantages of phrase-based over word-based:

1. **Local reorderings** - When using phrases instead of single words, local differences in word-order can be captured in the phrases themselves, whereas they would have otherwise been left to be resolved by the decoder, usually relying on the language model. This is apparent in both figure 3.2, as well as figure 3.3. In both cases the French and English phrases use a reversed word-order. Treating the phrases as an entity prevents us from having to deal with reordering these words later. We can thus predict to be less affected by word-ordering problems.

2. **Spurious words** - A phrase can contain spurious or null-aligned words, words that are not aligned with any word in the target sentence. Taking the phrase and its translation as an entity, solves the problem of having to deal with a word that has no counterpart in the other language. An example of this is the word *"à"* in figure 3.3. In the word-based approach described by Canisius & van den Bosch (2009), the decoder uses a special *null model* to try to compensate for this, from the null model one can infer which words are good candidates for deletion. I predict a phrase-based approach to be less sensitive to this, if only enough spurious words are within the span of a phrase.

3. **One-to-many alignments** - A phrase in a sense is a many-to-many alignment. It aligns two word-sequences which may have a different span. The word-based memory-based approach has difficulty resolving the one-to-many alignments that may be inherent in a phrase. The system is inclined to translate each word with one translation. The only way in which the system attempts to resolve this is through making use of the context overlap as shown in figure 2.11. However, here we are limited by the amount of context words in the output class, one word in this configuration. Of course it is possible to increase the amount of context, but as we will see later this brings quite some problems. The usage of phrases is predicted to solve a lot of the problems in situations where one word in the source language translates to multiple words in the target language, or vice versa for that matter.

One other argument in favour of a phrase-based method, is simply the observation of past results. In statistical machine translation, phrase-based methods have been shown to produce a significant improvement over word-based methods. Therefore the latest research in the field is mainly focussed on phrase-based methods, see for example Koehn (2004).

What we thus hope and expect to find, is that translations produced by a phrase-based memory-based machine translator show an improvement in translation quality compared to those produced by a word-based memory-based machine translator.

3.1.2 Setup

The task of phrase-based memory-based machine translation can be divided into several subtasks. Different software components are used to perform each task. New software components have been written for each of the tasks that are specific to phrase-based memory-based machine translation. We can distinguish two main phases, a training phase and a testing phase. The training phrase takes as input a parallel corpus and produces a memory-based model that is later used in the testing phase for classification. The testing phase takes as input a collection of input sentences in the source language, and eventually produces the translation of that input. The whole setup of the phrase-based memory-based machine translation system that will be proposed in this chapter, is illustrated in figure 3.4.

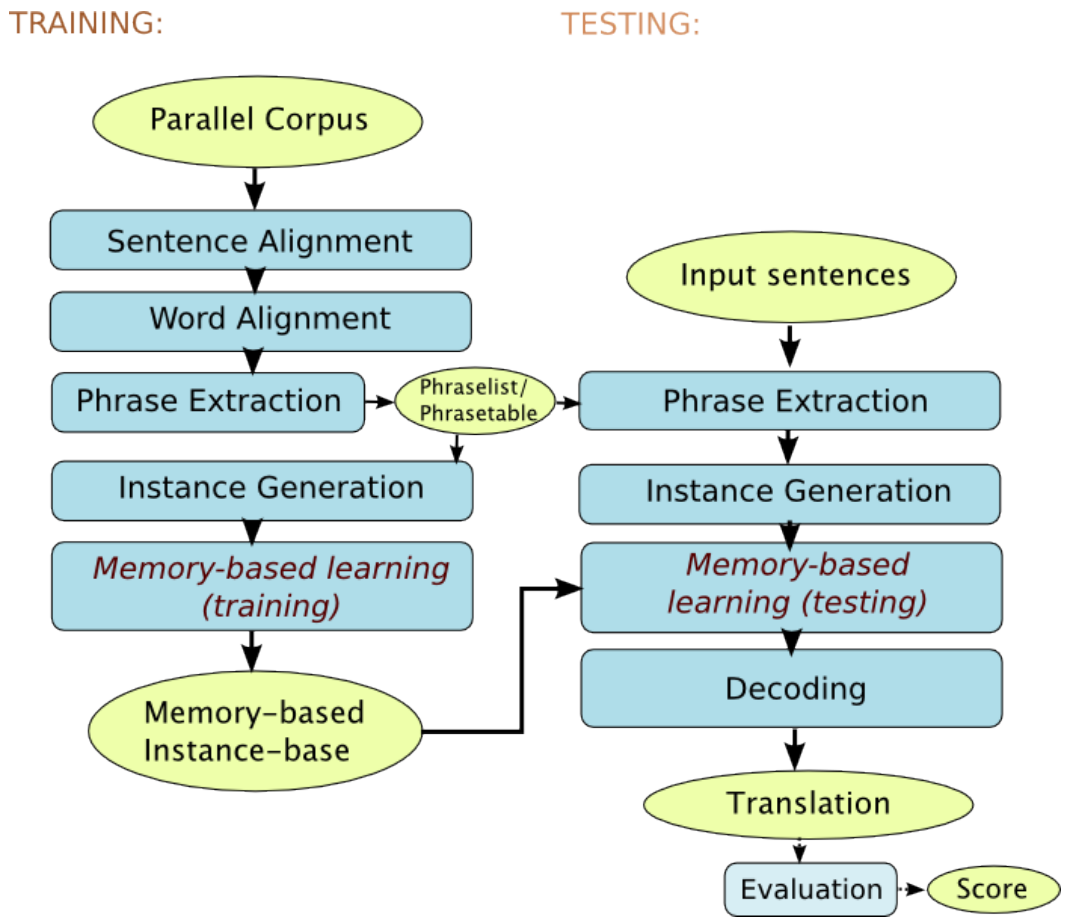


Figure 3.4: A scheme showing the overall setup of the proposed phrase-based memory-based machine translation system. The left-hand side corresponds to the training phase, and the right-hand side corresponds to the testing phase. Green round nodes denote data, and blue square nodes denote processes that manipulate the data

In the next sections of this chapter, we will discuss in detail all of the elements shown in figure 3.4. First however we will present a more general overview of this setup.

We start with the training phase. The input thereof is a tokenised parallel corpus. The next step is to establish which source sentences are paired up with which target sentences, and discard all those that can not be matched. This produces a collection of sentence pairs in source- and target- language. Next, we compute a *word-alignment* between all sentence pairs. We have seen how this is done in section 2.2.2. We use the software *GIZA++* (Och & Ney, 2003) for this task.

Up until this point, the procedure for word-based and phrase-based memory-based machine translation has been the same and no custom-written software has been needed yet. The divergence starts at the level of extracting phrases, which is of course only an issue in the phrase-based approach. We will see in the next section a discussion of various ways of extracting phrases. The extracted phrase-pairs or phrases will be stored in a data file and these in turn are read by the *instance generator* and form the basis for the generation of training instances. The instance generator is specific to the phrase-based approach, so custom software has been developed for this purpose.

After the training instances have been generated, the actual training can begin. As mentioned in section 2.3, we use the software TiMBL[1] (Daelemans, Zavrel, van der Sloot, & van den Bosch, 2007) to this end. The result of the training is a memory-based instance-base, more precisely, a losslessly compressed decision tree model of all instances. This model will be used by the testing phase, of which we will now sketch an overview.

The testing phase starts with a collection of input sentences in the source language. These are the sentences to be translated. The *instance generator* decomposes the sentences into test instances. It does so on the basis of the phrases extracted during the *training* stage, so this data serves as extra input to the instance generator, as shown in figure 3.4.

Once the test instances have been produced, it is the task of TiMBL to perform the actual classification using the memory-based model constructed during the training phase. For each test instance, a translation will be predicted. Last but not least, the *decoder* comes into play. The task of the decoder is as always to combine all translated fragments together into a coherent translation. For each input sentence it searches for the best configuration of the predicted translation fragments. Once complete, we end up with the final translation for each sentence and we are technically done. For research purposes, such as this thesis, we of course want to judge the quality of our translation and compare it to existing approaches. For that reason there is an extra evaluation stage, which assigns several scores to the produced translations. Chapter four will go into detail on this matter. We first limit ourselves to an extensive overview of the workings of the system we just sketched.

3.1.3 Implementation Setup

The phrase-based memory-based machine translation system proposed in this thesis has been implemented as chain of various technologies. This reflects the setup shown in figure 3.4. The implemented chain presupposes that the sentence-alignment and GIZA++ word-alignment have been performed already. If a phrase-translation table is used, it is also presupposed that this has been generated prior to running the processing chain. The subsection on that method will provide details on the software used for that purpose. Once all input requirements are met, the following custom scripts, written specifically for this thesis, can be called. They are listed chronologically in order of invocation:

1. `pbmbmt-make-phraselist` - A python program for the creation of a phrase list, necessary only for the phrase-list approach to phrase extraction.
2. `pbmbmt-make-instances` - A python program for the generation of training and test instances.
3. `pbmbmt-train` - A shell script wrapping the invocation of TiMBL for training.
4. `pbmbmt-test` - A shell script wrapping the invocation of TiMBL for testing.
5. `pbmbmt-decode` - The phrase-based memory-based decoder, a python program.

All custom software for this thesis has been implemented in Python 2.5 (van Rossum, 2006) and is targeted at Linux and other Unix-based operating systems. Small portions of the code have been designed originally in the context of the Constraint Satisfaction Inference Machine Translator by Canisius & van den Bosch (2009), and have been reused.

[1] Tilburg Memory-based Learner

The developed software is freely obtainable from the supplementary website for this master thesis, to be found at `http://proylt.anaproy.nl/projects/pbmbmt`. Furthermore, a practical "how-to", describing how to use the software, can be found on the webpage as well.

All code is available as open-source under the GNU General Public License v3[2].

3.2 Phrase extraction & Instance generator

One of the key questions in this research is the question on how to extract phrases. In our discussion on Statistical Machine Translation in section 2.2.2, we already saw a detailed answer to this question. We saw how two counter-directional word-alignments were combined into a phrase-alignment, which allowed for the construction of a *phrase-translation table*. It makes sense to use exactly this phrase-translation table as a source for extracting phrases for instance generation, so this is precisely what we shall do. But in addition to this, we will also try two other approaches of phrase extraction, yielding a total of three methods subject to investigation:

1. Phrase-translation table
2. Phrase list
3. Marker-based chunking

The second method of phrase extraction is the phrase-list approach, which is a simple method that extracts common n-grams in a corpus. The last approach is *marker-based chunking*, which segments a sentence into phrases, splitting whenever so-called *marker* words occur. One of the questions to be answered later in this thesis is how these three methods compare and which scores best.

Once we know what the phrases in a given sentence are, the procedure for generating instances is fairly simple. The algorithm for the generation of training instances can be formalised as shown in algorithm 1:

Algorithm 1 Training-Instance-Generator

Require: A set consisting of (S, T, A) tuples of source-sentence, target-sentence and a word-alignment between two. An integer n to specify the context-size.

Ensure: Set of training instances I, containing each a tuple consisting of a feature vector and a class label.

$I \Leftarrow \emptyset$
for all $(S, T, A) \in SentencePairs$ **do**
  for all $(P_s, P_t) \in ExtractPhases(S, T, A)$ **do**
    Features $\Leftarrow LeftContext(S, P_s, n) \cup P_s \cup RightContext(S, P_s, n)$
    Class $\Leftarrow LeftContext(T, P_t, n) \cup P_t \cup RightContext(T, P_t, n)$
    $I \Leftarrow I \cup (Features, Class)$
  end for
end for
return I

[2]The license can be obtained from http://www.gnu.org/copyleft/gpl.html

Here we delegate the hard work of extracting the phrases, to an abstract function that returns aligned phrase pairs, making use of one of the three aforementioned methods of phrase extraction. We will shortly see precisely how these methods work. We also make use of the functions $LeftContext$ and $RightContext$ to extract n extra context features. For simplicity, the formalisation above uses a single value of n, but in practise these are implemented as two or four different variables that can be set independently.

For the generation of test instances, the procedure is simpler, we after all need to consider only the source sentences, as the target sentences are completely unknown and are precisely the data we seek to obtain through classification. Note that for the generation of test instances we require the phrase list or phrase-translation table that has been generated using the *training* data, depending on the method chosen. In generating test instances using the phrase-list or phrase-table method, we can thus only extract phrases that we also detected during training. Only marker-based chunking operates in a different fashion. We will explain this somewhat more deviating method last.

The instance generation algorithm for testing can be formalised as in algorithm 2 below:

Algorithm 2 Test-Instance-Generator

Require: A set of input sentences. An integer n to specify the context-size.
Ensure: Set of test instances I, containing a feature vector each.

1: $I \Leftarrow \emptyset$
2: **for all** $S \in InputSentences$ **do**
3: **for all** $(P_s) \in ExtractPhrases(S, T)$ **do**
4: Features $\Leftarrow LeftContext(S, P_s, n) \cup P_s \cup RightContext(S, P_s, n)$
5: $I \Leftarrow I \cup (Features)$
6: **end for**
7: **end for**
8: **return** I

Phrase overlap

Thus far we have not yet mentioned one of the problems inherent in a phrase-based approach. An important issue is the fact that often, a sentence can simply not be decomposed entirely into phrases using the extraction methods we will discuss in the next section. In most cases there will be *gaps* between phrases, or in the worst case no phrase at all can be found in a sentence. If we return to the French-English sample alignment of figure 3.1, and we assume that the only two phrases we managed to extract are the two we used in subsequent examples, then we obtain a coverage as shown in figure 3.5 below:

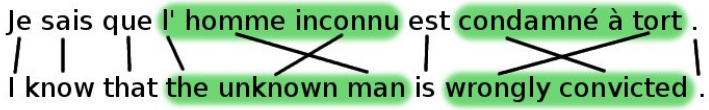


Figure 3.5: Word-alignment with coverage of extractable phrases, highlighted in green

In this case we see that four words in the source language are not covered. If we were to employ a method of instance generation that purely relies on extracted phrases, then we would miss out on a lot of data, depending on the quality of our phrase-translation table or phrase

list. However no phrase list or phrase-translation table will be good enough to prevent the occurrence of gaps. Gaps occur precisely in those places where the probability of the transition between words is lower, and/or the probability of a reoccuring alignment is lower. Phrase lists and phrase-translation tables on the contrary try to capture common patterns with higher probability.

The solution to this problem is straightforward. We already defined a phrase as consisting out of *one or more* words. So in addition to generating instances using one of the various phrase-extraction methods presented in the next section, we generate word-based instances as well. It is important to realise that this makes the phrase-based approach an extension of the word-based approach in a close sense: Given the same parallel corpus and input sentences, the training instances and test instances in phrase-based memory-based machine translation are a *superset* of those in word-based memory-based machine translation.

In our approach, the word-based instances are generated in the same fashion as explained in section 2.4, by moving a sliding window of length n over the source sentence. In addition to that we thus also have the extracted phrases. This leads to a certain overlap in instances. A word in the source sentence can be part of the focus of a feature vector multiple times. There is the word-based instance, but there may also be one *or more* extracted phrases that the word is a part of. We may thus generate multiple instances that all contain the same word or words as part of their focus.

I will elaborate on this by stressing the possibility of overlap in phrases. Figure 3.6 shows four phrase-pairs that can be hypothetically extracted, and there is overlap between two of them:

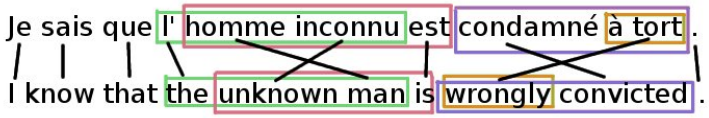


Figure 3.6: Overlap in coverage of extractable phrases

Here *"homme"*, *"inconnu"* and *"tort"* would each be part of the focus of a training instance *three* times[3]. The possibility of phrase overlap was also shown in figure 2.6 of the previous chapter, in the discussion on phrase-extraction in statistical machine translation. Since this is the source of the phrase-translation table we use, the likelihood of overlap becomes apparent.

Phrase overlap occurs in both training instances and test instances. The latter has an important side-effect that will have an impact on the decode process we describe later on. If there are multiple instances covering the same words, then there will be multiple possible *fragmentations* of the input sense. If we take the French input sentence in figure 3.6 as example, then we can decompose the source sentence into the following fragmentations using just the four highlighted phrases, as shown in figure 3.7:

[3] "à" would occur only two times since it will not generate a word-based instance as it is a null-aligned word

Je	sais	que	l'homme inconnu	est	condamné à tort
Je	sais	que	l'homme inconnu	est	condamné	à tort
Je	sais	que	l'homme inconnu	est	condamné	à	tort
Je	sais	que	l'	homme inconnu est	condamné à tort
Je	sais	que	l'	homme inconnu est	condamné	à tort
Je	sais	que	l'	homme inconnu est	condamné	à	tort
Je	sais	que	l'	homme	inconnu	est	condamné à tort
Je	sais	que	l'	homme	inconnu	est	condamné	à tort
Je	sais	que	l'	homme	inconnu	est	condamné	à	tort

Figure 3.7: Possible fragmentations of a sentence

3.2.1 Instance format

We have seen in the beginning of this section what exactly constitutes an instance. Training instances consist of a feature vector and a class, and test instances consist merely of a feature vector, as the class is what needs to be predicted. The feature vector consist out of a focus part, which can be either a phrase of arbitrary length or a single word, and left- and right-context features of fixed length. We will have to narrow in on this from the perspective of machine learning, because we have a certain issue to resolve. The careful reader might have already noticed that we if we take all features to be words, we end up with feature vectors of different sizes. If we leave it at that we have a problem, because many classification algorithms demand a fixed set of features in order to be able to relate the same feature across different instances to each other using some distance function.

There are at least three possible ways to resolve this problem:

1. **Consider the phrase as one single feature**: Instead of using multiple features for the focus part, we can put the phrase as an entity in one single feature rather than letting it span over multiple features. This thus leads to having once again a fixed number of features.
2. **Reserve a fixed number of features**: Reserve a fixed number (n) of features for the focus part and fill those with the phrase or word, assigning a dummy value to any slots that are unused. This solution restricts the maximum length of phrases to n.
3. **Output to multiple instance files**: Write instances with a particular focus-span to multiple training- and test files. Instances with a focus-span of n will be written to a file that contains all instances and only instances of length n. All files will be trained and tested separately.

The first solution appears to be the most straightforward one. It is therefore also the first approach implemented and tested. Using the training instance shown in figure 3.2, the instance would be generated as follows in a format that can be understood by TiMBL for training:

```
que l'_homme_inconnu est that^the_unknown_man^is
```

Figure 3.8: Output of one training instance, considering the phrase as a single feature

And figure 3.9 shows a test instance, which would contain a dummy question mark instead of a class:

```
que l'_homme_inconnu est ?
```

Figure 3.9: Output of one test instance, considering the phrase as a single feature

Three characters have special meaning in this output format. A *space* character demarcates columns of features, with the very last column representing the class label. This implies that it is not possible to use spaces to separate words in a phrase if we want the phrase to be considered a single feature, therefore *underscores* are used instead. Likewise, underscores are also used in the class label as a delimiter between the words of the target phrase. Lastly, we use *carets* to separate the context from the focus in the class label. The class by definition is a single entity so it cannot contain spaces. We will later see tests in which the context is left out on the *target* side, as shown in figure 3.10:

```
que l'_homme_inconnu est the_unknown_man
```

Figure 3.10: Output of the same training instance, but without context information in the class label

On the source side, we always maintain at least one context feature. There is a fourth special symbol not shown in the example, this is a double underscore, representing the beginning or the end of sentence. This double underscore may only occur in the context features or the context portion of the class.

During research and development it quickly became apparent that there were notable disadvantages to taking the phrase to be one single feature. In chapter 4 we will zoom in on the actual results. As explained in section 2.3, machine learning searches for common patterns amongst the same features of all instances. If phrases are taken as entities, then they can only be related to other phrases as a whole, either matching or not matching. If each word of a phrase has its own feature, then we have more variation and more grounds for comparison and pattern detection. Moreover, in machine learning it is generally a good idea to have a set of informative features that each contribute some amount of information not available from other features. So better results may be attained if we take the features separately.

Because of the poor results produced by the first method, I worked out and implemented the other methods as well. In chapter four the results of these different formats will be compared and discussed. The second solution takes a fixed amount of features (f) for the focus part, this implies that phrases are restricted to a maximum length. When a phrase P_s is generated, $f - |P_s|$ features will be unused; these receive a null value, symbolised by a single caret in this format. One additional and important feature that I introduced here is a feature that specifies the length of the phrase ($|P_s|$). I did this so the machine learning algorithm has a way of disambiguating on the phrase-length, and hopefully making it better at coping with the many null values which might otherwise have a negative impact on the accuracy. Let us take a look at figure 3.11, illustrating how the example training instance would look in this format (assuming $f = 6$):

```
n=3 que l' homme inconnu ^ ^ ^ est that^the_unknown_man^is
```

Figure 3.11: Output of a training instance with six fixed focus-features

The focus phrase consists of three words, and we reserved six slots for the focus, so three slots receive a null value. Note that the number of reserved features applies only to the *focus* part, in addition to this there are the context features, one left and one right.

The third solution deals with a fixed set of features as well. But as it outputs phrases of different length into different files, we receive a less complex file format, but it results in *multiple* files.

```
que l' homme inconnu est that^the_unknown_man^is
```

Figure 3.12: Output of a training instance into the training file for $n = 3$

There again is a certain maximum phrase-length f, as beyond a certain limit the probability of phrases gets so low that the data becomes extremely sparse and contains only a few instances. Each of the generated files is trained and tested *independently* of the others, Timbl is thus invoked f times for training and f times for testing, which could be reduced to f times in total if training and testing were combined.

As already mentioned, in chapter four we will return to this discussion and see in more detail the results of these different formats of instance generation.

3.2.2 Phrases from a phrase-translation table

In section 2.2.2 we saw the theory underlying the extraction of phrases in statistical machine translation. We can delegate the task of extracting phrases to tools provided by a statistical machine translation system. Given a parallel corpus, we use the `train-factored-phrase-model` script in the software toolkit *Moses* (Koehn et al., 2007) to generate a phrase table. We treat the phrase-translation table as a simple lookup table in which we can issue a lookup for a phrase in the source language, and obtain as a result its translation in the target language. If there are multiple translations for a given source phrase, then all possible translations will be returned and tried.

The script provided by *Moses* is actually a processing chain which will first invoke GIZA++ (Och & Ney, 2003) to compute the word-alignments, and subsequently derive phrase-alignments and extract phrases in the fashion already demonstrated. We thus essentially use a substantial portion of what is also used in statistical machine translation. It is therefore interesting to later on compare the performance of this phrase-translation table approach to statistical machine translation systems that also use this very same phrase-translation table, albeit in a very different fashion.

A simplified high-level description of phrase-extraction using a phrase table is provided in algorithm 3 below:

Algorithm 3 Extract-Phrases-Using-Phrasetable (for training)

Require: A source sentence S, target sentence T, word-alignment A between the two, a phrasetable, and a maximum phrase length m
Ensure: A set X of phrase-pairs (P_s, P_t) $X \Leftarrow \emptyset$

```
for all P_s ∈ set of all n-grams in S where n ≤ m ≤ |S| do
  if P_s ∈ phrasetable then
    P_t ⇐ QueryPhrasetable(P_s)
    if P_t ∈ set of all n-grams in T where n ≤ |T| then
      if Alignment between P_s and P_t does not violate A then
        X ⇐ X ∪ (P_s, P_t)
      end if
    end if
  end if
end for
return X
```

The algorithm decomposes a source sentence into all possible n-grams for all values of n up until a predefined maximum. Then the phrase-translation table is used to check if the n-gram is an actually defined phrase. The vast majority of n-grams can be discarded immediately. For the few that are found in the phrase-translation table, the matching translation is queried. Since the memory-based machine translation takes into consideration context, it is necessary to verify that the translation of this phrase is in fact the one that is used in the aligned target sentence and that it is consistent with the word-alignment. A good phrase-pair is only found and returned if all of these constraints are met.

Algorithm 3 is to be used in the generation of *training* instances. The algorithm for testing, shown in algorithm 4, is simpler and generates instances for all phrases that are found in the phrase table. This implies that a prerequisite for good test instances is that the phrase-translation table is generated from the very same dataset as used later in training. If the phrase-translation table were larger we might end up with a lot of test instances for phrases that were not seen in the training stage, and thus will perform poorly in classification.

Algorithm 4 Extract-Phrases-Using-Phrasetable (for testing)

Require: An input sentence S, a phrasetable, and a maximum phrase length m
Ensure: A set X of phrases $X \Leftarrow \emptyset$

```
for all P_s ∈ set of all n-grams in S where n ≤ m ≤ |S| do
  if P_s ∈ phrasetable then
    X ⇐ X ∪ (P_s)
  end if
end for
return X
```

3.2.3 Phrases from a phrase list

Extracting phrases using a phrase list is a rather simple method of phrase extraction. This simplicity is the principal advantage it has to offer over the phrase-translation table method of extraction. The generation of phrase lists does not rely on the extraction of phrase-pairs using relatively complicated algorithms such as those used for the generation of a phrase-translation table. A phrase list is a monolingual list of phrases. More specifically; I define a phrase list as a list of *common n-grams*, extracted from a corpus. Here all n-grams up until a certain

maximum length m are extracted ($2 \leq n \leq m$), and the occurrence of these n-grams are counted. All n-grams having a frequency count of over a predefined *threshold*, are included in the list.

The algorithm for the creation of phrase lists can be formalised and seen in algorithm 5 below. In this algorithm M is the memory store that contains phrases associated with their frequency.

Algorithm 5 Make-Phrase-list

Require: A corpus consisting of sentences, and a maximum phrase length m
Ensure: Outputs a list of common phrases

1: $M \Leftarrow \emptyset$
2: **for all** $S \in Corpus$ **do**
3: **for all** $P \in$ set of all n-grams in S where $n \leq m \leq |S|$ **do**
4: $M \Leftarrow IncrementCount(M, P)$
5: **end for**
6: **end for**
7: **for all** $(P, Count) \in M$ **where** $Count \geq t$ **do**
8: $OUTPUT(P)$
9: **end for**
10: **return** X

The above algorithm is a simplified version. The actual implementation is functionally equivalent but has been optimised to conserve memory-usage as otherwise the implementation would reach memory limits when processing huge corpora. It first counts bigrams, then trigrams, quadgrams, etc... At each each stage of n-gram counting, it uses the data from the $(n-1)$-grams to dismiss n-grams that do not pass the threshold. After all, if an n-gram occurs x times, then the *two* $(n-1)$-grams contained in it also need to occur *at least* x times. If this is not the case, we can dismiss the n-gram and save precious memory.

Once we have generated a phrase list. We can use it for the generation of training and test instances. The phrase extraction process proceeds in a similar manner as seen in the phrase-translation table algorithm. With the main distinction that the aligned phrase does not come from a predefined table, but is extracted directly from the word-alignment. Take a look at the algorithm (6) for the extraction of training instances:

Algorithm 6 Extract-Phrases-Using-Phraselist (for training)

Require: A source sentence S, target sentence T, word-alignment A between the two, a phrase list, and a maximum phrase length m
Ensure: A set X of phrase-pairs (P_s, P_t)

1: $X \Leftarrow \emptyset$
2: **for all** $P_s \in$ set of all n-grams in S where $n \leq m \leq |S|$ **do**
3: **if** $P_s \in$ phraselist **then**
4: $H \Leftarrow GetAlignedWords(P_s, T, A)$
5: **if** H forms a consecutive series of words in T without null alignments **then**
6: $P_t \Leftarrow H$
7: $X \Leftarrow X \cup (P_s, P_t)$
8: **end if**
9: **end if**
10: **end for**
11: **return** X

First this again iterates over all n-grams in S, for all values n. Then there is a test to see if

the n-gram occurs in the phraselist. If so, we follow the word-alignments and if the words they lead to form a consecutive entity without gaps, we take that to be the aligned phrase.

For the generation of test instances, we do exactly the same as phrase-translation table, only using a phrase list instead of a phrase table, as shown in algorithm 7 below:

Algorithm 7 Extract-Phrases-Using-Phraselist (for testing)

Require: An input sentence S, a phraselist, and a maximum phrase length m
Ensure: A set X of phrases

1: $X \Leftarrow \emptyset$
2: **for all** $P_s \in$ set of all n-grams in S where $n \leq m \leq |S|$ **do**
3: **if** $P_s \in$ phraselist **then**
4: $X \Leftarrow X \cup (P_s)$
5: **end if**
6: **end for**
7: **return** X

3.2.4 Marker-based chunking

Marker-based chunking is a phrase extraction strategy that differs considerably from the previous two. Moreover, it has already been employed in a previous study of memory-based machine translation (van den Bosch, Stroppa, & Way, 2007). In this thesis I want to continue on the foundation laid in that research and assess the performance of memory-based chunking as a method of phrase extraction in comparison to the previous two methods.

The architecture overview in figure 3.4 does not reflect well the inclusion of this particular method. Unlike the phrase-translation table and phrase-list approach, marker-based chunking does not rely on a large pre-generated data store of phrase-pairs or phrases. Instead the phrases can be directly extracted on-the-fly by the instance generator, without a big assault on system resources. The other two generation methods on the other hand are computationally expensive, consuming a considerable amount of memory and taking a fair amount of CPU time.

The idea behind memory-based chunking is based on the Marker Hypothesis (Green, 1979) from psycho-linguistics:

> *"The marker hypothesis posits that all languages are marked for surface syntax by a specific closed set of lexemes or morphemes which signify context."* (van den Bosch, Stroppa, & Way, 2007)

In other words, a language contains a set of closed-class words, so-called *markers*, which reveal a surface-syntax structure in a sentence. Markers are function words such as pronouns, determiners, particles, prepositions/postpositions and conjunctions, in section 4.1.2 of the next chapter we will see from what data these markers have been extracted. A superficial phrase-segmentation can be achieved by using the occurrence of marker words as boundaries. These segments are referred to as chunks. *Chunking* can be defined here as the process segmenting the entire sentence into a series of consecutive chunks. Chunks are thus phrases that do not overlap, which contrasts them with the phrases extracted using a phrase list or phrase-translation table.

The example in figure 3.13 shows an English sentence segmented into marker-based chunks, the markers are highlighted:

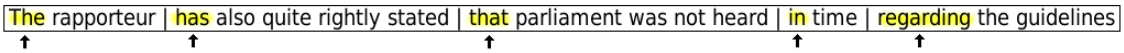


Figure 3.13: Example of marker-based chunking. (Adapted from van den Bosch, Stroppa, & Way (2007))

The chunks are generated whenever a new marker-word occurs, under the constraint that each chunk contains at least one content (non-marker) word (van den Bosch et al., 2007). In generation of training instances, both the source sentence as well as the target sentence are chunked independently. The English sentence in the above figure has a Dutch counterpart that can be chunked as follows:

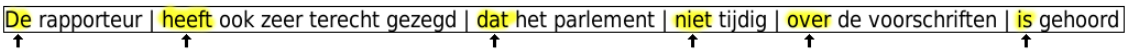


Figure 3.14: Marker-based chunking on a Dutch sentence. (Adapted from van den Bosch, Stroppa, & Way (2007))

The next step is to re-align the chunks of source- and target-language, using the word-alignment already at our disposition. This procedured is described in van den Bosch et al. (2007). What we essentially try to do is find the target chunk that has the highest probability of being aligned to the source chunk. We effectively estimate $P(C_t|C_s)$ for each source chunk , where $C_s \in S$ and $C_t \in T$, and align the source chunk with the most probable target chunk. For accurate results it is also necessary to perform the alignment in the other direction, estimating $P(C_s|C_t)$ and aligning each target chunk with the most probable source chunk. Then the *intersection* of both alignments can be taken to produce the final high-quality alignment.

The probability $P(C_s|C_t)$ and the reverse probability $P(C_t|C_s)$ are estimated directly using the word-alignment. A high-level description of the algorithm is provided as shown in algorithm 8.

In this algorithm, a matrix M is used to store the alignment. The algorithm consist of three phases, represented by the three loops:

1. For each source-chunk, find the best target chunk, the one that is aligned with most words in the source-chunk
2. For each target-chunk, find the best source chunk, the one that is aligned with most words in the target
3. Select and return the best matches from the intersection

Note that the product on line 12, in the second phase, effectively computes the intersection right-away.

3.3 Training & Testing

The previous section explained how training instances are generated from a parallel corpus and word-alignment, and how test-instances are generated from a test corpus. Three different

Algorithm 8 Chunk-Alignment

Require: A source-sentence S, and target-sentence T, both segmented into chunks.
Ensure: A set X of aligned chunks

for all $C_s \in S$ **do**
  for all $C_t \in T$ **do**
    $Count \Leftarrow$ count number of alignments that go from C_s to C_t
    $P(C_t|C_s) \Leftarrow \frac{Count}{|C_s|}$
    $M(C_s, C_t) \Leftarrow P(C_t|C_s)$
  end for
end for

for all $C_t \in T$ **do**
  for all $C_s \in S$ **do**
    $Count \Leftarrow$ count number of alignments that go from C_t to C_s
    $P(C_s|C_t) \Leftarrow \frac{Count}{|C_t|}$
    $M(C_s, C_t) \Leftarrow M(C_s, C_t) \cdot P(C_t|C_s)$
  end for
end for

$X \Leftarrow \emptyset$
for all $C_s \in S$ **do**
  for all $C_t \in T$ **do**
    if $\forall x(M(C_s, C_t) \geq M(C_s, x))$ **then**
      $X \Leftarrow X \cup (C_s, C_t)$
    end if
  end for
end for
return X

methods of phrase extraction have been introduced, and three different instance formats were presented. After all necessary instances have been created, the work of classifying, predicting translations for the various fragments is delegated to TiMBL (Daelemans et al., 2007). Section 2.3 explained the theory behind machine learning and memory-based learning, and laid out the algorithms used. The training stage will produce an instance base, and this instance based will subsequently be loaded and used by the testing stage for classification.

In all cases TiMBL is configured with $k = 1$, referring to the number of nearest neighbours to consider. We consistently configure TiMBL in such a way that it always outputs a *full* distribution of predicted classes, rather than resorting to a tie breaking mechanism and returning merely one class. The reason for doing this is due the fact that the classifier may assign the same probability, or a very similar probability to a number of classes. Disambiguation at this level might thus not always be possible or preferable, and it will be the task of the decoder to select the most suitable of the predictions, using the global context of the translation of the sentence as a whole.

The following real example in figure 3.15 illustrates the output of the classifier. Note that the instance displayed is a test instance generated using the split-file methodology, and that this particular fragment comes from the file containing phrases of *three* words. Moreover, there is no context in the class labels, unlike in the feature vector where the configuration is one left, one right:

Raw TiMBL output:

```
__ Stel dat je controle ? Assuming { Assuming 2.00000, lmag-
    ine_that_he_would 1.00000, Or_say_you 1.00000 }
```

Fragment (5 features):

left-context	focus	right-context
__	Stel dat je	controle

Distribution of predicted classes:

Assuming	0.5
imagine_that_he_would	0.25
Or_say_you	0.25

Figure 3.15: Example of classification with probability distribution. (split-file methodology, $n = 3$)

3.4 Decoder

One of the key components in any statistical or example-based machine translation system is the decoder. It is the task of the decoder to compose the final translation of a given input sentence. We recall that a decoder is needed because we can not translate the sentence as a whole, instead we translate *fragments* of the source sentence. The translation to each fragment is produced using classification, as shown in previous sections. This is what constitutes the *local phase*. The *global phase*, the decoder, seeks to find an optimal arrangement of these predicted translations, resulting in the translation of the whole sentence.

In sections 2.4.2 and 2.4.3, we discussed two memory-based machine translation decoders, the latter of which was based on constraint satisfaction inference. Decoding for phrase-based data introduces a new sets of challenges, therefore the existing decoders can not be used out of the box. For phrase-based machine translation we therefore propose a new decoder, inspired on the aforementioned memory-based decoders, but able to deal with phrases instead of words.

In the word-based approaches, test instances could be generated by moving a sliding window of length n over the source sentence and generating all n-grams it consists of (see section 2.4). In the phrase-based approach there is *phrase overlap* and such a straightforward approach is not possible. We do however still employ this approach to generate word-based instances *in addition* to the phrase-based instances, to prevent the occurrence of gaps.

3.4.1 Fragmentations, Fragments, and Hypothesis Fragments

Due to the overlapping nature of extractable phrases, the fact that we may end up with multiple instances covering the same words in the source sentence, we can speak of various possible *fragmentations* of the source sentence S. This was illustrated in figure 3.7. We define a *fragmentation* to be a chain of *non-overlapping fragments*, in which each *fragment* covers a certain range of consecutive words of arbitrary length n in source sentence S, where $1 \leq n \leq |S|$. In addition each fragment is associated with a left-context and right-context of a length predetermined during instance generation. The current implementation of the decoder does not allow for asymmetry in left- and right-context, the size of both should be equal and is defaulted at one. To indicate the beginning or end of a sentence, a special marker symbol (___) is used in the left or right context respectively. The idea now is to decode separately for each possible fragmentation, and then select the best result.

The decoder starts by reading the test corpus and the output files produced by the classifier *TiMBL* and containing the classified test instances. The test-corpus is read in a line-by-line fashion, in which each line is required to correspond to one sentence. Recall that depending on the method chosen, we may have instances split over multiple independent files. For each sentence S in the test-corpus, the output files are scanned for matching fragments. All matching fragments from the classifier output are associated with the sentence S. Note that this scanning procedure is implemented in a computationally inexpensive way, so it does not iterate over all instances in each file for each sentence. Instead, for each file it reads until an instance is found that no longer matches, also taking into account the fact that two consecutive sentences may be partially or wholly consisting of the same words and thus take the same subset of fragments.

Each instance in the classifier output is classified in the form of a distribution of classes, which are the various translations for the fragment and an associated probability score. These from the perspective of the decoder are referred to as *hypothesis fragments*. So each fragment will be associated with a collection of one or more hypothesis fragments, the associated scores denote the *translation probability* for the particular fragment being translated to the particular hypothesis fragment. It is important to understand the difference between fragments and *hypothesis* fragments, despite the possibly confusing nomenclature. The former are in the source-language, the latter are in the target language. The hypothesis fragments are eventually used to form *hypotheses*, which are translation candidates. Figure 3.16 illustrates the relation between the fragmentation, fragments and hypothesis fragments:

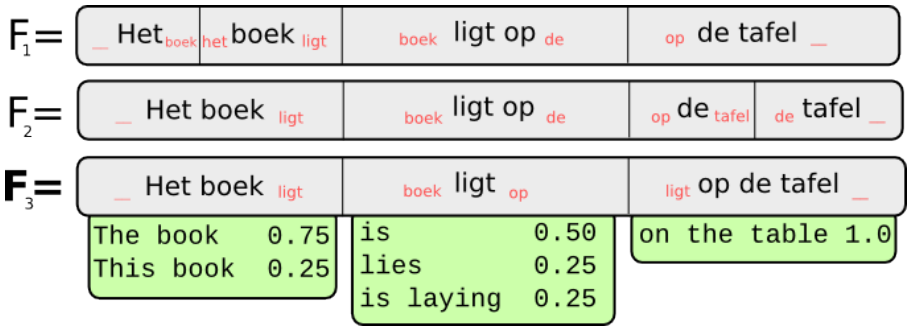


Figure 3.16: Schematic illustration of a fictitious input sentence *"Het boek ligt op de tafel"*, three fragmentations are shown, the third of which has been worked out to list the hypothesis fragments associated with each of the three fragments the fragmentation is composed of. In the fragments, context information is shown in small red text. Context has been set to zero for the hypothesis fragments.

Having gathered all matching fragments for a given source sentence, the task is to search for fragmentations, arrangements of fragments. We already demonstrated in figure 3.7 that the number of fragmentations can be huge, especially the longer sentences get. Therefore it proved to be an infeasible approach to generate all fragmentations in exhaustive fashion, and decoding for each fragmentation is not an option. Instead, we attempt to select a number of good fragmentations, and decode only for each of these.

Selecting a good fragmentation is a search problem with a large solution space. In order to solve this problem we resort to *local beam search*. This is a low-memory but non-optimal search algorithm often employed in Natural Language Processing tasks. Local beam search is a variant of the hill-climbing algorithm. If we apply hill-climbing to our problem, we start with an empty fragmentation. Then at each point in time we add the *best next fragment*,

until we eventually span the whole sentence. Now this solution need not be optimal, because the algorithm could be stuck in a local maximum in the search landscape. To increase the chances of finding good fragmentations, we keep k separate beams, instead of just one as in hill-climbing. The number of beams has been arbitrarily set to twenty. This implies we get at most twenty or k *best* fragmentations for each sentence. The formalisation can be seen in algorithm 9:

Algorithm 9 Search-Fragmentations

Require: A set $X_{current}$ containing fragmentations, initially containing only one element, an empty fragmentation. And a beam size k.

Ensure: The list containing the k best fragmentations found

```
1: X_next ⇐ ∅
2: for all F_current ∈ X_current do
3:   for all F_next ∈ Expand(F_current) do
4:     if Score(F_next) > Score(F_current) then
5:       X_next ⇐ X_next ∪ {F_current})
6:     end if
7:   end for
8: end for
9: if X_next == ∅ then
10:   return X_current
11: else
12:   X_next ⇐ Best k fragmentations in X_next
13:   return SearchFragmentations(X_next, k)
14: end if
```

Hill-climbing and local-beam search maximise a certain objective function that indicates the quality of a fragment. On the basis of this it is possible to decide which is the *best* next fragment, and express a level of quality for the fragmentation F as a whole. This to-be-maximised function is the product of the scores of each fragment, in which the score of each fragment ($\in F$) equals the *translation probability* of the highest scoring *hypothesis fragment*:

$$score(fragment_i \in F) = \text{max } Ptranslation(hfragments) \tag{3.1}$$

$$score(F) = \prod_{i=1}^{|F|} score(fragment_i) \tag{3.2}$$

For each of the fragmentations returned by the local beam search, the core decode procedure is started. It should be noted that this makes the phrase-based approach computationally more expensive compared to the word-based approach, as the latter by definition only has one fragmentation of the source sentence. The core decoder will return *a list* of the highest scoring translation hypotheses *for each fragmentation*. The length of this list can get as long as the *beam size* for the core decoder. The workings of the core decoder will be explained in the next section. However, before we go into that, we will take a look at how the lists returned by the core decoder are combined and used in the selection of one final translation.

One straightforward manner in which to select the final translation that is generated, would be to simply select the hypothesis that has the highest score, amongst all those returned by the possible fragmentations. This is one method that is indeed implemented, but it is somewhat naive and in addition to this, a more sophisticated method has been implemented as well.

First of all, it is the case that amongst the solution lists returned by different fragmentations, there may hypotheses that describe the very same output sentence. Thus two or more possible fragmentations may both predict the same output sequence of words, even though the underlying hypotheses may be composed differently (analagous to 3.7). What we thus do in the final solution search algorithm, is taking the *sum of the scores of all hypotheses that produce exactly the same output sequence of words in the target language*, accross all of the returned hypotheses for the various fragmentations. For example, the sentence T could be generated by any of the hypotheses H_1, H_2 or H_3, these are hypotheses returned by the core-decoder, possibly accross multiple fragementations. We have a disjunction and thus take the score of the sentence T itself to be the sum of the scores of these hypotheses: $score(T) = score(H_1) + score(H_2) + score(H_3)$. Then the only task that remains is selecting the sentence T for which the sum of the scores of the hypotheses that describe this sentence is maximised. This is the final translation generated by the system. Section 4.5 of the next chapter will show that this method indeed outperforms the simpler method.

3.4.2 Decoding

The core decode procedure takes a sentence S with a certain fragmentation F. It returns a list of the highest scoring *translation hypotheses*. A *translation hypothesis* is defined as a chain of *hypothesis fragments*, in which each hypothesis fragment pertains to a different *fragment*, and each fragment is a member of the fragmentation F.

The procedure starts by generating an *initial hypothesis*. This is a translation hypothesis in which we simply select for each fragment in the fragmentation, the highest scoring hypothesis-fragment, thus the hypothesis fragment with the highest translation probability. We simply order the hypothesis fragments for the initial hypothesis in the order we find the fragments in source fragmentation. The initial hypothesis in figure 3.16 thus is *"The book is on the table"*. In this example, the initial hypothesis happens to already be the best translation, but in most cases this is not the case.

Starting with an initial hypothesis, a global solution, has in this research proven to work considerably better than incrementally adding hypothesis fragments in a *Viterbi*-like fashion[4]. Any hypothesis can be modified in two main ways: the *order* in which the hypothesis fragments are assembled can be changed, and the *choice* of hypothesis fragments can be changed, i.e. other hypothesis fragments with an equal or lower translation probability could be tried. To this end, the decoder applies the following two operations to a hypothesis:

1. **Substitute** - Generate new hypotheses in which a hypothesis fragment of a particular fragment is substituted by another hypothesis fragment from the list. This is done in an exhaustive fashion. For each fragment, substitutions are made using all possible hypothesis fragments, each yielding a new hypothesis, each hypothesis fragment is used once to generate a new hypothesis. The only exception is that no substitutions are tried for a particular fragment if that very same fragment has already undergone a substitution operation in the previous decode round. This operation thus modifies the *choice* of hypothesis fragments.

[4]The first versions of the decoder were based on a different strategy than the current one. They did not start out with an initial hypothesis. Instead, there was more extensive *transition model* and decoding proceeded in a *Viterbi* like fashion. An empty hypothesis was the basis, and the decoder incrementally added hypothesis fragments, each time seeking the most probable next hypothesis fragment, which can be expressed as the conditional probability $P(hfragment_i|hfragment_{i-1})$. At the increment of i for each round of the decoder, this probability was estimated for all hypothesis fragments given the last hypothesis fragment in the partial chain.

2. **Swap** - Swap the location of two hypothesis fragments. This again is done in an exhaustive fashion such that all possible swaps are made. Each fragment swaps places with all neighbouring fragments within a certain range, each swap yielding a new hypothesis. An extra parameter specifies the maximum range over which a swap can occur. This thus modifies the *ordering* of hypothesis fragments.

Like the fragmentation search, the core decode process is also implemented as a *local beam search*. The decoder is called with the initial hypothesis, and then computes all possible substitutions and all possible swaps, this results in a high number of new hypotheses. The k best hypotheses are selected, with the restriction that they must also be better than the current hypothesis. For each of these k hypotheses the procedure is again repeated, calculating all possible substitutions and swaps, until the point that no better hypotheses can be found. The algorithm can be formalised as in algorithm 10 below and is very similar to algorithm 9 for searching fragmentations:

Algorithm 10 Core-Decoder

Require: A set $X_{current}$ containing hypotheses, initially called with only the initial hypothesis as element. And a beam size k.

Ensure: The list containing the k best hypotheses found

1: $X_{next} \Leftarrow \emptyset$
2: **for all** $H_{current} \in X_{current}$ **do**
3: **for all** $H_{next} \in Substitutions(H_{current}) \cup Swaps(H_{current})$ **do**
4: **if** $Score(H_{next}) > Score(H_{current})$ **then**
5: $X_{next} \Leftarrow X_{next} \cup \{H_{current}\})$
6: **end if**
7: **end for**
8: **end for**
9: **if** $X_{next} == \emptyset$ **then**
10: **return** $X_{current}$
11: **else**
12: $X_{next} \Leftarrow$ Best k hypotheses in X_{next}
13: **return** $CoreDecoder(X_{next}, k)$
14: **end if**

It is to be noted that unlike the constraint satisfaction inference decoder discussed in chapter 2, we do not define a *delete* operation alongside substitution and swap, and do not make use of a null model. All fragments from the fragmentation must be used, each one issues a hypothesis fragment that will be used in the final solution. As we describe a phrase-based approach, dealing with null-aligned words becomes of less importance. The intuition is that null-aligned words can be captured in the extracted phrases themselves. For instance, a source phrase of five words may be translated to a target phrase of just two. Forcing the decoder to use all fragments simplifies the score function as we need not introduce length penalties/bonuses. The validity of this intuition also relies of course on the quality and quantity of the extracted phrases during instance generation.

The success of the decode algorithm depends on the score function it maximises. For each hypothesis, a score is computed that is an expression of the quality of the hypothesis. Recall that a good translation can be said to maximise the product of *faithfulness* and *fluency*. These two components are also present in this decoder. The notion of fluency is expressed by the language model, and faithfulness by the translation model. The score function for a hypothesis H and fragmentation F is thus:

$$Score(H) = TranslationModel(H) \cdot LanguageModel(H) \tag{3.3}$$

Note that here the hypothesis produces the eventual translation T, and the fragmentation F covers the source sentence. We will investigate now these two components in more detail.

3.4.3 Language Model

Even though the language model is an important component, we need not devote much attention to it here. It was already thoroughly presented in section 2.2.2, on Statistical Machine Translation. The language model employed in phrase-based memory based machine translation is not different.

The language model produces a likelihood score for the whole target sentence derived from the hypothesis. The language model employed is based on *trigram* counts. Assume a hypothesis H, and derived target sentence T_H, consisting of words $w_1...w_{|T_H|}$. Any $w_i < 1$ or $w_i > |T_H|$ will hold the dummy element to indicate respectively the beginning or end of the sentence. The unsmoothed language model can then be formalised as:

$$LanguageModel(H) = P(T_H) = \prod_{i=1}^{|T_H|+2} P(w_i|w_i{-}1) \cdot P(w_i - 1|w_i{-}2) \tag{3.4}$$

We use the SRILM toolkit Stolcke (2002) to compute a language model on the training data, and we use this same software to query this from the decoder. A limited python binding to the SRILM has been developed by Canisius & van den Bosch (2009) to facilitate this. Initially WOPR was used, which is a language model based on memory-based learning van den Bosch & Berck (2009), but this proved to be far too slow for our purposes.

It should be noted that in the current implementation each hypothesis is scored separately and independently. For each hypothesis all trigrams are generated and looked up. Further conservation of CPU time may be achieved by instead limiting lookups to only those parts that changed in respect to the hypothesis it is derived from.

A limitation of the language model implemented is that it can not deal with unseen words. Therefore it has to be created from the same training data as used in instance generation, or possibly a superset thereof. This guarantees that all target-language words that can possibly be predicted by the classifier, also exist in the language model. Of course the language model can be larger than the training set, which is to be recommended to prevent overfitting on the training-data.

3.4.4 Translation Model

The translation model is at the heart of the phrase-based decoder. It consists in part of a *translation probability* for the hypothesis as a whole. This is defined as the product of the *translation probabilities* for all of the selected hypothesis fragments. Recall that the translation probabilities are extracted from the classifier output. For each fragment in the fragmentation, one of its hypothesis fragments is selected with the associated probability that is the translation probability. If we take the product of all of these we obtain the *translation probability*.

$$Ptranslation(H) = \prod_{i=1}^{|H|} Ptranslation_i \tag{3.5}$$

Two special parameters have been implemented, the first is a *translation weight* which is a constant that simply provides extra weight to the translation probability, allowing for the balance between language model and translation model to be tweaked:

$$Ptranslation(H) = \prod_{i=1}^{|H|} (Ptranslation_i)^{weight} \tag{3.6}$$

The second is a *translation threshold* value which determines which of the hypothesis fragments from the full class distribution returned by TiMBL are considered. Hypothesis fragments with a translation probability below a certain threshold will not be tried. Note that this parameter is in direct relation to the beam-size value of TiMBL itself, so either one could be used to achieve the same effect, but they should better not both be configured. The translation threshold is expressed as a number between 0 and 1, where a low value would have a similar effect as a high beam size in TiMBL.

The translation probability is not all that makes up the translation model though, there is also a *distortion score*. This distortion score assigns a score of one to hypothesis fragments that are chained in the same consecutive order as the fragments they are derived from in the source fragmentation. It is a function of the distance between fragments. A lower score is assigned to sequences of hypothesis fragments that are *distorted* in respect to the original order in the source fragmentation. To calculate a distortion score, a small constant is raised to the power of the distance two consecutive hypothesis fragments have in the source fragmentation. This constant is the *distortion constant*, and is set to any number between zero and one. A high distortion constant provides more liberty in reordering fragments, whilst a smaller distortion constant prefers to adhere more to the order dictated by the source fragmentation. Note from the conditions in the following equation that there is an assymetry implemented, hypothesis fragments in inverse order are considered to be more distorted.

$$DistScore(hfragment_i, hfragment_{i-1}) = \begin{cases} distortionconstant^{|I_i - I_{i-1}|-1} & \textbf{if } I_i - I_{i-1} > 0 \\ distortionconstant^{|I_i - I_{i-1}|} & \textbf{if } I_i - I_{i-1} < 0 \end{cases}$$

$$DistScore(H) = \prod_{i=1}^{|H|} DistScore(hfragment_i, hfragment_{i-1}) \tag{3.7}$$

In the above equation, I_i refers to the index of the hypothesis fragment's parent fragment in the fragmentation. I_0 is always 0. If we take the example hypothesis "the book is on the table" we have indices $(1, 2, 3)$ and the distortion score for the whole hypothesis is 1. If we swap the first and last fragments, we obtain "on the table is the book", with indices $(3, 2, 1)$. The distortion score would then be, assuming a high distortion factor of 0.9:

$$DistScore(\{\text{ on the table, is, the book}\}) = 0.9^3 \cdot 0.9^1 \cdot 0.9^1 = 0.59049 \tag{3.8}$$

The complete translation model is made up of the product of the translation probability and distortion score of the given hypothesis:

$$TranslationModel(H) = Ptranslation(H) \cdot DistScore(H) \tag{3.9}$$

3.4.5 Transitions and overlap

A hypothesis is a chain of hypothesis fragments and we can speak of a *transition* between each hypothesis fragment and the next. Whenever a swap or substitution is made, these transitions change. One special property that each transition holds, is an *overlap* value. This value indicates how many words to the left of the current hypothesis fragment in the chain, overlap with the right word of the previous one. Note that we are here considering only the *focus* and are *not* talking about context words yet. If we have a hypothesis fragment *A B C* and another *B C D*, then there are two possible transitions:

1. A B C B C D
2. A B C D

In swaps and substitutions, it is always the case that all these possibilities are generated. There is a deliberate bias for shorter fragments as the language model is naturally inclined to assign a better score to fewer fragments. In the latter of the two possible transitions, the second hypothesis fragment will be assigned an overlap value of 2, in the first case the overlap value is simply 0, as will be the case for most transitions.

An extra transition model can be activated in the decoder. This is a small extra language model that is a relic of one of the earlier versions of the decoder. As it has been proved to result in poorer scores, it by default has been disabled. We will come back to an actual review of these results in section 4.7.

What the transition model does is look up a trigram using the language model, of this trigram the last word is the first focus word of the hypothesis fragment, and the first two words of the trigram are the two words preceding the hypothesis fragment, and thus pertaining to one or two previous hypotheses fragments. The transition model thus essentially gives *extra* weight to the transitions between hypotheses fragments. The intuition behind this was that stressing these might contribute to better transitions and better results.

If context information in the hypothesis fragments is enabled, then we could make use of overlap between context words in a similar fashion as we have seen in the other two memory-based machine translation decoders and as illustrated by the two overlapping and words word pairs in figure 2.11. We could use these overlapping words as *fertility words* on the target-side. When applying swaps or substitutions and such overlap in context is found, then a hypothesis will be generated with fertility words, and one without, leaving it up to the decoder to select the one with the highest score.

Chapter 4

Experiments & Results

Translation quality assessment proceeds according to the lordly, completely unexplained, whimsy of "It doesn't sound right".
– Peter Fawcett

The previous chapter presented the theoretical setup and the implementation of phrase-based memory-based machine translation. In this chapter this theory and the developed implementation will be put to the test. We will now turn to the practical setup of the thesis. A series of experiments has been conducted to investigate different aspects of phrase-based memory-based machine translation. All will be presented and discussed in this chapter. This will eventually enable us to answer the research question: "Does a phrase-based approach improve memory-based machine translation?", and moreover show how the various phrase-extraction methods and instance formats perform.

4.1 Data

Memory-based learning relies heavily on the use of corpora, as do all forms of example-based learning. Thus far we have not yet mentioned the actual corpora used in this and prior research. This is what will be addressed in this section.

All research in this thesis focusses on Dutch to English translation. The choice for this language pair is practical in nature; a sizeable amount of data (including word alignments, phrase tables) was available for this language pair from prior research (van den Bosch & Berck, 2009; van den Bosch et al., 2007; Canisius & van den Bosch, 2009). And the fact that prior research in memory-based machine translation focusses on the same language pair, offers an excellent ground for comparison of the various approaches. Having said that, it remains interesting to try other language combinations in future research. Dutch and English are after all both member of the Germanic branch of Indo-European languages, and as such share quite some common characteristics. The selection of a much more distantly related language pair may likely result in poorer performance.

Four corpora have been at our disposition, and we have mainly restricted research to two of them[1]:

[1] Both two corpora are public domain and can be freely obtained from `http://urd.let.rug.nl/tiedeman/OPUS/`

1. **OpenSubtitles** - This is a corpus of aligned subtitles for movies, and is one of the corpora included in the OPUS parallel corpus (Tiedemann & Nygaard, 2004). It consists mostly of relatively short sentences in informal language. The used training set consists of $286,160$ sentence pairs.

2. **EMEA** - This corpus consists of texts made available by the European Medicines Agency, and is also distributed as part of the OPUS parallel corpus Tiedemann & Nygaard (2004). It is medical and formal in nature and contains also long sentences. The training set consists of $871,180$ sentence pairs.

Both corpora consist out of one sentence per line, which have been tokenised. Note that no de-capitalisation has taken place, and all software developed also keeps capitalisation intact.

In any machine learning setting, it is vital to split data in at least two different sets, one for training and one for testing. If testing occurs on the set the models are trained on, the results do not reflect the generalization abilities of the system in classifying new previously unseen data. Both corpora thus have been split in a part for training, and a part for testing. For testing, one thousand sentences have been reserved. An extra thousand sentences have been reserved for development purposes such as parameter optimisation. Anything that is left over is used for training, as it is important to train on as much data as possible.

In addition to different sets for training and testing, we use also a portion of the development data of the OpenSubtitles corpus. This is a set of a mere 150 sentence pairs. This set will be used for testing the decoder and seeking optimal parameters.

4.1.1 Language Model

The Language Model used in all experiments is computed over various corpora, using the software SRILM Stolcke (2002). In addition to the training portions of the two aforementioned corpora, two extra corpora have also been included: the **EuroParl** corpus (Koehn, 2005), extracted from the proceedings of the European Parliament, and the **JRC-Acquis** corpus (Steinberger et al., 2006), legislative texts of European Union Law. Both corpora are substantially larger than the OpenSubtitles and EMEA. Including these in the Language Model reduces the possibility of over-fitting the language model on the training data.

4.1.2 Marker-based Chunking

Marker-based chunking has been presented in section 3.2.4. However, an extra source of data is necessary from which we can extract the markers to use in chunking. A list of markers is needed in both languages, Dutch and English. For English several frequency lists from Leech et al. (2001) were used, which are based on the British National Corpus. The markers were extracted from the frequency lists for various closed-class words: interjections, pronouns, determiners, prepositions and conjunctions. In addition to this, punctuation was also considered as a marker.

For Dutch, the "Referentiebestand Nederlands", part of the Cornetto Database (Vossen, 2006), was used as the source for extracting markers. Extraction of set of closed-class function words of types similar to above-mentioned types was performed. In total 366 Dutch markers were extracted, and 313 English markers.

4.2 Output

In order to get a first impression of the translation quality, table 4.1 lists the first twenty-five sentences of the OpenSubtitles corpus, along with the predicted translation by the Phrase-Based Memory-based Machine Translator in the middle column. The last column shows the original reference translation, which is in not available to the translator and only serves as a reference in judging the translation quality. The table furthermore nicely depicts the nature of the OpenSubtitles corpus, containing a lot of informal language and lack of strictness in translations. The latter in particular makes it a difficult corpus to work with in machine translation. On the other hand, an advantage of this corpus is the relatively short sentence length.

	Dutch Source Sentence	**Automatic English Translation**	**Original Reference Translation**
1	Het is een simpele ontvoering .	Simple- minded kidnapping .	It' s a straight- up snatch .
2	We hebben één of twee kerels nodig die de wagen onder handen nemen .	That' s one or two guys need the car looks very take .	We got one , two guys take down the car .
3	Waar hebben we het dan over ?	What are we talking ?	Then what are we talking about ?
4	Een Conex container ?	A a container .	A Conex container ?
5	Eén kerel voor een dag ?	One guy for a day ?	One guy for a day ?
6	Wie zoekt er een hoer ?	who looking a ?	Who' s looking for a whore ?
7	Dit karwei is een weggevertje .	This squad s a you .	This job' s a gimme .
8	Het stelt niets voor .	It' s nothing .	This job' s nothing .
9	Waarom halen jullie haar dan niet ?	Why do you her t ?	Then why don' t your people go and get her ?
10	Ik zou iedereen kunnen huren .	I , everyone can rent .	- I could hire anybody .
11	- Doe het dan .	- Good .	- So go hire them .
12	Jij kwam naar mij .	You came to me .	You talked to me .
13	Dat is mijn prijs .	That' s my price .	That' s my price .
14	Er is een alternatief .	There' s an alternative .	- There' s another alternative .
15	- Wat ?	- What ?	- What ?
16	Zoals ik zei , we kennen dit huis .	Like I said , know this house .	As I said , we know this house .
17	Er zijn bepaalde mensen in dit huis ...	There are certain people in this house .	- There are some people in this house ...
18	- Avi die , als je daar toch bent kunt bezoeken ... en hallo van mij kunt zeggen , dan zou ik wat aan de prijs kunnen doen .	- And that , when you get there re can visit and say hello mine me can say , I got some the prize could do .	- Avi . ... that if you were to go while you were there ... ... if you said hello to them for me , I could cut your price .
19	Wat denk je wel niet dat ik ben ?	Where do you come think I am ?	- What do you think I am ?
20	- Ik weet niet wat je bent .	- I don' t know what you are .	- I don' t know what you are .
21	Je bent geen planner .	You' re not worry .	You ain' t a planner .
22	Je bent een schutter .	You' re a shot .	You' re a shooter .
23	Ik weet niet wat je hier doet .	I don' t know what here doing .	I don' t know what you are doing here .
24	Je hoort hier niet thuis .	You don´t belong here .	I think you' re off the reservation .
25	Er zijn mensen naar je op zoek .	There are people looking for you .	And I think there are people looking for you .

Table 4.1: Translation output for the first 25 sentences in the OpenSubtitles test set. (phrase-table extraction, split-file format, beam size 5, details on these become clear later in this chapter)

It is obvious that it is not feasible to study the output of a thousand sentences in such a manual way, let alone compare them to the thousands of sentences generated by the system in a different configuration. We therefore require an objective measure to give an indication of the quality, and to use this to compare results. Several metrics that have been proposed as approximate objective quality measurements will be presented in the next section.

4.3 Evaluation Metrics

To what extent a translation is a good translation of the original input is a very hard question to answer. In the introduction we already stressed that no two translations of a text are likely to be the same. Arguably the best way of evaluating the performance of machine translation systems would be to use *human judges*, who can make an assessment of the translation quality on one or more predefined scales, or who select which of the translations produced by different systems they regard as best. This remains inherently subjective, and it is a time-consuming and costly task. We therefore seek to evaluate translation quality through automatic means, which has the main advantage of being quick and consistent. It may not prove as reliable an evaluation as human judges would, but these all of these metrics try to obtain an as high as possible correlation with human judges. This correlation however is limited and the metrics are subject to criticism, which is why we shall rely on a set of metrics rather than on any single one of them. In this fashion, they prove an adequate solution considering practical constraints.

There are various metrics which aim to assess translation quality by assigning a certain score. The idea underlying these metrics is that they relate the translations of the machine translator to a reference set. For each corpus we have a test set of one thousand sentences in the source language, and we have a corresponding reference set of the translations thereof. These translations are in no way used by the system itself, but are only used in the evaluation stage. All metrics are based on a computation of similarity to the reference sentence, the more similar, the higher the score. We will use the following metrics in the evaluation of phrase-based memory-based machine translation:

1. **BLEU** - The BLEU metric Papineni et al. (2001) computes a weighted average of the number of n-gram overlaps between translation and reference. Simply put, the more n-grams in the translation also occur in the reference, the higher the score. A modified n-gram precision metric is used to prevent the algorithm from simply favouring repeated words. The higher the score, the more similar the translation is to the reference. BLEU is however regularly subject to criticism, and other metrics have been devised to improve upon this.

2. **NIST** - NIST is based on BLEU, but it takes into account how informative a particular n-gram is. Rare n-grams will receive more weight and common n-grams will receive less. The higher the score, the more similar the translation is to the reference.

3. **METEOR** - The METEOR metric Banerjee & Lavie (2005) attempts to outperform BLEU and obtain a higher correlation with the predictions by human judges. As in all metrics, METEOR generates a word-alignment between the translation and the reference. We use the "exact" module which maps two words when they are exactly the same. It then computes a harmonic mean of the precision and recall of unigrams. The higher the score, the more similar the translation is to the reference.

4. **Word-Error Rate (WER)** - This is a very simple metric derived from the minimum edit-distance algorithm, counting the number of substitutions, insertions and deletions necessary to transform the translation into the reference sentence. Here words are used as the basic unit. The lower the score, the more similar the translation is to the reference.

5. **Position Independent Word Error Rate (PER)** - This is the same metric as the above, but here the order of the words is not taken into account. Any ordering of the same words will have the same score. The lower the score, the more similar the translation is to the reference.

A more in-depth discussion on the actual computation of each of these metrics, and a comparison thereof, is beyond the scope of this thesis. We will only be interested in the comparison of the scores that come out of the experiments laid out in this chapter.

A special note has to be made regarding the relative nature of these metrics. The scores can generally only be meaningfully compared when applied to the same test test or the same system. That is also why the choice of corpora for this research depends mainly on the corpora used in prior research.

4.4 Compounding errors

A memory-based machine translation system consists of a chain of various components, as has been in illustrated in the setup scheme of figure 3.4. Each of these may introduce errors or inaccuracies, as none of the stages guarantee a 100% accuracy and none of the underlying models is a perfect solution. Machine translation is a hard problem in which we deal with models that attempt to divide this huge task into subtasks that we hope may get the job done to a certain extent. Since the outcome of one stage acts as the input of the next, we can speak of a compounding of errors from various levels. If we at the end of this chain receive a poor result, it is not apparent which component is to blame for the inaccuracy.

For example, in the word-alignment phase, there may be awkward alignments. Since the instance generation and phrase extraction are based on this, such "errors" will be carried over to these phases. The phrase-extraction itself may introduce new problems as well, so we can and eventually will end up with some incorrect instances that are used in training. Machine learning in turn also performs only with a certain accuracy, and there will always be wrong classifications. Finally, the decoder seeks to tie all hypothesis fragments together, but it relies on the quality of these fragments, and it is not at all guaranteed to find the composition and order that a human would judge best.

This compounding of errors thus stems from the enormous difficulty in translation, apparent in the fact that all devised solutions for the sub-tasks are models and limited approximations.

4.5 Parameter optimisation

Both TiMBL as well as the decoder developed in this thesis may take quite a variety of parameters that tweaks their behaviour. Finding an optimal combination of parameters is a hard search problem, as parameters are often highly interdependent. It is not possible to a-priori compute the best values for the parameters; they have to be found in an incremental search process that continually examines the outcome of the experiments.

For memory-based learning, we have consistently used the *TRIBL2* algorithm with $k = 1$, a beam size of 1, and with distribution output enabled. No further parameter optimisation has been performed for TiMBL, because this would increase the search space considerably and take substantially more time for processing and analysis. Possible future research might find potential for improving the results here, but for the comparison of approaches in this thesis such further optimisation is less relevant.

Since the decoder is new and specifically developed for this thesis, some parameter tests have been conducted using a small development set of the OpenSubtitles corpus, consisting of 150

phrase-pairs. The small size of this development set is due to wanting to limit CPU time necessary in this experiment and quickly obtain estimations, at the expense of being possibly somewhat less representative. Different value combinations have been tested[2] and compared for several parameters that have been introduced in section 3.4:

- The **beam size** of the core decoder - A larger beam size searches through more of the search space and is less likely to get caught in local maxima.
- The **distortion constant** - Determines the flexibility or rigidity in changing word order
- The **translation weight** - An exponential weight applied to the translation probabilities.
- The **maximum swap distance** - Defines the maximum distance over which a swap operation may be applied.

More variables could have been included, but an attempt was made to select the ones expected to have most impact on the end-result. Searching through more would increase the magnitude of the search problem. The decoder implementation technically allows more tweaking, but other less significant variables in the implementation were left at fixed arbitrary values. One notable fixed value that should be mentioned is the value for the *translation threshold*, as discussed briefly in section 3.4.4. We let TiMBL return the *full* distribution of classed and set the translation threshold of the decoder to 1.0, this is functionally equivalent to setting a TiMBL beam size of 1 and translation threshold of 0. The advantage of having the translation threshold set in the decoder rather than having a beam size configured in TiMBL is that in this way multiple experiments with different translation thresholds can be run using the same TiMBL output.

Experiments were also conducted with the core-decoder essentially disabled, i.e. simply returning the initial hypotheses without further decoding, no substitutions nor swaps. Needless to say, this speeds up the decoding process considerably, at the expense of poorer results. Table 4.2 presents the results of this first experiment:

	BLEU	NIST	METEOR	WER	PER
Enabled	**0.1928**	**3.4226**	**0.4499643**	**56.652**	**53.2159**
Disabled	0.1782	3.0024	0.4267648	59.5595	56.2996

Table 4.2: Performance with core decoder, and thus swaps and substitutions, enabled versus disabled (all other variables are fixed)

These scores unequivocally demonstrate that the decoding strategy of making substitutions and swaps to an initial hypotheses indeed produces better results. Nevertheless, the differences in the scores, although clear, are fairly modest. The other parameter optimisation experiments however show smaller differences in scores. Besides the enabling versus disabling of the core-decoder, overall tweaking of decoding parameters did not have a major impact on the various metrics.

Before we go into the discussion of the four to be optimised parameters, another small experiment will be presented first. Recall the discussion at the end of section 3.4.1 on the selection of the final translation, by selecting the translation for which the sum of the various hypotheses that describe it was maximised. This was a more sophisticated method that could be put in

[2] All parameter tests described in this section have been performed on the development set using the phrase list phrase-extraction method (with treshold 5) and split-file instance format

contrast to simply selecting the single highest ranking hypothesis amongst all. Table 4.3 shows empirically that this more sophisticated method, the standard method, indeed outperforms the simpler method:

	BLEU	NIST	METEOR	WER	PER
Simpler method	0.192	3.4136	0.4486506	56.7401	53.304
Standard method	**0.1928**	**3.4226**	**0.4499643**	**56.652**	**53.2159**

Table 4.3: Simple solution search method versus the standard disjunctive solution search method

The first of the actual parameter optimisation experiments is the changing of the distortion constant. Table 4.4 below shows the results for different distortion constants, from a very rigid setting allowing little reordering to take place, to a free setting of 1.0 in which all re-orderings go without penalty:

Distortion constant	BLEU	NIST	METEOR	WER	PER
0.00001	0.1837	3.3824	0.4459709	57.2687	53.304
0.01	0.1831	3.3632	0.4422661	57.3568	53.6564
0.1	**0.1924**	3.4137	0.447789	**56.4758**	53.304
0.15	0.1908	3.4119	0.4475151	56.5639	53.304
0.25	0.1909	3.4068	0.4476111	56.652	53.304
0.3	0.1915	3.4118	0.447885	56.5639	53.304
0.4	0.189	3.4169	0.4491424	56.5639	53.1278
0.5	0.1865	3.3991	0.4473788	57.0044	53.1278
0.6	0.1828	**3.429**	0.4505773	56.652	52.8634
0.7	0.1799	3.4215	0.4509476	56.7401	52.8634
0.75	0.1784	3.4277	**0.4511663**	56.7401	52.7753
0.8	0.1768	3.4007	0.4502236	56.652	52.7753
0.85	0.1745	3.3935	0.4500909	56.9163	**52.6872**
0.9	0.1733	3.3783	0.447916	57.2687	52.8634
0.95	0.1749	3.3899	0.4482342	57.4449	52.7753
1.0	0.1783	3.4046	0.4477861	57.4449	52.7753

Table 4.4: Scores for different distortion constants (all other variables are fixed)

Observe that different metrics peak at different points. The BLEU score tops at more restrained distortion constants, so if we were to base ourselves on that we could say a more restrained distortion constant proves to lead to better results than providing more freedom in reordering. However, NIST and METEOR score better slightly after the middle ($0.6 - -0.75$), whilst BLEU clearly performs worse there. The margins for most of the metrics are small. Two metrics that are of particular interest in this comparison are WER and PER. PER is a metric that does not take into account word order, whilst WER does take into account ordering. We see that WER performs worse at the very low-end and very high-end of the values. The best values for the distortion constant thus seem to be somewhere in the middle, in the range $0.1 - -0.8$, although 0.5 does stand out as a worse pick. There is no clear single winner emerging from this picture as the margins within this range remain quite small and different metrics point at different ends of this range and it becomes a matter of how much weight we assign to them. It is very hard to speculate about possible causes for these inconclusive results. In the actual test experiments the distortion constant is left at the arbitrary default of 0.25, well inside the aforementioned range, and which has one of the highest BLEU scores and is shown not to

under-perform by any large margin.

The above experiment was conducted with the beam size of the core decoder set to one, effectively reducing local beam search to hill-climbing. An experiment has also been conducted with different beam sizes. Table 4.5 shows the result of this experiment:

	BLEU	NIST	METEOR	WER	PER
1	0.1909	3.4068	0.4476111	56.652	53.304
2	0.1912	3.4092	0.4481591	**56.652**	53.3921
5	**0.1928**	**3.4226**	**0.4499643**	56.652	**53.2159**
7	0.1926	3.4101	0.4484333	56.8282	53.3921
10	0.192	3.4136	0.4486506	56.7401	53.304
25	0.1904	3.4047	0.4486506	56.8282	53.3921

Table 4.5: Scores for different beam sizes (all other variables are fixed)

Though only a few values were tried, five beams emerges as a good pick, albeit by very close margins. Hence, this is one of the beam sizes selected for the test experiments in later sections. A higher beam size does come with the disadvantage that it slows the search. In cases where computation time is limited, we will opt for simple hill-climbing search.

An idea implemented in the decoder, and discussed in section 3.4, was to give extra weight to the translation model by raising the translation probabilities to a certain power, the so-called translation weight. Table 4.6 shows that only a very minimal gain can be expected from this strategy. A translation weight of three was selected for the test experiments, since it has the highest NIST score and second-highest METEOR and BLEU scores.

Translation weight	BLEU	NIST	METEOR	WER	PER
1	0.1882	3.3688	0.4471329	56.7401	53.3921
2	0.189	3.3939	**0.447885**	56.7401	53.3921
3	0.1909	**3.4068**	0.4476111	56.652	**53.304**
5	**0.1911**	3.4062	0.4467497	**56.652**	53.3921
7	0.1848	3.3502	0.4417731	57.1806	53.7445

Table 4.6: Scores for various translation weights (all other variables are fixed)

The last parameter selected for optimisation is the maximum swap distance, discussed in section 3.4. This however produced an interesting result: increasing the maximum swap distance did not affect the score whatsoever. This would imply that either swaps over a larger distance were almost never favoured in the dataset, due to perhaps a too restrictive distortion constant, or that swaps over larger distances were performed using a series of swaps over shorter distance. The experiment was performed with a distortion constant of 0.25, but recall that the parameter optimisation experiment for the distortion constant (table 4.4) sketched a very ambiguous and inconclusive picture as well. Both parameters are of course intimately related, as both come into play only when the swap operation is applied. Investigating this issue, possibly attaining further optimisation, and improving the swap operation would be something to recommend for future research.

4.6 Context in classes

Throughout the discussion in chapters two and three, we have occasionally seen the usage of left- and right-context in the target class, whereas on other occasions we omitted context. Most notably all prior research on word-based methods included left- and right-context in the target class. Overlap between the context of classes was an important criterion for the decoder to re-assemble hypothesis fragments.

For the phrase-based approach it however was necessary to reconsider the usage of context. The reason for this is the *sparsity* of classes. Recall that a class is seen as a single entity; the machine learning algorithm is oblivious to the fact that it is made up out of context components and a focus phrase. The more words a class contains, the less likely it is to occur. The classes thus become *sparser* and many classes will occur only once. The problem with this sparsity is that the predicted output classes will be more ambiguous, as the machine learner has less grounds to disambiguate upon. There will thus be a larger amount of predicted classes of equal or similar weight.

Using phrases instead of words tends to naturally introduce more sparsity, as the classes simply get longer. If we were still to employ context in addition to using phrases, then this sparsity would be even greater. Therefore the idea rose to omit context in the target class, in an attempt to limit sparsity. This decision is furthermore justified by the decreased importance of context overlap in a phrase-based approach. Since phrases consist of multiple words, we may benefit from the overlap *in the phrases themselves*, something which is not possible in a word-based approach.

The decision to omit context in classes is empirically backed up by the comparison shown in table 4.7, showing very significant gains when performing an experiment omitting context:

	BLEU	NIST	METEOR	WER	PER
Without context (0X0)	**0.2184**	**4.9748**	**0.4529026**	**54.0885**	**48.7859**
With context 1 (1X1)	0.1211	3.3015	0.3493939	65.6648	61.5435

Table 4.7: A comparison of usage of context in classes, tested on the OpenSubtitles corpus (phrase list 25 extr., split-files format)

4.7 Transition Model

.

Before going in to the main results, some words are reserved for the discussion of the extra transition model mentioned in sections 3.4.5. Prior versions of the decoder employed a more extensive transition model, in which transitions between fragments were scored extra using the language model. Initially the decoder relied on these transitions to compute each time the *next most probable* hypothesis fragment, but now it instead constructs an initial hypothesis. The extra scoring of transition can however still be enabled, doing so however produces poorer scores, as shown in table 4.8:

	BLEU	NIST	METEOR	WER	PER
With extra transition scoring	0.173	3.3588	0.4438064	58.5022	53.3921
Without extra transition scoring	**0.1909**	**3.4068**	**0.4476111**	**56.652**	**53.304**

Table 4.8: Table illustrating the negative impact of the extra language model scoring transitions (tested on the OpenSubtitles development data, phrase list (25), split-file format)

4.8 Main Results

In this section the main results will be presented and discussed. These attempt to compare three different aspects:

1. **Phrase-extraction methods** - Three different phrase-extraction methods have been proposed and implemented: extraction using a phrase-translation table, extraction using a phrase list, and marker base chunking. How do these methods perform in comparison to each other, and which performs best?
2. **Instance formats** - We discussed three different ways for the representation of training- and test instances. The first format stored phrases as a single feature, the second reserved a fixed number of features, and the third split instances of different phrase-length into multiple files. How does the performance for the different formats relate to one another?
3. **Comparison to word-based systems** - How does the performance of the phrase-based approaches compare to that of pure word-based approaches?

Table 4.9 shows the outcome of all experiments on the OpenSubtitles corpus, table 4.10 does the same for the EMEA corpus. All experiments were run using a beam size of one (hill-climbing), a distortion constant of 0.25, a translation weight of 3, and a maximum swap distance of 5.

Decoder	Extract. Meth.	Instance F.	BLEU	NIST	METEOR	WER	PER
MBMT	-	-	0.148	3.817	0.3835	67.76	62.03
CSIMT	-	-	0.2002	4.750	0.4431	68.42	55.18
PBMBMT	-[3]	-	0.2163	**5.136**	**0.4644**	55.23	**48.22**
PBMBMT	phrase table	split-files	0.2256	5.004	0.4583	55.28	49.74
PBMBMT	phrase table	fixed-feat.	**0.2300**	5.055	0.4623	54.47	49.18
PBMBMT	phrase table	single-feat.	0.1142	3.026	0.3201	72.81	68.28
PBMBMT	phrase list (5)	split-files	0.2152	4.798	0.4445	54.57	49.89
PBMBMT	phrase list (25)	split-files	0.2184	4.975	0.4529	**54.09**	48.79
PBMBMT	phrase list (25)	fixed-feat.	0.2190	4.980	0.4543	**54.09**	48.77
PBMBMT	marker-based	split-files	0.1394	3.360	0.3437	66.40	62.38
PBMBMT	marker-based	fixed-feat.	0.1003	2.935	0.3057	76.79	71.16

Table 4.9: Main results on the **OpenSubtitles** corpus, Dutch to English

[3] No phrase-extraction method is selected, so the decoder will only receive word-based data and no phrases whatsoever

Decoder	Extract. Meth.	Instance F.	BLEU	NIST	METEOR	WER	PER
CSIMT	-	-	0.3013	5.938	0.5333	63.00	**50.85**
PBMBMT	-[3]	-	0.2715	5.600	0.5381	65.99	57.25
PBMBMT	phrase table	split-files	**0.3078**	**6.019**	0.5449	**58.76**	51.63
PBMBMT	phrase table	fixed-feat.	0.3075	6.011	**0.5455**	59.00	52.02
PBMBMT	phrase list (25)	split-files	0.2440	5.378	0.4967	62.82	56.86
PBMBMT	phrase list (25)	fixed-feat	0.2440	5.352	0.4946	62.74	56.67
PBMBMT	marker-based	split-files	0.2370	4.612	0.4513	74.37	66.78

Table 4.10: Main results on the **EMEA** corpus, Dutch to English

The tables start with the presentation of the *word-based* results of the decoders developed in prior research van den Bosch & Berck (2009); Canisius & van den Bosch (2009). Experiments with the classic MBMT decoder and constraint satisfaction inference decoder (CSIMT) were fully reproduced, rather than copied, as it was necessary to be certain they ran on exactly the same data, trained with exactly the same parameters as used in the remainder of the experiments.

Note that though the phrase-based decoder (PBMBMT) developed in this thesis is specifically designed for dealing with phrases, it can also be run using only word-based input, as our definition of phrases explicitly allowed phrases of length 1. This was deemed necessary to bridge unavoidable gaps in phrase coverage. The phrase-based decoder thus has no problems in dealing with with word-based instances. It is therefore also possible to conceive an experiment that does *not* use any of the three phrase-extraction methods, but instead generates only word-based instances in the very same fashion as done in prior research (2.4), and pass the results after classification to the PBMBMT decoder for final processing. In this case, the decoder will only have to decode for *exactly one* fragmentation. This is one of the experiments conducted and presented in the first section of tables 4.9 and 4.10, and this offers an excellent baseline in comparing the effect of phrase-based versus word-based, using the same decoder.

The remaining sections of the tables show the three extraction methods and three instance formats. For the phrase list extraction method, two different thresholds were tried, using a phrase list including n-grams occuring at least 5 respectively 25 times, respectively. Observe that the instance format in which the phrase is taken as a single-feature, is only tested once, as the results were quite conclusive already.

Now looking at the scores, how does the phrase-based approach compare to word-based approaches? We have three references for comparison here; the word-based results produced by the MBMT decoder, the word-based results produced by the CSIMT decoder, and the word-based results produced by the phrase-based decoder itself. An interesting observation can be made when looking at the results for the OpenSubtitles corpus; the new phrase-based decoder performs out better than the MBMT and CSIMT decoders, even without using phrases. However, this observation does not hold when we look at the EMEA corpus, where CSIMT maintains an edge. The most probable cause for the relatively high performance of the phrase-based decoder is in the absence of context in the class, and thus resulting in less sparsity compared to MBMT and CSIMT, which always rely on context. In table 4.7 we saw that the inclusion of context degraded performance significantly. The EMEA corpus has longer and more formal sentences, and is more repetitive in nature. It is conceivable that here the increase in sparsity is more limited and that context in classes does offer an overall advantage in decoding, giving CSIMT the edge.

The word-based results of the PBMBMT decoder can be used as a baseline for comparing the impact of the phrase-based approach compared to the word-based approach. If we take a look at the three phrase-extraction methods, then the clear winner is the phrase-translation-table method. The phrase-translation-table method scores slightly above the word-based baseline for both corpora, though not conclusively for all metrics. Both other methods perform worse; especially the marker-based chunking method scores notably poor on the OpenSubtitles corpus, far worse than any of the other methods and below baseline. The phrase-list approach also scores below baseline, only slightly for the OpenSubtitles corpus, but more significantly for the EMEA corpus.

It is surprising to see that the phrase-based approaches on the whole do not seem to offer a very significant gain over word-based approaches. In fact, depending on the metric used there may even be no gain at all. For OpenSubtitles, running the decoder without phrases even results in higher Meteor and NIST scores. Two extraction methods even show a clear loss compared to the word-based baseline. In fact only the phrase-translation-table approach was able to make a positive impact, and more clearly so for EMEA. What explanation is there to offer for this? Again the sparsity of phrases may play a role here, which led to too many ambiguous predictions made by the classifier, especially in the marker-based chunking approach. Another problem is the fact that there are relatively few phrases extracted for phrase list extraction and marker-based chunking, especially compared to the word-based approach. Table 4.11 illustrates the number of phrases extracted for training, split over various phrase-lengths n, and using the OpenSubtitles corpus. For the latter two methods a sharper decrease can be observed:

n	phrase table	phrase list (25)	marker-based chunking
1	1847887	1847887	2047875
2	1213597	619311	131886
3	845796	109338	118021
4	595531	13200	64915
5	420294	1357	31777
6	283107	326	13966
7	-	-	5779
8	-	-	2036
9	-	-	720
Total	5206212	2591419	2416975

Table 4.11: Overview of the number of phrases extracted for different phrase-lengths (determined using the split-file approach on the OpenSubtitles corpus)

The phrase-table approach succeeds in more successfully extracting longer phrases for training, whereas the other approaches show a steeper drop-off. This is one of several factors that may explain the relative success of the phrase-table approach, and the weaker performance of the other two. A more major factor is the fact that the phrase-table approach is more solidly anchored in both source and target language. For the phrase-table approach, the phrase aligned to the extracted phrase, is dictated by the phrase-translation table, which is in turn computed statistically over the whole training set. In the phrase list and marker-based chunking approach, the aligned counterpart is searched for in a more ad-hoc and dynamic fashion, on a per sentence basis rather than holisticly considering the whole corpus. This for the latter two extraction methods results in overall weaker ties between the extracted phrase-pairs. This very same issue is also the most likely explanation for the fact that the phrase list with threshold five, thus containing more phrases, is outperformed by the phrase list with a stricter threshold (25).

In explaining the relatively small gain a phrase-based approach seems to offer over a word-based approach, another important factor that has to be taken into consideration is the fragmentation search in the decoder (see algorithm 9). Even if there are quite a number of phrases, and even though the fragmentation search has a natural bias to select the fewest number of fragments, fragments with phrases will simply not be selected if their highest translation probability pales in comparison to that of fragments of mere single words.

When comparing the results of the three instance formats, the most apparent observation is the very poor performance of the format in which the phrase is taken as a single feature, and all are stored combined in a single file. This is however not unexpected, as this issue was already discussed in section 3.2.1. A notable problem of this format is that the context features get too much weight, whilst the focus itself loses weight. We end up with a truly enormous amount of classifications per instance, and thus a huge amount of ambiguity left for the decoder to resolve. The file size of the classified file in this format is 21 times larger than the combined file sizes of the files in the split-file format! The poor performance of this instance format is the reason for having listed only one experiment with it, which is more than enough to discard it.

The other two instance formats prove to be fairly equally matched. It is not really possible to proclaim one better than the other, as the scores are inconclusive. For the OpenSubtitles corpus, the phrase-table-based extraction results have been boosted a little by using a fixed predetermined number of features instead of splitting different phrase-lengths into different files. Yet, for the EMEA corpus the situation is reversed. For marker-based chunking, the comparison was only performed on OpenSubtitles and the split-file format clearly has the upper hand. For the other methods margins are simply too small to call.

All of the above results have been attained with the decoder using a beam size of 1, and thus effectively performing hill-climbing. In the section on parameter optimisation it became clear that a beam size of five might slightly boost results. Table 4.12 shows this is indeed the case for the OpenSubtitles corpus, for the EMEA corpus results are subtle and do not show any significant improvement (compare with tables 4.9 and 4.10).

Corpus	Extract. Meth.	Instance F.	BLEU	NIST	METEOR	WER	PER
OpenSub	phrase table	split-files	0.2274	5.010	0.4587	55.19	49.70
OpenSub	phrase table	fixed-feat.	**0.2314**	**5.061**	**0.4600**	54.44	49.17
OpenSub	phrase list (25)	split-files	0.2188	4.987	0.4536	**54.05**	**48.68**
OpenSub	phrase list (25)	fixed-feat.	0.2197	4.983	0.4547	54.06	48.72
EMEA	phrase table	split-files	0.3053	6.007	0.5438	**58.96**	**51.77**
EMEA	phrase table	fixed-feat.	**0.3084**	**6.015**	**0.5460**	59.03	52.02
EMEA	phrase list (25)	split-files	0.2419	5.368	0.4964	63.14	56.83
EMEA	phrase list (25)	fixed-feat.	0.2435	5.356	0.4952	62.76	56.60

Table 4.12: Boosted results using beam size 5 instead of 1

Note at this point that table 4.1 in the beginning of this chapter, which listed the first 25 sentences, was generated using a phrase-translation table, using the split-files instance format, and with beam size 5. It as such thus corresponds to the first line of scores in table 4.12.

How does phrase-based memory-based machine translation compare to the state-of-the-art in statistical and hybrid machine translation? Though an elaborate comparison with non-memory-based learning system is not part of the research question of this thesis, it may nevertheless prove insightful to show a brief comparison. Table 4.13 compares PBMBMT with three other

systems. The first two are founded upon statistical machine translation principles, and Systran is a largely rule-based system. It is clear that the performance of phrase-based memory-based translation does not come close to approaching the quality of state-of-the-art statistical machine translators, nor was this something that was to be expected. We however do notice that PBMBMT outperforms Systran on all metrics. To represent the phrase-based memory-based machine translation system, we use the best results of table 4.12:

No	Corpus	System	BLEU	NIST	METEOR	WER	PER
1	OpenSub	Moses	**0.3289**	**5.903**	**0.5408**	53.29	46.96
2	OpenSub	Google	0.3056	5.790	0.5224	**50.1**	**45.08**
3	OpenSub	PBMBMT	0.2314	5.061	0.4630	54.44	49.17
4	OpenSub	Systran	0.1749	4.583	0.4500	60.77	54.61
1	EMEA	Moses	**0.4701**	**7.059**	**0.6501**	**46.55**	**39.36**
2	EMEA	Google	0.3918	6.377	0.5830	57.57	50.44
3	EMEA	PBMBMT	0.3084	6.015	0.5460	59.03	52.02
4	EMEA	Systran	0.2895	5.472	0.5366	63.24	55.14

Table 4.13: A comparison with state-of-the-art systems

4.9 Computational considerations

The phrase-based decoder developed in this thesis has the advantage of being quite fast. It is so despite the fact that the core decoder has to be called for multiple fragmentations instead of just one, making it an inherently more complex task than a word-based approach. The reason for this speed advantage is to be sought in the fact that the substitution and swap operations are fairly simple, and in practise only a few decode rounds are needed to maximise the score function.

The bulk of processing time goes into memory-based training and subsequent memory-based learning. A comparison of the processing time for the various methods can be seen in table 4.14, run over the full test-set of 1000 sentences of OpenSubtitles corpus. The times for processes prior to training was not measured, this includes tokenisation, word-alignment, phrase-table or phrase-list generation and instance generation. Especially word-alignment and phrase-table generation are computationally expensive processes that take considerable time. Memory consumption was not explicitly measured either, but due to the very nature of the machine learning method, a substantial amount of memory is required during training and testing. The memory usage of the instance generator is predominantly determined by the size of the phrase table or phrase list, which is loaded into memory as a whole. The memory usage of the decoder is limited, and depends chiefly on the language model which has to be loaded into memory.

Extract. Meth.	Instance F.	**Training**	**Testing**	**Decoding**	**Total**
phrase table	split-files	*0:34:31*	*0:03:36*	*0:00:08 (0:00:31)*	*0:38:07*
phrase table	fixed-feat.	*0:56:54*	*0:03:40*	*0:00:08 (0:00:33)*	*1:00:44*
phrase table	single-feat.	*1:16:30*	*0:15:48*	*0:00:43 (0:01:24)*	*1:34:01*
phrase list (25)	split-files	*0:18:22*	*0:02:16*	*0:00:10 (0:00:40)*	*0:20:48*
phrase list (25)	fixed-feat.	*0:20:06*	*0:01:40*	*0:00:11 (0:00:42)*	*0:21:57*
marker-based	split-files	*0:23:14*	*0:02:29*	*0:00:08 (0:00:25)*	*0:25:51*

Table 4.14: A comparison in processing time on a thousand sentences from the OpenSubtitles corpus (using AMD Opteron 880, 2412Mhz)

In table 4.14, the decoder was run with beam size 1, and later also beam size 5, the duration for the latter is shown in parenthesis but not included in the final sum. Note that in testing, some extra time can be saved by invoking TiMBL to train and test in one run. However training and testing separately has a big advantage, as training and all prior steps need to be performed only once, whereas testing can be performed repeatedly on different data. In real-life applications such as for example on the web, it is only the combined time of testing and decoding that is most relevant.

It can be observed that taking the phrase as a single-feature, not only results in the worst result, but is also computationally the most expensive. This is however not surprising, since we already ascertained that this method saw a huge increase in the file size of the classified output file. A longer training time can also be observed for the fixed-feature method, but only for the phrase-table approach, the difference for the phrase-list approach is negligible in comparison.

Chapter 5

Conclusions & Future Research

This thesis focussed on *phrase-based* memory-based machine translation, the emphasis being on the phrase-based character as opposed to memory-based systems that consider words. The main research question was formulated: ***Does a phrase-based approach improve memory-based machine translation?*** In answering that, the question *how to extract phrases* played a pivotal role and resulted in three proposed methods of phrase-extraction for instance generation: using a phrase-translation table, using a phrase list, and through marker-based chunking.

All three methods have been implemented and compared. Phrase extraction using a phrase-translation table emerged unmistakably as the best method. This in itself need not be surprising, as it is after all also the method chosen by state-of-the-art statistical machine translators, and the other two approaches are notably less complex. This nevertheless implies that meaningful phrase-based memory-based machine translation can only be performed by resorting to the phrase-extraction tools provided by statistical machine translation toolkits such as MOSES. This dependency can be considered a disadvantage, as extra software is needed and it takes considerably more time to compute a phrase-translation table than the other two extraction methods. The other two methods are easier to compute, but scored worse and even below baseline, which indicates that these strategies do not contribute to a higher quality of memory-based machine translation.

It has also been shown that the omission of context in classes produces significantly better results in a phrase-based approach. This was credited to the decrease in sparsity and thus ambiguity. Moreover, this was justified by the fact that context becomes less relevant in phrase-based approaches, as phrases capture their own context, and can overlap with other phrases.

Nevertheless, the impact of phrase-based over word-based has been shown to be limited, in spite of the mentioned advantages (section 3.1.1) that phrases generally offer over words. Again it can be argued that sparsity plays a role here; phrases are by definition less prevalent than single words, i.e. they have a lower probability of occurrence, complicating the classification process. The omission of context in classes attempts to compensate for this to a certain extent.

A clear difference between the word-based character in memory-based machine translation, and that in for example statistical machine translation, is the use of source-side context, which is a main characteristic of the memory-based form of machine translation. This research and prior research have consistently used a context feature to the left and to the right of the word or phrase. So despite the word-based character, there already was a *significant role for context*. The main advantage of a phrase-based approach is also to be sought in context, but here we

speak of *context implicit in the phrase itself*. The limited gain may stem from the limited added value this implicit context provides over the explicit context features already inherent in memory-based machine translation.

A third factor in explaining the limited gain is the fact that the phrase-based method still makes extensive used of word-based instances, and it is up to the fragmentation search in the decoder to favour one over the other based on their translation probability. Because of ambiguity caused by sparsity, it is conceivable that the decoder favours words rather than phrases. So in practise a lot of fragments are still word-based. Including word-based support was necessary as there would otherwise be far too many gaps in the coverage of fragments on the input sentence, we thus simply considered words phrases of length one.

Regarding the instance format passed to the memory-based classifier, it has been empirically demonstrated that the phrase should never be taken as one single feature, but should instead be spread out over multiple features, one for each word of the phrase. This can be done by either splitting the file for training and testing over multiple files for differing phrase-lengths, or it can be done by reserving a fixed amount of features for the phrase and filling those left-to-right with the words of the phrase, writing a special null symbol for any features that are unused.

In decoding it has been shown that starting with an initial hypothesis is a better option than starting out with a null hypothesis and incrementally adding hypothesis fragments. It has also been shown that subsequently applying substitution- and swap-operations is a viable strategy for improving upon this initial hypothesis, even though the resulting gain is modest. Phrase-based decoding differs from word-based decoding in needing to take into consideration different fragmentations over the input sentence. A strategy has been employed that starts the core decode process for the best x fragmentations, and in the end combines the results to select the highest scoring candidate. There are thus essentially two inter-dependent search problems to be solved. Despite the fact that this increases the complexity of decoding, the strategy of applying substitutions and swaps manages to perform the decoding task in a very limited and efficient time-span.

5.1 Future Research

This thesis opens up various possibilities for future research. A main aim would of course be the further improvement of phrase-based memory-based machine translation. However, the findings with respect to the omission of context information in classes also opens up new research opportunities for the word-based approaches, most notably Constraint Satisfaction Inference Canisius & van den Bosch (2009).

Better performance in memory-based learning may be achieved through further parameter optimisation. The parameters of memory-based learning have not been tweaked in this thesis, but have been limited to a single setting. It is very possible that a gain might be achievable by trying different values for the number k of neighbours to consider, the beam size or translation threshold in the PBMBMT decoder, and the information metric used for the ordering of the tree. Possibly results may be improved also by selecting a different algorithm instead of the hybrid TRIBL2 used throughout this research, a pure $IB1$ approach or a pure $IGTree$ approach could be attempted.

Another very notable opportunity for further research would be the experimentation with different language pairs. Right now all research consistently focussed on Dutch to English trans-

lation, both members of the Germanic sub-branch of the Indo-European language family. Different results could be expected when selecting languages that are less closely related to each other, and which require a larger amount of re-ordering. Results may be significantly worse when translating from for example a Verb-Subject-Object language such as Arabic, to a Subject-Object-Verb language such as Japanese.

The phrase-list method can be extended to a "double phrase list" approach, in which the target alignment of a selected phrase, must also occur in a phrase list of the target-language, thus effectively requiring two phrase lists rather than only one for the source language. This would establish a stronger bond within each phrase-pair, but might also significantly reduce the amount of extractable phrases. The threshold for the phrase list could also be attempted with a higher value, since we saw an increase in performance when moving from a phrase list with threshold 5 to 25. Finding the right threshold is again an optimisation problem and depends on the size of the corpus and type-token ratio of the words in corpus.

The decoder right now incorporates swap and substitution operations. Canisius & van den Bosch (2009) in addition relied also on a null model to power a deletion operation. The necessity of this operation was considered less because of the phrase-based nature, but inclusion of such a model and such an operation might be beneficial. It is not without problem though, as it would be necessary to implement a good penalty function that prevents the decoder from favouring the deletion of as many fragments as possible.

The swap operation in the decoder showed an unexpected characteristic: different values for different maximum-swap distances did not result eventually in different scores. The cause of this has thus far been only speculative and might be a subject for further future research. Extending the fairly simple swap operation might also be a necessity in dealing with language pairs such as Arabic and Japanese. Such a research should start with an investigation of how often swaps occur, and over what distances swaps have taken places in the end.

Since the inclusion of context in classes proved to have aversive effects, it might also be interesting to reconsider the role of target-side context in word-based memory-based machine translation approaches such as CSIMT (Canisius & van den Bosch, 2009). It is conceivable that here too better results may be achievable by omitting target-side context, backed up by the fact that the phrase-based decoder outperformed CSIMT on word-based level for one of the two tested corpora.

Finally, a research suggestion for memory-based translation in general: new studies could be undertaken to investigate the effect of the inclusion of more features, such as syntactic features like Part-of-Speech tags or semantic features such as semantic roles.

Bibliography

Aha, D. W., Kibler, D., & Albert, M. K.(1991, January). Instance-based learning algorithms. *Machine Learning*, *06*(1), 37–66.

Banerjee, S., & Lavie, A. (2005, June). Meteor: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization* (pp. 65–72). Ann Arbor, Michigan: Association for Computational Linguistics.

Bar-Hillel, Y.(1962). The future of machine translation. *Times Literary Supplement*.

Bar-Hillel, Y. (1964). A demonstration of the nonfeasibility of fully automatic high quality translation. *Appendix III of The present status of automatic translation of languages, Advances in Computers, vol.1*.

Brown, P. F., Pietra, V. J. D., Pietra, S. A. D., & Mercer, R. L.(1993). The mathematics of statistical machine translation: parameter estimation. *Comput. Linguist.*, *19*(2), 263–311.

Canisius, S., & van den Bosch, A. (2009). A constraint satisfaction approach to machine translation. In *Proceedings of the 13th annual conference of the european association for machine translation (eamt-2009)* (pp. 182–189).

Chomsky, N. (1969). Quine's empirical assumptions. In D. Davidson & J. Hintikka (Eds.), *Words and objections: Essays on the work of W.V. Quine* (pp. 53–68). Dordrecht: D. Reidel.

Daelemans, W., & van den Bosch, A.(2005). *Memory-based language processing*. Cambridge, UK: Cambridge University Press.

Daelemans, W., van den Bosch, A., & Weijters, T.(1997). Igtree: Using trees for compression and classification in lazy learning algorithms. *Artif. Intell. Rev.*, *11*(1-5), 407-423.

Daelemans, W., Zavrel, J., van der Sloot, K., & van den Bosch, A. (2007). *Timbl: Tilburg memory based learner, version 6.1, reference guide. ilk technical report 07-07* (Tech. Rep.).

Dempster, A. P., Laird, N. M., & Rubin, D. B.(1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, *39*(1), 1–38.

Germann, U. (2003). Greedy decoding for statistical machine translation in almost linear time. In *NAACL '03: Proceedings of the 2003 conference of the north american chapter of the association for computational linguistics on human language technology* (pp. 1–8). Morristown, NJ, USA: Association for Computational Linguistics.

Green, T.(1979). The necessity of syntax markers. two experiments with artificial languages. *Journal of Verbal Learning and Behavior*, *18*, 481–496.

Jurafsky, D., & Martin, J. H. (2009). *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Upper Saddle River, NY: Prentice-Hall.

Koehn, P. (2004). Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In R. E. Frederking & K. Taylor (Eds.), *In the proceedings of the american machine translation association* (Vol. 3265, p. 115-124). Springer.

Koehn, P. (2005). Europarl: A parallel corpus for statistical machine translation. In *In proceedings of the machine translation summit X ([mt]'05)*. (pp. 79–86).

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., et al. (2007). Moses: Open source toolkit for statistical machine translation. In *ACL*. The Association for Computer Linguistics.

Leech, G., Rayson, P., & Wilson, A. (2001). *Word frequencies in written and spoken english: Based on the british national corpus*. London: Longman.

Mitchell, T. (1997). *Machine learning*. McGraw-Hill Education (ISE Editions).

Nirenburg, S. (1996). Bar-hillel and machine translation: then and now. *Koppel and Shamir*, 300–305.

Och, F. J., & Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational Linguistics*, *29*(1), 19-51.

Papineni, K., Roukos, S., Ward, T., & Zhu, W. (2001). *Bleu: a method for automatic evaluation of machine translation.*

Russell, S. J., & Norvig, P. (2003). *Artificial intelligence: a modern approach* (2nd international edition ed.). Prentice Hall.

Shannon, C. E., & Weaver, W. (1963). *A mathematical theory of communication*. Champaign, IL, USA: University of Illinois Press.

Somers, H. (1999). Review article: Example-based machine translation. *Machine Translation*, *14*(2), 113–157.

Steinberger, R., Pouliquen, B., Widiger, A., Ignat, C., Erjavec, T., Tufis, D., et al. (2006, Sep). The JRC-acquis: A multilingual aligned parallel corpus with 20+ languages. In *Proceedings of the 5th international conference on language resources and evaluation (LREC'2006)* (pp. 24–26). Genoa, Italy.

Stolcke, A. (2002). *Srilm – an extensible language modeling toolkit.*

Stroppa, N., van den Bosch, A., & Way, A. (2007). Exploiting source similarity for smt using context-informed features. In *Proceedings of the 11th international conference on theoretical issues in machine translation (tmi 2007)* (pp. 231–240).

Tan, P.-N., Steinbach, M., & Kumar, V. (2005). *Introduction to data mining*. Addison Wesley.

Tiedemann, J., & Nygaard, L. (2004). The opus corpus - parallel and free. In *Proceedings of the fourth international conference on language resources and evaluation (LREC04)* (pp. 26–28).

van den Bosch, A., & Berck, P. (2009). Memory-based machine translation and language modeling. *The Prague Bulletin of Mathematical Linguistics*(91), 17–26.

van den Bosch, A., Stroppa, N., & Way, A.(2007). A memory-based classification approach to marker-based ebmt. In *Proceedings of the metis-ii workshop on new approaches to machine translation* (pp. 63–72).

van Rossum, G.(2006). *Python reference manual, release 2.5* (Tech. Rep.). Amsterdam, The Netherlands, The Netherlands.

Vauquois, B.(1968). A survey of formal grammars and algorithms for recognition and transformation in mechanical translation. In *Ifip congress (2)* (p. 1114-1122).

Vossen, P. (2006). Cornetto: een lexicaal-semantische database voor taaltechnologie. *Dixit Special Issue*.

Wilks, Y.(1979). Machine translation and artificial intelligence. *Translating and the Computer, B.M. Snell (e.d)*.

Witten, I. H., & Frank, E.(2002). Data mining: practical machine learning tools and techniques with Java implementations. *ACM SIGMOD Record*, *31*(1), 76–77.