Context as Linguistic Bridges

Een wetenschappelijke proeve op het gebied van de Letteren

Proefschrift

ter verkrijging van de graad van doctor
aan de Radboud Universiteit Nijmegen
op gezag van de rector magnificus prof. dr. J.H.J.M van Krieken,
volgens besluit van het college van decanen.

door

Maarten van Gompel

geboren op 15 december 1982
te Etten-Leur

Promotores:	Prof. dr. A. van den Bosch
Manuscriptcommissie:	...

CONTENTS

1

INTRODUCTION

"To know an object is to lead to it through a context which the world provides." – William James (American Philospher, 1861-1947)

"Form is emptiness, emptiness is form" – Heart Sutra

1.1 Context as Linguistic Bridges

To even begin to merit the title of "Doctor of Philosophy", it is only proper to start this dissertation with some philosophical deliberations.

The study you are about to read sprung from the intuition that *context* is an important, if not the most important, characteristic that defines language. Language itself only exists only in the context of the world that surrounds us, as well our internal mental world. Without this context, there is nothing to communicate in the first place. Language is sometimes considered the epitome of human evolution. Our species evolved the remarkable ability to refer to the world outside and within by producing complex meaningful utterances, i.e. speech.

This was an unparalleled revolution in communication, which has undoubtedly played a leading role in us becoming the dominant intelligent species on the planet. It has put us in a position where we can communicate our thoughts and feelings about the world to one another on a more fine-grained level than any other species can. It has given us the ability of ever-increasing abstraction. The context of our language is no longer limited to just refer to objects in our immediate vicinity, but we can even refer to abstract thought itself. Whenever communication is attempted between people who do not share a language, the context to which can be referred is restricted. Just imagine tourists in a foreign bakery lustfully pointing at the pastries they desire to buy and devour.

Another major revolution in communication has been the development of writing, and much later that of print technology. This and the accompanied literacy of populations allows for our thoughts and ideas to be preserved more easily and accurately than oral traditions can accomplish. The ability to read and write broadens one's world, one's context, and receiving language education is therefore even deemed a fundamental human right.

Language is inherently ambiguous and context is the disambiguating factor without which it cannot exist. The context of a bakery and a neatly arranged line of pastries is essential for the baker to be able to disambiguate the pointing of the clueless foreign tourist and discern which pastry he actually wants. Demonstrative pronouns such as

"he", "she", "this", or even definite noun phrases such as "the house" convey little information, if not for the context they are employed in. In fact, any word by itself is pretty limited, can never exist in total isolation, and only derives meaning from the further context. It is defined through the context, just like a dictionary defines words in terms of other words.

Buddhist philosophy has a concept called *"shūnyatā"*. Shūnyatā is Sanskrit and can roughly be translated into English as "emptiness". It is an ontological notion that states that that everything, not just words, is interrelated and nothing has any existence or essence of or by itself: this philosophy posits that everything exists and is defined purely through *its context* and can by definition never be seen as independent from it.

Linguistic bridges

Whereas we humans all share an astounding capacity for language, history has given rise to numerous distinct and often mutually unintelligible languages. This brings us back to the predicament of our pastry-loving tourist in the foreign bakery, unable to express his choice in the language of the baker and therefore resorting to pointing. If we situate the bakery in France, the baker may reply "croissant" in response to the gesturing of his client. The perceptive client may then have learned that this is what the pastry is called. The famous utterance "Me Tarzan, You Jane", heavily supported by gestures as in the movies, is of a similar nature. Both protagonists breach the language barrier and gain new information. These examples show that context plays an important role in establishing translation, or any common vocabulary.

The *linguistic bridges* of this dissertation refer to these acts of translation. We can define translation as *a process that yields a representation of a message initially expressed in one language, in another*. Etymologically, the word *translation*, from the latin *translātiō*, refers to carrying (lātiō) across (trans) something. The focus is generally on accurate preservation of the semantics of message, although in some arts, such as poetry, form or emotion may take precendence over the substance.

The intuition underlying our the present study is that the context of a word or phrase is an important cue for the translation of that word or phrase. A word in context A may be translated differently than the same word in context B. Consider the Portuguese word "tempo" in the following sentences:

1. "O **tempo** é bom hoje." – "The **weather** is nice today."
2. "Não tenho **tempo** para estudar hoje." – "I don't have **time** to study today."

In the first sentence, "tempo" is translated as "weather", whereas in the second it is translated as "time". Portuguese uses the same word where English uses two distinct words; context provides cues to disambiguate into the proper translation. Another example shows where the relationship between English words and their French counterpart is not one-on-one and requires context to properly disambiguate:

1. "J'ai tué la **mouche**" – "I killed the **fly**"
2. "Je **vole** à Paris" - "I **fly** to Paris"
3. "Je **vole** l'horloge de mon père" – "I **steal** my father's watch"

In this latter example, disambiguation between the first two sentences is facilitated by the fact that the word "fly" has a different part-of-speech in both sentences, unlike the noun "tempo" in the earlier example. The presence of the article "the" in the noun phrase "the fly" already rules out translation to a verb. The French verb "voler" in turn does not just translate to "to fly", but can also mean "to steal", as shown in the last sentence.

These examples illustrate that languages are almost never translatable on a naive word-by-word basis, leaving aside the issue of word order, as determined by a language's syntax. Context plays a major role in determining the right translation. We can say context fulfills a bridging function in translation, and this dissertation investigates techniques for machines to exploit this to attain better automatic translations.

1.2 Natural Language Processing: Background

In our philosophical deliberations we briefly addressed the communication revolutions brought about by humanity evolving the faculty of language and speech, and later the technological development of writing and print. We all live in fortunate times to have just witnessed the next biggest revolution in communication since the invention of print: the information revolution. Text is not just printed anymore, it is digitised; i.e. made available, distributable, and searchable in digital form. The internet enables these digital resources to be collected, shared, and exploited at unpredecented rate and allows us to communicate instantaneously with people from all over the planet.

This wealth of digital textual data fuels our field of research: Natural Language Processing (NLP). We attempt to extract meaningful patterns from natural language data. Most contemporary techniques in NLP are grounded in Machine Learning, which employs statistical and information-theoretical principles to learn desired patterns from big collections of natural language data. These data-driven approaches

in NLP can be contrasted to rule-based approaches, not without their own merit, that are directly derived from human expert knowledge.

This dissertation strictly follows the data-driven trend and we therefore rely on large amounts of digitised text to conduct our research. In this section we give some theoretical background on *Machine Translation* and *Word Sense Disambiguation*, in order to be able to understand the subject matter of this dissertation.

Machine Translation

The area of research dedicated to the automatic translation of data from one natural language to another is called Machine Translation (MT). At the time the study of this dissertation is conducted, state-of-the-art techniques in this field are primarily based on statistical methods, in which case we can speak of *Statistical Machine Translation* (SMT). These can contrasted to earlier rule-based Machine Translation systems, which are characterised by an *analysis* stage, a *transfer* stage according to a set of rules, and a *generation* stage. Such transfer may occur on a shallow, a syntactic or a semantic level.

Statistical Machine Translation, in contrast, treats translation as a problem where an ordered sequence of words or phrases has to be found that is the most *probable* translation of the input sentence. It effectively tests many possible translations for individual fragments, and various ways in which these translated fragments can be ordered. Each test is a *translation hypothesis*, and many such hypotheses are evaluated by the system for a single sentence. The one with the highest probability score is eventually selected.

Statistical Machine Translation is an integrated solution that handles two problems at once; that of finding the best lexical translation for words or phrases, and that of reassembling those in an acceptable order. This corresponds to two key properties of translation:

1. **Semantic faithfulness** - A translation must accurately transfer meaning for all of its parts. The right lexical translation has to be found for the words and expressions in the source.

2. **Syntactic fluency** - The translated result must be fluently expressed, the sentence must be syntactically well-formed.

In Statistical Machine Translation, two key models tackle these issues. The *translation model* models the translation of words or phrases from the source language to the target language. The *language model*, for the target language, provides the means to order reassembled translated fragments. Both are probablistic models that are learned from data. The translation model requires a *parallel corpus* as input, and the language model can be trained on monolingual data.

As said, the act of translation is expressed as the search for the most probable translation. Take S to be a sentence in the source language, and T to be a translated sentence, i.e. a translation hypothesis, in the target language. The SMT process can then be formalised as in Equation 1.1, where the outcome $\hat{T}$ corresponds to the most probable translation.

$$\hat{T} = \arg\max_{T} p(T|S) \tag{1.1}$$

By applying Bayes' theorem we can invert this to Equation 1.2. This inversion is necessary as we cannot model $p(T|S)$ directly. Instead, we apply what is known as the *noisy channel model*. The system generates a large amount of translation hypotheses. This can be interpreted as if our original sentence in the source language entered a noisy channel and comes out scrambled in the target language at the other end. We then compute how probable a translation it is of the original input. Note that we can drop the denominator $p(S)$ as it will be constant throughout the process.

$$\hat{T} = \arg\max_{T} p(T|S) = \arg\max_{T} \frac{p(S|T)p(T)}{p(S)} = \arg\max_{T} p(S|T)p(T) \tag{1.2}$$

The two factors we end up with in Equation 1.2 correspond precisely to the *translation model* ($p(S|T)$) and *language model* ($p(T)$) respectively.

Translation Model

The translation model is learned from parallel corpus data. A parallel corpus has the same text in two or more languages, i.e. the corpora in a parallel corpus are considered translated versions of one another[1]. Early SMT systems were word-based, i.e. the translation model consisted of probabilities for word translation pairs. These systems have been superseded by phrase-based systems, in which multiple words at once may be translated as an entity (Koehn, 2004a). This allows for translations in which multiple words in the source language translate to a different number of words in the target language, or vice versa.

The basis for training such a translation model is a parallel corpus, i.e. the same text in two versions, i.e. two languages, carrying approximately the same meaning. First a *sentence alignment* is computed (Tiedemann, 2003). This identifies sentences that are translations of each other through statistical likelihood estimations. Subsequently, a *word alignment* is computed, according to a method pioneered by Brown et al. (1990). See also the work of Och and Ney (2003) and

[1] More accurately; often one side is the original language and the ois a translation thereof

Tiedemann (2003). The word alignment attempts to identify words that are translations of each other, by means of an *expectation maximisation* algorithm (A. P. Dempster, 1977). Finally, we can derive phrase translation pairs from these word alignments.

The outcome of the training stage is a *phrase translation table*, which constitutes the translation model. It is a collection of phrase pairs (s, t) in source and respectively target language language. Associated with each pair are the phrase-translation probabilities $p(s|t)$ and $p(t|s)$, derived from the joint frequency of the pair and the individual frequencies of the parts in the data. Additional probabilities, such as lexical weights and word/phrase penalties, such as a penalty for long phrases, may be included as well.

At test time, the translation probability $P(S|T)$ for a given translation hypothesis is estimated by computing the product over all phrase pairs that make up the hypothesis ($s \in S$, $t \in T$). A score is computed for each based on the product of $p(s|t)$ with optionally $p(t|s)$ and other scoring components. Each of the components in this *score function* is typically parametrised by a weight (λ). This is shown in equation 1.3.

$$P(S|T) = \prod_{i=1}^{n} p(s_i|t_i)^{\lambda_1} p(t_i|s_i)^{\lambda_2} \tag{1.3}$$

The values of the weights λ_1 and λ_2 for the score function cannot be determined a priori, but have to be computed experimentally on held-out tuning data. This is heuristcally tackled by trying a huge variety of parameter values and experimentally observe which performs best on the data. This final *parameter optimisation* step is an essential part of SMT (Och, 2003).

Language Model

Note that the translation model is not sensitive to the order in which fragments are assembled. Ordering is instead guided by the language model, the second key component in an SMT system.

The language model predicts the likelihood of a word given a previous history of predecessor words. This is a model of *context*; the difference with the context that is the focus in this dissertation, however, is that it models the target language rather than the source language. Moreover, it is a purely monolingual model whereas our investigation is aimed at modelling source-side context for translation directly.

A simple trigram-based language model is shown in equation 1.4, where we compute the probability of a sequence of words W.

$$p(W) = p(w_1, \ldots, w_n) = \prod_{i=1}^{n} p(w_i|w_{i-2}, w_{i-1}) \tag{1.4}$$

An actual language model will likely also implement *back-off* strategies and *smoothing*, to avoid the problem of encountering words unseen during training, which would result in a probability score of 0 otherwise.

The role of the language model is mainly to assign a higher probability to sentences in which the target-language phrases are ordered properly than to sentences in which phrases are in an unnatural order.

1.2.1 Word Sense Disambiguation

Word Sense Disambiguation (WSD) shares certain characteristics with translation. The idea in WSD is to disambiguate between multiple meanings of a polysemous words given a context. Consider the following example:

1. "My money is safely deposited in the **bank**."
2. "The ship avoided crashing into the **bank** of the river."

The first example refers to a bank as a financial institution, whereas the second refers to the boundary of a river. In Word Sense Disambiguation we attempt to automatically infer which sense is intended. The actual sense is typically represented in the form of a sort of identifier from a sense inventory such as WordNet.

We already saw in Section 1.1 that translation requires a similar disambiguation. After all, translation usually aims to preserve the semantics. If we translate "bank" from these two examples from English to Spanish, we will obtain "banco" for the first example, but "orilla" for the latter. In Chapter 3 we will take an in-depth look at using WSD techniques for localised translation problems.

1.3 Research Questions

Our intuition is that source-side context information, i.e. words in the source language from the surrounding context, provides valuable cues for translation. The examples we have discussed hitherto aim to illustrate this intuition. The over-arching question we posit for the dissertation is the following:

1. To what extent can we improve translation by considering source-side context information?
2. What context features prove most effective?
3. To what extent are linguistically uninformed features effective?
4. How can source-side classifiers be used in translation tasks?

Here *translation* is understood to be a generic term that encompasses the more localised translation of fragments in a context, comparable to Word Sense Disambiguation, as well as a full integration of source-side context in Statistical Machine Translation.

In the first chapters, some linguistically-informed features such as lemma features and part-of-speech tags are tested. Our emphasis in the overall study, however, is on assessing the efficiency of *linguistically uninformed* features, i.e. employing only the raw textual data. The reason for this is that we aim to assess the merit of the techniques as such, as the purest analysis with the rawest data. Moreover, by choosing this approach we do not need to rely on external language-specific tools such as as lemmatisers and part-of-speech taggers. In subsequent chapters, we will also show that the inclusion of linguistically-informed features is something already studied and described in the literature by others.

Our main hypothesis is that inclusion of source-side context information, without linguistically informed features, benefits translation quality.

1.4 Thesis structure

We begin our research on the level of translating simple words in context, and gradually move up in complexity and integration with SMT throughout the dissertation.

Before we can go into the matter of translation, Chapter 2 first introduces the software that was needed to efficiently extract patterns such as n-grams, with their context, from corpus data. This chapter was published in "van Gompel, M. and van den Bosch, A. (2016). Efficient n-gram, skipgram and flexgram modelling with Colibri Core. *Journal of Open Research Software*, 4(1)". The software was explicitly written in the scope of this research project and provides a necessary foundation that is in turn used by most of the software of the remaining chapters. It is a technical chapter that can optionally be skipped if the reader wishes to more quickly proceed to the heart of the matter. At the same time, the chapter offers tools that are applicable beyond the present dissertation.

Chapter 3 introduces *Cross-Lingual Word Sense Disambiguation* and presents a classifier-based system employing source-side context information. This system, when tested against competing systems, obtains the best or near-best accuracy. This chapter is based on two publications, "van Gompel, M. (2010). UvT-WSD1: A cross-lingual word sense disambiguation system. In *SemEval '10: Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 238–241, Morristown, NJ, USA. Association for Computational Linguistics", and "van Gompel, M. and van den Bosch, A. (2013). WSD2: Parameter optimisation for memory-based cross-lingual word-sense disambiguation. In *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval 2013), in conjunction with the Second Joint Conference on Lexical and Computational Semantics*"

Chapters 4, 5 and 6 are strongly linked and open up a novel line of investigation. We propose, test and present a translation assistance system capable of translating fragments in a cross-lingual context, i.e. fragments in language different from the language of the context. Translation subsequently proceeds to translate these other-language fragments to the language of the context. This system changes and investigates the notion of what source-side context is and may find practical application in Computer Assisted Translation and Computer Aided Language Learning. In Chapter 4 we describe a pilot study to assess the feasibility of the idea. This chapter is based on "van Gompel, M. and van den Bosch, A. (2014). Translation assistance by translation of L1 fragments in an L2 context. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 871–880, Baltimore, Maryland. Association for Computational Linguistics". It applies the lessons learned from Cross-Lingual Word Sense Disambiguation (Chapter 3) to a new task. In chapter 5 we manually construct a data set and evaluate the performance of several 3rd party participating systems on our data. This is set up as a shared task for the SemEval 2014 conference and was published as "van Gompel, M., Hendrickx, I., van den Bosch, A., Lefever, E., and Hoste, V. (2014). Semeval-2014 Task 5: L2 writing assistant. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 36–44, Dublin,Ireland". In Chapter 6 we finally present our own reworked "Colibrita" system on the basis of the pilot study and findings from the SemEval participants. Here we present the results and form conclusions regarding the role of context and the usefulness of classifiers versus Statistical Machine Translation techniques. This chapter also appeared in "the International Journal of Translation (2016)".

Chapter 7 is an attempt to directly integrate source-side context information in Statistical Machine Translation, by means of memory-based classifiers. It has not been published and is a novel contribution to this research. Chapter 8 aggregates and summarizes the conclusions from earlier chapters, and forms a final answer to our research questions.

2

EFFICIENT N-GRAM, SKIPGRAM AND FLEXGRAM MODELLING WITH COLIBRI CORE

In this chapter we focus on how we can computationally extract patterns in an efficient way that also allows for modelling their context. This is a necessary prerequisite for the further stages of the research. The chapter has a strong software-oriented focus and format. It explicitly introduces the software *Colibri Core*, which is the key component in the wider software employed in the research. This software, however, also finds application in a much broader context.

Counting n-grams lies at the core of any frequentist corpus analysis and is often considered a trivial matter. Going beyond consecutive n-grams to patterns such as skipgrams and flexgrams increases the demand for efficient solutions. The need to operate on big corpus data does so even more. Lossless compression and non-trivial algorithms are needed to lower the memory demands, yet retain good speed. Colibri Core is software for the efficient computation and querying of n-grams, skipgrams and flexgrams from corpus data. The resulting pattern models can be analysed and compared in various ways. The software offers a programming library for C++ and Python, as well as command-line tools.

This chapter is based on: van Gompel, M. and van den Bosch, A. (2016). Efficient n-gram, skipgram and flexgram modelling with Colibri Core. *Journal of Open Research Software*, 4(1)

2.1 Introduction

The n-gram, a sequence of n consecutive word tokens, is a core concept for many Natural Language Processing (NLP) applications. One of the most basic NLP tasks is to read corpus text and compute an n-gram frequency list, elementary for any kind of statistical analysis. The unigram frequency list, i.e. the word frequency list, is the simplest instance of this task which is especially ubiquitous. Computing n-gram frequency on a corpus text is fairly trivial, and any beginning computer science student will have no trouble to accomplish this in just a few lines of code in a modern high-level programming language. However, optimising this to reduce memory constraints, improve speed, and scale to large data, is a more complex matter. Colibri Core, the NLP software we introduce here, offers efficient algorithms to do this.

N-grams are typically distributed in a Zipfian fashion, implying there are only a few high-frequency patterns, with words such as

common function words in the lead, and there is a long tail of patterns that occur only very sparsely. This basic fact makes counting a notoriously memory-hungry enterprise, as patterns occurring below a minimum frequency threshold can not be discarded from memory until the entire corpus has been processed sequentially.

When working with large data sets and higher-order *n*-ngrams, this memory problem becomes apparent quickly when trivial solutions are employed. Colibri Core, on the other hand, offers tools and programming libraries that are heavily optimised to 1) reduce memory usage, and 2) offer high-speed performance.

The task of finding *n*-grams is generalised in Colibri Core to the task of finding *patterns*. Furthermore, once patterns are identified, resulting in a *pattern model*, Colibri Core can extract relations between the patterns.

As the name Colibri Core suggests, the software is geared to provide *core* functionality for modelling patterns and exposes this functionality as a programming library as well as through command line tools. It aims to provide a solid foundation upon which more specialised software can be built, such as software for language modelling. The software is aimed at NLP software developers and researchers with a solid technical background.

Related Work

Pattern extraction, and with it pattern matching, plays an important role in computer science. In the NLP literature, n-grams are a common type of pattern, and their modelling is often researched in the context of statistical language modelling. Software that springs from such studies is widespread in the field. Examples, by no means exhaustive, are SRILM (Stolcke, 2002), IRSTLM (Federico et al., 2008), and KenLM (Heafield, 2011). Focus on efficient modelling with regards to memory consumption and look-up speed is an important component in such studies. Others also focus on big data storage in this field (Guthrie and Hepple, 2010), though this study did not produce a usable open source software solution.

Colibri Core, however, takes a step back and is not a language modelling toolkit and therefore can't be readily compared with one. It is, however, very suitable to be used as a foundation to that end, which has been done by one study already (Onrust et al., 2016). What we do have in common with language modelling toolkits is that both types of systems contain a store of patterns (i.e. n-grams or beyond) and their frequencies, in which quick look-up is essential, and both provide a procedure to compute such a model. Both LM toolkits as well as Colibri Core employ various optimisations to reduce the memory usage of this store, and keep access times high.

Language Modelling toolkits do not generally offer any of the functionality Colibri Core offers when it comes to thresholding, indexing, nor the modelling of non-consecutive patterns such as skipgrams. The issue of skipgram extraction and modelling gained more traction in the past decade, see for instance Guthrie et al. (2006). The notion of skipgram (or rather flexgram in our terms, as shall become apparent later) also plays a part in vector space representations (Mikolov et al., 2013) that have become increasingly popular.

We think Colibri Core fills an interesting niche that is not covered by other readily available software. It provides solutions for the modelling of n-grams and skipgrams/flexgrams in an efficient and sufficiently extensible manner.

2.2 Implementation and Architecture

The overall architecture of Colibri Core is visualised in Figure 2.1, from a data flow perspective. The components shown here will be discussed in this section.

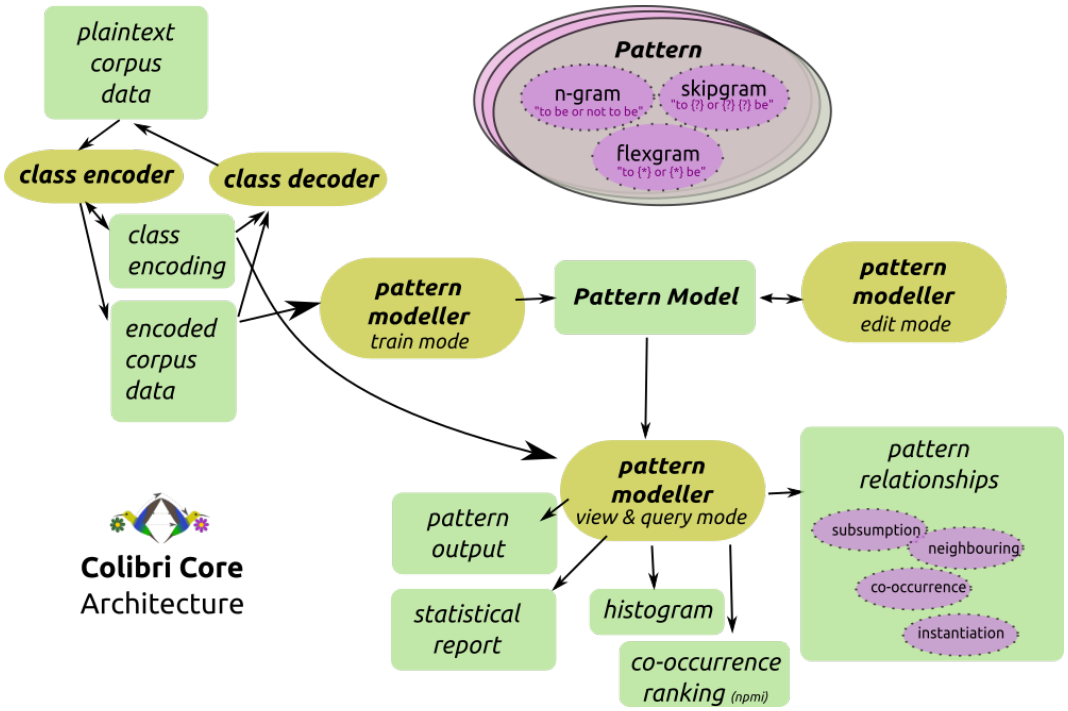

Figure 2.1: The Colibri Core architecture; light green squares represent data models, ochre (yellowish) rounded squares represent processes that manipulate data.

We will present the various features and optimisations that are implemented in Colibri Core. We start with a introduction of patterns and their encoding, Then discuss the implemented optimisation we use to count n-grams, and subsequently skipgrams. We then discuss more

advanced parametrisation and end with a section on the computation of flexgrams.

Feature: Patterns

We distinguish three categories of patterns, and define them as follows:

1. N-grams – A sequence of n word tokens that are all consecutive. For example: "`to be or not to be`"
2. Skipgrams – A fixed-length sequence of p word tokens and q token placeholders/wildcards with total length n ($n = p + q$), the placeholders constitute gaps, or skips, and a skipgram can contain multiple of these. In turn, a gap can span one or more tokens. For example: "`'to _ or _ _ be`"
3. Flexgrams – A sequence with one or more gaps of variable length, which implies the pattern by itself is of undefined length. For example: "`to * or * be`"

Our definitions are defined narrowly and, with exception of n-grams do not necessarily match up precisely to the way the concepts are used in other studies. Some may use the term skipgram to include what we call flexgram, or use another term such as "elastigram" to refer to flexgrams (Forsyth and Sharoff, 2014).

Skipgrams are used in the field to obtain a higher level of generalisation than can be offered by n-grams. They can, for instance, be found as features in classification tasks (D'hondt et al., 2012), or as a means in language modelling to decrease perplexity (Guthrie et al., 2006; Onrust et al., 2016).

Dealing with word tokens implies that the corpus data has to be in a tokenised form. We start from the basis of plain-text corpus data, containing one *unit* per line; a unit can correspond to a sentence, paragraph, tweet or whatever unit is deemed appropriate for the task at hand. Corpus data can alternatively be provided in FoLiA XML format (van Gompel and Reynaert, 2013) as well, although linguistic annotation will be ignored.

Text data is typically stored as a string of characters. The characters themselves draw their denotation from a character encoding. The storage of a huge amount of strings is inefficient from a memory perspective, considering the fact that words follow a Zipfian distribution. Colibri Core therefore works on the basis of a lossless compression, in which each unique word token is assigned a numeric type identifier, which we call a *class*. This effectively defines the *vocabulary* of your data, which we call a *class encoding*. A pattern is not represented as an array of characters, but as an array of these classes instead. Such methods are commonly employed in Language Modelling toolkits

as well, where 24 or 32-bit integers uniquely assigned to words are typically chosen (Guthrie and Hepple, 2010).

In Colibri Core, further lossless compression is achieved by holding this array of classes in a dynamic-length byte representation rather than fixed-sized integers. This allows for low class values to be stored in fewer bytes than high class values. Classes 0 to 127 can be stored in a single byte, higher classes require at least two bytes. To achieve maximum compression, classes are assigned to word tokens based on frequency (i.e. entropy encoding): words with the highest frequency receive the lowest classes. This is essentially a variant of Huffman coding (Huffman, 1952). Some of the lowest classes are reserved for special purposes, e.g. to delimit sentences (class 0) or as markers for out-of-vocabulary words (class 2) or skipped content (classes 3 and 4).

Of each byte in the class representation, the highest bit is reserved as a continuation marker. As long as the continuation marker is set, the next byte is still part of the class. When it is low, we know we are at the final byte of a class representation. The class itself is stored in the remaining 7-bits of each byte. In practise this results in good compression and reduces memory usage; an example corpus taking up 221 MB on disk in plaintext form is reduced to just 60 MB when compressed. A subset of the Google billion word corpus, 769 million words in 30 million sentences, takes 3.9 GB in plaintext form, and 1.2 GB when compressed in this matter.

To encode a text corpus, a class encoding needs to be computed first, as visualised in the top-left corner of Figure 2.1. To decode the encoded corpus back to plain text, the class encoding is needed again. Colibri Core provides tools and exposes library functions to do this.

Optimisation: Informed Iterative Counting

N-gram frequency lists are often parametrised by a certain threshold. All n-grams below this occurrence threshold are pruned. We can circumvent the problem of having to hold a huge amount of patterns in memory that do not meet the threshold, as is typical in a Zipfian distribution. We do this by employing informed counting. Informed counting is an iterative procedure, shown in pseudo code in Algorithm 1. Here we take m to be the maximum n-gram order we intend to extract. The whole corpus is then processed for each n where $1 < n \leq m$, extracting the respective n-grams at each iteration. This means that at each iteration, we can consult the results of the previous iteration. We can then readily discard any n-gram with $n > 1$ for which either of the two $n-1$-grams it contains does not meet the threshold, as it follows that the n-gram can never meet the threshold either. By outright discarding an n-gram we do not need to store it and its count in memory. After each iteration of n, we prune all the n-grams that did not reach the threshold.

Algorithm 1 Informed Iterative Counting for n-grams. Take m to be the maximum n-gram order we intend to extract, t to be the minimum occurrence threshold, and M to be the pattern model in memory, with unigrams already counted in the more trivial fashion.

for $n \in 2..m$ **do**
  for $line \in corpus$ **do**
    for $ngram \in extractNGrams(line, n)$ **do**
      $nm1gram_1, nm1gram_2 \leftarrow extractNGrams(ngram, n-1)$
      if $M(nm1gram_1) \geq t$ & $M(nm1gram_2) \geq t$ **then**
        $M(ngram) \leftarrow M(ngram) + 1$
      end if
    end for
  end for
  $M \leftarrow pruneNGrams(M, n, t)$
end for
return M

Though not expressed in the simplified algorithm above, the actual implementation accounts for more parameters, such as setting a lower bound to n. The amount of back-off, going all the way up to $m-1$ here, can also be fine-tuned.

This algorithm makes concessions to processing speed, as multiple passes over the data are needed, to conserve memory. A performance evaluation of this algorithm will be addressed later in the section on *Quality Control*.

Optimisation: Informed Skipgram Counting

The computation of skipgrams is parametrised by an upper limit $l \leq m$ in the number of tokens, i.e. the total length of the skipgram (including gaps) expressed in tokens. The possible configuration of gaps increases exponentially with the total length spanned. A skipgram of size three has only one possible gap configuration[1]; `a _ z`, a skipgram of size four already has three possible configurations; `a _ _ z`, `a b _ z` or `a _ y z`.

The algorithm, shown in Algorithm 2 considers all possible configurations in which skips can be inserted in all of the n-grams in the model. It can discard a skipgram candidate by looking at the non-skipped parts that make up the skipgram, and by checking whether those exceed the set threshold. Note that the computation of skipgrams first requires counts of all n-grams where $0 < n \geq l$.

[1] The initial and final token may never be gaps in the extracted skipgrams

Algorithm 2 Informed Counting for skipgrams. Take l to be the maximum skipgram order we intend to extract, t to be the minimum occurrence threshold, and M to be the pattern model in memory, with ngrams already counted.

```
for n ∈ 3..l do
    for ngram ∈ getNGrams(M, n, t) do
        for skipgram ∈ possibleConfigurations(ngram) do
            docount ← True
            for part ∈ parts(skipgram) do
                if M(part) < t then
                    docount ← False Break
                end if
            end for
            if docount then
                M(skipgram) ← M(skipgram) + 1
            end if
        end for
    end for
    M ← pruneSkipgrams(M, n, t)
end for
return M
```

In this algorithm, the $possibleConfigurations(ngram)$ function returns all possible skipgram configurations for the given n-gram. Note that the configuration of gaps depends only on the length of the n-gram, regardless of its content, and can therefore easily be precomputed. The $parts(skipgram)$ function returns all consecutive parts that are subsumed in the skipgram, i.e. the parts delimited by the gaps.

Like Algorithm 1, Algorithm 2 assumes a threshold $t > 1$. When $t = 1$, more trivial algorithms are invoked, as the user does not want to prune anything. These make only a single pass over the data. For skipgrams this leads to an explosion of resulting patterns, exponential with number of tokens, and is best avoided.

Parametrisation: What counts?

The counting algorithms are parametrised by various other parameters which are not shown in Algorithm 1 and Algorithm 2 to reduce complexity. The wide variety of parameters allow the user to influence precisely what is counted and this is one of the main assets of Colibri Core. Parameters exist to affect the following:

- The minimum and maximum length (in words/tokens) of the n-grams and/or skipgrams to be extracted. Setting minimum and maximum length to the same value will produce a model of homogeneous pattern length (e.g. only trigrams or words).
- A secondary *word* occurrence threshold can be configured. This is a value set higher than the primary occurrence threshold. Only patterns occurring above the primary threshold, and for which each of the individual words/unigrams in the pattern passes the secondary threshold as well, will be included in the model.
- N-grams that are not subsumed by higher order n-grams, i.e. which do not occur as part of a higher order n-gram in the data/model, can be pruned from the model. This allows you to extract for instance all trigrams and all bigrams and unigrams that make up the trigrams, but not the bigrams and unigrams that are not subsumed in trigrams.
- Skipgrams can be constrained using the *skip type threshold*. This requires that at least the specified number of distinct patterns must fit in the gaps of the skipgram. Higher values will produce less skipgrams, but typically more generic ones. For instance, a skipgram such as *The _ house* will then only be included in the model if the corpus has instances in which the gap can be filled by at least the specified number of distinct patterns. If the threshold is set to 2 for example, and the corpus contains *The big house* and *The small house*, then the skipgram *The _ house* is included. If the corpus only has one of these instantiations, and no other instantiations of the skipgram either, then the skipgram would not be included.
- Skipgrams and n-grams are typically computed using the same occurrence threshold, but it is also possible to use a different threshold for skipgrams.

Feature: Pattern Models

A pattern model is a $key \mapsto value$ store, where the keys correspond to patterns and the values typically correspond to occurrence counts, although any kind of other value is supported too. Our aim with pattern models is to have a data structure that allows for *quick* lookup and iteration, as well as quick insertion during training. Moreover, memory consumption should be as conservative as possible, to allow handling of big data.

The underlying C++ library allows for choosing the actual underlying container implementation and value type through *templating*. The default container datatype is a hash map[2], which guarantees $O(1)$

[2] using the `unordered_map` STL container in C++11

access and update times under ideal hashing conditions. The hash[3] is computed directly from the binary representation of a pattern. Storing each pattern individually results in a lot of redundant information to be stored, as patterns overlap to a large degree. To conserve memory, the models can store pattern pointers[4] instead. Instead of duplicating the content for each pattern, these point to the original corpus data which is held in memory.

The use of hash maps can be contrasted to the use of suffix (or prefix) tries (Weiner, 1973), a common datastructure for storing n-grams in which a tree is constructed and any path in the tree, complete or incomplete, corresponds to a suffix (or prefix). Although suffix tries also benefit from decreased memory use due to no overlap in pattern data, the strongly linked nature of tries causes a significant overhead in memory use[5] that quickly exceeds the memory footprint of hash maps. For this reason, hash maps are the default and tries are currently not implemented in Colibi Core. The templating, however, does allow for such an implementation to be added in the future. This flexibility to abstract over the underlying data structure is one of the assets of Colibri Core.

At this point, we need to address suffix arrays (Manber and Myers, 1990) as well, which is a sorted array of suffixes, derived from suffix tries but with significantly decreased space requirements. Suffix arrays with longest common prefix (LCP) arrays will consume less memory than our hash maps, but are typically much slower to construct and query. Though no exhaustive experiment was conducted to this end, we did compare a predecessor of Colibri Core with a suffix array implementation (Stehouwer and Van Zaanen, 2010) and found our implementation to be significantly faster in model construction.

We distinguish two types of pattern model, depending on the type of the values, which in the underlying C++ implementation is subject to templating as well:

1. **Unindexed Pattern Models** – Values are simple integers
2. **Indexed Pattern Models** – Values are arrays of indices where the pattern's occurrences in the corpus are stored.

Obviously, indexed pattern models make a considerably higher demand on memory. They do, however, allow for a broader range of computations, as shall become apparent in subsequent sections.

The command-line tool `colibri-patternmodeller` exposes most of the functionality for training, viewing and editing pattern models

[3] Spooky Hash v2 is used for hashing: http://burtleburtle.net/bob/hash/spooky.html
[4] Each pattern pointer takes up 16 bytes
[5] Each pointer consumes 8 bytes on 64-bit architectures, and one would be needed for every transition between two tokens.

(see also Figure 2.1). In the C++ and Python APIs, this functionality is generally available through methods on some flavour of the PatternModel class.

Optimisation: Two-step training

Training indexed patterns models is more memory intensive than training unindexed models, especially in very large corpora (say at least a hundred million words). To lower the demand on memory for such corpora, we implement a *two-step training* procedure. This involves first constructing an unindexed pattern model and subsequently constructing an indexed model on the basis of that, by making another pass over the corpus and gathering all indices. The gain here is in avoiding temporary storage of the indices that will not pass the occurrence threshold but that cannot be ruled out a priori by the informed counting algorithm. This conservation of memory comes at the cost of extended execution time.

Feature: Corpus Comparison

The computation of pattern models on two or more distinct corpora, provided the class encoding is the same for all of them, provides a basis for comparative corpus analysis. One measure for corpus comparison introduced in the software is the notion of *coverage*. This metric is expressed as the number of tokens in the test corpus that is covered by the patterns from the training corpus. This makes it an asymmetric metric, so the choice of training and test corpus matters. The metric can be represented either in absolute counts, or in normalised form as a fraction of the total amount of tokens in the test corpus.

To perform such comparisons, we first compute a pattern model on the training corpus, and subsequently compute a second pattern model on the test corpus, but *constrained* by the former pattern model. The ability to train constrained models is present throughout the software and can for instance also be used to train a pattern model based on a custom preset list of patterns, effectively limiting the model to this preselection. The previously described two-step training algorithm is also an example of constrained training.

Summarised statistics are computed at multiple levels. Measures such as occurrence count and coverage can be consulted for aggregates of n-grams, skipgrams, or flexgrams, as well as specific patterns. The former two can be inspected specifically for each of the different pattern sizes present in the model, i.e. for each value of n.

The coverage metric is a fairly crude metric of corpus overlap, despite the ability to assess it for different aggregates. A more widely established metric for corpus comparison is *log-likelihood*. Log likelihood expresses how much more likely any given pattern is for either

of the two models. It therefore allows you to identify how indicative a pattern is for a particular corpus. Our implementation follows the methodology of Rayson and Garside (2000). They compute the log-likelihood (L) given the frequency of a pattern in corpus 1 (a), and corpus 2 (b) as follows[6]:

$$L = 2a(\log \frac{a}{E_1}) + b(\log \frac{b}{E_2}), \text{ where } E_i = \frac{N_i(a+b)}{N_1 + N_2} \tag{2.1}$$

Feature: Relations between Patterns

Various relations can be extracted between patterns in a pattern model, either through the API or a dedicated query tool. For all but the first of the relation types an indexed pattern model is required.

- **Subsumption Relation** – n-gram xyz subsumes $n-1$-grams xy and yz.
- **Succession Relation** – Patterns that occur in a sequence in the corpus data. For example: pattern x precedes yz and pattern z succeeds xy.
- **Instantiation Relation** – Skipgrams or flexgrams may be *instantiated* by other patterns. For example, "to be _ not _" be is instantiated by "or _ to", resulting in the 6-gram "to be or not to be". This type of relation thus allows you to precisely determine what patterns occur in certain gaps.
- **Co-occurrence Relation** – Different patterns can naturally co-occur multiple times within the the structural "units" you decided upon for the corpus (e.g. sentences, paragraphs, tweets, etc). These units are newline delimited in your original input. The measure of such co-occurrence can be expressed by established metrics such as Jaccard and (normalised) mutual pointwise information.

For each of these categories, the relationship is bidirectional, i.e. you can query for the subsuming patterns as well as the subsumed patterns, the left neighbours as well as the right neighbours, the instantiations as well as the abstractions. The co-occurrence relationship is fully symmetrical.

These latter three relationships rely on both the *forward index* inherent in an indexed pattern model, as well as the *reverse index*, a function from positions in the corpus to an array of patterns that are found at said position. The reverse index is not modelled in memory as an explicit mapping from positions to patterns, that would use too

[6] Here, E_i expresses expected values, N_i is the total amount of tokens in the respective corpus

much memory, but as a simpler reverse index that only keeps track of where each unit/sentence starts. This can then be used to quickly resolve specific positions to all possible patterns.

Feature: Flexgram Counting

Thus far, we have explained the algorithms for n-gram counting and skipgram counting, but have not yet done so for flexgrams, i.e. patterns with variable-width gaps. The implementation allows flexgrams to be computed in two different ways. The first is by extracting skipgrams first, and then abstracting flexgrams from these skipgrams. In this case the flexgram computation is constrained by the same maximum-size limit under which the skipgrams have been extracted. The second method for flexgram extraction proceeds through the co-occurrence relation. A flexgram is formed whenever two patterns (within the same structural unit) co-occur above a set threshold. The implementation of this latter method is currently limited to flexgrams with a single variable-width gap. This method is recommended when the user is interested in long-distance flexgrams, whereas the abstraction method is recommended when the user is more interested in having multiple gaps or the relationship between flexgrams and skipgrams.

2.3 Quality Control

Unit tests

To ensure the software is working as intended, an extensive series of unit tests is available. The tool `colibri-test` tests the various functions of the C++ library. The `test.py` script tests the Python binding. Testing in the form of continuous integration is made possible through *Travis CI*, where all test results are publicly available for inspection. [7]

Performance Evaluation

In this section we conduct a performance evaluation of Colibri Core by measuring the time to compute a pattern model, and the memory resources used.

Comparisons can be made between Colibri Core's unindexed vs indexed models, between the optimised pointer models vs standard pattern models, and between preloading a corpus in memory or reading it directly from file. For the experiments, shown in Table 1, we

[7] https://travis-ci.org/proycon/colibri-core

used a corpus of Dutch translations of TED talks of 127,806 sentences and 2,330,075 words[8].

We perform tests without thresholding as well as with an occurrence threshold of 2, and extract anything from unigrams to 8-grams.

For reference, we include a naive Python implementation that simply counts n-grams and stores them in a Python dictionary where the keys are strings and the values integers. We also include a comparable implementation that uses NLTK[9]. Similarly, we include a naive C++ implementation based on a `std::unordered_map<string,uint32_t>`. The memory reduction that can be observed when comparing these against the Colibri Core models is attributed to the class encoding scheme we use. These memory optimisations do come with a performance drawback that is especially noticeable when no thresholding is applied.

The pattern pointer models prove capable of further reduction in memory consumption, and offer a clear speed advantage as well. The pointer models use a representation of patterns that refer to the original corpus data, which is fully loaded into memory, rather than storing a separate copy for each pattern.

For the experiments with thresholding, we added a simple Python implementation of iterative counting and assess the value of the algorithm as such by comparing it against the naive Python implementation that simply prunes values below threshold at the very end. We clearly observe a drastic reduction in memory usage, at the cost of a longer execution time due to the multiple iterations.

Though Colibri Core is not a Language Modelling toolkit in itself, a comparison with a popular third-party LM toolkit may be of interest. We compare an unindexed pattern model containing unigrams, bigrams and trigrams to a similar model in SRILM (Stolcke, 2002). [10]. Due to the many other features Colibri Core offers regarding thresholding, indexing, skipgrams and flexgrams, it can not yet rival the performance and memory efficiency of dedicated LM systems, which allow for more specific optimisations.

Indexed Pattern Models are by definition larger in memory than unindexed models, but they do offer a significant speed benefit in the computation of skipgrams, both of this can be observed in Table 1.

The C++ library allows for easy swapping of the underlying datastructure for pattern models. The default hashmap (`std::unordered_map`)

[8] The data is from the IWSLT 2012 Evaluation Campaign, http://hltc.cs.ust.hk/iwslt/index.php/evaluation-campaign/ted-task.html#MTtrack. Tokenisation was performed using ucto, https://languagemachines.github.io/ucto.

[9] Natural Language Toolkit, a popular platform for NLP on Python: http://www.nltk.org. Our implementation follows the naive approach but uses `ngrams()` from `nltk.util` and `FreqDist` from `nltk.probability`

[10] SRILM's `ngram-count` is run in vanilla form, i.e. no smoothing or interpolation, with the options `-no-bos -no-eos`. Note that the encoding of classes is a separate step in Colibri Core, so our total CPU time should be considered to be a second longer than reported for a more fair comparison.

Experiment	CPU time	Memory
IWSLT data, without thresholding ($t = 1, l = 8$)		
$*$ Naive Python implementation	20.0 s	1404 MB
$*$ Python NLTK implementation	18.7 s	1413 MB
$*$ Naive C++ implementation	10.5 s	1398 MB
Unindexed Pattern Model (from file)	29.7 s	775 MB (787 MB)
Unindexed Pattern Pointer Model (preloaded corpus)	19.8 s	615 MB (627 MB)
IWSLT data, with thresholding ($t = 2, l = 8$)		
$*$ Naive Python implementation	28.8 s	1485 MB
$*$ Python with iterative counting	70.9 s	171 MB
Unindexed Pattern Model (from file)	14.6 s	64 MB (76 MB)
Unindexed Pattern Model (preloaded corpus)	11.3 s	64 MB (76 MB)
Unindexed Pattern Pointer Model (preloaded corpus)	9.0 s	50 MB (62 MB)
Indexed Pattern Model (preloaded corpus)	13.6 s	148 MB (160 MB)
Indexed Pattern Pointer Model (preloaded corpus)	10.1 s	133 MB (146 MB)
Unindexed Pattern Model (ordered map)	30.5 s	84 MB (96 MB)
Unindexed Pattern Model (preloaded corpus), with skipgrams	62.4 s	105 MB (118 MB)
Unindexed Pattern Pointer Model (preloaded), skipgrams	50.0 s	89 MB (102 MB)
Indexed Pattern Model (preloaded corpus), with skipgrams	24.2 s	165 MB (178 MB)
IWSLT data, LM comparison, trigram model ($t = 1, l = 3$)		
Unindexed Pattern Model (from file)	5.9 s	152 MB (164 MB)
$*$ SRILM ngram-count	1.5 s	80 MB

Table 2.1: Colibri Core performance benchmarks on the IWSLT data set. Performed by the `colibri-benchmarks` tool and the `benchmarks.py` script for the Python baselines. All non-Colibri Core references are marked with an asterisk ($*$). Memory usage is measured as the difference in resident memory after training and before training. Peak memory usage is measured absolutely as reported by the OS and included in parentheses. All experiments were performed on a Linux system with an Intel Xeon CPU (E5-2630L v3) at 1.80GHz and 256GB RAM. Parameter t refers to the occurrence threshold, l to the maximum pattern length.

solution can be contrasted to a pattern model using using an ordered map (`std::map`[11]). As expected, the ordered map proves to be significantly slower and does not yield a memory advantage either.

As Colibri Core is suited for the processing of large data sets, provided sensible thresholds are set, we conduct extra experiments on the JRC-Acquis corpus, containing 31.3 million words in over 850,000 sentences, and on a portion of the Google Billion Word corpus, where we use a set of 768 million words in 30 million sentences. These results are shown in Table 2.

Noticeable in the big data experiments is the explosion in memory consumption when skipgrams are computed on an unindexed pattern model. This is attributed directly to the sheer amount of skipgrams that can be extracted from the data, and the fact that unindexed models can only compute skipgrams exhaustively and do not support

[11] the hash key is still computed as it determines the ordering

Experiment	CPU time	Memory
JRC-Acquis data, with thresholding ($t = 2, l = 8$)		
$*$ Naive Python implementation	8m 4s	11407 MB
Unindexed Pattern Model (from file)	4m 18s	1704 MB (1803 MB)
Unindexed Pattern Model (preloaded corpus)	3m 58s	1700 MB (1800 MB)
Unindexed Pattern Pointer Model (preloaded corpus)	3m 1s	1344 MB (1445 MB)
Indexed Pattern Model (preloaded corpus)	4m 36s	4117 MB (4217 MB)
Unindexed Pattern Model (preloaded corpus), skipgrams	61m 1s	30420 MB (30520 MB)
Indexed Pattern Model (prel.), skipgrams, skiptypes= 12	80m 11s	29746 MB (29846 MB)
JRC-Acquis data, LM comparison, trigram model ($t = 1, l = 3$)		
Unindexed Pattern Model (from file)	42 s	716 MB (847 MB)
$*$ SRILM ngram-count	12 s	348 MB
Google Billion Words corpus, with heavy thresholding ($t = 10, l = 4, y = 20$)		
Unindexed Pattern Pointer Model (preloaded corpus)	83m 14s	12279 MB (15317 MB)
Google Billion Words corpus, trigram model ($t = 1, l = 3$)		
Unindexed Pattern Model (from file)	$25m16s$	16568 MB (19788 MB)

Table 2.2: Benchmarks on two large data sets. Same setup as Table 1. Parameter y represents the occurrence threshold specifically for skipgrams.

the skip type threshold discussed in Section “Parametrisation: What counts?”.

Documentation

Extensive documentation, including API references for both Python and C++, is provided at https://proycon.github.io/colibri-core. An interactive tutorial for Python is also available. The next section will provide a small example of possible usage from the command line.

Usage Example

Although we refer to the aforementioned documentation and the Python tutorial for extensive usage instructions, we include a short use case in this section and show how to use the command line tools. Assume you have a corpus `corpus.txt` and you want to extract all n-grams, skipgrams and flexgrams that occur more than five times. We set the maximum length to 8. For skipgrams, we add the additional constraint that it abstracts over at least three distinct ngrams, i.e. there are three different ways of filling the gaps. Flexgrams we derive from skipgrams.

First we should ensure this corpus is properly tokenised and that sentence splitting has been performed, putting each sentence on one line to make this the basic unit Colibri Core can count with. This preprocessing is not done by Colibri Core but using an external tokeniser such as ucto[12]:

```
$ ucto ucto -L en -n corpus.txt > corpus.tok.txt
```

Once preprocessing is done, we can encode the corpus in a form suitable for Colibri Core, computing a *class encoding* and converting the data to it, as explained in section "Feature: Patterns". The flow of this and all subsequent steps is also visualised in the architectural scheme in Figure 2.1.

```
$ colibri-classencode corpus.tok.txt
```

This will result in the encoded corpus `corpus.tok.colibri.dat` and the corresponding vocabulary file `corpus.tok.colibri.cls`.

The main tool `colibri-patternmodeller` can now be invoked to extract ngrams, skipgrams and flexgrams and output a pattern model, as described in section "Feature: Pattern models", to file:

```
$ colibri-patternmodeller --datafile corpus.tok.colibri.dat \
--outputmodel corpus.colibri.patternmodel --threshold 5 \
--maxlength 8 --skipgrams --skiptypes 3 --flexgrams S
```

[12] https://languagemachines.github.io/ucto

We have opted for an indexed pattern model, which gives us more thresholding options such as `--skiptypes`, and makes skipgram extraction more efficient, but at the cost of a much higher memory footprint compared to the simpler unindexed models[13].

We can now read the model from file and produce output to standard output in the form of *1)* a print of all patterns in the model, *2)* a statistical report summarising counts for various groups of patterns, and *3)* a histogram of pattern frequencies:

```
$ colibri-patternmodeller \
--inputmodel corpus.colibri.patternmodel \
--classfile corpus.tok.colibri.cls \
--print --report --histogram
```

This step could have been combined with the previous one as well.

We again urge the reader to consult the documentation and tutorials for a more extensive description of all available options and use cases, as this example only covers one out of many use cases, and further in-depth examples fall beyond the scope of this paper.

Support

Support, including but not limited to bug reports, feature requests and general pleas for assistance, is provided through the Github issue tracker at https://github.com/proycon/colibri-core/issues. As such, the archive of issues is always publicly consultable.

2.4 Availability

Operating System

Colibri Core should be able to run on modern POSIX-compliant operating systems, including Linux, FreeBSD and Mac OS X. It is tested to compile with current versions of both `gcc` as well as `clang`.

Programming Language

Colibri Core is written in C++, adhering to the C++11 standard. The Python binding is written in Cython (0.23 or above) and supports both Python 2.7 as well as Python 3. The latter is recommended.

The Python binding ensures that the functionality of Colibri Core is easily accessible from Python without sacrificing the great performance benefit native code provides. Python was chosen as it is a high-level programming language in widespread use in the scientific

[13] Use the `--unindexed` flag to build an unindexed model

community, and the NLP community in particular. It demands less expertise from the developer than C++ and is more suitable for rapid prototyping.

Additional system requirements

Colibri Core provides memory-based techniques where models are held entirely in memory to guarantee maximum performance on lookup and computation. This approach can be contrasted to e.g. database approaches which have much higher latency. It does place considerable memory requirements on the system on which is it run, though this depends entirely on the size of the data and the thresholds the user uses. We recommend at least 16GB RAM. In practise, using Colibri Core on high-end computing servers with 256GB RAM is not uncommon for extensive computation on big data sets.

Colibri Core is single-threaded due to the non-distributable nature of most of the algorithms. A 64-bit architecture is required, 32-bit is not supported.

Dependencies

Colibri Core relies on the standard C/C++ library and a full build environment including autoconf and automake; Python 2.7 or 3; Cython 0.23 or above. Support for reading the *FoLiA* XML format for text is entirely optional and requires the `libfolia` library.[14]

List of contributors

Developed by Maarten van Gompel, with contributions and feedback from Louis Onrust and Antal van den Bosch (*Centre for Language Studies, Radboud University Nijmegen*).

Software Location

Archive

Name: Zenodo
Persistent Identifier: https://dx.doi.org/10.5281/zenodo.55641
Publisher: Maarten van Gompel
Licence: GNU General Public Licence v3
Date published: June 15th, 2016 (v2.4.1)

[14] https://proycon.github.io/folia

Code repository

Name: GitHub
Identifier: https://github.com/proycon/colibri-core
Website: https://proycon.github.io/colibri-core
Licence: GNU General Public Licence v3
Date published: since September 21st, 2013, latest release at the time of writing is v2.4.1 (June 15th, 2016)

Language

English

2.5 Reuse potential

Colibri Core explicitly aims to provide a foundation for researchers in the NLP community to build their tools and research on. The software is already being employed in ongoing research on Machine Translation[15], Bayesian Language Modelling[16], Kneser-Ney Language Modelling[17], spelling correction[18], and event prediction in social media streams[19]. This has culminated in the publication of several studies that use Colibri Core (van Gompel and van den Bosch, 2014; Onrust et al., 2016; Kunneman and van den Bosch, 2016).

As a programming library for both C++ and Python, Colibri Core can be potentially adopted by a wide variety of third party developers. As a set of tools and scripts, Colibri Core also has merit standalone. It is, however, focused on command-line usage and therefore still requires a certain technical expertise from the end-user.

To increase the accessibility of Colibri Core, a RESTful webservice as well as generic web interface is already provided through CLAM (van Gompel and Reynaert, 2014). With this we hope to meet the needs of less technical end-users, as well as automated networked clients. This webservice is hosted at https://webservices-lst.science.ru.nl.

Future work building upon Colibri Core may focus on offering high-level user-interfaces to reach a wider audience, and on further improvement of its performance.

[15] https://github.com/proycon/colibri-mt
[16] https://github.com/naiaden/cococpyp
[17] https://github.com/naiaden/apodiformes
[18] https://github.com/proycon/gecco
[19] https://github.com/fkunneman/ADNEXT_predict

3

CROSS-LINGUAL WORD SENSE DISAMBIGUATION

In this chapter we begin to investigate the use of source-language context to disambiguate a marked word in context; so our focus is a localised one. In contrast to later chapters, we also take into account some linguistic features.

We apply k-nearest neighbour classifiers to two similar and localised translation tasks organized for SemEval 2010: Cross-Lingual Word Sense Disambiguation and Cross-Lingual Lexical Substitution. Both tasks aim to disambiguate a marked word in context by finding an appropriate translation for it in another language. Our system, UvT-WSD1 participates for the two tasks and achieves winning scores for the former. With a later rewrite of our system, called WSD2, we focus on hyperparameter optimisation and participate again in the Cross-Lingual Word-Sense Disambiguation task for SemEval 2013. We then achieve winning scores for four of the five language pairs, when asked to predict the best translation(s). Our final results, however, indicate that hyperparameter optimisation did not lead to the best results, indicating overfitting by our optimisation method in this aspect. Feature selection does have a modest positive impact.

This chapter is based on the following two publications:
van Gompel, M. (2010). UvT-WSD1: A cross-lingual word sense disambiguation system. In *SemEval '10: Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 238–241, Morristown, NJ, USA. Association for Computational Linguistics
van Gompel, M. and van den Bosch, A. (2013). WSD2: Parameter optimisation for memory-based cross-lingual word-sense disambiguation. In *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval 2013), in conjunction with the Second Joint Conference on Lexical and Computational Semantics*

3.1 Introduction

The UvT-WSD1 system we introduce here took part in two similar SemEval 2010 tasks: Cross-Lingual Word Sense Disambiguation (Lefever and Hoste, 2010) and Cross-Lingual Lexical Substitution (Mihalcea et al., 2010). In each task, a number of polysemous words is selected for which the senses are to be determined for a number of instances of these words. For each word, a number of samples in context is provided, where each sample consists of one sentence, with the word to be disambiguated marked. An example adapted from Lefever and Hoste (2010) is shown below:

- INPUT: *"...giving fish to the people living on **the bank** of the river"*
  SENSE LABELS:

  Dutch: *oever/dijk*

  French: *rives/rivage/bord/bords*

  German: *Ufer*

  Italian: *riva*

  Spanish: *orilla*

- INPUT: *"The **bank** of Scotland ..."*
  SENSE LABELS:

  Dutch: *bank/kredietinstelling*

  French: *banque/établissement de crédit*

  German: *Bank/Kreditinstitut*

  Italian: *banca*

  Spanish: *banco*

Because of the cross-lingual nature of the tasks, a word sense corresponds to a translation in another language, rather than a sense description in the same language. In the Cross-lingual Lexical Substitution task, the target language is Spanish. The task is to find Spanish substitutes for the English words marked in the test samples. In the Cross-Lingual Word Sense Disambiguation task, we participate for English-Dutch and English-Spanish. The Word Sense Disambiguation task provides training data for all five languages, in the form of the sentence-aligned Europarl parallel corpus (Koehn, 2005). This is the source of training data our UvT-WSD1 system uses for both tasks.

The system may output several senses per instance, rather than producing just one sense prediction. These are evaluated in two different ways. The scoring type "**best**" expects that the system outputs the best senses, in the order of its confidence. The scoring type "**out of five/ten**" expects five or ten guesses, and each answer weighs the same. These metrics are more extensively described in Mihalcea et al. (2010). The UvT-WSD1 system participates in both scoring types, for both tasks. The system put forth in this paper follows a similar approach as described in earlier research by Hoste et al. (2002).

WSD2 is a rewrite and extension of our UvT-WSD1. In WSD2 we introduce and test a new level of hyperparameter optimisation. We now participate for all five target languages (Dutch, Spanish, Italian, French, and German). The task presents twenty polysemous nouns with fifty instances each to be mapped onto normalised (lemmatised) translations in all languages. The task is described in detail in Lefever and Hoste (2013a).

Trial data is provided and has been used to optimise system parameters. Due to the unsupervised nature of the task, no training data

is provided. However, given that the gold standard of the task is based exclusively on the Europarl parallel corpus (Koehn, 2005), we select that same corpus to minimise our chances of delivering translations that the human annotators preparing the test data could have never picked.

3.2 System Description

Our system uses machine learning techniques to learn what senses/-translations are associated with any of the target words. It does so on the basis of a variety of local and global context features, discussed in Section 3.2.2. At the core of the system are the classifiers, or so called "word experts", one per target word. These are built using the Tilburg Memory Based Learner (TiMBL) (Daelemans et al., 2009), making use of the IB1 algorithm, an implementation of the *k*-nearest neighbour classifier.

The core of the system can be subdivided into three stages. In the first stage, the word-aligned parallel corpus is read and for each found instance of one of the target words, features are extracted to be used in the classifier. The class consists of the word aligned to the found instance of the target word, i.e. the translation/sense. In this way a word expert is built for each of the target words in the task, yielding a total amount of classifiers equal to the total amount of target words. The test data is processed in a similar way, for each marked occurrence of any of the target words, features are extracted and test instances are created. Subsequently, the word experts are trained and tested, and on the basis of the training data, a parameter search algorithm (van den Bosch, 2004) determines the optimal set of classifier parameters for each word expert, including for example the value of *k* and the distance weighting metric used.

In the last stage, the classifier output of each word expert is parsed. The classifiers yield a distribution of classes per test instance, and these are converted to the appropriate formats for "best" and "out of five/ten" evaluation. For the latter scoring type, the five/ten highest scoring senses are selected, for the former scoring type, all classes scoring above a certain threshold are considered "best". The threshold is set at 90% of the score of the highest scoring class.

3.2.1 Word-Alignment, Tokenisation, Lemmatisation and Part-of-Speech-tagging

The Europarl parallel corpus, English-Spanish and English-Dutch, is delivered as a sentence-aligned parallel corpus. We subsequently run GIZA++ (Och and Ney, 2000) to compute a word-aligned parallel corpus.

This, however, is not the sole input. The target words in both tasks are actually specified as a lemma and part-of-speech tag pair, rather than words. In the Word Sense Disambiguation task, all target lemmas are simply nouns, but in the Cross-Lingual Lexical Substitution task, they can also be verbs, adjectives or adverbs. Likewise, both tasks expect the sense/translation output to also be in the form of lemmas. Therefore the system internally has to be aware of the lemma and part-of-speech tag of each word in the parallel corpus and test data, only then can it successfully find all occurrences of the target words. In order to get this information, both sides of the word-aligned parallel corpus are run through tokenisers, lemmatisers and Part-of-Speech taggers, and the tokenised output is realigned with the untokenised input so the word alignments are retained. The test data is also processed this way. For English and Spanish, the software suite Freeling (Atserias et al., 2006) performed all these tasks, and for Dutch it was done by Tadpole (van den Bosch et al., 2007a), a previous version of what is currently called Frog.

3.2.2 Feature Extraction

The system can extract a variety of features to be used in training and testing. A distinction can be made between *local context features* and *global context features*. Local context features are extracted from the immediate neighbours of the occurrence of the target word. One or more of the following local context features are extractable by the UvT-WSD1 system: word features, lemma features, and part-of-speech tag features. In each case, n features both to the right and left of the focus word are selected. Moreover, the system also supports the extraction of bigram features, but these did not perform well in the experiments.

The global context features are made up of a bag-of-words representation of keywords that *may* be indicative for a given word to sense/translation mapping. The idea is that words are collected which have a certain power of discrimination for the specific target word with a specific sense, and all such words are then put in a bag-of-word representation, yielding as many features as the amount of keywords found. A global count over the full corpus is needed to find these keywords. Each keyword acts as a binary feature, indicating whether or not that particular keyword is found in the context of the occurrence of the target word. The context in which these keywords are searched for is exactly one sentence, i.e. the sentence in which the target word occurs. This is due to the test data simply not supplying a wider context.

The method used to extract these keywords (k) is proposed by Ng and Lee (1996) and used also in the research of Hoste et al. (2002). Assume we have a focus word f, more precisely, a lemma and part-of-speech tag pair of one of the target words. We also have one of its

aligned translations/senses s, which in this implementation is also a lemma. We can now estimate $P(s|k)$, the probability of sense s, given a keyword k, by dividing $N_{s,k_{local}.}$ (the number of occurrences of a possible local context word k with particular focus word lemma-PoS combination and with a particular sense s) by $N_{k_{local}}$ (the number of occurrences of a possible local context keyword k_{loc} with a particular focus word-PoS combination regardless of its sense). If we also take into account the frequency of a possible keyword k in the complete training corpus ($N_{k_{corpus}}$), we get:

$$P(s|k) = \frac{N_{s,k_{local}}}{N_{k_{local}}}(\frac{1}{N_{k_{corpus}}}) \tag{3.1}$$

Hoste et al. (2002) select a keyword k for inclusion in the bag-of-words representation if that keyword occurs more than T_1 times in that sense s, and if $P(s|k) \geq T_2$. Both T_1 and T_2 are predefined thresholds, which by default were set to 3 and 0.001 respectively. In addition, UvT-WSD1 contains an extra parameter which can be enabled to automatically adjust the T_1 threshold when it yields too many or too few keywords. The selection of bag-of-word features is computed prior to the extraction of the training instances, as this information is a prerequisite for the successful generation of both training and test instances.

3.3 UvT-WSD1

3.3.1 Voting system

The local and global context features, and the various parameters that can be configured for extraction, yield a lot of possible classifier combinations. Rather than merging all local context and global context features together in a single classifier, they can also be split over several classifiers and have an arbiter voting system do the final classification step. UvT-WSD1 also supports this approach. A voter is constructed by taking as features the class output of up to three different classifiers, trained and tested on the training data, and mapping these features onto the actual correct sense in the training data. For testing, the same approach is taken: up to three classifiers run on the test data; their output is taken as feature vector, and the voting system predicts a sense. This approach may be useful in boosting results and smoothing out errors. In our experiments we see that a voter combination often performs better than taking all features together in one single classifier. Finally, also in the voter system there is a stage of automatic parameter optimisation for TiMBL.

Feature configuration	Accuracy
Voter: word2+lemma2 & global	**14.45**
word2+lemma2	14.09
word2	13.30
word2+global	13.09
global	12.78
word3	12.68
word1	12.61
word2+pos2+lemma2	11.03
baseline	10.82

*Table 3.1: Average scores for various configurations on the trial data of the English-*Dutch *Word Sense Disambiguation task, averaged over five target words. At the time this experiment was conducted, a bug existed in the output format specification, resulting in slightly lower scores.*

3.3.2 Experiments and Results

Both SemEval 2010 tasks have provided trial data upon which the system could be tested during the development stage. Considering the high configurability of the various parameters for feature extraction, the search space in possible configurations and classifier parameters is vast, also due to fact that the TiMBL classifier used may take a wealth of possible parameters. As already mentioned, for the latter an automatic algorithm of parameter optimisation was used (van den Bosch, 2004), but optimisation of the feature extraction parameters has not been automated. Rather, a selection of configurations has been manually chosen and tested during the development stage

In Table 3.1 some of the results on the *trial data* are listed, for the Dutch Word Sense Disambiguation task only. All scores are *averaged* over the five target words in the trial data. Included is also a baseline that has been computed by selecting simply the most common translation/sense for each target word, regardless of the context it appears in.

In Table 3.1,"word" refers to the surface forms of the words, "lemma" denotes lemma features, "pos" denotes Part-of-Speech information. The subsequent number refers to the local context size. The "global" configuration refers to global context features.

The following two configurations of features were assembled for submission to the final shared task:

1. **UvT-WSD1-v** (aka *UvT-v*) – An arbiter-voting system over three classifiers: 1) Word experts with two word features and lemma features on both sides of the focus word. 2) Word experts with

BEST	UvT-WSD1-v	UvT-WSD1-g
Precision & Recall	21.09	19.59
Mode Prec. & Rec.	43.76	41.02
Ranking (out of 14)	6	9
OUT OF TEN	UvT-WSD1-v	UvT-WSD1-g
Precision & Recall	58.91	55.29
Mode Prec. & Rec.	62.96	73.94
Ranking	3	4

Table 3.2: UvT-WSD1 results in the Cross-Lingual Lexical Substitution task

Dutch BEST	UvT-v	UvT-g	T3-COLEUR		
Precision & Recall	17.7	15.93	10.72 & 10.56		
Mode Prec. & Rec.	12.06	10.54	6.18 & 6.16		
Dutch OUT OF FIVE	UvT-v	UvT-g	T3-COLEUR		
Precision & Recall	34.95	34.92	21.54 & 21.22		
Mode Prec. & Rec.	24.62	19.72	12.05 & 12.03		
Spanish BEST	UvT-v	UHD-1	UvT-g	T3-COLEUR	FCC-WSD1
Precision & Recall	23.42	20.48 & 16.33	19.92	19.78 & 19.59	15.09
Mode Prec. & Rec.	24.98	28.48 & 22.19	24.17	24.59	14.31
Spanish OUT OF FIVE	UvT-g	UvT-v	FCC-WSD2	UHD-1	T3-COLEUR
Precision & Recall	43.12	42.17	40.76	38.78 & 31.81	35.84 & 35.46
Mode Prec. & Rec.	43.94	40.62	44.84	40.68 & 32.38	39.01 & 38.78

Table 3.3: UvT-WSD1 results in comparison to other participants in the Word-Sense Disambiguation task. Only whenever two values are given, precision and recall are not equal.

global features[1]. 3) Word experts with two word features, two lemma features *and* two part-of-speech tag features.

2. **UvT-WSD1-g** (aka *UvT-g*) – Word experts with global features only.

The UvT-v configuration was selected for its best performance of the trial chance[2]. The UvT-g configuration was included to judge the efficacy of global context features when used standalone.

Table 3.2 shows a condensed view of the results for the Cross-Lingual Lexical Substitution task. Table 3.3 shows the final results for the Word-Sense Disambiguation task. Note that UvT-WSD1-v and UvT-WSD1-g are two different configurations of the UvT-WSD1 system, and to conserve space these are abbreviated as UvT-v and UvT-g respectively. These are also the names used in both tasks (Lefever and Hoste, 2010; Mihalcea et al., 2010) to refer to our system.

[1] For the Cross-Lingual Lexical Substitution task only, the parameter to recompute the T_1 threshold automatically was enabled.

[2] An extra configuration with PoS information was included in the voter on the odd chance it can make a difference, despite the poor performance observed in Table 3.1

3.4 WSD2

3.4.1 Feature and Hyperparameter Optimisation

The size of the local context, the inclusion of global context features, and the inclusion of syntactic features are all features that can be selected, changed, or disabled, allowing for a variety of combinations to be tested. In addition, each word expert is a k-nearest neighbour classifier that can take on many hyperparameters beyond k. For our participation in the SemEval 2013 Cross-Lingual Word Sense Disambiguation task, with our WSD2 system, we performed both optimisations for all word experts. The optimisations, however, were performed independently to reduce complexity: we optimised classifier hyperparameters on the basis of the training examples extracted from our parallel corpus, producing optimal accuracy on each word-expert. We optimised feature selection on the basis of the trial data provided for the task. As has been argued before (Hoste et al., 2002), the joint search space of feature selection and hyperparameters is prohibitively large. Our current setup runs the risk of finding hyperparameters that are not optimal for the feature selection in the second optimisation step. Our final results indeed show that only feature selection produced improved results. We choose the feature selection with the highest score on the trial set, for each of the nouns and separately for both evaluation metrics in the task.

To optimise the choice of hyperparameters per word expert, a heuristic parameter search algorithm (van den Bosch, 2004)[3] was used that implements wrapped progressive sampling using cross-validation: it performs a large number of experiments with many hyperparameter setting combinations on small samples of training data, and then progressively zooms in on combinations estimated to perform well with larger samples of the training data. As a control run we also trained word experts with default hyperparameters, i.e. with $k = 1$ and with all other hyperparameters at their default values as specified in the TiMBL implementation.

3.4.2 Experiments & Results

To assess the accuracy of a certain configuration of our system as a whole, we take the average over all word experts. An initial experiment on the trial data explores the impact of different context sizes, with hyperparameter optimisation on the classifiers. The results, shown in Figure 3.1, clearly indicate that on average the classifiers perform best with a local context of just one word to the left and one to the right of the word to be disambiguated. Larger context sizes have a negative

[3] https://github.com/LanguageMachines/paramsearch

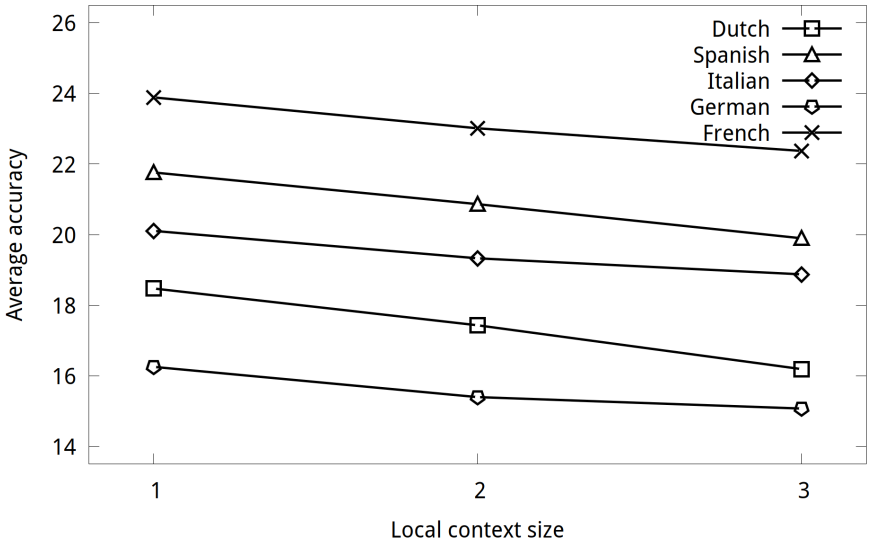

Figure 3.1: Average accuracy for different local context sizes

impact on average accuracy. These tests include hyperparameter optimisation, but the same trend shows without.

It is interesting to put this in contrast with the results from UvT-WSD1 in Table 3.1, where a context size of two leads to better results than context size one. This can only be explained by the difference in trial data.

We submitted three configurations of our system to the shared task, the maximum number of runs. Adding lemma features to the local context window of three words, i.e. left word, focus word and right word, proves beneficial in general. This is shown in Table 3.4. This is therefore the first configuration we submitted (`c1l`). As second configuration (`c1lN`) we submitted the same configuration without parameter optimisation on the classifiers. Note that neither of these include global context features.

The third configuration (`var`) we submitted includes feature selection, and selects per word expert the configuration that has the highest score on the trial data, and thus tests all kinds of configurations. Note that hyperparameter optimisation is also enabled for this configuration. Due to the feature selection on the trial data, we by definition obtain the highest scores on this trial data, but this carries the risk of overfitting. Results on the trial data are shown in Table 3.5.

The hyperparameter optimisation on classifier accuracy has a slightly negative impact, suggesting overfitting on the training data. Therefore a fourth configuration (`varN`) was tried later to independently assess the idea of feature selection, without hyperparameter optimisation on the classifiers. This proves to be a good idea. However,

BEST	ES	FR	IT	NL	DE
baseline	19.65	21.23	15.17	15.75	13.16
plain	21.76	23.89	20.10	18.47	16.25
+lem (c1l)	21.88	**23.93**	**19.90**	**18.61**	**16.43**
+pos	22.09	23.91	19.95	18.02	15.37
lem+pos	**22.12**	23.61	19.82	18.18	15.48
glob.context	20.57	23.34	17.76	17.06	16.05
OUT-OF-5	**ES**	**FR**	**IT**	**NL**	**DE**
baseline	48.34	45.99	34.51	38.59	32.90
plain	49.81	**50.91**	42.30	41.74	**36.86**
+lem (c1l)	**49.91**	50.65	**42.41**	**41.83**	36.45
+pos	47.86	49.72	41.91	41.31	35.93
lem+pos	47.90	49.75	41.49	41.31	35.80
glob.context	48.09	49.68	40.87	37.70	34.47

Table 3.4: Feature exploration on the trial data

BEST	ES	FR	IT	NL	DE
c1lN	22.60	24.09	19.87	18.70	16.43
c1l	21.88	23.93	19.90	18.61	16.43
var	23.79	**25.66**	**21.65**	**20.19**	**19.06**
varN	**23.90**	25.65	21.52	19.92	18.96
OUT-OF-5	**ES**	**FR**	**IT**	**NL**	**DE**
c1lN	50.14	50.98	42.92	42.08	36.45
c1l	49.91	50.65	42.41	41.83	36.45
var	51.95	**53.66**	45.59	**44.66**	**39.81**
varN	**52.91**	53.61	**45.92**	44.32	39.40

Table 3.5: Results on the trial data

the fourth configuration was not yet available for the actual competition. This incidentally would have had no impact on the final ranking between competitors. When we run these systems on the actual test data of the shared task, we obtain the results in Table 3.6. The best score amongst the other competitors is mentioned in the last row for reference, this is the HLTDI team (Rudnick et al., 2013) for all but Best-Spanish, which goes to the NRC contribution (Carpuat, 2013).

A major factor in this task is the accuracy of lemmatisation, and to lesser extent of PoS tagging. We conducted additional experiments on German and French without lemmatisation, tested on the trial data. Results immediately fell below baseline.

Another main factor is the quality of the word alignments, and the degree to which the found word alignments correspond with the translations the human annotators could choose from in preparing the gold standard. An idea we tested is, instead of relying on the mere intersection of word alignments, to use a phrase-translation table generated by and for the Statistical Machine Translation system Moses (Koehn et al., 2007a), which uses the grow-diag-final heuristic to extract phrase pairs. This results in more phrases, and whilst this

BEST	ES	FR	IT	NL	DE
baseline	23.23	25.74	20.21	20.66	17.42
c1l	28.40	29.88	25.43	23.14	20.70
c1lN	28.65	30.11	**25.66**	**23.61**	20.82
var	23.3	25.89	20.38	17.17	16.2
varN	29.05	**30.15**	24.90	23.57	**21.98**
best competitor	**32.16**	28.23	24.62	22.36	19.92
OUT-OF-5	**ES**	**FR**	**IT**	**NL**	**DE**
baseline	53.07	51.36	42.63	43.59	38.86
c1l	58.23	59.07	52.22	**47.83**	43.17
c1lN	57.62	**59.80**	52.73	47.62	43.24
var	55.70	59.19	51.18	46.85	41.46
varN	58.61	59.26	50.89	50.42	43.34
best competitor	**61.69**	58.20	**53.57**	46.55	**43.66**

Table 3.6: Results on the test set

is a good idea for MT, in the current task it has a detrimental effect, as it creates too many translation options and we do not have an MT decoder to discard ineffective options in this task. The grow-diag-final heuristic incorporates unaligned words to the end of a translation in the translation option, a bad idea for CLWSD.

3.5 Discussion and Conclusion

In our UvT-WSD1 system we used the same configuration of feature extraction, or a voter over a predetermined set of configurations, for all word experts. The actual classifier parameters however, do differ per word expert, as they are the result of the automatic parameter optimisation algorithm. Our WSD2 system takes parameter optimisation one step further by selecting system parameters per word expert from the best configurations on the trial data. Optimising the hyperparameters of the classifiers on the training data proves to have a slightly negative effect though, especially when combined with the selection of features. This is likely due to the fact that feature selection was performed after hyperparameter optimisation, causing certain optimisations to be rendered ineffective.

Keeping in mind the fact that different word experts may perform differently, some *general* conclusions can be drawn from the experiments on the trial data. It appears to be beneficial to include lemma features, rather than just word features. Both the 2010 and 2013 tasks corroborate this. However, adding Part-of-Speech features tends to have a negative impact. For these local context features, the optimum context size is often two features to the left and two features to the right of the focus word, cf. Hendrickx et al. (2002).

The global keyword features perform well for UvT-WSD1, but best results are achieved if they are not mixed with the local context features in one classifier. For WSD2 in the 2013 task, however, the global context features make less an impact than even just the linguistically-uninformed local context feature, as became clear from the the feature exploration stage in Table 3.4. As the keyword selection algorithm is the same as in UvT-WSD1, this is surprising and must then be attributed to the dataset. Taking into account the earlier discrepancy between the results of UvT-WSD1 and WSD2, we may conclude that optimal configurations differ accross datasets.

For UvT-WSD1, an arbiter voting approach over multiple classifiers helps to smooth out errors and yields the highest scores (see Tables 3.2 and 3.3). When compared to the other participants, the UvT-WSD1 system in the voting configuration ranks first in the Word Sense Disambiguation task, for the two language pairs in which we participated. WSD2 does not use a voting approach, as it already explicitly attempts to assess the efficacy of various hyperparameter optimisations and feature selections.

When asked to predict the best translation(s), our WSD2 system comes out on top for four out of five languages in the SemEval 2013 task; only for Spanish we are surpassed by two competitors. Our out-of-five predictions win for two out of five languages, and are fairly close the the best competitor for the others, except again for Spanish. It is interesting to observe that, due to our feature selection without hyperparameter optimisation on the classifier not being available yet at the time of submission, our simplest system `c1lN` emerged as best in the contest.

We assumed independence between hyperparameter optimisation and feature selection, where the former was conducted using cross-validation on the training data rather than on the development set. As this independence assumption is a mere simplification to reduce algorithmic complexity, future research could focus on a more integrated approach and test hyperparameter optimisation of the classifiers on the trial set which may produce better scores.

The WSD2 system is available as open-source under the GNU Public License v3. It is implemented in Python (van Rossum, 2006) and can be obtained from `https://github.com/proycon/wsd2`[4]. The experimental data and output of our participation in the 2013 SemEval task are included in this git repository as well.

[4] git commit f10e796141003d8a2fbaf8c463588a6d7380c05e represents a fair state of the system at the time of submission

4

TRANSLATION ASSISTANCE BY TRANSLATION OF L1 FRAGMENTS IN AN L2 CONTEXT

In this chapter we present new research in translation assistance. We describe a pilot system capable of translating native language (L1) fragments to foreign language (L2) fragments in an L2 context. Practical applications of this research can be framed in the context of second language learning. The type of translation assistance system under investigation here encourages language learners to write in their target language while allowing them to fall back to their native language in case the correct word or expression is not known. These code switches are subsequently translated to L2 given the L2 context. We study the feasibility of exploiting cross-lingual context to obtain high-quality translation suggestions that improve over statistical language modelling and word-sense disambiguation baselines. A classification-based approach is presented that is indeed found to improve significantly over these baselines by making use of a contextual window spanning a small number of neighbouring words.

This chapter is based on: van Gompel, M. and van den Bosch, A. (2014). Translation assistance by translation of L1 fragments in an L2 context. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 871–880, Baltimore, Maryland. Association for Computational Linguistics

4.1 Introduction

Whereas machine translation generally concerns the translation of whole sentences or texts from one language to the other, this study focusses on the translation of native language (henceforth L1) words and phrases, i.e. smaller fragments, in a foreign language (L2) context. Despite the major efforts and improvements, automatic translation does not yet rival human-level quality. Vexing issues are morphology, word-order change and long-distance dependencies. Although there is a morpho-syntactic component in this research, our scope is more constrained; its focus is on the faithful preservation of meaning from L1 to L2, akin to the role of the translation model in Statistical Machine Translation (SMT).

The cross-lingual context in our research question may at first seem artificial, but its design explicitly aims at applications related to computer-aided language learning (Laghos and Panayiotis, 2005; Levy, 1997) and computer-aided translation (Barrachina et al., 2009). Currently, language learners need to refer to a bilingual dictionary

when in doubt about a translation of a word or phrase. Yet, this problem arises in a context, not in isolation; the learner may have already translated successfully a part of the text into L2 leading up to the problematic word or phrase. Dictionaries are not the best source to look up context; they may contain example usages, but remain biased towards single words or short expressions.

The proposed application allows code switching and produces context-sensitive suggestions as writing progresses. In this research we test the feasibility of the foundation of this idea.The following examples serve to illustrate the idea and demonstrate what output the proposed translation assistance system would ideally produce. The parts in bold correspond to respectively the inserted fragment and the system translation.

- Input (L1=English,L2=Spanish): *"Hoy vamos a **the swimming pool**."*
  Desired output: *"Hoy vamos a **la piscina**."*
- Input (L1-English, L2=German): *"Das wetter ist wirklich **abominable**."*
  Desired output: *"Das wetter ist wirklich **ekelhaft**."*
- Input (L1=French,L2=English): *"I **rentre à la maison** because I am tired."*
  Desired output: *"I **return home** because I am tired."*
- Input (L1=Dutch, L2=English): *"Workers are facing a massive **aanval op** their employment and social rights."*
  Desired output: *"Workers are facing a massive **attack on** their employment and social rights."*

The main research question in this research is how to disambiguate an L1 word or phrase to its L2 translation based on an L2 context, and whether such cross-lingual contextual approaches provide added value compared to baseline models that are not context informed or compared to standard language models.

4.2 Data preparation

Preparing the data to build training and test data for our intended translation assistance system is not trivial, as the type of interactive translation assistant we aim to develop does not exist yet. We need to generate training and test data that realistically emulates the task. We start with a parallel corpus that is tokenised for both L1 and L2. No further linguistic processing such as part-of-speech tagging or lemmatisation takes place in our experiments; adding this remains open for future research.

1. using phrase-translation table T and parallel corpus split S
2. **for** each aligned sentence pair $(sentence_s \in S_s, sentence_t \in S_t)$ in the parallel corpus split (S_s,S_t):
3. **for** each fragment $(f_s \in sentence_s, f_t \in sentence_t)$ where $(f_s, f_t) \in T$:
4. **if** $P(f_s|f_t) \cdot P(f_t|f_s) \geq \lambda_1$ **and** $P(f_s|f_t) \cdot P(f_t|f_s) \geq \lambda_2 \cdot P(f_s|f_{strongest\_t}) \cdot P(f_{strongest\_t}|f_s)$:
5. Output a pair $(sentence'_t, sentence_t)$ where $sentence'_t$ is a copy of t but with fragment f_t substituted by f_s, i.e. the introduction of an L1 word or phrase in an L2 sentence.

Figure 4.1: Algorithm for extracting training and test data on the basis of a phrase-translation table (T) and subset/split from a parallel corpus (S). The indentation indicates the nesting.

The parallel corpus is randomly sampled into two large and equally-sized parts. One is the basis for the training set, and the other is the basis for the test set. The reason for such a large test split shall become apparent soon.

From each of the splits (S), a phrase-translation table is constructed automatically in an unsupervised fashion. This is done using the scripts provided by the Statistical Machine Translation system Moses (Koehn et al., 2007b). It invokes GIZA++ (Och and Ney, 2000) to establish statistical word alignments based on the IBM Models and subsequently extracts phrases using the `grow-diag-final` algorithm (Och and Ney, 2003). The result, independent for each set, will be a phrase-translation table (T) that maps phrases in L1 to L2. For each phrase-pair (f_s, f_t) this phrase-translation table holds the computed translation probabilities $P(f_s|f_t)$ and $P(f_t|f_s)$.

Given these phrase-translation tables, we can now extract both training data and test data using the algorithm in Figure 4.1. In our discourse, the source language (s) corresponds to L1, the fallback language used for by the end-user for inserting fragments, whilst the target language (t) is L2.

Step 4 is effectively a filter: two thresholds can be configured to discard weak alignments, i.e. those with low probabilities, from the phrase-translation table so that only strong couplings make it into the generated set. The parameter λ_1 adds a constraint based on the

product of the two conditional probabilities $(P(f_t|f_s) \cdot P(f_s|f_t))$, and sets a threshold that has to be surpassed. A second parameter λ_2 further limits the considered phrase pairs (f_s, f_t) to have the product of their conditional probabilities not deviate more than a fraction λ_2 from the joint probability for the strongest possible pairing for f_s, the source fragment. In Figure fig:algo, $f_{strongest\_t}$ corresponds to the best scoring translation for a given source fragment f_s. This metric thus effectively prunes weaker alternative translations in the phrase-translation table from being considered if there is a much stronger candidate. Nevertheless, it has to be noted that even with λ_1 and λ_2, the test set will include a certain amount of errors. This is due to the nature of the unsupervised method with which the phrase-translation table is constructed. For our purposes however, the test set suffices to test our hypothesis.

In our experiments, we choose fixed values for these parameters, by manual inspection and judgement of the output. The λ_1 parameter was set to 0.01 and λ_2 to 0.8. Whilst other thresholds may possibly produce cleaner sets, this is hard to evaluate as finding optimal values causes a prohibitive increase in complexity of the search space, and again this is not necessary to test our hypothesis.

The output of the algorithm in Figure 4.1 is a modified set of sentence pairs $(sentence'_t, sentence_t)$, in which the same sentence pair may be used multiple times with different L1 substitutions for different fragments. The final test set is created by randomly sampling the desired number of test instances.

Note that the training set and test set are constructed on their own respective and independently generated phrase-translation tables. This ensures complete independence of training and test data. Generating test data using the same phrase-translation table as the training data would introduce a bias. The fact that a phrase-translation table needs to be constructed for the test data is also the reason that the parallel corpus split from which the test data is derived has to be large enough, ensuring better quality.

We concede that our current way of testing is a mere approximation of the real-world scenario. An ideal test corpus would consist of L2 sentences with L1 fallback as crafted by L2 language learners with an L1 background. However, such corpora do not exist yet at the time of this research. Nevertheless, we hope to show that our automated way of test set generation is sufficient to test the feasibility of our core hypothesis that L1 fragments can be translated to L2 using L2 context information.

4.3 System

We develop a classifier-based system composed of so-called "classifier experts". Numerous classifiers are trained and each is an expert in

translating a single word or phrase. In other words, for each word type or phrase type that occurs as a fragment in the training set, and which does not map to just a single translation, a classifier is trained. The classifier maps the L1 word or phrase in its L2 context to its L2 translation. Words or phrases that always map to a single translation are stored in a simple mapping table, as a classifier would have no added value in such cases. The classifiers use the IB1 algorithm (Aha et al., 1991) as implemented in TiMBL (Daelemans et al., 2009).[1] IB1 implements *k*-nearest neighbour classification. The choice for this algorithm is motivated by the fact that it handles multiple classes with ease, but first and foremost because it has been successfully employed for word sense disambiguation in other studies (Hoste et al., 2002; Decadt et al., 2004), in particular in cross-lingual word sense disambiguation as seen in Chapter 3. It has also been used in machine translation studies in which local source context is used to classify source phrases into target phrases, rather than looking them up in a phrase table (Stroppa et al., 2007; Haque et al., 2011b). The idea of local phrase selection with a discriminative machine learning classifier using additional local (source-language) context was introduced in parallel to Stroppa et al. (2007) by Carpuat and Wu (2007a) and Giménez and Màrquez (2007); cf. Haque et al. (2011b) for an overview of more recent methods.

The feature vector for the classifiers represents a local context of neighbouring words, and optionally also global context keywords in a binary-valued bag-of-words configuration. The local context consists of an X number of L2 words to the left of the L1 fragment, and Y words to the right.

When presented with test data, in which the L1 fragment is explicitly marked, we first check whether there is ambiguity for this L1 fragment and if a direct translation is available in our simple mapping table. If so, we are done quickly and need not rely on context information. If not, we check for the presence of a classifier expert for the offered L1 fragment; only then we can proceed by extracting the desired number of L2 local context words to the immediate left and right of this fragment and adding those to the feature vector. The classifier will return a probability distribution of the most likely translations given the context and we can replace the L1 fragment with the highest scoring L2 translation and present it back to the user.

In addition to local context features, we also experimented with global context features. These are a set of L2 contextual keywords for each L1 word/phrase and its L2 translation occurring in the same sentence, not necessarily in the immediate neighbourhood of the L1 word/phrase. The keywords are selected to be indicative for a specific translation. We used the method of extraction by Ng and Lee (1996), as we have seen in Chapter 3, and encoded all keywords in a binary

[1] https://languagemachines.github.io/timbl

bag of words model. The experiments however showed that inclusion of such keywords did not make any noticeable impact on any of the results, so we restrict ourselves to mentioning this negative result.

Our full system, including the scripts for data preparation, training, and evaluation, is implemented in Python and freely available as open-source from `https://github.com/proycon/colibrita/` . Version tag `v0.2.1` is representative for the version used in this research.

4.3.1 Language Model

We also implement a statistical language model as an optional component of our classifier-based system and also as a baseline to compare our system to. The language model is a trigram-based back-off language model with Kneser-Ney smoothing, computed using SRILM (Stolcke, 2002) and trained on the same training data as the translation model. No additional external data was brought in, to keep the comparison fair.

For any given hypothesis H, results from the L1 to L2 classifier are combined with results from the L2 language model. We do so by normalising the class probability from the classifier ($score_T(H)$), which is our translation model, and the language model ($score_{lm}(H)$), in such a way that the highest classifier score for the alternatives under consideration is always 1.0, and the highest language model score of the sentence is always 1.0. Take $score_T(H)$ and $score_{lm}(H)$ to be log probabilities, the search for the best (most probable) translation hypothesis $\hat{H}$ can then be expressed as:

$$\hat{H} = \arg\max_{H}(score_T(H) + score_{lm}(H)) \tag{4.1}$$

If desired, the search can be parametrised with variables λ_3 and λ_4, representing the weights we want to attach to the classifier-based translation model and the language model, respectively. In the current study we simply left both weights set to one, thereby assigning equal importance to translation model and language model.

4.4 Evaluation

Several automated metrics exist for the evaluation of L2 system output against the L2 reference output in the test set. We first measure absolute accuracy by simply counting all output fragments that exactly match the reference fragments, as a fraction of the total amount of fragments. This measure may be too strict, so we add a more flexible *word accuracy* measure which takes into account partial matches at the word level. If output o is a subset of reference r then a score of $\frac{|o|}{|r|}$ is assigned for that sentence pair. If instead, r is a subset of o, then a score of $\frac{|r|}{|o|}$ will be assigned. A perfect match will result in a score

of 1 whereas a complete lack of overlap will be scored 0. The word accuracy for the entire set is then computed by taking the sum of the word accuracies per sentence pair, divided by the total number of sentence pairs.

We also compute a recall metric that measures the number of fragments that the system provided a translation for as a fraction of the total number of fragments in the input, regardless of whether the fragment is translated correctly or not. The system may skip fragments for which it can find no solution at all.

In addition to these, the system's output can be compared against the L2 reference translation(s) using established Machine Translation evaluation metrics. We report on BLEU, NIST, METEOR, and word error rate metrics WER and PER. These scores should generally be much better than the typical MT system performances as only local changes are made to otherwise "perfect" L2 sentences.

4.5 Baselines

A context-insensitive yet informed baseline was constructed to assess the impact of L2 context information in translating L1 fragments. The baseline selects the most probable L1 fragment per L2 fragment according to the phrase-translation table. This baseline, henceforth referred to as the 'most likely fragment' baseline (MLF) is analogous to the 'most frequent sense'-baseline common in evaluating WSD systems.

A second baseline was constructed by weighing the probabilities from the translation table directly with the L2 language model described earlier. It adds a LM component to the MLF baseline. This LM baseline allows the comparison of classification through L1 fragments in an L2 context, with a more traditional L2 context modelling (i.e. target language modelling) which is also customary in MT decoders. Computing this baseline is done in the same fashion as previously illustrated in Equation 4.1, where $score_T$ then represents the normalised $p(t|s)$ score from the phrase-translation table rather than the class probability from the classifier.

4.6 Experiments & Results

The data for our experiments were drawn from the Europarl parallel corpus (Koehn, 2005) from which we extracted two sets of 200,000 sentence pairs each for several language pairs. These were used to form the training and test sets. The final test sets are a randomly sampled 5,000 sentence pairs from the 200,000-sentence test split for each language pair.

All input data for the experiments in this section are publicly available[2].

Let us first zoom in to convey a sense of scale on a specific language pair. The actual Europarl training set we generate for English (L1) to Spanish (L2), i.e. English fallback in a Spanish context, consists of 5,608,015 sentence pairs. This number is much larger than the 200,000 we mentioned before because single sentence pairs may be reused multiple times with different marked fragments. From this training set of sentence pairs over 100,000 classifier experts are derived. The eleven largest classifiers are shown in Table 4.1, along with the number of training instances per classifier. The full table would reveal a Zipfian distribution.

Fragment	Training instances	Translations
the	256,772	la, el, los, las
of	139,273	de, del
and	128,074	y, de, e
to	66,565	a, para, que, de
a	54,306	un, una
is	40,511	es, está, se
for	34,054	para, de, por
this	29,691	este, esta, esto
European	26,543	Europea, Europeo Europeas, Europeos
on	23,147	sobre, en
of the	22,361	de la, de los

Table 4.1: The top eleven classifier experts for English to Spanish. The eleventh entry is included as an example of a common phrasal fragment

Among the classifier experts are only words and phrases that are ambiguous and may thus map to multiple translations. This implies that such words and phrases must have occurred at least twice in the corpus, though this threshold is made configurable and could have been set higher to limit the number of classifiers. The remaining 246,380 unambiguous mappings are stored in a separate mapping table.

For the classifier-based system, we tested various different feature vector configurations. The first experiment, of which the results are shown in Figure 4.2, sets a fixed and symmetric local context size across all classifiers, and tests three context widths. Here we observe that a context width of one yields the best results. The BLEU scores, not included in the figure but shown in Table 4.2, show a similar trend. This trend holds for all the MT metrics.

Table 4.2 shows the results for English to Spanish in more detail and adds a comparison with the two baseline systems. The various `lXrY` configurations use the same feature vector setup for all classifier

[2] Download and unpack `http://lst.science.ru.nl/~proycon/colibrita-acl2014-data.zip`

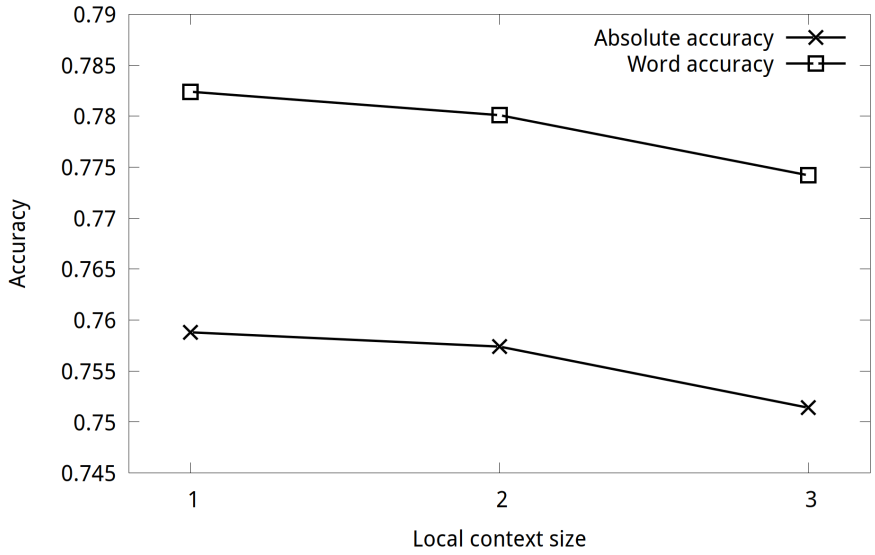

Figure 4.2: Accuracy for different local context sizes, Europarl English to Spanish

experts. Here X indicates the left context size and Y the right context size. The `auto` configuration does not uniformly apply the same feature vector setup to all classifier experts but instead seeks to find the optimal setup per classifier expert. This shall be further discussed in Section 4.6.1.

As expected, the LM baseline substantially outperforms the context-insensitive MLF baseline. Second, our classifier approach attains a substantially higher accuracy than the LM baseline. Third, we observe that adding the language model to our classifier leads to another significant gain (configuration `l1r1+LM` in the results in Table 4.2). It appears that the classifier approach and the L2 language model are able to complement each other.

Statistical significance on the BLEU scores was tested using pair-wise bootstrap sampling (Koehn, 2004b). All significance tests were performed with 5,000 iterations. We compared the outcomes of several key configurations. We first tested `l1r1` against both baselines; both differences are significant at $p < 0.01$ for both. The same significance level was found when comparing `l1r1+LM` against `l1r1`, `auto+LM` against `auto`, as well as the LM baseline against the MLF baseline. Automatic feature selection `auto` was found to perform statistically better than `l1r1`, but only at $p < 0.05$. Conclusions with regard to context width may have to be tempered somewhat, as the performance of the `l1r1` configuration was found to not be significantly better than

Configuration	Accuracy	Word Accuracy	BLEU	METEOR	NIST	WER	PER
MLF baseline	0.6164	0.6662	0.972	0.9705	17.0784	1.4465	1.4209
LM baseline	0.7158	0.7434	0.9785	0.9739	17.1573	1.1735	1.1574
l1r1	0.7588	0.7824	0.9801	0.9747	17.1550	1.1625	1.1444
l2r2	0.7574	0.7801	0.9800	0.9746	17.1550	1.1750	1.1569
l3r3	0.7514	0.7742	0.9796	0.9744	17.1445	1.1946	1.1780
l1r1+LM	**0.7810**	**0.7973**	**0.9816**	**0.9754**	**17.1685**	**1.0946**	**1.077**
auto	0.7626	0.7850	0.9803	0.9748	17.1544	1.1594	1.1424
auto+LM	0.7796	0.7966	0.9815	0.9754	17.1664	1.1021	1.0845
l1r0	0.6924	0.7223	0.9757	0.9723	17.1087	1.3415	1.3249
l2r0	0.6960	0.7245	0.9759	0.9724	17.1091	1.3364	1.3193
l2r1	0.7624	0.7849	0.9803	0.9748	17.1558	1.1554	1.1378

Table 4.2: Europarl results for English to Spanish (i.e English fallback in Spanish context). Recall = 0.9422

that of the `l2r2` configuration. However, `l1r1` performs significantly better than `l3r3` at $p < 0.01$, and `l2r2` performs significantly better than `l3r3` at $p < 0.01$.

In Table 4.3 we present some illustrative examples from the English → Spanish Europarl data. We show the difference between the most-likely-fragment baseline and our system.

Likewise, Table 4.4 exemplifies small fragments from the `l1r1` configuration compared to the same configuration enriched with a language model. We observe in this data that the language model often has the added power to choose a correct translation that is not the first prediction of the classifier, but one of the weaker alternatives that nevertheless fits better. Though the classifier generally works best in the `l1r1` configuration, i.e. with context size one, the trigram-based language model allows further left-context information to be incorporated that influences the weights of the classifier output, successfully forcing the system to select alternatives. This combination of a classifier with context size one and trigram-based language model proves to be most effective and reaches the best results so far. We have not conducted experiments with language models of other orders.

Input: Mientras no haya prueba en contrario , la financiación de partidos políticos **European** sólo se justifica , incluso después del tratado de Niza , desde el momento en que concurra a la expresión del sufragio universal , que es la única definición aceptable de un partido político .
MLF baseline: Mientras no haya prueba en contrario , la financiación de partidos políticos **Europea** sólo se justifica , incluso después del tratado de Niza , desde el momento en que concurra a la expresión del sufragio universal , que es la única definición aceptable de un partido político .
l1r1: Mientras no haya prueba en contrario , la financiación de partidos políticos **europeos** sólo se justifica , incluso después del tratado de Niza , desde el momento en que concurra a la expresión del sufragio universal , que es la única definición aceptable de un partido político .

Input: Esta Directiva es nuestra oportunidad **to** marcar una verdadera diferencia , reduciendo la trágica pérdida de vidas en nuestras carreteras .
MLF baseline: Esta Directiva es nuestra oportunidad **a** marcar una verdadera diferencia , reduciendo la trágica pérdida de vidas en nuestras carreteras .
l1r1: Esta Directiva es nuestra oportunidad **para** marcar una verdadera diferencia , reduciendo la trágica pérdida de vidas en nuestras carreteras .

Input: Es la **last** vez que me dirijo a esta Cámara .
MLF baseline: Es la **pasado** vez que me dirijo a esta Cámara .
l1r1: Es la **última** vez que me dirijo a esta Cámara .

Input: Pero el enfoque actual de la Comisión no puede conducir a una buena política ya que es tributario del funcionamiento del mercado y de las normas establecidas por la OMC , el FMI y el Banco Mundial , normas que siguen siendo desfavorables para los **developing countries** .
MLF baseline: Pero el enfoque actual de la Comisión no puede conducir a una buena política ya que es tributario del funcionamiento del mercado y de las normas establecidas por la OMC , el FMI y el Banco Mundial , normas que siguen siendo desfavorables para los **los países en desarrollo** .
l1r1: Pero el enfoque actual de la Comisión no puede conducir a una buena política ya que es tributario del funcionamiento del mercado y de las normas establecidas por la OMC , el FMI y el Banco Mundial , normas que siguen siendo desfavorables para los **países en desarrollo** .

Table 4.3: Some illustrative examples of MLF-baseline output versus system output, in which system output matches the correct human reference output. The actual fragments concerned are highlighted in bold. The first example shows our system correcting for number agreement, the second a correction in selecting the right preposition, and the third shows that the English word last *can be translated in different ways, only one of which is correct in this context. The last example shows a phrasal translation, in which the determiner was duplicated in the baseline*

Input: Sin ese tipo de protección la gente no aprovechará la oportunidad **to** vivir , viajar y trabajar donde les parezca en la Unión Europea .
l1r1: Sin ese tipo de protección la gente no aprovechará la oportunidad **para** vivir , viajar y trabajar donde les parezca en la Unión Europea .
l1r1+LM: Sin ese tipo de protección la gente no aprovechará la oportunidad **de** vivir , viajar y trabajar donde les parezca en la Unión Europea .

Input: La Comisión también está acometiendo medidas en el ámbito social y **educational** con vistas a mejorar la situación de los niños .
l1r1: La Comisión también está acometiendo medidas en el ámbito social y **educativas** con vistas a mejorar la situación de los niños .
l1r1+LM: La Comisión también está acometiendo medidas en el ámbito social y **educativo** con vistas a mejorar la situación de los niños .

Table 4.4: Some examples of l1r1 versus the same configuration enriched with a language model.

4.6.1 Context optimisation

It has been argued that classifier experts in a word sense disambiguation ensemble should be individually optimised (Decadt et al., 2004). This we have also shown in Chapter 3, where we find a positive impact when conducting feature selection per classifier. This intuitively makes sense; a context of one may seem to be better than any other when uniformly applied to all classifier experts, but it may well be that certain classifiers benefit from different feature selections. We therefore proceed with this line of investigation as well.

Automatic configuration selection was done by performing leave-one-out testing (for small number of instances) or 10-fold-cross validation (for larger number of instances, $n \geq 20$) on the training data per classifier expert. Various configurations were tested. Per classifier expert, the best scoring configuration was selected, referred to as the `auto` configuration in Table 4.2. The `auto` configuration improves results over the uniformly applied feature selection. However, if we enable the language model as we do in the `auto+LM` configuration we do not notice an improvement over `l1r1+LM`, surprisingly. We suspect the lack of impact here can be explained by the trigram-based Language Model having less added value when the (left) context size of the classifier is two or three; they are now less complementary.

Table 4.5 lists what context sizes have been chosen in the automatic feature selection. A context size of one prevails in the vast majority of cases, which is not surprising considering the good results we have already seen with this configuration.

66.5%	l1r1
19.9%	l2r2
7.7%	l3r3
3.5%	l4r4
2.4%	l5r5

Table 4.5: Frequency of automatically selected configurations on English to Spanish Europarl dataset

In this study we did not yet conduct optimisation of the classifier parameters. We used the IB1 algorithm with $k = 1$ and the default values of the TiMBL implementation. In Chapter 3, we reported a decrease in performance due to overfitting when this is done, so we do not expect it to make a positive impact. The second reason for omitting this is more practical in nature; to do this in combination with feature selection would add substantial search complexity, making experiments far more time consuming, even prohibitively so.

The bottom lines in Table 4.2 represent results when all right-context is omitted, emulating a real-time prediction when no right context is available yet. This has a substantial negative impact on

results. We experimented with several asymmetric configurations and found that taking two words to the left and one to the right yields even better results than symmetric configurations for this data set. This result is in line with the positive effect of adding the LM to the `l1r1`.

In order to draw accurate conclusions, experiments on a single data set and language pair are not sufficient. We therefore conducted a number of experiments with other language pairs, and present the abridged results in Table 4.6.

There are some noticeable discrepancies for some experiments in Table 4.6 when compared to our earlier results in Table 4.2. We see that the language model baseline for English→French shows the same substantial improvement over the baseline as our English→Spanish results. The same holds for the Chinese→English experiment. However, for English→Dutch and English→Chinese we find that the LM baseline actually performs slightly worse than baseline. Nevertheless, in all these cases, the positive effect of including a Language Model to our classifier-based system again shows. Also, we note that in all cases our system performs better than the two baselines.

Another discrepancy is found in the BLEU scores of the English→-Chinese experiments, where we measure an unexpected drop in BLEU score under baseline. However, all other scores do show the expected improvement. The error rate metrics show improvement as well. We therefore attach low importance to this deviation in BLEU here.

In all of the aforementioned experiments, the system produced a single solution for each of the fragments, the one it deemed best, or no solution at all if it could not find any. Alternative evaluation metrics could allow the system to output multiple alternatives. Omission of a solution by definition causes a decrease in recall. In all of our experiments recall is high (well above 90%), mostly because train and test data lie in the same domain and have been generated in the same fashion, lower recall is expected with more real-world data.

Dataset	L1	L2	Configuration	Accuracy	Word Accuracy	BLEU
europarl200k	en	nl	baseline	0.7026	0.7283	0.9771
europarl200k	en	nl	LM baseline	0.6958	0.7195	0.9773
europarl200k	en	nl	l1r1	0.7790	0.7941	0.9814
europarl200k	en	nl	l1r1+LM	**0.7838**	**0.7973**	**0.9818**
europarl200k	en	nl	auto	0.7796	0.7947	0.9815
europarl200k	en	nl	auto+LM	0.7812	0.7954	0.9816
europarl200k	en	fr	baseline	0.5874	0.6403	0.9709
europarl200k	en	fr	LM baseline	0.7054	0.7319	0.9787
europarl200k	en	fr	l1r1	0.7416	0.7698	0.9797
europarl200k	en	fr	l1r1+LM	**0.7680**	**0.7885**	**0.9815**
europarl200k	en	fr	auto	0.7484	0.7737	0.9801
europarl200k	en	fr	auto+LM	0.7654	0.7860	0.9813
iwslt12ted	en	zh	baseline	0.6622	0.7122	**0.6421**
iwslt12ted	en	zh	LM baseline	0.6550	0.6982	0.6416
iwslt12ted	en	zh	l1r1	0.7150	0.7531	0.5736
iwslt12ted	en	zh	l1r1+LM	**0.7296**	**0.7619**	0.5826
iwslt12ted	en	zh	auto	0.7150	0.7519	0.5746
iwslt12ted	en	zh	auto+LM	0.7280	0.7605	0.5833
iwslt12ted	zh	en	baseline	0.5784	0.6167	0.9634
iwslt12ted	zh	en	LM baseline	0.6148	0.6463	0.9656
iwslt12ted	zh	en	l1r1	0.7104	0.7338	0.9709
iwslt12ted	zh	en	l1r1+LM	**0.7270**	**0.7460**	**0.9721**
iwslt12ted	zh	en	auto	0.7078	0.7319	0.9709
iwslt12ted	zh	en	auto+LM	0.7230	0.7428	0.9719

Table 4.6: Results on different datasets and language pairs. The `iwslt12ted` *set is the dataset used in the IWSLT 2012 Evaluation Campaign (Federico et al., 2012), and is formed by a collection of transcriptions of TED talks. Here we used of just over* 70,000 *sentences for training. Recall for each of the four datasets is* 0.9498 *(en-nl),* 0.9494 *(en-fr),* 0.9386 *(en-zh), and* 0.9366 *(zh-en)*

4.7 Discussion and conclusion

In this study we have shown the feasibility of a classifier-based translation assistance system in which L1 fragments are translated in an L2 context, in which the classifier experts are built individually per word or phrase. We have shown that such a translation assistance system scores both above a context-insensitive baseline, as well as an L2 language model baseline.

Furthermore, we found that combining this cross-language context-sensitive technique with an L2 language model boosts results further.

The presence of a one-word right-hand side context proves crucial for good results, which has implications for practical translation assistance application that translate as soon as the user finishes an L1 fragment. Revisiting the translation when right context becomes available would be advisable.

We tested various configurations and conclude that small context sizes work better than larger ones. Automated configuration selection had positive results, yet the system with context size one and an L2 language model component often produces the best results. In static configurations, the failure of a wider context window to be more succesful may be attributed to the increased sparsity that comes from such an expansion.

The idea of a comprehensive translation assistance system may extend beyond the translation of L1 fragments in an L2 context. There are more NLP components that might play a role if such a system were to find practical application. Word completion or predictive editing (in combination with error correction) would for instance seem an indispensable part of such a system, and can be implemented alongside the technique proposed in this study. A point of more practically-oriented future research is to see how feasible such combinations are and what techniques can be used.

An application of our idea outside the area of translation assistance is post-correction of the output of some MT systems that, as a last-resort heuristic, copy source words or phrases into their output, producing precisely the kind of input our system is trained on. Our classification-based approach may be able to resolve some of these cases operating as an add-on to a regular MT system – or as a independent post-correction system.

Our system allows L1 fragments to be of arbitrary length. If a fragment was not seen during training stage, and is therefore not covered by a classifier expert, then the system will be unable to translate it. Nevertheless, if a longer L1 fragment can be decomposed into subfragments that are known, then some recombination of the translations of said sub-fragments may be a good translation for the whole. We are currently exploring this line of investigation, in which the gap with MT narrows further.

Finally, an important line of future research is the creation of a more representative test set. Lacking an interactive system that actually does what we emulate, we hypothesise that good approximations would be to use gap exercises, or cloze tests, that test specific aspects difficulties in language learning. Similarly, we may use L2 learner corpora with annotations of code-switching points or errors. Here we then assume that places where L2 errors occur may be indicative of places where L2 learners are in some trouble, and might want to fall back to generating L1. By then manually translating gaps or such problematic fragments into L1 we hope to establish a more realistic test set.

5

SEMEVAL-2014 TASK 5: L2 WRITING ASSISTANT

In this chapter, we present a new cross-lingual task for SemEval concerning the translation of L1 fragments in an L2 context. The task is derived from the ideas and findings explored in Chapter 4. The task is at the boundary of Cross-Lingual Word Sense Disambiguation and Machine Translation and finds application in the field of computer-assisted translation, particularly in the context of second language learning. Translating L1 fragments in an L2 context allows language learners when writing in a target language (L2) to fall back to their native language (L1) whenever they are uncertain of the right word or phrase.

This chapter is based on: van Gompel, M., Hendrickx, I., van den Bosch, A., Lefever, E., and Hoste, V. (2014). Semeval-2014 Task 5: L2 writing assistant. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 36–44, Dublin,Ireland

5.1 Introduction

We present a new cross-lingual and application-oriented task for SemEval that is situated in the area where Word Sense Disambiguation and Machine Translation meet. Finding the proper translation of a word or phrase in a given context is much like the problem of disambiguating between multiple senses.

In this task participants are asked to build a translation/writing assistance system that translates specifically marked L1 fragments in an L2 context to their proper L2 translation. This type of translation can be applied in writing assistance systems for language learners in which users write in a target language, but are allowed to occasionally back off to their native L1 when they are uncertain of the proper lexical or grammatical form in L2. The task concerns the NLP back-end rather than any user interface.

Full-on machine translation typically concerns the translation of complete sentences or texts from L1 to L2. This task, in contrast, focuses on smaller fragments, side-tracking the problem of full word reordering.

We focus on the following language combinations of L1 and L2 pairs: *English-German*, *English-Spanish*, *French-English* and *Dutch-*

English. Task participants could participate for all language pairs or any subset thereof.

5.2 Task Description

We frame the task in the context of second language learning, yielding a specific practical application.

Participants build a *translation assistance system* rather than a full machine translation system. The L1 expression, a word or phrase, is translated by the system to L2, given the L2 context already present, including right-side context if available. The aim here, as in all translation, is to carry the semantics of the L1 fragment over to L2 and find the most suitable L2 expression given the already present L2 context.

Other than a limit on length (6 words), we do not pose explicit constraints on the kinds of L1 fragments allowed. The number of L1 fragments is limited to one fragment per sentence.

The task addresses both a core problem of WSD, with cross-lingual context, and a sub-problem of Phrase-based Statistical Machine Translation; that of finding the most suitable translation of a word or phrase. In MT this would be modelled by the translation model. In our task the full complexity of full-sentential translation is bypassed, putting the emphasis on the semantic aspect of translation. Our task has specific practical applications and a specific intended audience, namely intermediate and advanced second language learners, whom one generally wants to encourage to use their target language as much as possible, but who may often feel the need to fall back to their native language.

Currently, language learners are forced to fall back to a bilingual dictionary when in doubt. Such dictionaries do not take the L2 context into account and are generally more constrained to single words or short expressions. The proposed application would allow more flexible context-dependent lookups as writing progresses. The task tests how effectively participating systems accomplish this.

The following examples illustrate the task for the four language pairs we offer:

- Input (L1=English,L2=Spanish): *"Todo ello,* ***in accordance*** *con los principios que siempre hemos apoyado."*
  Desired output: *"Todo ello,* ***de conformidad*** *con los principios que siempre hemos apoyado."*

- Input (L1-English, L2=German): *"Das, was wir heute machen,* ***is essentially*** *ein Ärgernis."*
  Desired output: *"Das, was wir heute machen,* ***ist im Grunde genommen*** *ein Ärgernis."*

- Input (L1=French,L2=English): *"I **rentre à la maison** because I am tired."*
  Desired output: *"I **return home** because I am tired."*

- Input (L1=Dutch, L2=English): *"Workers are facing a massive **aanval op** their employment and social rights."*
  Desired output: *"Workers are facing a massive **attack on** their employment and social rights."*

The task can be related to the two tasks that were offered in previous years of SemEval and already introduced in Chapter 3: Lexical Substitution (Mihalcea et al., 2010) and most notably Cross-lingual Word Sense Disambiguation (Lefever and Hoste, 2013b).

When comparing our task to the Cross-Lingual Word-Sense Disambiguation task, one notable difference is the fact that our task concerns not just words, but also phrases. Another essential difference is the nature of the context; our context is in L2 instead of L1. Unlike the Cross-Lingual Word Sense Disambiguation task, we do not constrain the L1 words or phrases that may be used for translation, except for a maximum length which we set to 6 tokens, whereas Lefever and Hoste (2013b) only tested a select number of nouns. Our task emphasizes a correct meaning-preserving choice of words in which translations have to fit in the L2 context. There is thus a clear morphosyntactic aspect to the task, although less prominent than in full machine translation, as the remainder of the sentence, already in L2, does not need to be changed. In the Cross-Lingual Word Sense Disambiguation tasks, the translations/senses were lemmatised. We deliberately chose a different path that allows for the envisioned application to function directly as a translation assistance system.

Chapter 4 described a pilot study conducted to test the feasibility of the proposed translation system. It shows that L2 context information can be a useful cue in translation of L1 fragments to L2, improving over a non-context-informed baseline.

5.3 Data

We did not provide training data for this task, as we did not want to bias participating systems by favouring a particular sort of material and methodology. Moreover, it would be a prohibitively large task to manually collect enough training data for the task itself. Participants were therefore free to use any suitable training material such as parallel corpora, wordnets, or bilingual lexica.

Trial and test data has been collected for the task, both delivered in a simple XML format that explicitly marks the fragments. System output of participants adheres to the same format. The trial set, released early on in the task, was used by participants to develop

and tune their systems on. The test set corresponds to the final data released for the evaluation period; the final evaluation was conducted on this data.

The trial data was constructed in an automated fashion in the way described in Chapter 4. In summary, first a phrase-translation table is constructed from a parallel corpus. We used the Europarl parallel corpus (Koehn, 2005) and the Moses tools (Koehn et al., 2007b), which in turn makes use of GIZA++ (Och and Ney, 2000). Only strong phrase pairs (exceeding a set threshold) were retained and weaker ones were pruned. This phrase-translation table was then used to create input sentences in which the L2 fragments are swapped for their L1 counterparts, effectively mimicking a fall-back to L1 in an L2 context. The full L2 sentence acts as reference sentence. Finally, to ensure all fragments are correct and sensible, a manual selection from this automatically generated corpus constituted the final trial set.

In the pilot study in Chapter 4, such a data set, even without the manual selection stage, proved adequate to demonstrate the feasibility of translating L1 fragments in an L2 context. One can, however, rightfully argue whether such data is sufficiently representative for the task and whether it would adequately cover instances where L2 language learners might experience difficulties and be inclined to fall back to L1. We therefore created a more representative test set for the task.

The actual test set conforms to much more stringent constraints and was composed entirely by hand from a wide variety of written sources. Amongst these sources are study books and grammar books for language learners, short bilingual on-line stories aimed at language learners, gap-exercises and cloze tests, and contemporary written resources such as newspapers, novels, and Wikipedia. We aimed for actual learner corpora, but finding suitable learner corpora with sufficient data proved hard. For German we could use the the Merlin corpus (Abel et al., 2013). In example (a) we see a real example of a fragment in a fallback language in an L2 context from the Merlin corpus.

(a) **Input:** Das Klima hier ist **Tropical** und wir haben fast keinen Winter
Reference: Das Klima hier ist **tropisch** und wir haben fast keinen Winter.

For various sources bilingual data was available. For the ones that were monolingual (L2) we resorted to manual translation. To ensure our translations were correct, these were later independently verified, and where necessary corrected by native speakers.

A large portion of the test set comes from off-line resources because we wanted to make sure that a substantial portion of the test set could not be found verbatim on-line. This was done to prevent systems from solving the actual problem by just attempting to just look up the sources through the available context information.

Note that in general we aimed for the European varieties of the different languages. However, for English we did add the US spelling variants as alternatives.

A complete list of all sources used in establishing the test set is available on our website[1].

We created a trial set and test set/gold standard of 500 sentence pairs per language pair. Due to the detection of some errors at a later stage, some of which were caused by the tokenisation process, we were forced to remove some sentences from the test set and found ourselves slightly below our aim for some of the language pairs. The test set was delivered in both tokenised[2] and untokenised form. The trial set was delivered only in tokenised form. Evaluation was conducted against the tokenised version, but our evaluation script was designed to be as lenient as possible regarding differences in tokenisation. We explicitly took cases into account where participant's tokenisers split contractions (such as Spanish "del" to "de" + "el"), whereas our tokeniser did not.

For a given input fragment, it may well be possible that there are multiple correct translations possible. In establishing our test set, we therefore paid special attention to adding alternatives. To ensure no alternatives were missed, all participant output was aggregated in one set, effectively anonymising the systems, and valid but previously missed alternatives were added to the gold standard.

5.4 Evaluation

Several metrics are available for automatic evaluation. First, we measure the absolute accuracy $a = c/n$, where c is the number of fragment translations from the system output that precisely match the corresponding fragments in the reference translation, and n is the total number of translatable fragments, including those for which no translation was found. We also introduce a word-based accuracy, which unlike the absolute accuracy gives some credits to mismatches that show partial overlap with the reference translation. It assigns a score according to the longest consecutive matching substring between output fragment and reference fragment and is computed as follows:

$$wac = \frac{|longestsubmatch(output, reference)|}{max(|output|, |reference|)} \tag{5.1}$$

The system with the highest word-based accuracy wins the competition. All matching is case-sensitive.

Systems may decide not to translate fragments if they cannot find a suitable translation. A recall metric simply measures the number of fragments for which the system generated a translation, regardless of

[1] https://github.com/proycon/semeval2014task5

[2] Using ucto, available at https://github.com/proycon/ucto

whether that translation is correct or not, as a proportion of the total number of fragments.

In addition to these task-specific metrics, standard MT metrics such as BLEU, NIST, METEOR and error rates such as WER, PER and TER, are included in the evaluation script as well. Scores such as BLEU will generally be high (> 0.95) when computed on the full sentence, as a large portion of the sentence is already translated and only a specific fragment remains to be evaluated. Nevertheless, these generic metrics are proven in our pilot study to follow the same trend as the more task-specific evaluation metrics, and will be omitted in the result section for brevity.

It regularly occurs that multiple translations are possible. As stated, in the creation of the test set we have taken this into account by explicitly encoding valid alternatives. A match with any alternative in the reference counts as a valid match. For word accuracy, the highest word accuracy amongst all possible alternatives in the reference is taken. Likewise, participant system output may contain multiple alternatives as well, as we allowed two different types of runs, following the example of the Cross-Lingual Lexical Substitution and Cross-Lingual Word Sense Disambiguation tasks:

- **Best** - The system may only output one, its best, translation;
- **Out of Five** - The system may output up to five alternatives, effectively allowing 5 guesses. Only the best match is counted. This metric does *not* count how many of the five are valid.

Participants could submit up to three runs per language pair and evaluation type.

5.5 Participants

Six teams submitted systems, three of which participated for all language pairs. In alphabetic order, these are:

1. **CNRC** - Cyril Goutte, Michel Simard, Marine Carpuat - National Research Council Canada – *All language pairs*
2. **IUCL** - Alex Rudnick, Liu Can, Levi King, Sandra Kübler, Markus Dickinson - Indiana University (US) – *all language pairs*
3. **UEdin** - Eva Hasler - University of Edinburgh (UK) – *all language pairs except English-German*
4. **UNAL** - Sergio Jiménez, Emilio Silva - Universidad Nacional de Colombia – *English-Spanish*
5. **Sensible** - Liling Tan - Universität des Saarlandes (Germany) and Nanyang Technological University (Singapore) – *all language pairs*

6. **TeamZ** - Anubhav Gupta - Université de Franche-Comté (France) – *English-Spanish, English-German*

Participants implemented distinct methodologies and implementations. One obvious avenue of tackling the problem is through standard Statistical Machine Translation (SMT). The CNRC team takes a pure SMT approach with few modifications. They employ their own Portage decoder and directly send an L1 fragment in an L2 context, corresponding to a partial translation hypothesis with only one fragment left to decode, to their decoder (Goutte et al., 2014). The UEdin team applies a similar method using the Moses decoder, marking the L2 context so that the decoder leaves this context as is. In addition they add a context similarity feature for every phrase pair in the phrase translation table, which expresses topical similarity with the test context. In order to properly decode, the phrase table is filtered per test sentence (Hasler, 2014). The IUCL and UNAL teams do make use of the information from word alignments or phrase translation tables, but do not use a standard SMT decoder. The IUCL system combines various information sources in a log-linear model: phrase table, L2 Language Model, Multilingual Dictionary, and a dependency-based collocation model, although this latter source was not finished in time for the system submission (Rudnick et al., 2014). The UNAL system extracts syntactic features as a means to relate L1 fragments with L2 context to their L2 fragment translations, and uses memory-based classifiers to achieve this (Silva-Schlenker et al., 2014). The two systems on the lower end of the result spectrum use different techniques altogether. The Sensible team approaches the problem by attempting to emulate the manual post-editing process human translators employ to correct MT output (Tan et al., 2014), whereas TeamZ relies on Wiktionary as the sole source (Gupta, 2014).

5.6 Results

The results of the six participating teams can be viewed in consensed form in Table 4.2. This table shows the highest word accuracy achieved by the participants, in which multiple system runs have been aggregated. A ranking can quickly be distilled from this, as the best score is marked in bold. The system by the University of Edinburgh emerges as the clear winner of the task. The full results of the various system runs by the six participants are shown in Tables 5.1 and 5.2, two pages down, all three aforementioned evaluation metrics are reported there and the systems are sorted by word accuracy per language pair and evaluation type.

For the lowest-ranking participants, the score is negatively impacted by the low recall; their systems could not find translations for a large number of fragments.

Figures 5.1 (next page) and 5.2 (last page) show the results for the *best* evaluation type for each system run. Three bars are shown; from left to right these represent *accuracy* (blue), *word-accuracy* (green) and *recall* (red). Graphs for *out-of-five* evaluation were omitted for brevity, but tend to follow the same trend with scores that are somewhat higher. These scores can be viewed on the result website at https://github.com/proycon/semeval2014task5/. The result website also holds the system output and evaluation scripts with which all graphs and tables can be reproduced.

We observe that the best scoring team in the task (UEdin), as well as the CNRC team, both employ standard Statistical Machine Translation and achieve high results. From this we can conclude that standard SMT techniques are suitable for this task. Teams IUCL and UNAL achieve similarly good results, building on word and phrase alignment data as does SMT, yet not using a traditional SMT decoder. TeamZ and Sensible, the two systems ranked lowest do not rely on any techniques from SMT. To what extent the context-informed measures of the various participants are effective can not be judged from this comparison, but can only be assessed in comparison to their own baselines. For this we refer to the system papers of the participants.

5.7 Discussion

We did not specify any training data for the task. The advantage of this is that participants were free to build a wider variety of systems from various sources, rather than introducing a bias towards for instances statistical systems. The disadvantage, however, is that a comparison of the various systems does not yield conclusive results regarding the merit of their methodologies. Discrepancies might at least be partly due to differences in training data, as it is generally well understood in MT that more training data improves results. The baselines various participants describe in their system papers provide more insight to the merit of their approaches than a comparison between them.

In the creation of the test set, we aimed to mimic intermediate to high-level language learners. We also aimed at a fair distribution of different part-of-speech categories and phrasal length. The difficulty of the task differs between language pairs, though not intentionally so. We observe that the Dutch-English set is the hardest and the Spanish-English is the easiest in the task. One of the participants implicitly observes this through measurement of the number of Out-of-Vocabulary words (Goutte et al., 2014). This implies that when comparing system performance between different language pairs, one can not simply ascribe a lower result to a system having more difficulty with said language pair. This could rather be an intrinsic property of the test set or the distance between the languages.

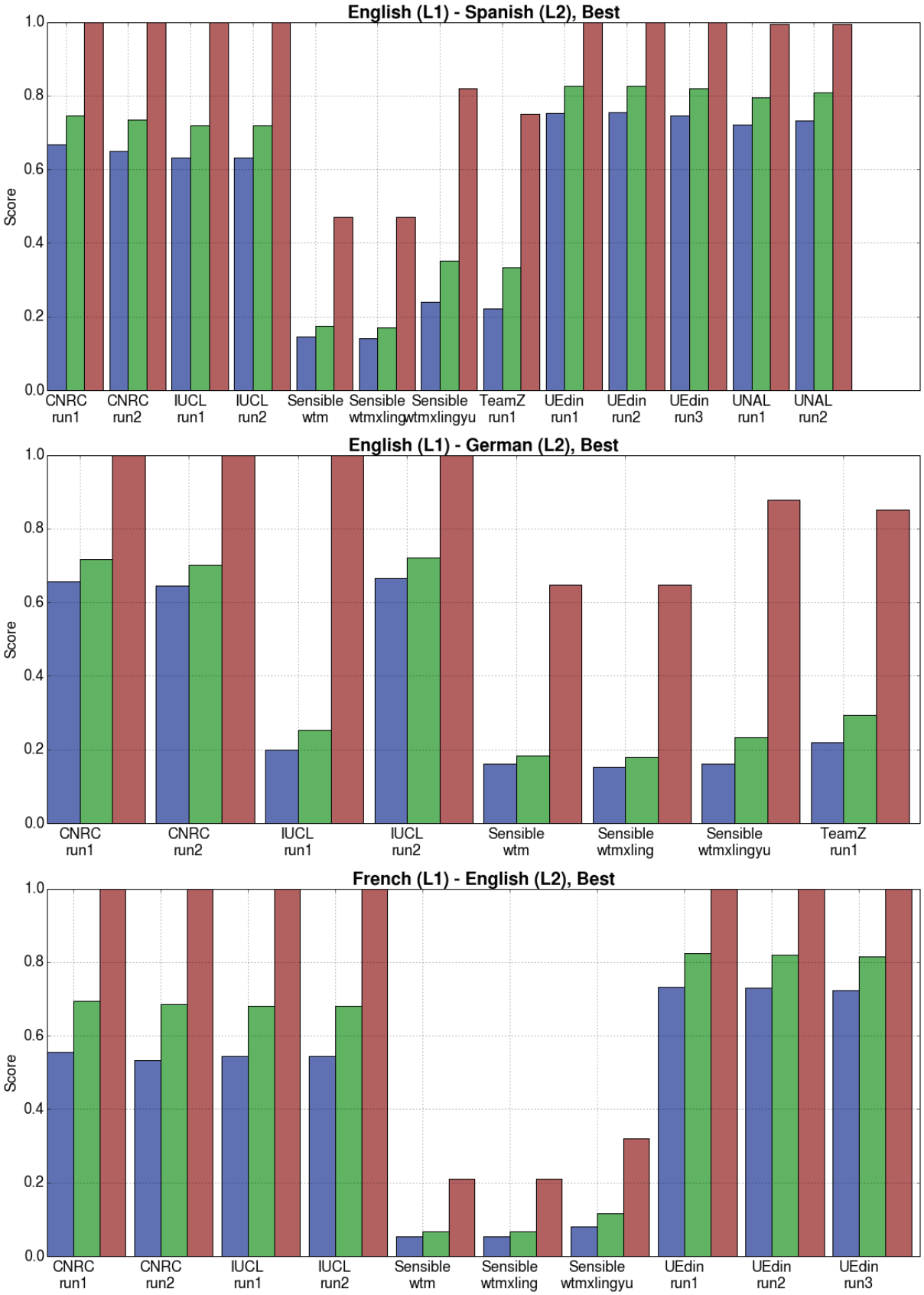

Figure 5.1: English to Spanish (top), English to German (middle) and French to English (bottom). The three bars, left-to-right, represent Accuracy (blue), Word Accuracy (green) and Recall (red).

Distance in syntactic structure between languages also defines the limits of this task. During composition of the test set it became clear that backing off to L1 was not always possible when syntax diverged to much. An example of this is separable verbs in Dutch and German. Consider the German sentence *"Er **ruft** seine Mutter **an**"* (translation: *"He **calls** his mother"*). Imagine a German language learner wanting to compose such a sentence but wanting to fall back to English for the verb *"to call"*, which would translate to German as *"anrufen"*. The possible input sentence may still be easy to construe: *"Er **calls** seine Mutter"*, but the solution to this problem would require insertion at two different points, whereas the task currently only deals with a substitution of a single fragment. The reverse is arguably even more complex and may stray too far from what a language learner may do. Consider an English language learner wanting to fall back to her native German, struggling with the English translation for *"anrufen"*. She may compose a sentence such as *"He **ruft** his mother **an**"*, which would require translating two dependent fragments into one.

We already have interesting examples in the gold standard, such as example (b), showing syntactic word-order changes confined to a single fragment.

(b) **Input:** I always wanted **iemand te zijn** , but now I realize I should have been more specific.
Reference: I always wanted **to be somebody** , but now I realize I should have been more specific.
Participant output (aggregated): to be a person; it to be; someone to his; to be somebody; person to be; someone to; someone to be; to be anybody; to anyone; to be someone; a person to have any; to be someone else

Another question we can ask, but have not investigated, is whether a language learner would insert the proper morphosyntactic form of an L1 word given the L2 context, or whether she may be inclined to fall back to a normal form such as an infinitive. Especially in the above case of separable verbs someone may be more inclined to circumvent the double fragments and provide the input: *"He **anrufen** his mother"*, but in simpler cases the same issue arises as well. Consider an English learner falling back to her native Croatian, a Slavic language which heavily declines nouns. If she did not know the English word *"book"* and wanted to write *"He gave the book to him"*, she could use either the Croatian word *"knjigu"* in its accusative declension or fall back to the normal form *"knjiga"*. A proper writing assistant system would have to account for both options.

We can analyse which of the sentences in the test data participants struggled with most. First we look at the number of sentences that produce an average word accuracy of zero, measured per sentence over all systems and runs in the out-of-five metric. This means no participant was close to the correct output. There were 6 such sentences in English-Spanish, 17 in English-German, 6 in French-English, and 32 in Dutch-English.

A particularly difficult context from the Spanish set is when a subjunctive verb form was required, but an indicative verb form was submitted by the systems, such as in the sentence: *"Espero que los frenos del coche* ***funcionen*** *bien."*. Though this may be deduced from context (the word *"Espero"*, expressing hope yet doubt, being key here), it is often subtle and hard to capture. Another problematic case that recurs in the German and Dutch data sets is compound nouns. The English fragment *"work motivation"* should translate into the German compound *"Arbeitsmotivation"* or *"Arbeitsmoral"*, yet participants were not able to find the actual compound noun. Beside compound nouns, other less frequent multi-word expressions are also amongst the difficult cases. Sparsity or complete absence in training data of these expressions is why systems struggle here.

Another point of discussion is the fact that we enriched the test set by adding previously unavailable alternative translations from an aggregated pool of system output. This might draw criticism for possibly introducing a bias, also considering the fact that the decision to include a particular alternative for a given context is not always straightforward and at times subjective. We, however, contend that this is the best way to ensure that valid system output is not discarded and reduce the number of false negatives. The effect of this measure has been an increase in (word) accuracy for all systems, without significant impact on ranking.

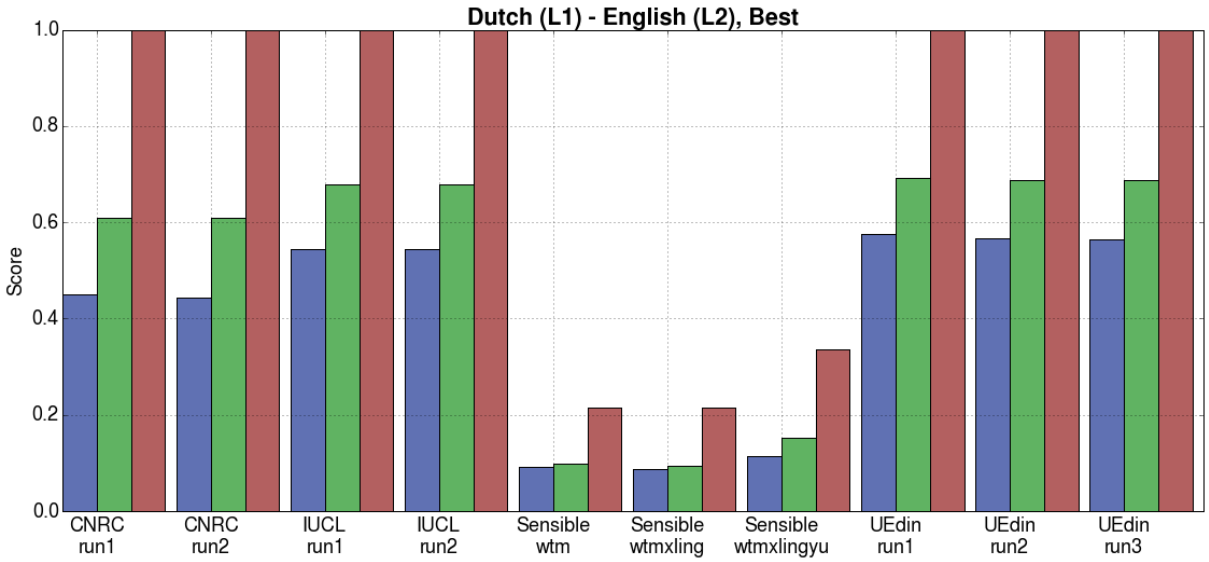

Figure 5.2: Dutch to English.

5.8 Conclusion

In this SemEval task we showed that systems can translate L1 fragments in an L2 context, a task that finds application in computer-

assisted translation and computer-assisted language learning. The localised translation of a fragment in a cross-lingual context makes it a novel task in the field. Though the task has its limits, we argue for its practical application in a language-learning setting: as a writing assistant and dictionary replacement. Six contestants participated in the task, and used an ensemble of techniques from Statistical Machine Translation and Word Sense Disambiguation. Most of the task organizers' time went into manually establishing a gold standard based on a wide variety of sources, most aimed at language learners, for each of the four language pairs in the task. We have been positively surprised by the good results of the highest ranking systems.

5.9 Acknowledgements

We would like to thank Andreu van Hooft and Sarah Schulz for their manual correction work, and Sean Banville, Geert Joris, Bernard De Clerck, Rogier Crijns, Adriane Boyd, Detmar Meurers, Guillermo Sanz Gallego and Nils Smeuninx for helping us with the data collection.

System	Acc	W.Acc.	Recall
English-Spanish (best)			
UEdin-run2	0.755	0.827	1.0
UEdin-run1	0.753	0.827	1.0
UEdin-run3	0.745	0.82	1.0
UNAL-run2	0.733	0.809	0.994
UNAL-run1	0.721	0.794	0.994
CNRC-run1	0.667	0.745	1.0
CNRC-run2	0.651	0.735	1.0
IUCL-run1	0.633	0.72	1.0
IUCL-run2	0.633	0.72	1.0
Sensible-wtmxlingyu	0.239	0.351	0.819
TeamZ-run1	0.223	0.333	0.751
Sensible-wtm	0.145	0.175	0.470
Sensible-wtmxling	0.141	0.171	0.470
English-Spanish (out-of-five)			
UEdin-run3	0.928	0.949	1.0
UEdin-run1	0.924	0.946	1.0
UEdin-run2	0.92	0.944	1.0
CNRC-run1	0.843	0.887	1.0
CNRC-run2	0.837	0.884	1.0
UNAL-run1	0.823	0.88	0.994
IUCL-run1	0.781	0.847	1.0
IUCL-run2	0.781	0.847	1.0
Sensible-wtmxlingyu	0.263	0.416	0.819
TeamZ-run1	0.277	0.386	0.751
Sensible-wtm	0.173	0.231	0.470
Sensible-wtmxling	0.169	0.228	0.470
English-German (best)			
IUCL-run2	0.665	0.722	1.0
CNRC-run1	0.657	0.717	1.0
CNRC-run2	0.645	0.702	1.0
TeamZ-run1	0.218	0.293	0.852
IUCL-run1	0.198	0.252	1.0
Sensible-wtmxlingyu	0.162	0.233	0.878
Sensible-wtm	0.16	0.184	0.647
Sensible-wtmxling	0.152	0.178	0.647
English-German (out-of-five)			
CNRC-run1	0.834	0.868	1.0
CNRC-run2	0.828	0.865	1.0
IUCL-run2	0.806	0.857	1.0
TeamZ-run1	0.307	0.385	0.852
IUCL-run1	0.228	0.317	1.0
Sensible-wtmxlingyu	0.18	0.306	0.878
Sensible-wtm	0.182	0.256	0.647
Sensible-wtmxling	0.174	0.25	0.647

Table 5.1: Full results for English-Spanish and English-German.

System	Acc	W.Acc.	Recall
French-English (best)			
UEdin-run1	0.733	0.824	1.0
UEdin-run2	0.731	0.821	1.0
UEdin-run3	0.723	0.816	1.0
CNRC-run1	0.556	0.694	1.0
CNRC-run2	0.533	0.686	1.0
IUCL-run1	0.545	0.682	1.0
IUCL-run2	0.545	0.682	1.0
Sensible-wtmxlingyu	0.081	0.116	0.321
Sensible-wtm	0.055	0.067	0.210
Sensible-wtmxling	0.055	0.067	0.210
French-English (out-of-five)			
UEdin-run2	0.909	0.939	1.0
UEdin-run1	0.905	0.938	1.0
UEdin-run3	0.907	0.937	1.0
CNRC-run1	0.739	0.839	1.0
CNRC-run2	0.731	0.834	1.0
IUCL-run1	0.691	0.8	1.0
IUCL-run2	0.691	0.8	1.0
Sensible-wtmxlingyu	0.085	0.14	0.321
Sensible-wtmxling	0.061	0.09	0.210
Sensible-wtm	0.061	0.089	0.210
Dutch-English (best)			
UEdin-run1	0.575	0.692	1.0
UEdin-run2	0.567	0.688	1.0
UEdin-run3	0.565	0.688	1.0
IUCL-run1	0.544	0.679	1.0
IUCL-run2	0.544	0.679	1.0
CNRC-run1	0.45	0.61	1.0
CNRC-run2	0.444	0.609	1.0
Sensible-wtmxlingyu	0.115	0.152	0.335
Sensible-wtm	0.092	0.099	0.214
Sensible-wtmxling	0.088	0.095	0.214
Dutch-English (out-of-five)			
UEdin-run1	0.733	0.811	1.0
UEdin-run3	0.727	0.808	1.0
UEdin-run2	0.725	0.808	1.0
IUCL-run1	0.634	0.753	1.0
IUCL-run2	0.634	0.753	1.0
CNRC-run1	0.606	0.723	1.0
CNRC-run2	0.602	0.721	1.0
Sensible-wtmxlingyu	0.123	0.171	0.335
Sensible-wtm	0.099	0.115	0.214
Sensible-wtmxling	0.096	0.112	0.214

Table 5.2: Full results for French-English and Dutch-English.

6

THE ROLE OF CONTEXT INFORMATION IN L2 TRANSLATION ASSISTANCE

In this chapter we conclude the work started in Chapters 4 and 5, and investigate to what extent L2 context information can aid the translation of L1 fragments in an L2 context, and what techniques are most suitable. We focus on two approaches: the classifier-based approach from Chapter 4, and one rooted in Statistical Machine Translation. Various mixtures between the two are investigated. We explicitly investigate the the role of context information (in L2) and of the L2 language model. We test the incorporation of memory-based classifiers, as it is proven method in Word Sense Dismambiguation, as a means of better disambiguating the L1 fragments. We find Statistical Machine Translation to be the most adequate solution to the problem, and show how it can be applied with a cross-lingual context. Integrating classifiers in such a framework has little merit, as there is considerable overlap with the functioning of the L2 language model.

This chapter also appeared in: the International Journal of Translation (2016), Volume 28, No 1–2, pp. 71–101.

6.1 Introduction

In this study we test and compare two approaches to translate L1 fragments in an L2 context, focussing on the use of memory-based classifiers to disambiguate the L1 fragments using their L2 context information. The first approach is purely classifier-based, whereas the second integrates classifiers in phrase-based statistical machine translation. This latter approach enhances the translation model component by taking context directly into account in translation.

The MT-integrated system described is described in Section 6.5. The pure classifier-based approach is mainly described in Chapter 4, we offer a recap in Section 6.4. In the present study, it will be re-evaluated on the data from our SemEval-2014 "L2 Translation Assistant" task described in Chapter 5, which provides a a representative test set where none existed yet.

Section 6.2 briefly summarises the most important findings from the participants in the SemEval task, and what we have learned from it. In Section 6.3 we describe the data used for our experiments, as well as the evaluation metrics used throughout the study. We discuss the results of our experiments and formulate our conclusions in Section 6.6.

6.2 SemEval Results

As we have seen in Chapter 5, six participants took part in the SemEval task by writing a translation assistance system. This provides an interesting basis for comparison, although direct comparison is complicated by the fact that the SemEval task did not specify a training set.

The study by SemEval participant Goutte et al. (2014) shows that a solution based on standard Phrase-based Statistical Machine Translation is viable strategy of tackling the L2 Translation Assistant task. They present a clean and vanilla approach using their Portage decoder (Sadat et al., 2005), in which the L2 context is explicitly passed as context, and the remaining L1 fragment is translated, yielding a full sentence translation in L2. They score well in the SemEval task, finishing in second or third place for all language pairs.

The best-performing system (Hasler, 2014), finishing in first place for the three language pairs in which it participated, follows a very similar strategy, using the Moses MT decoder (Koehn et al., 2007b). They pass the entire sentence, i.e the L1 fragment in the L2 context, to the decoder. The L2 context is specifically marked to be left untouched by the decoder, using XML syntax supported by Moses. In this XML format, the L2 context has its translations explicitly passed, and reordering *walls* prevent the decoder from changing the word order, leaving only the L1 fragment to be translated. This method was proposed earlier in a predecessor of Moses (Cabezas and Resnik, 2005). Consider Example 6.2.1 showing the XML input example from Hasler (2014), in which L2 is English and L1 is French:

Example 6.2.1

```
<wall/>les manifesteurs<wall/>
<np translation="want">want</np><wall/>
<np translation="to">to</np><wall/>
<np translation="replace">replace</np><wall/>
<np translation="the">the</np><wall/>
<np translation="government">government</np><wall/>
<np translation=".">.</np><wall/>
```

They show that this results in a considerable improvement over a baseline that translates just the L1 fragment without any L2 context. In fact, we will see later that this simple and intuitive method outperforms the classifier-based method from the pilot study, which does not directly invoke an MT decoder. We will therefore expand our system using this method, and extend it to take classifier output into account.

The remaining participants in the SemEval task do not use a traditional MT decoder. Nevertheless, two of them achieve good scores. Rudnick et al. (2014), who finish first for the English – German pair,

built their own distinct log-linear model for the task, which combines various information sources. Amongst the features in this model are the translation probability and inverse translation probability, directly taken from a phrase-translation table built using GIZA++ (Och and Ney, 2000) and the Moses training scripts for phrase extraction using the *grow-diag-final-and* algorithm.

Silva-Schlenker et al. (2014) use a classifier-based approach using Timbl (Daelemans et al., 2009), which is the same memory-based classifier as used in the pilot study in Chapter 4, and upon which we build in this chapter as well. Silva-Schlenker et al. (2014) hypothesize that the relevance of the context words for determining a correct translation is proportional to their syntactic relatedness to the target, rather than their physical closeness in the sentence, a hypothesis also investigated by Haque et al. (2011b). Silva-Schlenker et al. (2014) run a dependency parser on the data and use syntax as a feature selector rather than as a feature itself. This results in a feature vector of relevant words. However, they do not find improvement over their baseline consisting of local context information (2 words left, 2 words right) using this method. This team participated only on the English – Spanish pair, but they do rank second. This reconfirms that there is merit in classifier-based approaches.

The SemEval participants employ various different methods to relate to the L2 context. Goutte et al. (2014) fully rely on the L2 Language Model, Hasler (2014) introduces an additional context-similarity feature to the log-linear equation at the heart of the decoder. They derive this from a phrase pair topic model, and find this improves results.

6.3 Methodology

From the systems participating in the SemEval task we observe that Statistical Machine Translation (SMT) techniques as well as classifier-based techniques are capable of achieving good performance on the L2 Translation Assistant Task. The various participants' systems can not be readily compared and assessed on the merit of their approach, as training data and training methods differ. In this study, we do have two comparable systems. We assess the impact of L2 context information in a purely classifier-based system, and in a MT-based hybrid system enhanced with similar classifiers.

It has to be strongly emphasised that our focus is on text data in its surface form, without any further associated linguistic information requiring external language-specific resources, such as taggers or parsers. We want to assess the efficacy of context information in its purest unmodified form as we aim for an approach that is as language-independent as possible. The inclusion of certain well-chosen linguistic features may likely achieve better results than our approach. However,

results from both Silva-Schlenker et al. (2014) as well as Rudnick et al. (2014) already show that intuitively promising linguistically-informed features, dependency features in this case, do not always lead to an improvement over local lexical context information. Unlike those studies, we do not introduce a new linguistically-informed feature, but we deliberately constrain ourselves to assess whether the context as-is has merit in the translation task.

This approach is motivated by the fact that local context information of surface forms proves a powerful cue in word sense disambiguation, a field of research with notable similarities to the task at hand. Often fairly small window sizes lead to the best results, as shown in Chapter 3. These good results were achieved using memory-based classifiers. We therefore bring these techniques to the present issue of translating L1 fragments in an L2 context, and investigate their efficacy.

To test whether we do not constrain ourselves too much with mere local context features, we also conduct a small experiment with global context features, i.e. identifying powerful keywords for a given translation from the entire sentential context.

6.3.1 Data & Software

The training data for our translation model is composed of two major parallel corpora joined together: Europarl v7 (Koehn, 2005) and OpenSubtitles 2012[1] (Tiedemann, 2012). Table 6.1 lists the size of these corpora.

Table 6.1: Corpora sizes for all of the language pairs

L1	L2	Sentences
English	Spanish	41,797,582
English	German	7,542,070
French	English	24,862,173
Dutch	English	26,347,136

For all our systems, a phrase-translation table was built using the Moses scripts, first invoking GIZA++ for word-alignment and subsequently performing phrase-extraction using the *grow-diag-final* algorithm. This phrase-translation table serves as the foundation for building the classifiers, as shall be explained in Section 6.4.

An L2 language model is generated using SRILM (Stolcke, 2002). We use a trigram language model with Kneser-Ney smoothing. The model is trained on the L2 side of the parallel corpus. It plays an essential role in the MT-based hybrid system, and can be enabled as an extra component in the classifier-based system.

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

Both systems are implemented in our software package *colibrita*.[2] All input and output data of our experiments is also available for download. [3]

6.3.2 Evaluation

The translation results are automatically evaluated against multiple human reference translations in the SemEval test set from Chapter 5. This test set explicitly takes alternatives into account; the evaluation metrics do so too. The evaluation metrics[4] have been shown in Section 5.4. When matching with multiple reference translations, the highest score is taken.

6.4 Classifier-based System

6.4.1 System Description

The classifier-based system is the system as described in Chapter 4. Recall that we employ classifier experts; a separate classifier for each possible input phrase, i.e. word or n-gram, and each classifier thus specialises in the translation of a single word or phrase, in all of the distinct contexts and with all of the different translations that it is seen with in the training data. In this section we recap how this system works, and add some details on its improved implementation and application in the present study.

Classifier experts are built for each L1 phrase that has multiple L2 translation candidates, i.e. are ambiguous, provided that the phrase pair exceeds a certain threshold. This threshold must be met by the product of the translation probability and its inverse, i.e. $P(source|target) \cdot P(target|source)$. In our experiments here, the threshold is set to 0.0001 to prune the worst candidates and keep file sizes and memory consumption manageable. Recall from Chapter 4 that phrases that are unambiguous (in the training data) are translated directly by a simple mapping table.

Given the phrase-translation table we use for training, the task of collecting the necessary training data for classification requires finding

[2] *colibrita* is written in Python 3, is open source (GPL v.3), and available from `https://github.com/proycon/colibrita/`. Version tag `0.3.1` is representative for the version used in this research and includes scripts for running the experiments. Colibita dependents on *colibri-core* and *colibri-mt*, both obtainable from the same github source, version tag `colibrita-v0.3.1` represents the state at the time of research.

[3] Freely available through Academic Torrents: `http://academictorrents.com/details/ab6e61059b7f7879e027ca33fb0f9e82980cd855`

[4] Although common in MT research, we do not report on the BLEU (Papineni et al., 2002) score. In Chapter 4 we already ascertained that this measure correlates well with our task-specific accuracy metrics, but it has little added value because most of the sentence is already translated and therefore BLEU scores are unnaturally high.

all positions in the corpus where the L1 and L2 phrase from each of the phrase pairs occur. When we have these positions we can extract the context information we need for the classifiers' feature vectors. This position data, however, is not present in the phrase-translation table itself. We therefore proceed to construct an index of all phrases on either side of the parallel corpus, including only those that occur in the phrase-translation table. This is done using *colibri-core*[5], introduced in Chapter 2, and constitutes our forward index. If both the source phrase and target phrase are found in a sentence pair, we assume we found a translation instance. This is an assumption because at this stage we do not have the actual word alignments loaded. If multiple translation candidates are present in the L2 sentence, the one with the highest probability according to the phrase table takes precedence. Our system could be improved by consulting the actual GIZA++ word alignments at this stage.

In addition to this forward index, *colibri-core* is also used to build a reverse index, i.e a mapping of positions to words, only for the L2 side of the parallel corpus in this case. This allows us to quickly extract local context information in the form of m words to the left and n words to the right. This is done using *colibri-mt*[6]. Each instance found is added to the memory of the classifier expert dedicated to the L1 phrase, in which the L2 context information constitutes the feature vector, and L2 phrase is the class label. This procedure potentially generates a large number of classifiers, which we constrain drastically by only generating those that will be actually needed in the test stage. When the memories are constructed for all, the actual training begins and proceeds independently for each of the classifier experts. For the variant of the IB1 algorithm as implemented in Timbl, this entails the computation of feature weights for each of the local context features.

The test phase is straightforward. Each test L1 fragment in L2 context is explicitly marked in the SemEval test data. Per sentence, we check if we have a classifier for the marked fragment, and if so, we extract the context features and pass it to the classifier. This produces a translation, or rather, a probability distribution of translation options. Prior to checking whether a classifier is present, we first check if the L1 fragment is in the direct mapping table established for unambiguous translations, and resort to that instead if found. It is possible that a certain fragment has not been seen during training and is neither in the simple mapping table, nor do we have a classifier for it. In such cases, we are unable to translate the fragment. This is one of the main weaknesses of this classifier-based approach we later attempt to mitigate using an MT approach.

[5] https://github.com/proycon/colibri-core
[6] https://github.com/proycon/colibri-mt

Recall that a statistical language model is implemented as an optional component of the classifier-based system. We also use it as a baseline to compare our system to Chapter 4.

Various parts of the search and classification process can in theory be parametrised with weights. We follow the practices established in Chapter 4: We assign equal importance to translation model and language model. We use default parameters[7] for Timbl, the k-Nearest Neighbour classifier, as parameter optimisation at this stage has been shown not to yield a positive impact and prone to overfitting in a comparable WSD setting, as shown in Chapter 3. Whereas we conducted automatic feature selection per classifier expert in Chapter 4, and found marginal improvements with it, we omit this in the present chapster to reduce experimental complexity. We deal with larger data now and it quickly becomes computationally prohibitive.

6.4.2 Experiments and Results

We conducted a series of experiments to test whether the conclusions from Chapter 4 hold on the more representative test set of the SemEval Task.

Two baselines were created. The first is a "most likely fragment" baseline (MLF). This baseline does not use any context information and merely selects the most frequent translation option for a given L1 fragment, according to the phrase-translation table. It is used as a control for our context-informed experiments, which in the pilot study surpassed this uninformed baseline, and we hypothesise the same will occur with the improved test set.

A second baseline is constructed by weighing the probabilities from the translation table directly with the aforementioned L2 language model (LM). This effectively adds an LM component to the MLF baseline. This LM baseline allows the comparison of classification through L1 fragments in an L2 context, with a more traditional form of context modelling (i.e. target language modelling) as is also customary in SMT decoders. The computation of the baseline proceeds as in Equation 4.1, with $score_T$ representing the normalised $p(t|s)$ score from the phrase-translation table rather than the classifier.

We experimented with several configurations for the local context window. Starting with one word left, one word right, up to three words left and right. We also included an asymmetric configuration of two words to the left and one word to the right. Omitting right context alltogether was already discarded in the pilot study.

Tables 6.2 and 6.3 show the results of these experiments, for each of the four language pairs in the SemEval task. Statistical significance

[7] $k = 1$, weighted overlap (`-m O`), and weighted using GainRatio (`-w gr`). Note that $k = 1$ in Timbl refers to the neigbours at the shortest distance, it may still contain multiple instances if they are at the same distance.

was computed on the Word Accuracy metric, using the paired t-test, an asterisk indicates significance with respect to the MLF baseline, and a ‡ indicates significance with respect to the LM baseline, both at $p < 0.05$.

Table 6.2: Results of classifier-based experiments for English – Spanish (left) and English – German (right)

System	Accuracy	W.Acc.
MLF baseline	0.544	0.651
LM baseline	0.643	0.720∗
l1r1	0.615	0.707∗
l1r1-lm	0.651	0.736∗
l2r1	0.629	0.717∗
l2r1-lm	**0.657**	**0.742**∗‡
l2r2	0.628	0.722∗
l2r2-lm	0.655	0.739∗
l3r3	0.627	0.715∗
l3r3-lm	0.649	0.729∗

System	Accuracy	W.Acc.
MLF baseline	0.571	0.627
LM baseline	0.605	0.657
l1r1	0.617	0.672
l1r1-lm	0.603	0.656
l2r1	**0.635**	**0.690**∗
l2r1-lm	0.613	0.665
l2r2	0.625	0.681∗
l2r2-lm	0.611	0.664
l3r3	0.620	0.672∗
l3r3-lm	0.615	0.667∗

Table 6.3: Results of classifier-based experiments for French – English (left) and Dutch – English (right)

System	Accuracy	W.Acc.
MLF baseline	**0.519**	**0.614**
LM baseline	0.507	0.596
l1r1	0.503	0.603
l1r1-lm	0.511	0.604
l2r1	0.503	0.601
l2r1-lm	0.511	0.604
l2r2	0.493	0.595
l2r2-lm	0.503	0.598
l3r3	0.499	0.600
l3r3-lm	0.503	0.598

System	Accuracy	W.Acc.
MLF baseline	0.419	0.480
LM baseline	0.370	0.445∗
l1r1	**0.437**	0.496‡
l1r1-lm	0.386	0.460‡
l2r1	0.435	**0.497**‡
l2r1-lm	0.386	0.461‡
l2r2	0.431	0.495‡
l2r2-lm	0.384	0.458
l3r3	0.419	0.485‡
l3r3-lm	0.378	0.454∗

The main conclusion from the pilot study was that classifiers informed by simple local context features outperformed both the non-context-informed baseline as well as the LM baseline. We see that this former conclusion holds for two out of four language pairs: English – Spanish and English – German. For the remaining two language pairs, it is either below baseline or fails to pass significance tests.[8] This puts the conclusion from the pilot study in question. Similarly, improvement over LM baseline is only observed in two out for four language pairs.

[8] All mentions of significance in this chapter are statistically backed using a paired t-test with $p < 0.05$

The results for English – Spanish support the idea of memory-based classifiers as a means to solve this task the strongest. We first perceive a significant gain in accuracy for the LM Baseline, compared to the MLF baseline, and all classifier-based systems again consistently improve over the MLF baseline. For improvement over the LM baseline we only have one system, the best one, which passes the significance test. The pilot study concluded that the language model and classifiers were able to complement each other, despite significant overlap in what they model. We do see that the best system for English – Spanish is the system that combines classifiers and adds a language model as well. The highest score is achieved for the asymmetric configuration with two words to the left, and one word to the right.

The English – German pair reconfirms the classifier's ability to beat the non-context-informed baseline, but adding a language model component proves to have little added value here. The LM baseline does not significantly improve upon the MLF baseline either.

For the last two language pairs, both translating to English, the classifiers make no significant impact over the MLF baseline whatsoever. We speculate that morphological richness may be a factor here: when translating from a morphologically simpler language to a morphologically richer[9] language, such as English to Spanish and English to German, the classifiers have a bigger role to perform and may prove beneficial. That being said, we can not jump to this conclusion too quickly, as the different sets do not just differ in language pair, but despite our best efforts to deliver a balanced test, also differ in difficulty, the English-Spanish set being the easier one, and the Dutch-English set being the hardest.

We do not at all observe a strong effect of the language model being complementary with the classifier system, as was demonstrated in the pilot study. When adding an LM component to our classifier-based system, we see a positive impact over both MLF and LM baseline for only the English – Spanish language pair. Although the classifiers model L1 fragments in and L2 context, and the LM models only L2 without integrating the translation step, these are different methods to the same end, and the models will inevitably overlap to a degree, especially if they model roughly the same size of local context.

Regarding the size of the local context window, we see best performance for small context sizes, with all of the winning scores in either *l1r1* (one word to the left, one word to the right) or *l2r1* (two words to the left, one word to the right). Results for *l3r3* consistently show a performance drop. However, the configurations are too similar to pass statistical significance tests.

[9] i.e. distinguishing a higher diversity of surface forms for a given lemma

6.4.3 Additional experiments

It may be argued that a trigram language model, as used in all our experiments, has little to offer over classifiers that model roughly the same size of local context window. We therefore conducted a small extra experiment for English – German, *l1r1*, with a 5-gram language model with Kneser-Ney smoothing. This does not result in a significant improvement over *l1r1-lm*, with a word accuracy of 0.6587, an absolute accuracy of 0.6052, and does not significantly differ from the *l1r1* configuration either (without language model).

One of the main points of contention for this present study is our emphasis on simple, and often even small, local context features. Intuitively, there is great appeal in incorporating global context keywords, i.e. finding high-frequency words in the whole sentential context that prove salient for a given translation pair. These can be incorporated into the feature vector using a bag-of-word configuration with binary values indicating presence of absence. Surprisingly, our studies with memory-based classification in word sense disambiguation in Chapter 3 show that these global keywords do not always have enough disambiguation power to beat local context. Our WSD2 system on SemEval 2013 data (Section 3.4) did not lead to an improvement over local context, whereas our very similar and earlier UvT-WSD1 system on SemEval 2010 data (Section 3.3) did. We suspect this may be attributed to increased sparsity due to the difference in data, as the systems are as good as identical. We put this to the test again by once again conducting an extra experiment using global context keywords in the same fashion (Ng and Lee, 1996). We find results significantly below our MLF baseline, with a drop in accuracy and word accuracy to 0.4930 and 0.5686, respectively.

6.5 MT-based hybrid System

6.5.1 Introduction

During this research, thanks to insights from SemEval task participants Hasler (2014) and Goutte et al. (2014), it became apparent that we could improve results by switching to an SMT-based system. We are not parting from using classifiers to disambiguate L1 fragments in an L2 context to the proper L2 translation, but we are integrating these classifiers in a full statistical machine translation framework, analogous to Haque et al. (2011b). The idea is that SMT has no facilities to explicitly model context, let alone cross-lingual context. The classifiers fill this void and we aim to assess whether incorporating classifiers has merit.

6.5.2 System Description

The training and construction of the classifiers does not differ from the classifier-based system. For the phrase-translation table we already used the pipeline offered by SMT system Moses (Koehn et al., 2007b), so that remains identical as well.

The difference lies in the test phase, where we actually use the Moses SMT decoder to do the job, rather than just invoking classifiers. Moses cannot explicitly model context in its translation model, so we still rely on the classifiers for that. In order to integrate these classifiers into the decoder, we make use of Moses' ability to read XML input in which translation for parts can be explicitly provided. An example of such XML input was shown in Example 6.2.1 in Section 6.2.

Because we use Moses, we have an extra parameter tuning step to perform prior to testing. We use Minimum Error Rate Training (MERT) (Och, 2003) to tune the weights that drive the log-linear combination at the heart of the decoder, determining precisely how much importance to give to the various scores of the translation model, the language model, and other models. We tune our system on data from the News Commentary corpus[10] for three out of the four language pairs: English – Spanish, English – German and French – English. For Dutch – English we have to resort to another corpus, a collection of transcribed bilingual TED talks compiled for the IWSLT 2012 Evaluation Campaign[11]. We use 10,000 sentence pairs for each. The choice for these corpora, as opposed to a subset of Europarl or Opensubtitles, is motivated by our aim for more generalisation.

When obtaining a test sentence in L2 with a marked fragment in L1, we first pass the fragment through the classifier, if a classifier exists for the fragment. The classifier outputs a distribution of translation options, with associated confidence scores. We can pass multiple translation options with associated probabilities to Moses with XML syntax, consider the excerpt in Example 6.5.1, in which three translation options are received from the classifier and passed to the Moses decoder using XML:

Example 6.5.1

```
<w translation="es">es</w><wall/>
<w translation="importante">importante</w><wall/>
<w translation="cuidarse">cuidarse</w><wall/>
<w translation="y">y</w><wall/>
<f translation="la␣única||la␣única␣forma||la␣única␣manera"
 prob="0.555556||0.333333||0.111111">the only way</f><wall/>
<w translation="es">es</w><wall/>
```

[10] Part of the WMT 2013 shared task training data, http://www.statmt.org/wmt13/translation-task.html

[11] http://hltc.cs.ust.hk/iwslt/index.php/evaluation-campaign/ted-task.html

The Moses decoder now deals with the scores derived from the classifier, rather than those from the phrase translation table, and is then mainly driven by the language model, as the translations are fixed.

The scores from the classifier are much coarser than those from the phrase-translation table, as those in the phrase-translation table consist of four components,[12] and we are tweaking only a single component, namely $p(t|s)$. As this would result in a skewed integration of classifier scores, we implement a "weighted" method that performs part of the decoder's log-linear equation, parametrised by the weights obtained from MERT, and passes this weighted result to Moses using the XML syntax. This workaround is necessary because the XML input method itself is limited and offers no facilities to specify just a single component of the translation model.

To assess whether the information from our classifiers outperforms a baseline without such classifiers, we establish a new MT-based baseline that does not incorporate classifiers and just passes the fragment as-is for the decoder to translate, as shown in Example 6.5.2:

Example 6.5.2

```
<w translation="es">es</w><wall/>
<w translation="importante">importante</w><wall/>
<w translation="cuidarse">cuidarse</w><wall/>
<w translation="y">y</w><wall/>
the only way<wall/>
<w translation="es">es</w><wall/>
```

From our experiments it becomes apparent that it is hard to surpass this baseline. We therefore added another configuration that uses the classifier output in a first-past-the-post fashion; if a classifier makes a prediction that reaches a certain threshold, we use that classifier prediction, otherwise we use Moses and do not pass it any information from the classifiers.

In order to pass numerous test fragments to Moses at high speed, we run Moses as a server, and use our software *colibrita* to connect to it after having classified each fragment.

6.5.3 Experiments & Results

Tables 6.4 and 6.5 show the results of these experiments, for each of the four language pairs in the SemEval task. The *"mosescut"* configurations, refer to the first-past-the-post approach, with the number indicating the threshold. It is the coarsest form of integration. Next is the *"moses"* configuration which just passes the classifier score to

[12] conditional probabilities $p(s|t)$, $p(t|s)$ and corresponding lexical weights

the decoder. Last, *"mosesweighted"* is the most fine-grained form of integration and computes part of the log-linear equation prior to passing it to the decoder. For reference we include the MLF baseline as well as the best scoring classifier-based system from the earlier tables. Statistical significance was computed on the Word Accuracy metric, using the paired t-test; an asterisk indicates significance with respect to the MLF baseline, and a ‡ indicates significance with respect to the MT baseline, both at $p < 0.05$.

Table 6.4: Results of MT-based experiments for English-Spanish (left) & English-German (right)

System	Accuracy	W.Acc.
MLF baseline	0.544	0.651
MT baseline	0.651	0.750∗
l2r1-lm	0.657	0.742∗
l1r1-moses	0.625	0.736∗
l1r1-mosescuto.7	0.657	0.761∗
l1r1-mosesweighted	0.610	0.724∗
l2r1-moses	0.647	0.755∗
l2r1-mosescuto.7	**0.669**	**0.774**∗‡
l2r1-mosesweighted	0.635	0.744∗
l2r2-moses	0.649	0.756∗
l3r3-moses	0.657	0.756∗

System	Accuracy	W.Acc.
MLF baseline	0.571	0.627
MT baseline	**0.689**	0.745∗
l2r1	0.635	0.690
l1r1-moses	0.603	0.672∗‡
l1r1-mosescuto.7	0.681	0.739∗
l1r1-mosesweighted	0.569	0.651‡
l2r1-moses	0.619	0.685∗‡
l2r1-mosescuto.7	**0.689**	**0.748**∗
l2r1-mosesweighted	0.599	0.675∗‡
l2r2-moses	0.617	0.684∗‡
l3r3-moses	0.633	0.693∗‡

Table 6.5: Results of MT-based experiments for French-English (left) and Dutch-English (right)

System	Accuracy	W.Acc.
MLF baseline	0.519	0.614
MT baseline	**0.620**	**0.742**∗
l1r1-lm	0.511	0.604
l1r1-moses	0.568	0.709∗‡
l1r1-mosescuto.7	0.590	0.721∗‡
l1r1-mosesweighted	0.562	0.703∗‡
l2r1-moses	0.576	0.711∗‡
l2r1-mosescuto.7	0.588	0.717∗‡
l2r1-mosesweighted	0.568	0.706∗‡
l2r2-moses	0.572	0.708∗‡
l3r3-moses	0.570	0.707∗‡

System	Accuracy	W.Acc
MLF baseline	0.419	0.480
MT baseline	0.534	0.677∗
l2r1	0.435	0.497
l1r1-moses	0.487	0.647∗‡
l1r1-mosescuto.7	**0.548**	0.686∗
l1r1-mosesweighted	0.476	0.634∗‡
l2r1-moses	0.495	0.654∗‡
l2r1-mosescuto.7	**0.548**	**0.687**∗
l2r1-mosesweighted	0.489	0.647∗‡
l2r2-moses	0.495	0.653∗‡
l3r3-moses	0.497	0.654∗‡

First, we observe that the MT baseline is considerably higher compared to the classifier-based systems, and significantly higher than the MLF baseline on all accounts. It surpasses all of the classifier-based scores consistently for all language pairs. The MT baseline seems hard to beat, as both English – German and French – English do not achieve results above baseline at all and the positive impact for Dutch-

English fails to pass the significance threshold. Only English – Spanish shows a clear positive impact of integrating context information using classifiers, but only using the coarsest *"mosescut"* system, that places all bets on a single classifier prediction. The most fine-tuned adjustment of the weights, *"mosesweighted"*, consistently underperforms, also for the higher configurations omitted from the tables.

We experimented with various different threshold values for the *"mosescut"* configuration, although these are omitted from the tables for brevity. We found a threshold of 0.7 to be performing amongst the best, for all language pairs.

This MT-based hybrid approach may seem to challenge our conclusion that classifiers modelling context improve over a non-context informed baseline. But that is not the case. The Moses baseline here is context-informed; it makes use of a language model. We conducted a modified MT-baseline experiment on English – Spanish with the language model disabled. Word accuracy plummeted to 0.44, even below the MLF baseline. What is challenged, however, is the role classifiers modelling source-side context can play when integrated in an SMT decoder, and to what extent explicit modelling of context information is not already implicitly captured by the language model. Still, however, we see classifiers making a positive impact for two out of four language pairs.

6.6 Discussion & Conclusion

Table 6.6, shows the results of our best classifier-based and best MT-based hybrid run[13], compared to the other six participants in the SemEval task as described in Chapter 5.

Table 6.6: Highest word accuracy per team, per language pair. The best score for each language-pair is marked in bold.

Team	**en-es**	**en-de**	**fr-en**	**nl-en**
MT-based	0.774 (3^{rd})	**0.748** (1^{st})	0.742 (2^{nd})	0.687 (2^{nd})
Classifier-based	0.742 (4^{th})	0.690 (4^{th})	0.610 (4^{th})	0.497 (4^{th})
CNRC (Goutte et al., 2014)	0.745	0.717	0.694	0.610
IUCL (Rudnick et al., 2014)	0.720	0.722	0.682	0.679
UEdin (Hasler, 2014)	**0.827**	-	**0.824**	**0.692**
UNAL (Silva-Schlenker et al., 2014)	0.809	-	-	-
Sensible (Tan et al., 2014)	0.351	0.233	-	-
TeamZ (Gupta, 2014)	0.333	0.293	-	-

We have not been able to surpass the score of the winning system by Hasler (2014). What is different between our approaches? Rather than incorporating classifiers modelling local context, Hasler (2014)

[13] For French-English, the best run is the baseline system

add a context similarity feature derived from a topic model that learns topic distributions for phrase pairs, which improves performance over their baseline. Their baseline is constructed in the same way as ours, except that it uses a different parameter tuning method (the MIRA Tuning Algorithm) instead of MERT. For English – Spanish they do not surpass their own baseline, which attains an even higher word accuracy of 0.839. For French – English their baseline attains a score of 0.823 and for Dutch – English 0.686, which is the score our best MT-based hybrid system attains as well. The difference between our respective baselines can be attributed to different training and tuning corpora. Our training data includes the OpenSubtitles corpus and is therefore much larger, however, they train a larger language model on different data than the translation model. This may likely account for better generalisation and better scores. We confirm this by running an enriched MT-baseline experiment on English – Spanish with a substantially larger five-gram language model[14], and indeed obtain scores that are higher than our standard MT-baseline.

This confirms the important role of the language model. We already saw that for the language pairs translating from English, the language model baseline beats the MLF baseline. Language modelling in the MT decoder is clearly superior to our simpler implementation in the LM baseline, as we see the MT baseline consistently above the LM baseline. We conclude that it is very hard to attain results above this MT baseline, for an MT-based system enriched with classifiers modelling just local context.

The role of the language model can also be assessed by inspecting the weight it has received from the MERT parameter tuning algorithm. We express this score relative to the weight of the $p(t|s)$ component of the translation model[15] and obtain 1.12 for English – Spanish, 1.37 for English – German, 0.62 for French – English and 0.90 for Dutch – English. Overall, this means the language model weight and translation model weight are fairly balanced and, as expected, both are essential components. The language model carries more weight when translating from English to Spanish or German, which we believe is due to the target languages being morphologically richer and the stronger role the LM then plays to select the proper morphological variant given the context.

Our best performing system is the first-past-the-post *"mosescut"* system, with slight but significant gains above the MT baseline for English – Spanish. To gain some insight in the data, we take the English – Spanish output of the system *l2r1-mosescuto.7* , and investigate the instances where there is a difference with the MT baseline,

[14] Trained on four corpora: Europarl, OpenSubtitles, MultiUN (http://www.euromatrixplus.net/multi-un/) and NewsCrawl (http://www.statmt.org/wmt13/translation-task.html)

[15] The $p(t|s)$ component of the translation model consistently has a higher weight than the other (three) components

which amounts to 70 instances (14%). The remaining instances were translated the same. We decided to perform a limited[16] human evaluation of these seventy instances, to form an impression whether the automatic evaluation metrics indeed correspond with human judgement, and whether our system output can rightfully be considered to be better than the MT baseline. Out of the 70 differing instances, the baseline was deemed to have the better solution for 14 of them, and the *l2r1-mosescuto.7* system had the better solution for 28. The remaining 28 instances were judged to be equally good or equally bad. This is in line with the automatic evaluation metrics but illustrates the very small margin we are working with.

Examples 6.6.1, 6.6.2 and 6.6.3 show a comparison of some interesting output sentences where the role of context made a noticeable difference. We also added the MLF baseline, which matches the MT baseline in a lot of the 28 cases where the mosescut system was found to perform better.

Example 6.6.1
MLF baseline: *Le aconsejo este vino* ***rojo*** *de Argentina .*
MT baseline: *Le aconsejo este vino* ***rojo*** *de Argentina .*
l2r1-mosescuto.7: *Le aconsejo este vino* ***tinto*** *de Argentina .*

Example 6.6.2
MLF baseline: *El embajador* ***firmado*** *hoy el acuerdo .*
MT baseline: *El embajador* ***firmado*** *hoy el acuerdo .*
l2r1-mosescuto.7: *El embajador* ***ha firmado*** *hoy el acuerdo .*

Example 6.6.3
MLF baseline: *En la actualidad se cuenta con la medicina* ***más avanzados*** *.*
MT baseline: *En la actualidad se cuenta con la medicina* ***más avanzados*** *.*
l2r1-mosescuto.7: *En la actualidad se cuenta con la medicina* ***más avanzada*** *.*

Still, we find examples of fragments where the wrong morpho-syntactic form was selected. In the English – German language pair, this selection is complicated by the fact there are also cases in addition to gender, and number, which are typically harder to resolve from a small local context. Example 6.6.4 shows such an error where our system, as well as the baseline, selected a dative form rather than the nominative:

Example 6.6.4
Reference: *Ich habe einen guten Chef und* ***das Büro*** *liegt in der Nähe von*

[16] Single person, non-blind

meinem Haus .
l2r1-mosescuto.7: *Ich habe einen guten Chef und* ***dem Büro*** *liegt in der Nähe von meinem Haus .*

We can conclude that L2 context information provides valuable cues for the translation of L1 fragments in an L2 context. However, we revisit some the conclusions from the pilot study of Chapter 4, which concluded that memory-based classifiers are a viable means of solving this translation task. We do obtain significant results over a non-context informed baseline with a purely-classifier based approach, but for just two out of four language pairs. Also, complementing the classifier model with an extra language model component only leads to improved results for a single language pair, contrary to earlier findings. We hypothesized there is a large overlap between what is modelled by the LM and the classifiers.

Standard statistical machine translation proves to be a far better solution for the task at hand, even though many translations in a given sentence are already fixed and only a single L1 fragment remains. The Moses SMT decoder can be used in such a fashion, by using the XML input facility and passing the L2 context translations explicitly. We extensively studied the incorporation of memory-based classifiers in this framework, seeking to combine the translation step and context-awareness in a single model, but found little to no positive impact; only a first-past-the-post system where a classifier prediction above a certain majority threshold overrides the invocation of the MT decoder has a positive effect for one out of four language pairs.

Why do we not attain better results with the classifiers? In the MT framework, our analysis has shown the classifiers do not effectuate much change after the translation model and language model have already done their job. This implies that the classifier model overlaps considerably with what the translation model and language model already model jointly. A contributing factor in this is also that all models were deliberately trained on the same training data in our experiments, as we wanted a fair comparison and assess the merit of the technique as-such. It is still conceivable that classifiers make an impact if they are modelled on different data. Nevertheless, we have a bit of a thirdwheel scenario, where adding a classifier component to explicitly model context in as part of the translation model provides no clear benefit over the already existing combination of translation model and language model. Our classifiers, moreover, may be sensitive to the translation direction as they perform better mainly for directions that proceed from morphologically simpler to morphologically richer languages: our positive results are all for translation from English to Spanish and English to German.

A simpler and more straightforward alternative to attain better results in the task in general is to improve the language model, by

training it on more and different data and increasing its potential for generalisation, as done for example by Hasler (2014), and possibly by optimising its parameters. Classifiers may be worth reintroducing only when incorporating linguistic information from the context, that can not be directly taken into account by the SMT decoder. Such approaches, however, often do not produce the expected increase in translation quality either (Silva-Schlenker et al., 2014; Rudnick et al., 2014).

In a final experiment, we show that there is still room for improvement with regard to choosing the right morpho-syntactic variant in the given context. We apply stemming to both our best system output for English – Spanish as well as to the reference translation. We subsequently see a word accuracy increase from 0.77 to 0.81. From this we can derive that even our best system fails to select the correct morpho-syntactic surface form for a number of cases.

As for the classifiers, minor gains may still be expected when implementing automatic feature selection as in Chapters 4 and 3. This would allow different classifiers to use different, optimised, local context configurations, rather than using a single configuration for all classifier experts. That approach, however, proved not to scale well and could not cope with the amount of data we use now.

In summary, the main conclusions to draw from this study are that **1)** L2 context information provides valuable cues for translating L1 fragments in L2 context, also without additional linguistically-informed features; **2)** classifiers modelling local L2 context information have little to no added value to the components already present in an SMT framework and **3)** We reconfirmed that a traditional phrase-based SMT system is a good solution in the L2 Translation Assistance task.

7

UNINFORMED SOURCE-SIDE CONTEXT IN STATISTICAL MACHINE TRANSLATION

In this chapter we present a series of experiments focusing on the modelling of source-side context to improve phrase-based Statistical Machine Translation. We attempt to independently reproduce a line of existing research and test whether considering source-side context information directly in the translation model has a positive effect on translation quality. We furthermore investigate various ways in which discriminative classifier-based models can be integrated into Statical Machine Translation. We use proven techniques from Word Sense Disambiguation, effectively integrating these techniques in Statistical Machine Translation. Our approach is language-independent and knowledge-poor or uninformed: we do not employ any explicit linguistic features computed by part-of-speech taggers, word sense disambiguation systems, supertaggers, or parsers, as used by previous work. We observe improvements in translation quality by our method, but only clearly so for domain-specific corpora with a high degree of formulaic language. Lowering the expectations raised by previous work, we conclude that the explicit modeling of source-side context information does not generally add much information to that already implicitly available in the SMT decoding process.

7.1 Introduction

In phrase-based Statistical Machine Translation the problem of translating an input sentence from a source language to a target language is implemented as a game of probabilities and a search for the most probable translation option. These probabilities are expressed in a number of models that specialise in a certain aspect relevant to the translation process. The method is called "phrase-based" because phrases are the building blocks of the translation model, generalising earlier approaches in Statistical Machine Translation based on words. Phrases in this sense are sequences of one or more words, i.e. n-grams of variable length (including unigrams). They are not constrained to form a proper linguistic constituent of any kind.

We introduced phrase-based SMT in Chapter 1, Section 1.2. To recapitulate, two models are at the core of phrase-based SMT: first there is the translation model which maps the translation phrases in the source language (S) to phrases in target language (T). Each mapping between phrases is associated with a vector of probabilities, $p(phrase_S|phrase_T)$ and $p(phrase_T|phrase_S)$. They can be seen to

model the notion of *"semantic faithfulness"*; if a phrase is translated from one language to another, the meaning should be preserved as accurately as possible.

The second core model is the language model. This model is monolingual in nature and models the target language. It models what words are likely to follow others and can be interpreted as modelling the *"syntactic fluency"* notion of translation; a translation should be in a natural word order and be a typical sequence of words for the target language.

A Machine Translation *decoder* optimises a log-linear model of the combination of these two, and optionally other models as well. Given an input sentence in the source language, it searches through a vast space of all "possible" translation options, most nonsensical, for a path maximising the probabilities according to each of the models, taking into account different weights they may be assigned.

This chapter focusses on the role of additional surface-form source-language context information in the SMT process. The Language Model models context for the target language. Yet in SMT there is no component modelling context for the source language, whereas intuitively source-side context may provide a powerful cue for translation. Consider the word "bank" and its Spanish translation in examples 1 and 2.

(1) **English:** I don't trust the bank.
Spanish: No me fio del banco.

(2) **English:** The boat headed towards the bank of the river.
Spanish: El barco se dirigió hacia la orilla del río.

The same English word, "bank", may express multiple lexical semantic senses, some of which are expressed by different words in Spanish. Source-side context information may provide valuable clues to what sense is being intended, and therefore what translation is correct. The words "boat" and the phrase "of the river" in example 2 provide clues that we are using bank in its aquatic sense. Example 1 is less obvious, but the word "trust" could be seen to be a clue for "bank" denoting a financial institution.

We hypothesise that the inclusion of source-side context information in the translation model of a phrased-based SMT system improves translation results, as the context helps in providing a more accurate disambiguation. A counter-hypothesis to this would be that although source-side context information is not modelled explicitly in this paradigm, it is implicitly captured by the combination of translation model and target language model, and explicit modelling has no added value.

There is an obvious overlap between what we aim to do here and the field of Word Sense Disambiguation (WSD). We effectively test

an integration of proven techniques from WSD in Statistical Machine Translation, and apply these to phrases rather than just words.

WSD systems often employ a variety of linguistic features. The focus of our study, however, is to be independent of the explicit computation of linguistically abstract features with language-dependent tools such as lemmatisers, part-of-speech taggers, supertaggers, or dependency parsers. Integration of such techniques has already been researched by Haque et al. (2011a). We attempt to assess the merit of source-side context information as purely as possible.

Not introducing extra data for the translation system means our goals have to be set more modest as well. We may not expect the same gains as are achieved by introducing extra data from linguistic preprocessors.

In the next section, we will sketch an overview of previous research. Section 7.3 describes how our system is set up. In Section 7.4 we introduce the data sets used and the experiments we conducted. The results are presented, analysed and discussed in Section 7.5. In Section 7.6 we come to the final conclusion.

7.2 Previous research

The idea to integrate WSD approaches in SMT is not new, nor is the idea to use source-side context information to disambiguate in translation. Various studies have been conducted, offering mixed results. In the early days of SMT, Varea et al. (2002) explicitly modelled source-side context in a maximum entropy model for word-based SMT, and report slightly improved error rates on a translation task.

Carpuat and Wu (2005) were the first to tackle the question whether full-scale WSD models were beneficial to translation quality when integrated in SMT systems, and thus their work forms an important foundation for our own study. Their approach uses an ensemble classification model that integrates position-sensitive, syntactic, and local collocational features, a combination with proven merit in competitive WSD tasks. This includes linguistic features such as part-of-speech tags and lemmas, as well as more complex syntactic relations. They test on a single Chinese-to-English test set only, and only evaluate with BLEU, which leaves open the question whether their conclusions would hold on different language pairs, test sets, and using different evaluation metrics. Their approach is word-based and the level of integration is also limited.

Carpuat and Wu focus on the WSD model rather than on the SMT model, whereas we place more focus on the SMT model and the integration method, and keep our WSD model relatively simple.

Despite their efforts, they reach the surprising conclusion that inclusion of WSD techniques does *not* yield better translation quality. Will these results hold in a more modern phrase-based Statistical

Machine Translation approach? Indeed, two years later they expanded their study to full phrasal units (Carpuat and Wu, 2007b) and, for the first time, found results that did support the hypothesis that SMT benefits from the integration of WSD techniques. They now focus on better integration in *phrase-based* SMT: "Rather than using a generic SenseEval model as we did in Carpuat and Wu (2005), here both the WSD training and the WSD predictions are integrated into the phrase-based SMT framework." (Carpuat and Wu, 2007b). They also broaden their use of evaluation metrics, yet still test on only Chinese to English.

The work of Giménez and Màrquez (2007), from the same year, follows a similar strategy. They use support vector machines to predict the phrase translation probabilities for the phrase-translation table component of SMT, rather than relying on the context-unaware Maximum Likehood Estimate the statistical process produces. The feature vector for their classifiers consists of local context and global context features. In addition to the surface forms of the words, they rely on shallow linguistic features such as part-of-speech tags and lemmas. They conduct a manual evaluation judging fluency and adequacy, and conclude that considering context improves semantic adequacy, yet does not benefit syntactic fluency. They remark that the integration of the classifier probabilities in an SMT framework needs further research, which is something that will indeed be a focus in our present study.

The year 2007 saw a culmination of various studies integrating WSD techniques in SMT using classifiers. A third study in this trend was Stroppa et al. (2007). They focus on the word form, as does this present study, and add only part-of-speech features. On a dataset from the IWSLT 2006 challenge, drawn from a corpus of Basic Travel Expression, for Chinese–English and Italian–English, they achieve a significant improvement for the former language pair, whereas the BLEU score for the latter language pair fails to pass the significance test. We attempt to reproduce the Chinese–English experiment in this study, Carpuat and Wu (2007b) also based their research on the same data.

Source-context aware translation has also been attempted outside of the predominant SMT paradigm. For instance, van den Bosch and Berck (2009) implement a form of example-based machine translation that is word-based and relies chiefly on classifiers for the translation model component. Two studies derive from the same concept while transcending a word-based paradigm: van den Bosch et al. (2007b) use chunks delimited by common markers, and van Gompel et al. (2009) attempt a full extension to phrases similar to SMT. Although positive results are achieved in the latter study, it does not rival state-of-the-art SMT.

The most complete study we build upon is Haque et al. (2011a), which in turn draws from the majority of the aforementioned studies, and provides an extensive comparison between them. Their study finds that including linguistically-informed contextual features produces improvements, but not always; also, different contextual features have different, unpredictable results, some of which positive. The main contrast between our study and theirs is that Haque et al focus on a variety of linguistically-informed contextual features, whereas we depart from a language-independent angle, i.e. we do not presuppose the presence of natural language processing tools for computing linguistically abstract features. Furthermore, we intend to settle the conflicting reports on whether this may lead to an improvement in translation quality. A notable focus in our study will be possible methods of integrating the classifier probabilities in SMT, as recommended also by Giménez and Màrquez (2007). Furthermore, we take into account and compensate for the instability of parameter optimisation techniques.

7.3 Methodology

Like most of the previous work, we approach the machine translation problem in a phrase-based fashion, which has superseded the simpler word-based paradigm for quite some time. This means that phrases, defined as a sequence of one or more words (that need not form a linguistic entity in any way), form the basic building blocks of our translation model. The problem of translating a sentence is decomposed into the problem of translating its phrasal sub-parts and combining the results in the best order.

In describing our methodology, we first focus on the problem of phrasal translation, adding in the source-side context component. This shall be done using discriminative classifiers. Then we address how this can be integrated into a phrase-based Statistical Machine Translation decoder, which takes care of the ordering aspect.

7.3.1 Modelling source-side context with classifiers

In line with several previous studies (Haque et al., 2011a; van Gompel et al., 2009; Stroppa et al., 2007; van den Bosch et al., 2007b), we make use of memory-based classifiers to build a translation model informed by source-side context information. More specifically, we will be using IB1 (Aha et al., 1991), an implementation of k-Nearest Neighbour classification; IGTree (Daelemans et al., 1997), an optimised and lossless tree-based approximation thereof; and TRIBL2, a hybrid between IB1 and IGTree.

These algorithms are all implemented in the software package TiMBL (Daelemans et al., 2009)[1] and are well-suited for symbolic data and highly multivariate class spaces. Moreover, memory-based classification has been a successful method in the field of Word Sense Disambiguation (Hendrickx and van den Bosch, 2001), and as demonstrated as well in Chapter 3.

When speaking of the k nearest neighbours in the implementation of IB1, IGTree and TRIBL2; we are actually referring to the neighbours at the nearest distance k. So even with $k = 1$ we may be talking about multiple data points that are all at equal distance.

IGTree compresses the instance base into an ordered decision tree structure at training time, and issues look-ups in this tree structure at test time. Unlike other top-down induced decision tree methods such as C4.5, features in IGTree are tested in a fixed order, computed only once for all features. This order is determined using metrics such as *information gain* or *gain ratio*[2]. They determine the relative informativeness or disambiguating power of each feature and provide a ranking of all features.

IGTree's performance relies on the differences in gain ratio between features. If these are small then IGTree may perform significantly below IB1 (Daelemans et al., 2009). A hybrid approach called TRIBL2 (Daelemans et al., 2009) starts out with IGTree and switches to a IB1 variant when a value mismatch occurs in the tree. In this study, we therefore opt to use TRIBL2 over plain IGTree, but only when using IB1 would have a prohibitively large impact on performance.

In our classifier-based translation model we model the probability of a target phrase (t) given a source phrase (s) and context information (C). We can express this as $p(t|s, C)$. Stroppa et al. (2007) state that direct estimation of this probability using relative frequencies would result in overestimation of large phrases, and that therefore a smoothing factor is required. They proceed to say that through memory-based classification they introduce precisely such a smoothing factor implicitly. Given a source phrase and context information, the classifier yields classes corresponding to target phrases, with an associated weight. After normalization, these can be considered a posterior distribution of target phrases.

We focus on the modelling of local context, i.e. words in the immediate vicinity of the source phrase. Take w_0 to be the first word of the source phrase s, then for a local context size of n words to the left and m to the right we construct the feature vector C as follows[3]:

$$C = \langle w_{-n}..w_{-1}, s, w_{|s|+1}..w_{|s|+m} \rangle \tag{7.1}$$

Now there are two ways in which we can construct a classifier:

[1] https://languagemachines.github.io/timbl, version 6.4.5 is the version we used.
[2] This is the default that is also used in our experiments
[3] For context words out of the sentence's bounds, placeholders are used instead

- **Monolithic classifier** – One aggregated classifier for all source phrases. For this type of large-scale classifier we use the TRIBL2 algorithm, as the IB1 algorithm would be prohibitively slow.
- **Classifier experts** – One classifier per source phrase. For these smaller-type classifiers we use the IB1 algorithm.

In this study we will use and compare both methods, which is, for the task at hand, the first such a comparison in the literature as far as we know.

For the monolithic classifier, the first feature in the ranking will always be s. Nevertheless, it is quite conceivable that a match for the context is not found and the classifier proceeds to match on another feature. To prevent situations in which the classifier falls back to a completely different source phrase, and thus comes up with unrelated translation options, we enforce that the source phrases need to match, which is what Stroppa et al. (2007) do as well.

For the classifier experts, on the other hand, the source phrase carries no discriminative power, as it is shared amongst all instances. We therefore omit it from the feature vector.

7.3.2 Training

The translation model is trained on parallel corpus data. We follow a common MT pipeline, and additionally derive classifier training data from the resulting phrase-translation table and the parallel corpus.

Given a tokenised and sentence-aligned parallel corpus, we iteratively learn word alignments using GIZA++ (Och and Ney, 2000). Then we identify and extract phrase pairs using the *grow-diag-final* algorithm (Och and Ney, 2003). The result is a phrase-translation table mapping source phrases to target phrases, along with associated scores which we will discuss in the next section.

The translation model would be finished if we would want to leave it to be uninformed about source-side context. We have some additional steps to perform to train our context-informed classifiers. We take the phrase-translation table, along with the parallel corpus, as a basis for extracting training instances.

We build indexed pattern models of all source phrases and target phrases that occur on their respective side of the parallel corpus, and which also occur in the phrase-translation table. An indexed pattern model maps each distinct phrase to the locations in the corpus where it occurs. Additionally, a reverse index is included in the model for the target-side of the corpus, which maps any given $(sentence, token)$ position to a set of phrases that begins at that position. This is computed

using the software package *colibri-core*,[4], as introduced in Chapter 2, which takes care of a losslessly compressed in-memory representation for all phrases, and allows us to cope with large corpora. Given these two pattern models M_{source} and M_{target} we can extract the context for each phrase pair quickly and efficiently, as shown in simplified form in Algorithm 3.

Algorithm 3 Algorithm for feature extraction for training classifiers. Take n again to be the left context, m to be the size of the right context, and $w(i,j)$ to denote the word in the source corpus in sentence i, token j. The vector C represents the context information and constitutes the feature vector. The algorithm will return a list containing two-tuples (C,t).

```
instances ← []
for (s ∈ M_source, t ∈ M_target) do
    for i ∈ (M_source[s] ∩ M_target[t]) do
        for j ∈ M_source[s][i] do
            C ← ⟨w_{i,j−n} ... w_{i,j−1}, s, w_{i,j+|s|+1} ... w_{i,j+|s|+m}⟩
            INSTANCES.APPEND((C, t))
        end for
    end for
end for
return instances
```

This algorithm is implemented in *colibri-mt*.[5]

The returned instances can be stored directly, either in a single model for the monolithic approach or in separate models for each s for the classifier expert approach. A final training phase then computes the feature ranking and transforms this data into the instance base format required for TiMBL.

It may well happen that either 1) an (s,t) pair only occurs once, or 2) a pattern s occurs in multiple contexts but all map to the same t. In such cases, a context-informed classifier has no added value and therefore such instances are omitted from the training data, i.e. they are not included in case of the monolithic classifier or a classifier expert for them is not generated, meaning the classifier bypass is not needed for such instances and normal behaviour suffices.

7.3.3 Integration in an SMT Decoder

The task of an SMT decoder is to find the best translation among a pool of possible translation hypotheses. The best translation hypothesis is

[4] https://proycon.github.io/colibri-core, release v2.1.2 represents a fair state of the software version during this study, although some experiments were conducted with older versions

[5] https://github.com/proycon/colibri-mt
version v0.2 represents a fair state of the software version during this study.

the hypothesis that maximises a log-linear combination and is sought after in a beam-search algorithm. This log-linear combination draws from various models, such as a translation model (i.e. the phrase-translation table), a target-language model, and optionally additional models such as a distortion model and a word-reordering model.

As you may recall from Chapter 1, the SMT model is generally expressed as in Equation 7.2, taking sentence e to be the translation in the target language, and f to be the sentence to be translated, in the source language.

$$\arg\max_e p(e|f) = \arg\max_e p(f|e)p(e) \tag{7.2}$$

Bayes' rule inverts the problem into two factors[6], the former corresponding to the translation model, and the latter corresponding to the language model.

The translation model is a mapping of the set of source phrases (S) to the set of target phrases (T). Each phrase pair (s, t) where $s \in S$ and $t \in T$ is described by a score vector indicating the likelihood of translation. This score vector most notably consists of the probabilities $p(s|t)$ and $p(t|s)$. In addition, the lexical weighting probabilities $lex(s|t)$ and $lex(t|s)$ express the probability of a phrase pair word-by-word, and are often included as components in the score vector. During decoding, the total score of the translation model and other models is expressed as a log-linear combination, in which different weights can be assigned to each of the components of the score vector. These weights are hyperparameters to the task and are typically *optimised* automatically on development data using for instance Minimum Error Rate Reduction (MERT) (Och, 2003).

The probability $p(t|s)$ is the one we are interested in. Recall that the classifiers seek to model $p(t|s, C)$, where C constitutes the vector of context information. The hypothesis under investigation in the present study is that $p(t|s, C)$ is a more accurate measure than $p(t|s)$.

The state-of-the-art SMT decoder used in the majority of MT studies is Moses[7] (Koehn et al., 2007b). Moses offers no facilities to take source-side context information in account. We considered three options to achieve our goal of integrating source-side context: 1) creating a new decoder; 2) enhancing Moses; or 3) using a bypass method. Although we initially set out to create a new decoder, it proved to be too time-consuming to attain the same quality as Moses. We therefore decided, in line also with most of the literature, to follow the third option and use a bypass method; this allows us to use Moses as a black box.

[6] The normalisation denominator p(f) can be dropped as it would not alter the outcome of the *argmax* function

[7] `http://statmt.org/moses/`,
git commit `1615d56b69df5d959de9c5416cf628d76ac99169` represents a fair state of the version used in this study.

The bypass method is a *discriminative translation filtering* step (Haque et al., 2011a). It performs context-aware classification in a pre-processing step, namely through alteration of the phrase-translation table, and bypasses the need to alter the decoder. Taking each sentence individually, we ensure that the entries in the phrase-translation table are explicitly tuned to the source-side context. The output of the classifier(s) acts as the filter and constrains the translations options, as well as adjusts the score vector. Each instance of a source phrase receives a separate entry in the phrase-translation table, as opposed to one source phrase applying to all instances in the test data.

To achieve this, each source phrase in the phrase-translation table is replaced by a representation of its position in the test data, e.g. $(1, 0)$ for first sentence, first word. This is done using the software package *colibri-mt*. It creates an indexed pattern model on the test data, constrained by the phrases in the phrase-translation table. This thus constitutes a mapping of all distinct source phrases in the phrase-translation table to the indices in the test corpus that are instances of these phrases.

Subsequently we invoke the classifier(s) and construct the altered phrase-translation table as shown in Algorithm 4.

Algorithm 4 Classifier invocation on test data. Take M_{test} to be the pattern model of the test data, i.e. a map of source phrases occuring in the test data, and $[(t, p(t|s, C))]$ to be a list of translation options (t) with associated probability $p(t|s, C)$.

for $s \in M_{\text{test}}$ **do**
  for $(i, j) \in M_{\text{test}}[s]$ **do**
    $C \leftarrow \langle w_{i,j-n} \dots w_{i,j-1}, s, w_{i,j+|s|+1} \dots w_{i,j+|s|+m} \rangle$
    $[(t, p(t|s, C))] \leftarrow$ CLASSIFY(s, i, j, C)
    APPENDTOPHRASETABLE$(s, i, j, [(t, p(t|s, C)])$
  end for
end for

In Algorithm 4 we examine each source phrase in turn, find where it occurs in the test data (i, j) using the pattern model (M_{test}), and extract context information (C). The context information constitutes our feature vector, with which we invoke the appropriate classifier. This is either the monolithic classifier, or the classifier expert pertaining to the source phrase under consideration. The result of this classification is a distribution of translation options for that source phrase in the given context, along with a classifier score for each option. After normalisation, this score is $p(t|s, C)$. We now have two alternative score weighting methods for integrating this in the score vector for the phrase pair:

- **Replace** - Replace the $p(t|s)$ probability with $p(t|s, C)$;

- **Append** - Leave the existing $p(t|s)$ as it was, and append $p(t|s, C)$ as a new score to the score vector.

The score weighting of choice is applied and the data is entered into the altered phrase-translation table. The source phrase takes the form of a sequence of (i, j) indices, rather than the actual words. The test data is replaced with a series of consecutive positions as well. These two altered data sets are now the input to Moses, which can now run unmodified.

7.4 Experiments

We conduct a series of experiments to assess whether integration of context-informed classifiers in Statistical Machine Translation leads to an improvement in translation quality.

7.4.1 Data sets

For the translation model, we rely on a parallel corpus as our main source of input. Whenever the source is untokenised, we tokenise the data using *ucto*.[8] For the language model, we simply reuse the target-side portion of the parallel corpus; we do not introduce additional data for the language model in our experiments.

We use various corpora[9] and test various language pairs as we hope to come to a general conclusion. Table 7.1 lists them all, along with the amount of sentence pairs we used for training, development and testing.

All our experimental end results and all non-restricted data sets used can be freely downloaded.[12]

7.4.2 Evaluation

We assess translation quality along three widely-used automated metrics, as human evaluation is prohibitively expensive:

1. **BLEU** - BLEU (Papineni et al., 2002) is probably the most widely-used metric in Machine Translation. It computes a weighted average of n-gram overlap between the system output and reference output. The score thus increases as more n-grams in the reference translation are found in the system output. how

[8] An open-source regular-expression based tokeniser with unicode support, https://languagemachines.github.io/ucto, we used version 0.7.0

[9] EMEA and JRC-Acquis can be obtained through http://opus.lingfil.uu.se/

[10] This corpus is not publicly available unfortunately

[11] This corpus is not publicly available unfortunately

[12] Download archive of all experimental data & results: http://academictorrents.com/details/3cc11d076c62bfa0480afc187a4bb6e4759e5c16

Languages	Sentence Pairs
EMEA – *Documents of the European Medicines Agency* (Tiedemann, 2012)	
Spanish to English	1,088,333 / 5000 / 5000
English to Dutch	1,080,894 / 5000 / 5000
Fryske Akademy Parallel Corpus - *A collection of texts in Frisian and Dutch, contains numerous books and other sources* (van Gompel et al., 2014) [11]	
Dutch to Frisian	137,937 / 2000 / 2000
Frisian to Dutch	137,937 / 2000 / 2000
JRC-Acquis - *A collection of legislative documents of the European Union* (Tiedemann, 2012)	
English to Spanish	1,233,670 / 5000 / 5000
English to Spanish	250,000 / 5000 / 5000
IWLST 2012 TED Talks - *Transcripts and translations of TED talks, used for subtitling, as used in the IWSLT 2012 Evaluation Campaign* (Cettolo et al., 2012; Federico et al., 2012)	
English to Dutch	127,806 / 885 / 1569
IWLST 2006 Evaluation Campaign - *Collection of phrases from a phrase-book / traveller's guide*	
Chinese to English	40,274 / 489 / 486
Yandex 1M Web Corpus - *Phrases crawled from the web by Russian search engine Yandex*	
English to Russian	990,000 / 5000 / 5000

Table 7.1: Overview of parallel corpora used for experiments. The three values for sentence pairs correspond to the size of the training, development and test set, respectively.

informative a particular n-gram is by assigning more weight to rare n-grams and less to highly-frequent n-grams.

2. **METEOR** - METEOR (Banerjee and Lavie, 2005) also attempts to improve upon BLEU and attempts to emulate human judgement better. Unlike the prior metrics, it places emphasis on recall rather than precision.

3. **TER** - Translation Error Rate Snover et al. (2006) aims to measure the amount of editing that a human would have to perform to change a system output so it exactly matches a reference translation.

For BLEU and METEOR, higher scores are better, for TER, however, a lower score corresponds with a better translation quality.

For each experiment we construct a non-context-informed baseline. This baseline does make use of our full pipeline, i.e. the bypass method described in Section 7.3.3, but it does not invoke the classifiers. We do this to ensure the only difference is the actual integration of context information, and that differences in results can not be attributed to minor idiosyncrasies of the implementation.

7.4.3 Parameter optimisation

An SMT system relies heavily on parameter optimisation. Each component of the score vector is weighted by a separate parameter. These parameters are experimentally determined on development data using Minimum Error Rate Reduction (Och, 2003), in which we optimise on the BLEU score.

MERT is, however, known to have high variance across multiple runs, due to many local optima in the search space. We therefore can not simply run MERT once for each experiment, as that gives no basis for determining whether our results can be considered statistically significant. To remedy this issue, we have to control for optimiser instability. We do so by applying the methodology of Clark et al. (2011), which runs MERT multiple times, employs *bootstrap resampling* to compute standard deviations and *approximate randomisation* to compute p-values for the assessment of statistical significance. All this is implemented in the software *MultEval*[13] by Jonathan Clark.

For each of the evaluation metrics, BLUE, METEOR and TER. MultEval reports three numbers in parentheses, these are the following, from Clark et al. (2011):

1. $\bar{s}_{sel}$ - Variance due to *test set selection*, computed using bootstrap resampling for each optimiser run. This metric reports the average variance over all MERT runs. Higher values indicate

[13] Available from https://github.com/jhclark/multeval

that more replications, a larger test test, is desireable to draw reliable inferences.

2. s_{opt} - Variance due to *optimiser instability*. Calculated directly as the variance of aggregate metric score over all MERT runs. Higher values are indicative of optimisation on development data generalising well to previously unseen test data, i.e. it leads to overfitting.

3. p, - The p-value is calculated through approximate randomisation. It roughly expresses the probability of the difference between the system output and baseline occurring due to chance. Approximate randomisation uses random permutations to simililate chance occurrences. The quality of this measure depends on the number of separate optimisation runs, and is conditioned on the test test.

In all our experiments, we use three seperate optimisation runs, to keep complexity and experimental run time to a minimum whilst still reaping the benefits of multiple runs.

While we conduct parameter optimisation for the parameters of the SMT decoder, we do not do so for the classifiers. Yet, these too could be optimised. Optimisation of classifier parameters comes with a number of challenges. Ideally, it should not be independent of the optimisation of the decoder parameters, and the classifier parameters should be evaluated on the end result, i.e.
the full sentential translations, according to one or more MT metrics. This, however, is prohibitively expensive as it vastly increases the parameter search space and thus the complexity of the problem. As a concession for the sake of computability, classifier parameter optimisation can be considered independently and assessed through simpler cross-validation or leave-one-out methods (depending on the number of instances in the classifier expert). It still introduces a significant computational bottleneck. This has been extensively researched in Chapter 3, Section 3.4 in the context of cross-lingual Word Sense Disambiguation, rather than full Machine Translation. There classifier parameter optimisation showed an overall negative effect, most likely due to overfitting, especially when combined with variable context selection.

Optimisation can be taken a step further by also optimising the local context size, i.e. the make up of the feature vector, on a by-classifier basis, as opposed to using the same context size for all classifiers. Variable context selection by itself did make a small positive impact in a WSD context (See Section 3.4).

Given the results of this prior study, and due to the inherent computational complexity of different layers of optimisation (even under an independence assumption) we choose to limit ourselves to

just optimisation of the SMT decoder parameters in this study. These parameters have the most immediate and largest impact as they are directly optimised on an evaluation metric that is used. Note that although we forego on explicit variable context selection, an implicit form is still there for the classifier experts; the feature weights are computed on a by-classifier basis.

7.5 Results

In this section we present the results of our experiments. Our largest issue with the present MT literature on this subject is that we too often see that hypotheses are tested only on one or two corpora, with just one or two language pairs, and sometimes without proper significance testing. We feel that from such results, generic conclusions are drawn too quickly. To prevent this pitfall, our experiments are distributed amongst various corpora and language pairs, and are subject to rigid significance tests.

Section 7.5.1 focuses on our main hypothesis; does inclusion of surface-form source-side context information lead to improved translation quality? We posit a number of hypotheses to determine in which cases source-side-context might work. In Section 7.5.2 we conduct a quantitative analysis to assess whether the classifiers make an impact even if this does not reflect in the translation quality.

Our secondary hypotheses concern the ways in which classifiers can be integrated in SMT. We investigate the impact of the classifier type and weighting methods. These are assessed in Sections 7.5.3 and 7.5.4, respectively. We conclude the results section with a qualitative analysis in Section 7.5.5.

7.5.1 Source-side context vs. no context

The primary objective of our study is to assess the role of source-side context information. To this end we conducted a series of experiments on various corpora and language pairs.

In all tables in this section, the context-aware configurations are named lxry, in which x and y refer to the size (in words/tokens) of the left and right contexts respectively. Recall that the classifiers may be built in two distinct ways, a single *mono*lithic configuration (using TRIBL2), or a classifier *expert* configuration (using IB1). Secondly, score weighting can be done either through the *replace* or the *append* method, as shown in respectively the upper and lower parts of the tables.

All experiments have been optimised with MERT on BLEU as described in Section 7.4.3. The score triple in parentheses corresponds to $(\bar{s}_{sel}, s_{opt}, p)$ and represents the measures discussed in Section 7.4.3:

System	BLEU	METEOR	TER
Baseline (replace)	26.8 (1.4/0.6)	29.8 (0.7/0.2/-)	56.8 (1.5/1.2/-)
l1r1 (mono/replace)	27.7 (1.4/1.1/p=0.01)	**30.2** (0.7/0.1/p=0.01)	**55.5** (1.5/1.6/p=0.0)
l2r1 (mono/replace)	27.2 (1.4/0.3/p=0.15)	29.9 (0.7/0.2/p=0.47)	56.5 (1.5/0.6/p=0.56)
l2r2 (mono/replace)	**28.0** (1.5/0.4/p=0.0)	30.0 (0.7/0.2/p=0.26)	55.8 (1.5/0.6/p=0.02)
l3r3 (mono/replace)	27.2 (1.4/0.3/p=0.28)	29.8 (0.7/0.1/p=0.95)	57.6 (1.5/0.4/p=0.06)
l1r1 (experts/replace)	26.6 (1.4/0.2/p=0.58)	**30.2** (0.7/0.1/p=0.03)	57.0 (1.5/0.1/p=0.57)
l2r1 (experts/replace)	27.0 (1.4/0.8/p=0.5)	30.0 (0.7/0.1/p=0.39)	56.9 (1.5/0.8/p=0.71)
l2r2 (experts/replace)	27.2 (1.4/0.3/p=0.21)	30.1 (0.7/0.2/p=0.09)	56.6 (1.5/0.9/p=0.7)
l3r3 (experts/replace)	26.2 (1.4/0.5/p=0.2)	29.8 (0.7/0.2/p=0.83)	58.1 (1.5/0.8/p=0.0)
Baseline (append)	26.8 (1.4/0.5/-)	29.8 (0.7/0.1/-)	57.2 (1.5/0.6/-)
l1r1 (mono/append)	26.8 (1.4/0.3/p=0.92)	30.0 (0.7/0.1/p=0.23)	57.3 (1.5/0.5/p=0.84)
l2r1 (mono/append)	27.6 (1.4/0.2/p=0.01)	30.2 (0.7/0.0/p=0.04)	55.9 (1.5/0.2/p=0.0)
l2r2 (mono/append)	27.4 (1.4/0.8/p=0.09)	29.7 (0.7/0.2/p=0.44)	57.0 (1.5/0.9/p=0.62)
l3r3 (mono/append)	27.1 (1.4/0.8/p=0.35)	29.8 (0.7/0.2/p=0.91)	57.2 (1.5/1.3/p=0.93)
l1r1 (experts/append)	25.9 (1.4/0.5/p=0.02)	30.0 (0.7/0.0/p=0.34)	57.6 (1.5/0.7/p=0.37)
l2r1 (experts/append)	**27.9** (1.4/0.2/p=0.0)	**30.3** (0.7/0.2/p=0.01)	**55.6** (1.5/0.4/p=0.0)
l2r2 (experts/append)	27.3 (1.4/0.5/p=0.15)	30.0 (0.7/0.3/p=0.25)	56.5 (1.5/0.4/p=0.05)
l3r3 (experts/append)	26.3 (1.4/0.7/p=0.25)	29.5 (0.7/0.1/p=0.14)	58.7 (1.5/0.4/p=0.0)

Table 7.2: Results for Chinese to English on IWSLT 2006 Evaluation Campaign data. The numbers in the context-aware configurations (lxry), refer to the size of the left respectively right context.

test set selection variance, optimiser instability variance, and p-value with respect to the baseline.

In this section we focus exclusively on the question whether context-informed systems are beneficial; a comparison of the score weighting methods and classifier configurations is postponed until Section 7.5.4 and Section 7.5.3, respectively.

The first results, on the IWSLT 2006 Evaluation Campaign, Chinese to English, are listed in Table 7.2. A dataset from the same source[14] was used in Stroppa et al. (2007), who reported positive results. We observe scores outperforming the baselines for all experiments. However, not all pass the strict significance tests we impose. Only configurations l1r1/monolithic/replace, l2r1/monolithic/append and l2r2/experts/append pass the significance tests on all metrics, where we consider a $p <= 0.05$ to be the passing criterium. Other configurations either do not pass the significance tests at all, or paint a conflicting picture for certain metrics. Nevertheless, we can conclude that in this batch of experiments incorporating source-side context

[14] This dataset is not publicly available. As we do not have the exact set used by Stroppa et al. (2007), we have to assume our training and test sets, whilst drawn from the same source, are different subsets than the ones used by Stroppa et al. (2007).

System	BLEU	METEOR	TER
Baseline (replace)	**31.3** (0.5/0.2/p=-)	**32.7** (0.2/0.1/p=-)	47.9 (0.5/0.6/p=-)
l2r1 (mono/replace)	30.8 (0.5/0.1/p=0.0)	32.5 (0.2/0.0/p=0.0)	48.1 (0.5/0.2/p=0.04)
l2r2 (mono/replace)	30.5 (0.5/0.1/p=0.0)	32.4 (0.2/0.0/p=0.0)	48.5 (0.4/0.2/p=0.0)
l3r3 (mono/replace)	30.4 (0.5/0.1/p=0.0)	32.3 (0.2/0.1/p=0.0)	48.4 (0.4/0.2/p=0.0)
l1r1 (experts/replace)	**31.3** (0.5/0.0/p=0.63)	32.6 (0.2/0.0/p=0.05)	**47.1** (0.4/0.3/p=0.0)
l2r1 (experts/replace)	30.9 (0.5/0.1/p=0.0)	32.5 (0.2/0.0/p=0.0)	48.3 (0.5/0.1/p=0.0)
l2r2 (experts/replace)	31.0 (0.5/0.1/p=0.0)	32.5 (0.2/0.1/p=0.0)	47.9 (0.5/0.1/p=0.82)
l3r3 (experts/replace)	30.8 (0.5/0.2/p=0.0)	32.4 (0.2/0.1/p=0.0)	48.2 (0.5/0.4/p=0.04)
Baseline (append)	31.1 (0.5/0.3/p=-)	32.6 (0.2/0.0/p=-)	48.1 (0.5/0.7/p=-)
l2r1 (mono/append)	30.8 (0.5/0.1/p=0.0)	32.4 (0.2/0.0/p=0.0)	47.4 (0.4/0.4/p=0.0)
l2r2 (mono/append)	30.6 (0.5/0.4/p=0.0)	32.4 (0.2/0.1/p=0.0)	48.3 (0.4/1.1/p=0.12)
l3r3 (mono/append)	30.5 (0.5/0.1/p=0.0)	32.3 (0.2/0.0/p=0.0)	48.5 (0.4/0.1/p=0.0)
l1r1 (experts/append)	**31.3** (0.5/0.1/p=0.11)	**32.6** (0.2/0.0/p=0.49)	**47.1** (0.4/0.5/p=0.0)
l2r1 (experts/append)	30.9 (0.5/0.3/p=0.05)	32.5 (0.2/0.0/p=0.0)	47.9 (0.5/0.8/p=0.01)
l2r2 (experts/append)	30.6 (0.5/0.3/p=0.0)	32.4 (0.2/0.1/p=0.0)	48.4 (0.5/0.8/p=0.01)
l3r3 (experts/append)	30.7 (0.5/0.2/p=0.0)	32.4 (0.2/0.0/p=0.0)	48.3 (0.4/0.2/p=0.08)

Table 7.3: Results on IWSLT 2012, Dutch to English

information using machine learning proves beneficial for certain configurations.

In Table 7.3 we see results on the IWSLT 2012 corpus, for Dutch to English. Here we observe that we do not manage to surpass the non-context informed baseline with an acceptable degree of significance.

We move on to larger corpora, such as the EMEA corpus and the JRC corpus, and run experiments for English to Spanish and in the former corpus also for English to Dutch. These results are shown in Table 7.4 for EMEA, and Table 7.5 for JRC.

From all sets of experiments, we again observe that the results do not surprass the baseline, although for JRC some configurations such as *l2r1 (experts/append)* do score slightly above threshold, with statistical significance, on two out of three metrics. It is clear by now that surpassing the non-context informed baseline seems a very tough challenge.

We have tried additional experiments on different corpora and language pairs. A Frisian-Dutch parallel corpus was assembled on mostly literary texts, in the scope of another collaborative study between the authors and the Fryske Akademy (van Gompel et al., 2014). This language pair benefits from high BLEU scores due to the high degree of similarity between the two languages. Running our experiments on this corpus, both for Dutch to Frisian as well as Frisian to Dutch

[15] METEOR scores for this experiment were not properly computed due to an error with the evaluation script

System	BLEU	METEOR	TER
EMEA-en-es			
Baseline (replace)	71.5 (0.5/0.0/p=-)	80.9 (0.4/0.0/p=-)	25.8 (0.6/0.0/p=-)
l1r1 (mono/replace)	71.5 (0.5/0.0/p=0.89)	**81.0** (0.4/0.1/p=0.49)	**25.7** (0.6/0.1/p=0.02)
l2r1 (mono/replace)	71.5 (0.5/0.0/p=0.43)	80.9 (0.4/0.0/p=0.75)	25.8 (0.6/0.1/p=0.78)
l2r2 (mono/replace)	71.5 (0.5/0.1/p=0.65)	80.9 (0.4/0.1/p=0.62)	25.8 (0.6/0.1/p=0.44)
l3r3 (mono/replace)	71.5 (0.5/0.0/p=0.72)	80.9 (0.4/0.1/p=0.93)	25.8 (0.6/0.1/p=0.32)
Baseline (append)	71.5 (0.5/0.0/p=-)	80.9 (0.4/0.0/p=-)	25.8 (0.6/0.0/p=-)
l1r1 (mono/append)	71.5 (0.5/0.0/p=0.74)	80.9 (0.4/0.1/p=0.43)	25.8 (0.6/0.1/p=0.92)
l2r1 (mono/append)	**71.6** (0.5/0.0/p=0.03)	**81.0** (0.4/0.0/p=0.42)	25.8 (0.6/0.1/p=0.39)
l2r2 (mono/append)	71.5 (0.5/0.1/p=0.4)	80.9 (0.4/0.0/p=0.07)	25.9 (0.6/0.1/p=0.17)
l3r3 (mono/append)	71.5 (0.5/0.1/p=0.51)	80.9 (0.4/0.1/p=0.74)	25.8 (0.6/0.1/p=0.9)
EMEA-en-nl			
Baseline (replace)	**68.9** (0.6/0.1/p=-)	-[15]	28.3 (0.7/0.1/p=-)
l1r1 (mono/replace)	**68.9** (0.6/0.0/p=0.83)	-	**28.2** (0.7/0.0/p=0.04)
l2r1 (mono/replace)	68.8 (0.6/0.0/p=0.28)	-	28.3 (0.7/0.1/p=0.2)
l2r2 (mono/replace)	68.8 (0.6/0.0/p=0.32)	-	**28.2** (0.7/0.1/p=0.03)
l3r3 (mono/replace)	**68.9** (0.6/0.1/p=0.8)	-	**28.2** (0.7/0.0/p=0.01)
Baseline (append)	68.8 (0.6/0.1/p=-)	-	28.3 (0.7/0.0/p=-)
l1r1 (mono/append)	**68.9** (0.6/0.1/p=0.12)	-	28.3 (0.7/0.1/p=0.56)
l2r1 (mono/append)	68.8 (0.6/0.1/p=0.9)	-	28.3 (0.7/0.1/p=0.32)
l2r2 (mono/append)	68.8 (0.6/0.1/p=0.76)	-	**28.2** (0.7/0.1/p=0.02)
l3r3 (mono/append)	68.8 (0.6/0.0/p=0.69)	-	**28.2** (0.7/0.0/p=0.04)

Table 7.4: Results on EMEA, English to Spanish and English to Dutch.

System	BLEU	METEOR	TER
JRC-en-es			
Baseline (replace)	57.6 (0.5/0.2/p=-)	**73.1** (0.4/0.0/p=-)	32.7 (0.5/0.2/p=-)
l1r1 (mono/replace)	57.2 (0.5/0.1/p=0.0)	72.2 (0.4/0.1/p=0.0)	33.1 (0.5/0.4/p=0.0)
l2r1 (mono/replace)	57.6 (0.5/0.1/p=0.74)	72.5 (0.4/0.1/p=0.0)	32.7 (0.5/0.2/p=0.86)
l2r2 (mono/replace)	57.7 (0.5/0.1/p=0.0)	72.6 (0.4/0.1/p=0.0)	32.8 (0.5/0.2/p=0.23)
l3r3 (mono/replace)	**57.8** (0.5/0.0/p=0.01)	72.7 (0.4/0.0/p=0.0)	**32.5** (0.5/0.2/p=0.0)
l1r1 (experts/replace)	57.2 (0.5/0.1/p=0.0)	72.1 (0.4/0.0/p=0.0)	33.4 (0.5/0.2/p=0.0)
l2r1 (experts/replace)	57.0 (0.5/0.4/p=0.0)	72.2 (0.4/0.1/p=0.0)	33.5 (0.5/0.4/p=0.0)
l2r2 (experts/replace)	57.6 (0.5/0.2/p=0.26)	72.5 (0.4/0.1/p=0.0)	32.8 (0.5/0.4/p=0.03)
l3r3 (experts/replace)	**57.8** (0.5/0.0/p=0.0)	72.7 (0.4/0.1/p=0.0)	32.7 (0.5/0.1/p=0.89)
Baseline (append)	57.6 (0.5/0.1/p=-)	**73.1** (0.4/0.0/p=-)	32.8 (0.5/0.1/p=-)
l1r1 (mono/append)	57.6 (0.5/0.1/p=0.07)	72.7 (0.4/0.0/p=0.0)	32.7 (0.5/0.1/p=0.18)
l2r1 (mono/append)	57.6 (0.5/0.1/p=0.02)	72.8 (0.4/0.1/p=0.0)	32.7 (0.5/0.2/p=0.05)
l2r2 (mono/append)	57.5 (0.5/0.1/p=0.25)	72.7 (0.4/0.1/p=0.0)	32.9 (0.5/0.0/p=0.04)
l3r3 (mono/append)	57.5 (0.5/0.3/p=0.58)	72.7 (0.4/0.1/p=0.0)	32.9 (0.5/0.3/p=0.02)
l1r1 (experts/append)	**57.7** (0.5/0.1/p=0.07)	72.8 (0.4/0.1/p=0.0)	32.7 (0.5/0.1/p=0.29)
l2r1 (experts/append)	**57.7** (0.5/0.1/p=0.0)	72.8 (0.4/0.0/p=0.0)	**32.6** (0.5/0.2/p=0.0)
l2r2 (experts/append)	57.5 (0.5/0.3/p=0.44)	72.7 (0.4/0.1/p=0.0)	32.9 (0.5/0.2/p=0.01)
l3r3 (experts/append)	57.6 (0.5/0.3/p=0.0)	72.8 (0.4/0.0/p=0.0)	32.7 (0.5/0.4/p=0.05)

Table 7.5: Results on JRC, English to Spanish

scores below baseline, and the TER score that does exceed baseline does not pass the significance tests. Table 7.6 shows an excerpt of these experimental results.

A possible hypothesis we entertained is that source-side context modelling stands to gain most when translating from morphologically simpler languages to morphologically richer languages, as the source-side context information may help in determining the proper morphological form. We test this with English to Russian on the Yandex corpus, but consistently score below baseline. An excerpt of these results is shown in Table 7.7.

7.5.2 Quantitative Analysis

Are we placing our bets on the wrong horse by focussing solely on $p(t|s)$ and *replacing* it with $p(t|s, C)$? No, if we aggregate the results from all experiments that use the replace score weighting method, we find that the $p(t|s)$ or $p(t|s, C)$ feature receives the highest weight of the translation model scores in 63% of the experiments[16], the next runner-up with 25% being the forward probability $p(s|t)$.

[16] Multiple optimiser runs (3) are counted as separate experiments in this test

System	BLEU	METEOR	TER
FryskeAkademy-nl-fy			
Baseline (replace)	**60.7** (0.6/0.1/p=-)	**73.9** (0.4/0.0/p=-)	33.6 (0.6/0.1/p=-)
l1r1 (mono/replace)	60.3 (0.6/0.1/p=0.0)	73.7 (0.5/0.0/p=0.0)	**33.5** (0.6/0.1/p=0.2)
l2r1 (mono/replace)	60.3 (0.6/0.1/p=0.0)	73.7 (0.5/0.0/p=0.0)	**33.5** (0.6/0.1/p=0.2)
FryskeAkademy-fy-nl			
Baseline (replace)	**55.3** (0.7/0.0/p=-)	-	**28.6** (0.6/0.0/p=-)
l1r1 (mono/replace)	54.9 (0.6/0.0/p=0.0)	-	28.8 (0.5/0.0/p=0.0)
l2r1 (mono/replace)	54.9 (0.6/0.1/p=0.0)	-	28.9 (0.6/0.0/p=0.0)

Table 7.6: An excerpt of the results on the Frisian parallel corpus, Dutch to Frisian and Frisian to Dutch.

System	BLEU	TER
Yandex English-Russian		
Baseline (replace)	**15.7** (0.3/0.0/p=-)	**72.0** (0.3/0.0/p=-)
l1r1 (mono/replace)	14.7 (0.3/0.0/p=0.0)	73.3 (0.3/0.0/p=0.0)
l2r1 (mono/replace)	15.0 (0.3/0.0/p=0.0)	73.0 (0.3/0.1/p=0.0)

Table 7.7: An excerpt of the results on the Yandex corpus, English to Russian.

We run the same test for the append score weighting method, which leaves the $p(t|s)$ feature as is, and *appends* the $p(t|s, C)$ feature to the translation model. We find that $p(t|s, C)$ still receives most weight in 33% percent of the experiments. The unaltered $p(t|s)$ score wins in 29% of the cases and $p(s|t)$ in 23%[17].

These tests show that our focus on this particular feature in the translation model is justified.

We have measured the impact of the classifier on translation quality. To get a deeper understanding of the impact of the classifier(s), we measure to what extent the phrases used in the test set output of a context-informed system differ from the phrases used in the non-context informed output, irrespective of whether that difference contributes positively or negatively to the translation quality. To this end, we ask Moses to verbosely output, for each output sentence, which phrases from the intermediate phrase-translation table are used. We then analyse how many phrase pairs are unique to the context-informed system.

These phrases are indicative of the SMT decoder choosing a different translation option than it did for the baseline, due to either the classifier information or optimiser differences. Table 7.8 shows the results for a few of the data sets we used. We also show, in the last column, how many of the source-side fragments are not in the base-

[17] The remaining probability mass is distributed over the lexical weighting features

Corpus	Language Pair	Configuration	Diff. pairs	Diff. source
Yandex (1M)	English→Russian	l1r1 (mono/replace)	37.72%	26.58%
Fryske Akademy	Frisian→Dutch	l1r1 (mono/replace)	33.57%	32.50%
IWSLT 2012 TED	Dutch→English	l1r1 (mono/replace)	26.86%	20.07%
IWSLT 2006	Chinese→English	l1r1 (mono/replace)	24.55%	12.49%

Table 7.8: The number of phrases in the context-informed test output that are not in the non-context-informed baseline. We measure both the number of differing phrase pairs (as a fraction of all phrase pairs), as well as the number of differing source phrases without regard for translation (as a fraction of all source phrases).

line, implying that the SMT decoder chose a different source-phrase to translate than the baseline did.

For this experiment, however, we had to take the optimiser out of the equation by optimising only once on the non-context-informed baseline, and use those parameters on the context-informed experiments as well. Although this leads to sub-optimal translation quality, it guarantees that the difference we measure can be attributed solely to the inclusion of context information:

We observe from this data that our system does have a decent impact on phrase selection. A correlation between translation quality and the difference ratio of phrase pairs or source phrases, can not be discerned, however.

7.5.3 Classifier type

We have introduced two types of classifiers, integrated in a Statistical Machine Translation framework by means of the "bypass" method described in Section 7.3.3. The monolithic classifier, modelling all phrase-pairs in a single classifier, is used also by Stroppa et al. (2007) and Haque et al. (2011a), whereas the classifiers experts, one classifier per source-phrase, is a new addition that more closely resembles how such classifiers are employed in Word Sense Disambiguation.

Classifier experts were built using the IB1 algorithm, whereas we choose for TRIBL2 for the monolithic classifier. This was done because the monolithic classifier is a lot bigger by definition; a pure IB1 approach would be too slow and TRIBL2 seeks to combine the accuracy of IB1 with the speed of a decision-tree classifier, IGTree.

Table 7.9 lists the number of classifier experts that were built for some of the data sets. The actual number of source phrases is always much higher, as classifiers are only built when there is ambiguity regarding the translation. To see how these classifiers are typically distributed, based on the number of instances and classes, we zoom

Corpus	Sentence pairs	Languages	Classifier Experts
JRC-Acquis	1,233,670	English-Spanish	12,652,153
IWSLT 2012 TED	127,806	Dutch-English	193,811
IWSLT 2006	40,274	Chinese-English	22,212
Fryske Akademy	137,937	Dutch-Frisian	276,819
Fryske Akademy	137,937	Frisian-Dutch	213,573

Table 7.9: Number of classifier experts generated per data set.

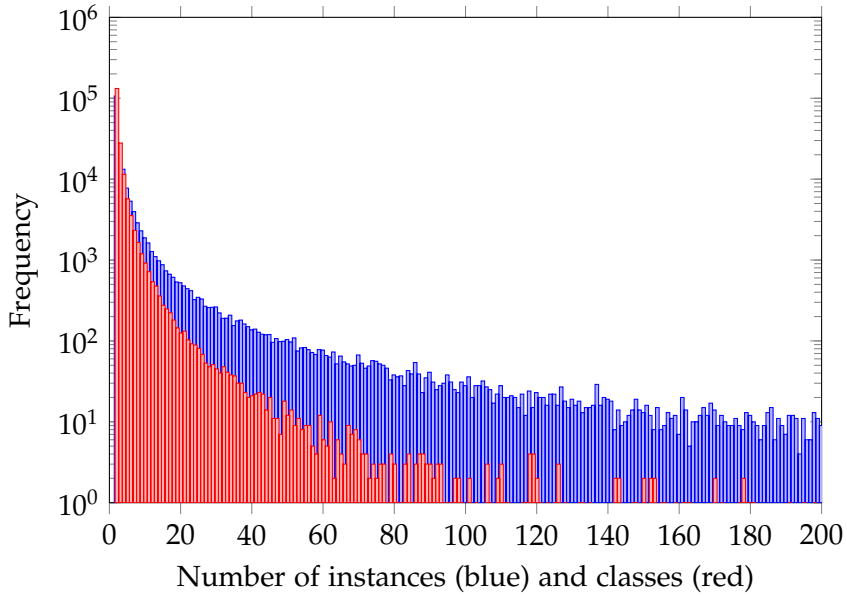

Figure 7.1: Histogram showing the distribution of the number of instances (blue) and number of classes (red) in the classifier experts for IWSLT 2012 TED, Dutch to English (logarithmic).

in on the IWSLT 2012 data for Dutch to English, and plot a histogram in Figure 7.1. A typical Zipfian curve can be discerned from this figure, with the vast majority of classifier experts being based on a small amount of instances (left) and a long tail of experts, only partially shown, with fewer than 100 instances. Only a small subset of these experts has more than one translation option, seen by the small amount of red in the right half of Figure 7.1. In other words, most ambiguity is present in classifier experts trained on less than 100 examples of a source phrase in different contexts.

Various of the tables in Section 7.5.1 list results for both classifier types. As both methods generally fail to surpass the baseline, and the differences in translation score between classifier experts and monolithic classifier appear minimal, we can not reach a strong conclusion regarding their respective merit.

7.5.4 Weighting methods and score analysis

Two weighting methods have been implemented: the *append* method, adding $p(t|s,C)$ as an extra score to the score vector, and the *replace* method, which replaces the existing $p(t|s)$ score with $p(t|s,C)$.

Comparing the two weighting methods, however, is not trivial. The very act of adding a score, for the append method, shifts the weights for the score vector. Any shift may likely affect the outcome. This is also the reason that the replace and append methods have their own respective baselines in the result tables seen so far.

For the append method, the $p(t|s,C)$ score is often simply equal to the $p(t|s)$ score. This is by definition so for any source phrases for which no classifier was needed, due to not being ambiguous in translation or context.

As seen before, given the over-all lack of impact above baseline observed across most experiments, we can not draw any satisfactory conclusion on the merit of the *append* method versus the *replace* method, given the experiments we conducted. We can, however, use the append scoring method to provide further insight into the classifier results. How often does the classifier assign a $p(t|s,C)$ that differs from $p(t|s)$, and how big is this difference?

For each occurrence of each source phrase in a given test set, we gather all possible translations from the phrase-translation table and compute the difference Δ between the context-informed translation probability and the non-context informed one as per Equation 7.3.

$$\Delta = \begin{cases} \frac{p(t|s,C)}{p(t|s)} - 1, & \text{if } p(t|s,C) \geq p(t|s) \\ -(\frac{p(t|s)}{p(t|s,C)} - 1), & \text{otherwise} \end{cases} \tag{7.3}$$

We compute this on the test data of the JRC-Acquis corpus, from English to Spanish, as shown in Figure 7.2. For training we used a reduced subset of 250,000 sentences. The measurements are taken from the phrase-translation table directly so the decoder and optimizer plays no role here. Our measurement merely assesses to what extent context-informed classifiers produce an outcome that differs from when no context is taken into consideration.

For Figure 7.2, we set up bins of size 0.1, so each point on the x axis aggregates results of the specified $\Delta \pm 0.05$. A value of zero indicates there is no difference whatsoever. This includes all source phrases where either translation or context are unambiguous. A spike and outlier can be observed at this point, covering 9% of all phrase-pairs. We observe that the positive side of the graph drops much less sharply than the negative side. The positive numbers cover the phrase pairs to which the classifier assigns a higher probability; the value expresses the factor by which the probability is increased. No less than 75% of the phrase pairs are in this region. This is a direct result

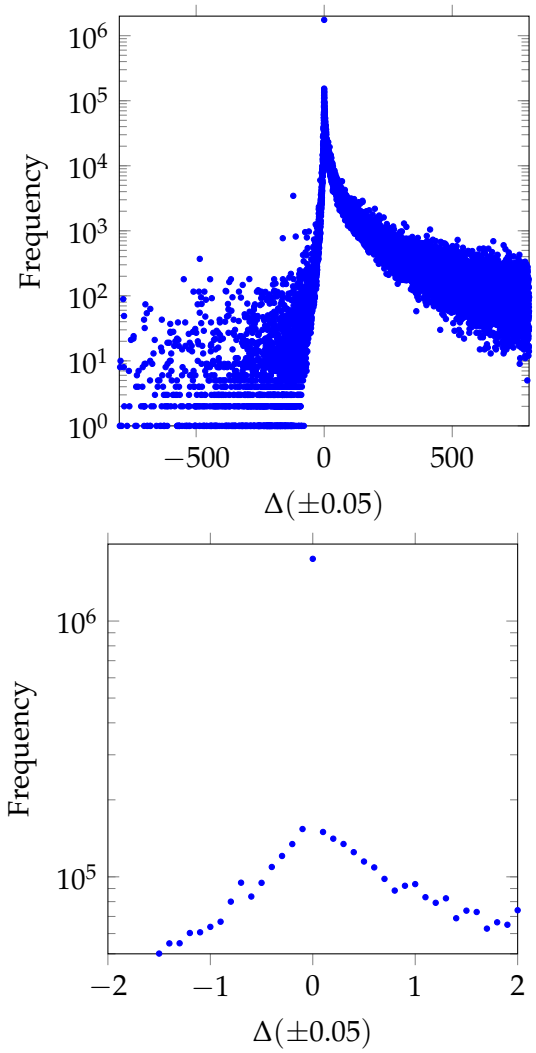

Figure 7.2: Scatter plot (logarithmic scale) illustrating the difference Δ between the classifier score and the original SMT score on the JRC-Acquis test set, English to Spanish, trained on 250,000 sentences. The second figure provides a zoomed view and illustrates that small changes are most prevalent.

of the discriminative filtering done by the classifiers, which prunes the weakest translation options, and therefore on the whole raises the probability of the remaining options.

7.5.5 Qualitative analysis

In this section we take a closer look at the translations themselves and see if we can discern patterns that can tell us what the effect of source-side context information is and when it pays off.

We zoom in on the EMEA corpus – English to Spanish, one million sentence pairs. We compare a special run of the context-informed `l1r1` system with the non-context informed baseline in order to assess the impact of context information. In this special run, the decoder parameters have not been optimised independently, but are set equal to those of the baseline. Although this leads to sub-optimal translation quality, it ensures we compare only the effect of the classifier rather than of the optimiser. Out of 5,000 sentences in the test set, 3,863 (77%) are translated identically. This high number again shows that it is difficult to make a difference. Nevertheless, 23% of the test sentences are translated differently.

The next few examples will show some comparisons in which the context-informed system improves over the baseline.

Example 3 shows three examples of the selection of a better translation given the context, matching with the reference translation, even though the baseline translation could be considered correct as well. This is the most common type of difference. Example 4 shows the dropping of a word. Example 5 illustrates the selection of a grammatically better phrase in the context.

(3) **Baseline:** El cambio continuo del lugar de inyección dentro de un área determinada puede ayudar a
l1r1: El cambio continuo del lugar de inyección dentro de una región determinada puede ayudar a
Reference: La contínua rotación de los puntos de inyección dentro de una región determinada puede ayudar a reducir o prevenir estas reacciones

---

Baseline: Baraclude reduce la cantidad de virus en su cuerpo y mejora el estado del hígado .
l1r1: Baraclude reduce la cantidad de virus en su organismo y mejora el estado del hígado
Reference: Baraclude reduce la cantidad de virus en su organismo y mejora el estado del hígado .

Baseline: Como consecuencia , el fenilbutirato de sodio , reduce los niveles plasmáticos de amonio y glutamina en pacientes con trastornos del ciclo de la urea .
l1r1: Como consecuencia , el fenilbutirato de sodio reduce las concentraciones plasmáticas elevadas de amonio y glutamina en pacientes con trastornos del ciclo de la urea .
Reference: Como consecuencia , el fenilbutirato de sodio reduce las concentraciones plasmáticas elevadas de amoníaco y glutamina en pacientes con trastornos del ciclo de la urea .

(4) **Baseline:** Los estudios en animales , no pueden excluir el desarrollo potencial de toxicidad (ver sección 5.3) .
l1r1: Estudios en animales , no pueden excluir el desarrollo potencial de toxicidad (ver sección 5.3) .
Reference: Estudios en animales , no pueden excluir el desarrollo potencial de toxicidad (ver sección 5.3) .

(5) **Baseline:** Las reacciones adversas consideradas relacionadas con el uso de Agenerase son síntomas gastrointestinales , erupción y parestesia oral / peri-oral .

l1r1: Las reacciones adversas que se consideran relacionadas con el uso de Agenerase son síntomas gastrointestinales , erupción y parestesia oral /
Reference: Las reacciones adversas que se consideran relacionadas con el uso de Agenerase son síntomas gastrointestinales , erupción y parestesia oral / perioral .

In addition to the positive examples, there are also neutral and negative examples. In example 6, all translations can technically be considered correct and convey the same message with different nuances; a different construction is chosen. Example 7 shows the inverse situation of example 3; a different synonym was chosen but it does not match with the reference translation. This too is a common pattern in the data. This raises the question whether the impact of context-information would not be lower if the test set would have contained multiple reference translations. Such data unfortunately is hard to come by and was not available in this study.

(6) **Baseline:** Asegúrese de que el polvo esté completamente disuelto .
l1r1: Asegúrese de que el polvo se disuelva completamente .
Reference: Asegúrese de que el polvo se ha disuelto completamente .

(7) **Baseline:** Los pacientes se asignaron aleatoriamente a recibir 500 μ g de Aranesp una vez cada tres semanas o 2,25 μ g / kg una vez a la semana .
l1r1: Los pacientes fueron aleatorizados a recibir 500 μ g de Aranesp una vez cada tres semanas o 2,25 μ g / kg una vez a la semana .
Reference: Los pacientes se asignaron aleatoriamente a recibir 500 μ g de Aranesp una vez cada tres semanas o 2,25 μ g / kg una vez a la semana .

We also zoom on the IWSLT 2006 Chinese to English experiment, which is the only one that managed to make a positive impact. We wonder whether we can find what causes these positive results, whereas most other experiments fail to surpass baseline. As we used three optimisation runs, we have three outputs available for the baseline and for three for the best performing system (l2r1 (experts/append)), we select one of each at random and manually inspect the output. Out of 485 test sentences, 44 are translated perfectly with respect to the reference, for both the system and the baseline. Often, however, the baseline and system choose different words that still make sense, as in Example 8. In some cases, the baseline and the system are hindered by out-of-vocabulary words, and retain some chinese characters in the output.

(8) **Baseline:** where is the currency exchange office ?
l2r1: where is the currency exchange office ?
Reference: where is the money exchange ?

The baseline and the system are in complete agreement for 241 sentences (50%). Our focus goes to the remainder where they differ. We first look at instances where the system output is a better match with the reference translation than the baseline. Example 9 illustrates this.

(9) **Baseline:** my watch in ten minutes slow a day .
l2r1: my watch loses about ten minutes a day .
Reference: my watch loses ten minutes a day .

(10) **Baseline:** how much is the his room number ?
l2r1: what 's his room number ?
Reference: what is his room number ?

We can not directly ascribe the differences to the presence of context information, as we selected output from just one out of three optimiser runs. In this example however, all optimiser runs of the l2r1 system, as well as the baseline, produce the same output for Example 9 and therefore we can conclude that the inclusion of context information has a positive result in this case. Almost the same goes for Example 10, where only the baseline output slightly differs for one optimiser run.

In our selected system outputs, when manually inspecting the data and comparing it the reference, we count that the l2r1 system performs better than baseline in 36 cases, the baseline performs better in 25 cases.[18]. For the remaining instances, no easy judgement of one above the other could be made.

Again, the absence of multiple reference translations has to be reiterated. Example 11 shows two translations that both seem fine, but both do not match the reference well. Different lexical choices that can be considered good translations in their own right prevalent throughout this dataset.

(11) **Baseline:** my stomach hurts
l1r1: i have a pain in my stomach
Reference: i have a stomach ache

The IWSLT 2006 dataset is characterised by short and relatively simple sentences, whereas the other data sets tend to contain longer phrases. We can hypothesize that whilst source-side context information may have a slight local positive effect, this is often lost in translation quality as sentences grow longer and other local mismatches are made.

7.6 Conclusions and Discussion

We have conducted a large number of experiments to assess whether surface-form source-side context information, i.e. without the use of any explicit linguistic features that require supervised parsers or taggers, can improve translation quality. Memory-based classifiers were used, and integrated in an SMT framework, following techniques commonly employed in WSD. Different classifier types, context sizes,

[18] This analysis does not compensate for different optimiser, and was moreover conducted in non-blinded fashion by a single person. It is only meant to give an impression of the data.

and score weighting methods have been researched. Multiple parameter optimisation runs were conducted to obtain the decoder parameter weights, compensating for decoder instability. Optimisation of classifier parameters and feature selection was omitted in order to contain the computational complexity of the problem, and considering the lack of positive results for optimisation of classifier parameters in Chapter 3.

Various distinct corpora and language pairs have been employed for the experiments in order to ensure that conclusions are of a generic enough nature rather than incidental artifacts of the type that have arguably steered conclusions in previous work. Despite all efforts, clear positive results were only attained for IWSLT 2006 (Chinese to English). Other scores did not manage to unequivocally surpass the baseline. The IWSLT 2006 Chinese-English data set was used by both Carpuat and Wu (2007b), as well as Stroppa et al. (2007), albeit a different subset. The study of Carpuat and Wu (2007b) reports positive results on this data set using various linguistic features from WSD. But most interesting is an experiment without linguistic features conducted by Stroppa et al. (2007) where they find positive results above baseline, which we in turn have indeed reproduced with the same positive result. On their other language pair (Italian-English) from the same dataset, however, they too fail to make a statistically significant impact above baseline. What sets the IWSLT 2006 Chinese-English set apart? Can the slight positive results be attributed to the language pair or the fact that the data set is characterised by short and simple sentences? Possibly a combination of these two factors is most probable, as Stroppa et al. (2007) did not attain positive results with the Italian-English variant of this same dataset either, although it did consist of the same short sentences.

In this study, we effectively replicated and expanded upon the part of the work of Stroppa et al. (2007) and Haque et al. (2011a) that does not depend on further linguistic information. Despite a small number of positive results reported in these studies, we have to conclude that inclusion of surface-form source-side context information is not a good route for the improvement of MT quality. It is likely that improvement can be more easily achieved through for example optimisation of the target-side language model.

Upon inspection of the output, we do at times find that inclusion of context information produces a better lexical choice than the baseline. These changes are fairly localised, however, and fail to show in the evaluation metrics or pass significance tests. An important factor in this is the lack of multiple reference translations, as such datasets are unfortunately rare. More balances judgement could be made if we could compare against multiple reference translations. Moreover, in longer sentences it often occurs that a modest gain at one place may be lost against mistranslations elsewhere.

Finally, we would like to focus deeper on why the results of our study overall do not exceed the baseline. There may be an explanation for this if we look at the interplay between the SMT decoder and the classifiers. The classifier instances are generated with the phrase-translation table as input, looking up the context in the corpus. There will therefore never be phrases in the classifier training data that do not occur in the classifier. If we have a classifier training instance $(s, C) \mapsto t$, i.e. a source phrase s with context C and mapping to translation t, and C is a context that occurs multiple times in the corpus, then there is often a source fragment s' in the phrase-translation table that is the conjunction of s and C, and which maps to a t' of which t is at least a substring. In other words, if a context for a source phrase is sufficiently attested in the data, then this context may be an integral part of a larger source phrase. Only if the context is not prevalent at all, then no such s' exists, in which case one may also posit that the context is less likely to occur in test data and the training instance is therefore considered fairly weak. It is thus likely that the context information we try to *explicitly* model, is often already *implicitly* available to the decoder. Rather than considering a source phrase in context, the decoder may thus opt to choose to include the context as integral part of the phrase. The target-language model already plays an important role here, as it encourages the decoder to choose translation options that fit well together. Source-side context modelling, without additional linguistic features, seems to have little to nothing to offer.

8

SUMMARY & CONCLUSION

In this dissertation, we posited the main hypothesis that inclusion of source-side context information, without linguistically informed features, benefits translation quality. We have assessed this question from various different angles. We set out to answer the following inter-related questions and have indeed largely already done so in Section 7.6 of Chapter 7:

1. To what extent can we improve translation by considering source-side context information?
2. What context features prove most effective?
3. To what extent are linguistically uninformed features effective?
4. How can source-side classifiers be used in translation tasks?

Aside from the answers to the research questions provided by this dissertation, other notable contributions this research project delivers are 1) the Colibri Core software that was developed and described in Chapter 2, application of which goes well beyond what it was used for in this research; and 2) the cross-lingual translation assistance task we proposed in Chapter 5, along with interesting system submissions by the various participants.

Chapter 3 demonstrated an application in cross-lingual word sense disambiguation, where word expert classifiers succesfully tackled the problem of translating a word in context. Our system emerged as the winning system in the SemEval 2010 Cross Lingual Word Sense Disambiguation task, and achieved either winning or high scores in the same task in 2012. This second time around, we placed focus on hyperparameter optimisation of the classifier parameters and automatic feature selection per classifier expert, which supersedes the voting approach we used the first time. Automatic feature selection indeed leads to a modest improvement, whereas classifier parameter optimisation does not.

At this stage, we experimented with some basic linguistically informed features as well, i.e. part-of-speech tags and lemmas. Lemmas proved to be beneficial indeed, whereas part-of-speech tags failed to make a positive impact. Nevertheless, our study's explicit focus is to assess the efficacy of the surface features in their pure form, i.e. not enriched with any linguistic information. We employed multiple classifiers or word experts in which the feature vector consists of a simple local context window. In the cross-lingual WSD task our

linguistically-uninformed classifiers easily surpass the non-context-informed baselines. In other words, for this task, we find evidence that corroborates our main hypothesis and gives positive support to our first and third question.

As for our second question, what context features prove most effective? We extensively experimented with global context features in addition to the local context features found in the immediate neighbourhood of the phrase to be translated. We found a discrepancy in results between the SemEval 2010 and 2012 runs. Subsequent attempts in later chapters support the SemEval 2012 findings, in which global context features do not lead to an improvement in translation quality. In fact, we often find most gains with a small local context window.

The initial successes with translation aided by source-side context information led us to propose this technique for a novel application in computer-aided translation assistance. In this task, we translate L1 fragments in an L2 context. In Chapter 4 we tested the feasibility of this idea and found our context-sensitive classifiers improve both upon a non-context informed baseline, as well a baseline enriched with a language model. We then proposed our task for SemEval 2014 in Chapter 5, and attracted participation by six participants, some of whom achieve high results. It becomes apparent, specifically, that phrase-based statistical machine translation is able to achieve better results than pure classifier-based approaches, despite the localised and cross-lingual nature of the task.

We confirm this in Chapter 6, where we conducted an extensive comparison of the pure classifier-based and SMT-based solutions to our L2 Translation Assistance task. We then proceed to integrate classifiers in the SMT-based solution, in an attempt to gain the best of both worlds. However, we find that this does not lead to improved translation quality. Nevertheless, our hypothesis that translation can be improved by considering source-side context information is still upheld here.

In Chapter 7 we moved away from the translation of mere localised fragments in a context, and experimented with full Statistical Machine Translation, i.e. translation of whole sentences. We incorporated source-side context information using classifiers in the SMT decoder, using a bypass method. We tested various forms of integration, but found that we could only really improve upon the baseline for one corpus (IWSLT 2006, Chinese to English), which was also a positive result in a previous study (Stroppa et al., 2007). All other results turned out on or below baseline, or failed to pass rigourous significance tests. Our main hypothesis therefore does not hold for full statistical machine translation. We ask ourselves why this is the case and suggest that that the information we attempt to explicitly model in the translation model, is already implicitly and satisfactorily covered by both the target-side language model as well as the trans-

lation model. Integration of linguistically-uninformed classifiers in statistical machine translation therefore has no added value.

Our efforts only prove fruitful when used standalone or for localised problems, whereas statistical machine translation proves to be a superior technique, leading to higher translation quality, for those translation tasks that it can tackle.

When it comes to classifiers and their integration in SMT, future research could continue in the direction that past studies such as Haque et al. (2011a) have already proceeded; that of incorporating linguistic features. The technical foundation for such studies is well prepared in this one: The integration and score weighting methods from Chapters 6 and especially 7 lay a foundation upon which expansions with specific linguistic features would make this a feasible option. We do not think, however, that this is a promising line of continued investigation as the novelty of such further studies would be limited, depending on the type of linguistic features investigated and the way they are incorporated. Moreover, results from participants of our SemEval task, show that intuitively promising linguistically-informed features, such as dependency features, do not always lead to the expected improvement (Silva-Schlenker et al., 2014; Rudnick et al., 2014).

We especially have to concede that the state-of-the-art in Machine Translation has changed considerably during and after the years this research has taken place; techniques such as Neural Machine Translation have appeared on stage and often prove superior to the SMT paradigm we employed. What the possibilities of source-side context in Neural Machine Translation are, is already an active subject of research (Jean et al., 2017; Wang et al., 2017; Bawden et al., 2017; Maruf and Haffari, 2017). These studies do mostly focus on wider discourse context than we did.

The academic merit of our study is primarily in deepening and closing a line of investigation; that of modelling source-side context information, mostly through memory-based classifiers, and integration thereof in Statistical Machine Translation.

BIBLIOGRAPHY

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

Abel, A., Nicolas, L., Hana, J., Štindlová, B., Wisniewski, K., Woldt, C., Meurers, D., and Bykh, S. (2013). A trilingual learner corpus illustrating european reference levels. In *Proceedings of the Learner Corpus Research Conference*, Bergen, Norway.

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

Atserias, J., Casas, B., Comelles, E., González, M., Padró, L., and Padró., M. (2006). FreeLing 1.3: Syntactic and semantic services in an open-source NLP library . In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC 2006)*, Genoa, Italy. ELRA.

Banerjee, S. and Lavie, A. (2005). 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*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Barrachina, S., Bender, O., Casacuberta, F., Civera, J., Cubel, E., Khadivi, S., Lagarda, A. L., Ney, H., Tomás, J., Vidal, E., and Vilar, J. (2009). Statistical approaches to computer-assisted translation. *Computational Linguistics*, 35(1):3–28.

Bawden, R., Sennrich, R., Birch, A., and Haddow, B. (2017). Evaluating Discourse Phenomena in Neural Machine Translation. *arXiv e-prints*, page arXiv:1711.00513.

Brown, P., Cocke, J., Della Pietra, S., Della Pietra, V., Jelinek, F., Mercer, R., and Roossin, P. (1990). A statistical approach to language translation. 16(2):79–85.

Cabezas, C. and Resnik, P. (2005). Using WSD Techniques for Lexical Selection in Statistical Machine Translation. Technical Report LAMP-TR-124,CS-TR-4736,UMIACS-TR-2005-42, University of Maryland, College Park.

Carpuat, M. (2013). NRC: A Machine Translation Approach to Cross-Lingual Word Sense Disambiguation (semeval-2013 task 10). In *Proceedings of the 7th International Workshop on Semantic Evaluation*

(SemEval 2013), in conjunction with the Second Joint Conference on Lexical and Computational Semantics.

Carpuat, M. and Wu, D. (2005). Word sense disambiguation vs. statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 387–394, Stroudsburg, PA, USA. Association for Computational Linguistics.

Carpuat, M. and Wu, D. (2007a). Improving statistical machine translation using word sense disambiguation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 61–72.

Carpuat, M. and Wu, D. (2007b). Improving statistical machine translation using word sense disambiguation. In *In The 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)*, pages 61–72.

Cettolo, M., Girardi, C., and Federico, M. (2012). Wit^3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*, pages 261–268, Trento, Italy.

Clark, J. H., Dyer, C., Lavie, A., and Smith, N. A. (2011). Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *In Proceedings of ACL*.

Daelemans, W., van den Bosch, A., and Weijters, A. (1997). IGTree: using trees for compression and classification in lazy learning algorithms. *Artificial Intelligence Review*, 11:407–423.

Daelemans, W., Zavrel, J., van der Sloot, K., and van den Bosch, A. (2009). TiMBL: Tilburg memory based learner, version 6.2, reference guide. Technical Report ILK 09-01, ILK Research Group, Tilburg University.

Decadt, B., Hoste, V., Daelemans, W., and Van den Bosch, A. (2004). GAMBL, genetic algorithm optimization of memory-based WSD. In Mihalcea, R. and Edmonds, P., editors, *Proceedings of the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (Senseval-3)*, pages 108–112, New Brunswick, NJ. ACL.

D'hondt, E., Verberne, S., Weber, N., Koster, K., and Boves, L. (2012). Using skipgrams and PoS-based feature selection for patent classification. *Computational Linguistics in the Netherlands Journal*, 2:52–70.

Federico, M., Bertoldi, N., and Cettolo, M. (2008). IRSTLM: an open source toolkit for handling large scale language models. In *INTERSPEECH*, pages 1618–1621. ISCA.

Federico, M., Cettolo, M., Bentivogli, L., Paul, M., and Stüker, S. (2012). Overview of the IWSLT 2012 evaluation campaign. In *Proceedings of the seventh International Workshop on Spoken Language Translation (IWSLT)*, pages 12–33.

Forsyth, R. S. and Sharoff, S. (2014). Document dissimilarity within and across languages: A benchmarking study. *LLC*, 29(1):6–22.

Giménez, J. and Màrquez, L. (2007). Context-aware discriminative phrase selection for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 159–166, Prague, Czech Republic. Association for Computational Linguistics.

Goutte, C., Simard, M., and Carpuat, M. (2014). CNRC-TMT: Second language writing assistant system description. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval-2014)*, Dublin, Ireland.

Gupta, A. (2014). Team Z: Wiktionary as L2 writing assistant. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval-2014)*, Dublin, Ireland.

Guthrie, D., Allison, B., Liu, W., Guthrie, L., and Wilks, Y. (2006). A closer look at skip-gram modelling. In *Proceedings of the Fifth international Conference on Language Resources and Evaluation (LREC-2006)*, Genoa, Italy.

Guthrie, D. and Hepple, M. (2010). Storing the web in memory: Space efficient language models with constant time retrieval. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 262–272, Stroudsburg, PA, USA. Association for Computational Linguistics.

Haque, R., Naskar, S., van den Bosch, A., and Way, A. (2011a). Integrating source-language context into phrase-based statistical machine translation. *Machine Translation*, 25(3):239–285.

Haque, R., Naskar, S. K., van den Bosch, A., and Way, A. (2011b). Integrating source-language context into phrase-based statistical machine translation. *Machine Translation*, 25(3):239–285.

Hasler, E. (2014). UEdin: Translating L1 phrases in L2 context using context-sensitive smt. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval-2014)*, Dublin, Ireland.

Heafield, K. (2011). KenLM: faster and smaller language model queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, United Kingdom.

Hendrickx, I. and van den Bosch, A. (2001). Dutch word sense disambiguation: Data and preliminary results. In *Proceedings of*

SENSEVAL-2 Second International Workshop on Evaluating Word Sense Disambiguation Systems, pages 13–16, Toulouse, France. Association for Computational Linguistics.

Hendrickx, I., Van den Bosch, A., Hoste, V., and Daelemans, W. (2002). Dutch word sense disambiguation: Optimizing the localness of context. In *Proceedings of the Workshop on word sense disambiguation: Recent successes and future directions*, pages 61–65, Philadelphia, PA.

Hoste, V., Hendrickx, I., Daelemans, W., and Van den Bosch, A. (2002). Parameter optimization for machine learning of word sense disambiguation. *Natural Language Engineering*, 8(4):311–325.

Huffman, D. A. (1952). A method for the construction of minimum-redundancy codes. *Proceedings of the Institute of Radio Engineers*, 40(9):1098–1101.

Jean, S., Lauly, S., Firat, O., and Cho, K. (2017). Does Neural Machine Translation Benefit from Larger Context? *arXiv e-prints*, page arXiv:1704.05135.

Koehn, P. (2004a). Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In Frederking, R. E. and Taylor, K., editors, *In the proceedings of the American Machine Translation Association*, volume 3265 of *Lecture Notes in Computer Science*, pages 115–124. Springer.

Koehn, P. (2004b). Statistical significance tests for machine translation evaluation. In Lin, D. and Wu, D., editors, *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.

Koehn, P. (2005). Europarl: A parallel corpus for statistical machine translation. In *In Proceedings of the Machine Translation Summit X ([MT]'05).*, pages 79–86.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007a). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007b). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*

Companion Volume Proceedings of the Demo and Poster Sessions, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Kunneman, F. and van den Bosch, A. (2016). Open-domain extraction of future events from twitter. *Natural Language Engineering*.

Laghos, A. and Panayiotis, Z. (2005). Computer assisted/aided language learning. pages 331–336.

Lefever, E. and Hoste, V. (2010). Semeval-2010 task 3: Cross-lingual word sense disambiguation. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 15–20, Stroudsburg, PA, USA. Association for Computational Linguistics.

Lefever, E. and Hoste, V. (2013a). SemEval-2013 Task 10: Cross-Lingual Word Sense Disambiguation. In *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval 2013), in conjunction with the Second Joint Conference on Lexical and Computational Semantics*.

Lefever, E. and Hoste, V. (2013b). SemEval-2013 Task 10: Cross-Lingual Word Sense Disambiguation. In *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval 2013), in conjunction with the Second Joint Conference on Lexical and Computational Semantics*.

Levy, M. (1997). *Computer-assisted language learning: Context and conceptualization.* Oxford: Clarendon Press.

Manber, U. and Myers, G. (1990). Suffix arrays: A new method for on-line string searches. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '90, pages 319–327, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

Maruf, S. and Haffari, G. (2017). Document Context Neural Machine Translation with Memory Networks. *arXiv e-prints*, page arXiv:1711.03688.

Mihalcea, R., Sinha, R., and McCarthy, D. (2010). Semeval 2010 task 2: Cross-lingual lexical substitution. In *Proceedings of the 5th International Workshop on Semantic Evaluations (SemEval-2010)*, Uppsala, Sweden.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Ng, H. T. and Lee, H. B. (1996). Integrating multiple knowledge sources to disambiguate word sense: An exemplar-based approach. In *ACL*, pages 40–47.

Och, F. and Ney, H. (2000). Giza++: Training of statistical translation models. Technical report, RWTH Aachen, University of Technology.

Och, F. J. (2003). Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 160–167, Stroudsburg, PA, USA. Association for Computational Linguistics.

Och, F. J. and Ney, H. (2003). A systematic comparison of various statistical alignment models. *Comput. Linguist.*, 29(1):19–51.

Onrust, L., van den Bosch, A., and van Hamme, H. (2016). To appear in: Improving cross-domain n-gram language modelling with skipgrams. In *In Proceedings of ACL*.

Papineni, K., Roukos, S., Ward, T., and Zhu, W. (2002). Bleu: a method for automatic evaluation of machine translation. In *Computational Linguistics (ACL), Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia*, pages 311–318.

Rayson, P. and Garside, R. (2000). Comparing corpora using frequency profiling. In *In proceedings of the workshop on Comparing Corpora, held in conjunction ACL 2000. October 2000, Hong Kong*, pages 1–6.

Rudnick, A., King, L., Liu, C., Dickinson, M., and Kübler, S. (2014). IUCL: Combining information sources for semeval task 5. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval-2014)*, Dublin, Ireland.

Rudnick, A., Liu, C., and Gasser, M. (2013). HLTDI: CL-WSD using Markov Random Fields for SemEval-2013 Task 10. In *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval 2013), in conjunction with the Second Joint Conference on Lexical and Computational Semantics*.

Sadat, F., Johnson, H., Agbago, A., Foster, G., Kuhn, R., Martin, J., and Tikuisis, A. (2005). Portage: A phrase-based machine translation system. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pages 129–132, Ann Arbor, Michigan. Association for Computational Linguistics.

Silva-Schlenker, E., Jimenez, S., and Baquero, J. (2014). UNAL-NLP: Cross-lingual phrase sense disambiguation with syntactic dependency trees. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval-2014)*, Dublin, Ireland.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. (2006). A study of translation edit rate with targeted human annotation. In *In Proceedings of the 7th Conference of the Association for Machine Translation in the Americas (AMTA*, pages 223–231.

Stehouwer, H. and Van Zaanen, M. (2010). Finding patterns in strings using suffix arrays. In Ganzha, M. and Paprzycki, M., editors,

Proceedings of the International Multiconference on Computer Science and Information Technology, pages 505–511, Wisła, Poland. IEEE.

Stolcke, A. (2002). Srilm - an extensible language modeling toolkit. In Hansen, J. H. L. and Pellom, B. L., editors, *7th International Conference on Spoken Language Processing, ICSLP2002 - INTERSPEECH 2002, Denver, Colorado, USA, September 16-20, 2002*. ISCA.

Stroppa, N., van den Bosch, A., and Way, A. (2007). Exploiting source similarity for SMT using context-informed features. In Way, A. and Gawronska, B., editors, *Proceedings of the 11th International Conference on Theoretical Issues in Machine Translation (TMI 2007)*, pages 231–240, Skövde, Sweden.

Tan, L., Schumann, A., Martinez, J., and Bond, F. (2014). Sensible: L2 translation assistance by emulating the manual post-editing process. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval-2014)*, Dublin, Ireland.

Tiedemann, J. (2003). *Recycling Translations – Extraction of Lexical Data from Parallel Corpora and their Application in Natural Language Processing*. PhD thesis, Uppsala University, Uppsala, Sweden. Anna Sågvall Hein, Åke Viberg (eds): Studia Linguistica Upsaliensia.

Tiedemann, J. (2012). Parallel data, tools and interfaces in opus. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC12)*.

van den Bosch, A. (2004). Wrapped progressive sampling search for optimizing learning algorithm parameters. In Verbrugge, R., Taatgen, N., and Schomaker, L., editors, *Proceedings of the Sixteenth Belgian-Dutch Conference on Artificial Intelligence*, pages 219–226, Groningen, The Netherlands.

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

van den Bosch, A., Busser, G., Canisius, S., and Daelemans, W. (2007a). An efficient memory-based morpho-syntactic tagger and parser for Dutch. In Dirix, P., Schuurman, I., Vandeghinste, V., , and van Eynde, F., editors, *Computational Linguistics in the Netherlands: Selected Papers from the Seventeenth CLIN Meeting*, pages 99–114, Leuven, Belgium.

van den Bosch, A., Stroppa, N., and Way, A. (2007b). A memory-based classification approach to marker-based EBMT. In van Eynde, F., Vandeghinste, V., and Schuurman, I., editors, *Proceedings of the METIS-II Workshop on New Approaches to Machine Translation*, pages 63–72, Leuven, Belgium.

van Gompel, M. (2010). UvT-WSD1: A cross-lingual word sense disambiguation system. In *SemEval '10: Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 238–241, Morristown, NJ, USA. Association for Computational Linguistics.

van Gompel, M., Hendrickx, I., van den Bosch, A., Lefever, E., and Hoste, V. (2014). Semeval-2014 Task 5: L2 writing assistant. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 36–44, Dublin,Ireland.

van Gompel, M. and Reynaert, M. (2013). FoLiA: A practical XML format for linguistic annotation - a descriptive and comparative study. *Computational Linguistics in the Netherlands Journal*, 3.

van Gompel, M. and Reynaert, M. (2014). CLAM: Quickly deploy NLP command-line tools on the web. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics*.

van Gompel, M. and van den Bosch, A. (2013). WSD2: Parameter optimisation for memory-based cross-lingual word-sense disambiguation. In *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval 2013), in conjunction with the Second Joint Conference on Lexical and Computational Semantics*.

van Gompel, M. and van den Bosch, A. (2014). Translation assistance by translation of L1 fragments in an L2 context. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 871–880, Baltimore, Maryland. Association for Computational Linguistics.

van Gompel, M. and van den Bosch, A. (2016). Efficient n-gram, skipgram and flexgram modelling with Colibri Core. *Journal of Open Research Software*, 4(1).

van Gompel, M., van den Bosch, A., and Berck, P. (2009). Extending memory-based machine translation to phrases. In Forcada, M. and Way, A., editors, *Proceedings of the Third Workshop on Example-Based Machine Translation*, pages 79–86, Dublin, Ireland.

van Gompel, M., van den Bosch, A., and Dykstra, A. (2014). Oersetter: Frisian-dutch statistical machine translation. *Philologia Frisica anno 2012*, pages 287–296.

van Rossum, G. (2006). Python reference manual, release 2.5. Technical report, Amsterdam, The Netherlands, The Netherlands.

Varea, I. G., Informática, D. D., Och, F. J., Ney, H., and Casacuberta, F. (2002). Refined lexicon models for statistical machine translation using a maximum entropy approach. In *In Association for Computational Linguistics*.

Wang, L., Tu, Z., Way, A., and Liu, Q. (2017). Exploiting Cross-Sentence Context for Neural Machine Translation. *arXiv e-prints*, page arXiv:1704.04347.

Weiner, P. (1973). Linear pattern matching algorithms. In *SWAT (FOCS)*, pages 1–11. IEEE Computer Society.