Final thesis

Using Alignment Methods to Reduce Translation of Changes in Structured Information

by

Daniel Resman

LIU-IDA/LITH-EX-A--12/025--SE

2012-06-14
Final Thesis

Using Alignment Methods to Reduce Translation of Changes in Structured Information

by

Daniel Resman

LIU-IDA/LITH-EX-A--12/025--SE

2012-06-14

Supervisor: Magnus Merkel
Examiner: Lars Ahrenberg
Abstract

In this thesis I present an unsupervised approach that can be made supervised in order to reduce translation of changes in structured information, stored in XML-documents. By combining a sentence boundary detection algorithm and a sentence alignment algorithm, a translation memory is created from the old version of the information in different languages. This translation memory can then be used to translate sentences that are not changed. The structure of the XML is used to improve the performance.

Two implementations were made and evaluated in three steps: sentence boundary detection, sentence alignment and correspondence. The last step evaluates the using of the translation memory on a new version in the source language. The second implementation was an improvement, using the results of the evaluation of the first implementation. The evaluation was done using 100 XML-documents in English, German and Swedish. There was a significant difference between the results of the implementations in the first two steps. The errors were reduced by each step and in the last step there were only three errors by first implementation and no errors by the second implementation.

The evaluation of the implementations showed that it was possible to reduce text that requires re-translation by about 80%. Similar information can and is used by the translators to achieve higher productivity, but this thesis shows that it is possible to reduce translation even before the texts reaches the translators.
Acknowledgement

Without any real application data, this thesis would have been without correct context and the results not that interesting. I would like to thank Ingvar Waldebrink at Volvo Construction Equipment for letting me use their data.

I would also like to thank my supervisor, Associate Professor Magnus Merkel at Natural Language Processing Laboratory (NLPLab) at the Department of Computer Science at Linköping University. Thanks for our good discussions, all knowledge and feedback that you have provided.

I also owe thanks to Kaj Nyström, my supervisor at Combitech who helped me with many practical issues so I could focus on the thesis. Also many thanks for your feedback and our good discussions, which gave me a different perspective on the problem from Combitech’s point of view.

Last but not least I would like to extend my thanks to all at Combitech who have always helped me when I have had questions or problems about AllAddin and/or something completely different.
List of Figures and Tables

Figure 1-1: Alignment between documents ................................................................. 2
Figure 2-1: Editing of service information XML-document in Arbortext Editor .............. 4
Figure 2-2: Viewing a service information module ...................................................... 5
Figure 2-3: Export to translation (select categories) .................................................. 5
Figure 2-4 Export to translation (select language) .................................................... 5
Figure 2-5: Sentence alignment .................................................................................. 6
Table 2-6: Results on Wall Street Journal by Strunk et al. (2006) ............................... 8
Table 2-7: Results on Lacio-Web Corpus by Strunk et al. (2006) ............................... 8
Table 2-8: Results by Singh and Husain (2005) .......................................................... 10
Table 2-9: Results by Sánchez-Villamil et al. (2006) .................................................. 10
Table 2-10: Results by Fattah et al. (2006b, 2007) ..................................................... 11
Table 3-1: Number of duplicates .............................................................................. 12
Table 3-2: Non-equal element structures ................................................................. 12
Table 3-3: Manual sentence alignment (frequency) .................................................... 13
Table 3-4: Manual sentence alignment (percentage) ................................................... 13
Table 3-5: Changes .................................................................................................... 14
Figure 4-1: Pseudo code of the orthographic heuristic ................................................. 18
Figure 5-1: Data flow diagram .................................................................................. 21
Figure 5-2: Split by XML-tags .................................................................................. 22
Figure 5-3: Split by space character ......................................................................... 22
Figure 5-4: Data flow diagram of sentence boundary detection (training) ................ 24
Figure 5-5: Calculation of $D_2(i,j)$ and $B(i,j)$ ....................................................... 25
Figure 5-6: Path of beads example .......................................................................... 26
Table 5-7: Results of first sentence boundary detection algorithm .......................... 26
Table 5-8: Results of first sentence alignment implementation .................................. 27
Table 5-9: Results of first sentence alignment implementation (multi-sentence paragraphs) .... 27
Table 5-10: Results of correspondence in first implementation .................................. 27
Table 5-11: Results of second sentence boundary detection implementation .......... 29
Table 5-12: Results of correspondence in second implementation ........................... 29
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.4</td>
<td>Translation Export</td>
<td>23</td>
</tr>
<tr>
<td>5.5</td>
<td>First Implementation</td>
<td>23</td>
</tr>
<tr>
<td>5.5.1</td>
<td>Sentence Boundary Detection</td>
<td>23</td>
</tr>
<tr>
<td>5.5.2</td>
<td>Sentence Alignment</td>
<td>24</td>
</tr>
<tr>
<td>5.6</td>
<td>Results of First Implementation</td>
<td>26</td>
</tr>
<tr>
<td>5.6.1</td>
<td>Sentence Boundary Detection</td>
<td>26</td>
</tr>
<tr>
<td>5.6.2</td>
<td>Sentence Alignment</td>
<td>27</td>
</tr>
<tr>
<td>5.6.3</td>
<td>Correspondence</td>
<td>27</td>
</tr>
<tr>
<td>5.7</td>
<td>Second Implementation</td>
<td>28</td>
</tr>
<tr>
<td>5.7.1</td>
<td>Sentence Boundary Detection</td>
<td>28</td>
</tr>
<tr>
<td>5.7.2</td>
<td>Sentence Alignment</td>
<td>28</td>
</tr>
<tr>
<td>5.8</td>
<td>Results of Second Implementation</td>
<td>28</td>
</tr>
<tr>
<td>5.8.1</td>
<td>Sentence Boundary Detection</td>
<td>29</td>
</tr>
<tr>
<td>5.8.2</td>
<td>Sentence Alignment</td>
<td>29</td>
</tr>
<tr>
<td>5.8.3</td>
<td>Correspondence</td>
<td>29</td>
</tr>
<tr>
<td>6</td>
<td>Discussion</td>
<td>30</td>
</tr>
<tr>
<td>6.1</td>
<td>Limitations</td>
<td>30</td>
</tr>
<tr>
<td>6.2</td>
<td>First Implementation</td>
<td>30</td>
</tr>
<tr>
<td>6.3</td>
<td>Second Implementation</td>
<td>30</td>
</tr>
<tr>
<td>6.4</td>
<td>Further Development</td>
<td>30</td>
</tr>
<tr>
<td>6.5</td>
<td>Other Types of Texts</td>
<td>31</td>
</tr>
<tr>
<td>7</td>
<td>Conclusion</td>
<td>32</td>
</tr>
<tr>
<td>References</td>
<td></td>
<td>33</td>
</tr>
<tr>
<td>Appendix A</td>
<td>Results of Implementations</td>
<td>35</td>
</tr>
<tr>
<td>Appendix B</td>
<td>Module Swedish-English (XML)</td>
<td>36</td>
</tr>
<tr>
<td>Appendix C</td>
<td>Module Swedish-English</td>
<td>40</td>
</tr>
</tbody>
</table>
1 Introduction

Professional translators are consulted when it is important to get fluent and correct translations of information. Many companies produce a lot of information that need these kinds of translations, especially for the after sales market. These translations constitute a considerable expense so any reduction of translation is welcome. If a text has a translation, but some changes are made to the text, then you do not want send the whole text to translation. This thesis presents a solution to reduce translation of changed information and especially information stored in the semi-structured format XML.

1.1 Background

Combitech develops a system called UpTime where it is possible to produce, manage and distribute after sales information in multiple languages from a central management server. UpTime consists of a number of applications customized for different customers and for different purposes like: viewing, production and management or acting as server. One of these applications is called AllAddin and is used for production and management of information. There are two customizations of AllAddin: one for Volvo Construction Equipment (VCE) and one for Volvo Penta. The VCE version was used in this thesis.

AllAddin contains a lot of different types of information: training information, decision trees, service information, installation instructions and much more. Much of the information is represented as documents. Every document consists of a tree of modules and every module can be used in many different documents. It is the author’s job to make the modules general enough so they can be reused. A module can contain other modules, but also other objects such as pictures, phrases and warnings that also are reusable. Except for other modules and objects, a module can contain text with structures such as tables, lists and paragraphs, but also formatting such as bold, italic and superscript. A module is stored as XML-documents, one for every version and language.

The main advantage of this module approach is that much information can be reused. Similar products often have similar information. New documents can use modules in already existing documents and therefore speed up the process of writing documents. Another advantage is that when a document needs translation, those modules that already are translated to the target language do not need to be translated.

This type of information needs to be correct and therefore professional translators are used. Translation firms often get paid by the number of words that needs to be translated. The module approach saves a lot of money, but it is possible to do even better. Every time a module is changed and translated again, the whole module is translated even if it is just a word that is changed. So Combitech wants a way to reduce the translation cost even more.

1.2 Problem

Four different XML-documents play a part in this problem: the old version in the source language, $D_{s1}$, its translation, $D_{t1}$ and the new version in the source language, $D_{s2}$. By solving the problem a correct new translation, $D_{t2}$ will be possible at lower cost.
The problem is to identify the change between $D_{s1}$ and $D_{s2}$; get some sort of alignment between $D_{t1}$ and $D_{s2}$, where the alignment shows what in one of the documents corresponds to what in the other document. The second problem is hard to solve directly because it is hard to determine an alignment when both information and languages are different. $D_{s1}$, $D_{t1}$ should have the same information and $D_{s2}$ are written in the same language, so it is easier to align $D_{s1}$, $D_{t1}$ and $D_{s2}$ than aligning $D_{t1}$, $D_{s2}$ directly. Figure 1-1 shows an example of a solution. The parts marked with stripes could not be translated automatically from the previous versions and need to be manual translated. When these problems are solved, then it is easy to answer what has to be translated and where to put it.

![Alignment between documents](image)

**Figure 1-1: Alignment between documents**

### 1.3 Purpose

The purpose of the thesis was to develop an algorithm that determines what has to be translated, where the amount of what has to be translated is proportional to the change. It was also important that the algorithm determines where to put the new translation without getting an incorrect final translation.

### 1.4 Method

Texts are written in different ways in different languages and contexts. These ways needed to be identified to make the problem more specific and to understand what could be used in an implementation. One key feature of these texts is that they are stored in XML. The structure of the XML and how it is used are as important as the texts itself. With a good view of the problem, better implementations could be made and therefore analyzing the texts is a good step before the implementation.
Two types of implementations were developed: implementation to extract the right data from the database and implementations to test a given solution. These implementations were not integrated into AllAddin, but were run as standalone programs that could use some of AllAddins features. Two solutions were implemented in an iterative way where the first implementation was evaluated to be able to do the next implementation better.

1.5 Limitations
Only versions of modules in English, Swedish and German and where the source language was in one of them have been used. This is because there was only knowledge about these languages and the knowledge was needed to do a good evaluation of the implementations.

There are a lot of different types of documents in AllAddin. The different types are represented a little different and have different types of information. The solution needs to know how the documents are represented to make the right decisions. There was a limited amount of time provided for this thesis so it was better to focus on one type of document, rather than the different types of documents and their representation. Still it should be easy to modify the solution so it can handle other types of documents in AllAddin. The service information modules had a substantial amount of text and the number of modules was also relative large. Service information modules were therefore chosen to be used in this thesis.
2 Background

2.1 AllAddin

AllAddin is a Windows application written in C#. All information is stored in a database and can be viewed and modified using the application. Every version and language of a module is stored as XML. The XML can easily be edited in the application, but the editing is restricted by a Document Type Definition (DTD). A DTD defines the structure of an XML-document with a list of legal elements and their attributes (Refsnes Data, 2012). Some visual help is present, like showing a table instead of displaying the tags. This can be seen in Figure 2-1, where a service information module is edited using the integrated Arbortext Editor.

Figure 2-1: Editing of service information XML-document in Arbortext Editor

Before viewing a module some steps need to be taken by the application. Because modules can contain other objects and modules, these need to be inserted into the XML. A new XML-document is constructed. The XML is used for storing information, not viewing; therefore an XSLT-document is used to transform the XML into HTML. To the left in Figure 2-2 is the result after the insertion and transformation. In the same figure to the right is the tree structure of the modules in AllAddin.
If you want to export information to translation in AllAddin you first select what type of information, for example service information, and then which information, source language and target language shown in Figure 2-3 and Figure 2-4 respectively.

The export results in XML-documents with modules that need translation marked.

2.2 Alignment
There are many different ways to align documents. A common way is to divide the documents into sentences and then align these. With a division of the documents into sentences, an alignment
consists of groups of consecutive sentences in the source document, $S$, that correspond to groups of consecutive sentences in target document, $T$. Let $S_i, i = 1, ..., I$, be the groups of consecutive sentences in $S$ and $T_j, j = 1, ..., J$, be the groups of consecutive sentences in $T$, where $S_i$ and $T_j$ are the $i^{th}$ and $j^{th}$ group of sentences in $S$ and $T$ respectively. A crossing alignment is when correspondent groups are not consecutive, for example groups $S_i$ and $S_j$ of the source language correspond to the groups $T_j$ and $T_i$ respectively. Crossing alignments are rarely encountered in practice (Li, et al., 2010).

Because of this it is often assumed that correspondent groups are consecutive, to simplify the problem. By making the assumption, a correspondence between two groups can be written as $n-m$, where $n$ sentences in source document correspond to $m$ sentences in target document. Brown et al. (1991) call these correspondences between two groups “beads”. The good thing with using sentences is that the most common bead is the 1-1 bead. Gale and Church (1991) used a trilingual corpus (English-French, English-German) of 15 economic reports issued by the Union Bank of Switzerland where 89 % were 1-1 beads. 92 % were 1-1 beads in the English-Arabic corpus used by Fattah et al. (2006) and Li et al. (2010) used a English-Chinese corpus of Internet articles with 83 % 1-1 beads. Depending on the documents used it is harder or easier to do an alignment. Figure 2-5 shows two texts from VCE that are sentence aligned with beads: 1-1, 1-2 and 1-1.

![Figure 2-5: Sentence alignment](image)

### 2.2.1 Definition of Sentence

The purpose of a sentence in alignment of documents is to be a unit of tokens that is small and have a good correspondence to other unit of tokens (sentences). So it is a bad idea to define a sentence in a grammatical way that can be very hard to detect and does not need to have a good correspondence. Instead I define a sentence as one part of a division of a unit of text, divided by finding what Nunberg (1990) calls a “text-sentence”. He explains that “The text-sentence is that unit of written texts that is customarily presented as bracketed by a capital letter and a period (though those properties are not criterial)”. A single number in a table cell is not a text-sentence, but a sentence according to my definition because it is a unit of text that could not be divided into two or more text-sentences.

### 2.2.2 Problem of Sentence Boundary Detection

To divide a text into sentences may seem at first as an easy task. Just compare characters in the text to a list of sentence boundary punctuation marks such as “.”, “!” , “?” and divide. These punctuation marks are however ambiguous in English, Swedish, German and many other languages. A period can be used in internet addresses, abbreviations and to indicate ellipsis (“…”). In English, periods are also used as decimal points and in German, ordinal numbers written in digits end with a period¹.

¹ Examples of this are: “1. Gang” that means “1st gear” or “Von 7. bis zum 12. August” that means “From 7th to 12th August”. 
tricky part is that abbreviations and ellipses for example, can both end a sentence and be used in the middle of one. Exclamation marks and question marks are less ambiguous, but can be used multiple times for emphasis. Exclamation marks can also appear in proper names, such as “Yahoo!” and “Jeopardy!”. Internal periods in tokens, for example in internet addresses and decimal numbers can easily be ignored, because the periods are not at the end of the token.

2.2.3 Approaches on Sentence Boundary Detection

Previous work has been done in the area of detecting sentence boundaries and with good results. Many different approaches have been made that suits different types of data, but it is just not the error rate of the different systems on specific corpora that differs. What also differs is how much and what information that is needed to get good results. The different systems explained here use a similar definition of sentence as in 2.2.1.

One of the first very successful systems for sentence boundary detection was developed by Riley (1989). It uses regression trees (Breiman, et al., 1984) to classify periods to some features about the preceding and following token. The probabilities of a token occurring next to a sentence boundary are compiled from a sentence boundary labeled corpus.

The SATZ system (Palmer & Hearst, 1997) uses a similar approach as the system developed by Riley (1989), but instead of using the tokens themselves surrounding a period it uses part-of-speech estimates of the token. Every token is represented by a vector of part-of-speech distribution of the token. The part-of-speech distribution is obtained from a lexicon containing part-of-speech frequency data. If a token is not in the lexicon, different guessing heuristics is used for estimation. SATZ also uses capitalization information and an abbreviation list. The vectors are used as input to a machine learning algorithm to classify periods. Palmer & Hearst (1997) used both the C4.5 decision tree classifier (Quinlan, 1993) and neural networks. To train the system a corpus with sentence boundaries labeled is needed.

Like the previous systems described, the MxTerminator system (Reynar & Ratnaparkhi, 1997) uses the contexts of end-of-sentence punctuation marks and a machine learning algorithm to identify sentence boundaries. Maximum entropy modeling in combination with an abbreviation list is used as the machine learning algorithm. A corpus with sentence boundaries labeled is used for training. The abbreviation list is induced from the training corpus by treating every token with a period that is not labeled as a sentence boundary as an abbreviation.

Another system, developed by Gillick (2009), uses the left and the right token of a period and features of those tokens to decide if it is a sentence boundary. This system uses an implementation of support vector machine (SVM) with linear kernel called SVM$^{\text{light}}$ (Joachims, 1999) as machine learning algorithm. As input to the SVM a corpus with sentence boundaries labeled is used.

The SATZ, MxTerminator and the systems developed by Gillick and Riley are all using a supervised learning algorithm, meaning that pre-labeled data is needed, but the Punkt system (Kiss & Strunk, 2006) is instead using an unsupervised learning algorithm to detect sentence boundaries. It uses a log-likelihood ratio based heuristic to detect abbreviations. The context and other log-likelihood ratio heuristics decide if an ellipsis or an abbreviation is a sentence boundary or not. Every other token that ends with a period is classified as a sentence boundary.
2.2.4 Evaluation of Sentence Boundary Detection Systems

The system developed by Riley (1989) was trained by data from Associated Press news with 25 million words and then tested on the Brown Corpus with an error rate of 0.2%. This is very good results, but 25 million words is a lot of data and the data was also pre-labeled with sentence boundaries. Tests on smaller sets of data were not reported.

In (Gillick, 2009) it was reported of tests on Wall Street Journal with training data of 500, 5,000 and 42,317 sentences from Wall Street Journal. The error rates were 1.36 %, 0.52 % and 0.25 % respectively.

To compare different sentence boundary algorithms, three different measures can be used: precision, recall and F-measure. Precision is calculated as the number of correctly classified sentence boundaries divided by the number of identified sentence boundaries. Recall is the number of correctly classified sentence boundaries divided by the actual number of sentence boundaries. The F-measure is a combination of both precision and recall, calculated in Equation 1.

Equation 1: F-measure

\[ F = 2 \cdot \frac{\text{recall} \cdot \text{precision}}{\text{recall} + \text{precision}} \]

SATZ, MxTerminator and Punkt were evaluated in (Strunk, et al., 2006). Table 2-6 shows the results with 156 English articles from Wall Street Journal containing 3,455 sentences as test corpus. Punkt was tested with a training of all the articles and with a new training for every article that were tested.

<table>
<thead>
<tr>
<th>System</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Punkt (all)</td>
<td>90.70 %</td>
<td>92.34 %</td>
<td>91.51 %</td>
</tr>
<tr>
<td>Punkt (individual articles)</td>
<td>80.43 %</td>
<td>83.40 %</td>
<td>81.88 %</td>
</tr>
<tr>
<td>MxTerminator</td>
<td>91.19 %</td>
<td>91.25 %</td>
<td>91.22 %</td>
</tr>
<tr>
<td>SATZ</td>
<td>98.67 %</td>
<td>85.98 %</td>
<td>91.88 %</td>
</tr>
</tbody>
</table>

Table 2-6: Results on Wall Street Journal by Strunk et al. (2006)

The systems were also tested on the Lacio-Web Corpus with 21,822 Portuguese sentences using a 10-fold cross-validation. Table 2-7 shows the results of the test.

<table>
<thead>
<tr>
<th>System</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Punkt</td>
<td>97.58 %</td>
<td>96.87 %</td>
<td>97.22 %</td>
</tr>
<tr>
<td>MxTerminator</td>
<td>96.31 %</td>
<td>96.63 %</td>
<td>96.46 %</td>
</tr>
<tr>
<td>SATZ</td>
<td>99.59 %</td>
<td>98.74 %</td>
<td>99.16 %</td>
</tr>
</tbody>
</table>

Table 2-7: Results on Lacio-Web Corpus by Strunk et al. (2006)

2.2.5 Approaches on Sentence Alignment

As with sentence boundary detection some previous work has been done on sentence alignment and with different approaches. These approaches differs mostly in how much and what information that is needed to get good results. The different algorithms often align sentences by maximizing a probability or minimizing a cost/distance of combining different sentences. This is often solved by a dynamic programming search.

The first successful approach was to use the length of sentences. A sentence with a specific length in one language typically corresponds to a sentence with a similar length in another language. Brown et
al. (1991) developed a method that uses the number of words in a sentence as the length of the sentence. A probability is calculated by taking the probability of a bead and the probabilities of the lengths of the sentences. The probabilities of beads and the probabilities of the lengths were determined by analyzing texts. 0-1, 1-0, 1-1, 1-2 and 2-1 were the different beads used. For 1-1, 1-2 and 2-1, the probability of the lengths of the target language were not determined by the analysis. Instead the probabilities are assumed to be distributed according to a normal distribution of the ratios between the total lengths of sentences in the different languages. The mean and variance of the normal distribution were also determined by analyzing texts. A method that builds upon this method is described in (Moore, 2002). The difference is that a Poisson distribution is used instead and the method is just the first step of three. In the second step, sentences that got a high probability in the first step are used to train a modified version of IBM Translation Model 1 (Brown, et al., 1993), which is a word aligner. In the last step the first method in combination with the word aligner is used to align sentences.

Another sentence aligner using the lengths of sentences were developed by Gale & Church (1991). The number of characters is used to specify the length of a sentence. The method assumes a model where every character in one language gives rise to a random number of characters in another language. The random variables are independent and are identically distributed according to a normal distribution. The mean and variance were gained by analyzing texts. A value is calculated on the distribution by taking the difference of the total length of sentences in a bead. The value is used to integrate a standard normal distribution (with mean zero and variance one) to get the probability of the magnitude of the value. This probability is multiplied with the probability of the bead to get a final probability that is used to calculate a distance.

Different types of classifiers have been used. A probabilistic neural network (P-NNT) classifier in (Fattah, et al., 2006a), a feed forward neural network (FF-NNT) classifier in (Fattah, et al., 2006b) and a Gaussian mixture model (GMM) classifier in (Fattah, et al., 2007) use the same three features for training and classification. The first feature is the lengths of sentences in number of characters. Alignment of symbols (";", "(" , "%" etc.) are used in the second feature. The best alignment is used and from that a factor is calculated depending on how probable the alignment is. This factor is then used as the second feature. Another factor called the cognate factor is calculated from shared words, the more shared words the higher the factor. This is the last feature. There are different inputs to the methods from the three different features depending on the bead.

Sánchez-Villamil et al. (2006) have developed different methods for aligning sentences in HTML texts. The HTML is actually converted to XML (XHTML), using the tidy\(^2\) program, before used in the methods. Tags are divided into 4 categories: structural, format, content and irrelevant. The different methods try to minimize an edit distance cost. A table with costs of inserting, deleting and changing the different tags and normal text is used to sum up the cost. The differences between the different methods are in what order things are done and how the cost between text segments is calculated.

Another method using HTML texts were developed by Zhu et al. (2011). The method uses a graph-based approach instead of a dynamic programming search when maximizing a weight. Nodes in the graph represent sentences and edges represent correspondence between sentences. LIBSVM\(^3\) a

\(^2\) http://tidy.sourceforge.net/  
\(^3\) http://www.csie.ntu.edu.tw/~cjlin/libsvm/
library for support vector classification is used to get weights on edges. Many different features are used as input to the classifier, both features between sentences and the sentences themselves. HTML tags, length of the sentences in number of characters, pair of mutual translated terms are some of the features used.

2.2.6 Evaluation of Sentence Alignment Methods

Precision, recall and F-measure can be used for performance measurement of sentence alignment methods in the same way as for sentence boundary detection systems. Correctly classified beads divided by identified beads is the precision. Recall is correctly classified beads divided by the actual number of beads. F-measure is the same and is calculated by Equation 1.

In (Singh & Husain, 2005) the method developed by Gale and Church (1991) and the method developed by Brown et al. are compared to the method developed by Moore (2002). Three English-Hindi corpora: EMILLE, ERDC and India Today and one English-French corpus were used. 2500 sentences were extracted from each corpus. The result is shown in Table 2-8. Why the poor performance by the method by Moore (2002) with the EMILLE corpus is not discussed in (Singh & Husain, 2005).

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Measure</th>
<th>Brown</th>
<th>Gale &amp; Church</th>
<th>Moore</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMILLE</td>
<td>Precision</td>
<td>99.3 %</td>
<td>99.1 %</td>
<td>66.8 %</td>
</tr>
<tr>
<td></td>
<td>Recall</td>
<td>96.0 %</td>
<td>93.0 %</td>
<td>63.2 %</td>
</tr>
<tr>
<td></td>
<td>F-measure</td>
<td>97.6 %</td>
<td>96.0 %</td>
<td>64.9 %</td>
</tr>
<tr>
<td>ERDC</td>
<td>Precision</td>
<td>99.6 %</td>
<td>99.5 %</td>
<td>100.0 %</td>
</tr>
<tr>
<td></td>
<td>Recall</td>
<td>99.9 %</td>
<td>99.1 %</td>
<td>97.0 %</td>
</tr>
<tr>
<td></td>
<td>F-measure</td>
<td>99.3 %</td>
<td>99.3 %</td>
<td>98.4 %</td>
</tr>
<tr>
<td>India Today</td>
<td>Precision</td>
<td>91.8 %</td>
<td>93.9 %</td>
<td>99.5 %</td>
</tr>
<tr>
<td></td>
<td>Recall</td>
<td>81.0 %</td>
<td>83.0 %</td>
<td>81.5 %</td>
</tr>
<tr>
<td></td>
<td>F-measure</td>
<td>86.1 %</td>
<td>88.1 %</td>
<td>89.6 %</td>
</tr>
<tr>
<td>English-French</td>
<td>Precision</td>
<td>100.0 %</td>
<td>100.0 %</td>
<td>100.0 %</td>
</tr>
<tr>
<td></td>
<td>Recall</td>
<td>100.0 %</td>
<td>99.3 %</td>
<td>99.3 %</td>
</tr>
<tr>
<td></td>
<td>F-measure</td>
<td>100.0 %</td>
<td>99.6 %</td>
<td>99.6 %</td>
</tr>
</tbody>
</table>

Table 2-8: Results by Singh and Husain (2005)

Sánchez-Villamil et al. (2006) evaluated their method using three different corpora. The first one is taken from elPeriódico, an online daily newspaper with texts in Spanish and Catalan. An English-Spanish corpus containing a small fragment from Quitoxe was the second corpus. The last corpus consisted of texts from help pages of the chatting client mIRC\(^4\) in Spanish, Portuguese, Italian, Catalan and Galician. The five languages resulted in ten language pairs that were all aligned. The result is displayed in Table 2-9. Both Quitoxe and mIRC corpus contains lots of 0-1 and 1-0 beads, therefore the difference in performance.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>elPeriódico</td>
<td>94 %</td>
<td>94 %</td>
<td>94 %</td>
<td>71 %</td>
<td>77 %</td>
<td>74 %</td>
<td>51 %</td>
<td>57 %</td>
<td>54 %</td>
</tr>
<tr>
<td>Quitoxe</td>
<td>71 %</td>
<td>77 %</td>
<td>74 %</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mIRC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2-9: Results by Sánchez-Villamil et al. (2006)

Zhu et al. (2011) used a training corpus and a test corpus with 200 bilingual web pages each. This resulted in a precision of 86 % and a recall of 82 %, which lead to an F-measure of 84 %.

\(^4\) http://www.mirc.co.uk/
The feed forward neural network (FF-NNT) classifier in (Fattah, et al., 2006b) were trained and tested on English-Arabic corpus with text from web pages. In (Fattah, et al., 2007) a Gaussian mixture model (GMM) classifier method is described, but it is also compared to many other methods, among others: the probabilistic neural network (P-NNT) classifier method in (Fattah, et al., 2006a), a length-based method similar to the one by Gale and Church (1991) and the method developed by Moore (2002). The same training corpus and test corpus were used in the two cases. Features were extracted from 7653 sentence pairs to train the classifiers and 1200 sentence pairs were used for testing. The result is displayed in Table 2-10.

<table>
<thead>
<tr>
<th>Measure</th>
<th>Length-based</th>
<th>Moore</th>
<th>FF-NNT</th>
<th>P-NNT</th>
<th>GMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Errors</td>
<td>77</td>
<td>57</td>
<td>32</td>
<td>57</td>
<td>39</td>
</tr>
<tr>
<td>Error rate %</td>
<td>6.4 %</td>
<td>4.7 %</td>
<td>2.6 %</td>
<td>4.7 %</td>
<td>3.2 %</td>
</tr>
</tbody>
</table>

Table 2-10: Results by Fattah et al. (2006b, 2007)

2.3 Translation Memory

According to Macklovitch and Russel (2000) there are at least two different definitions of the term “translation memory”. According to the narrower of the two, a translation memory (TM) “is a particular type of translation support tool that maintains a database of source and target-language sentence pairs, and automatically retrieves the translation of those sentences in a new text which occur in the database”. The other “definition regards TM simply as an archive of past translations, structured in such way as to promote translation reuse”. The broader definition of TM is the idea of the sentence alignment between $D_{s1}$ and $D_{t1}$, where the translation reuse is the translation of $D_{s2}$.

A translation memory tool is used by the translators of the XML-documents in this thesis to achieve higher productivity. In a survey by Lagoudaki (2006), 82.5 % of 874 translation professionals confirmed use of TM and the usage of TM correlated with technical texts. When a translator uses a TM tool, the correctness of the translation is not crucial. The tool can suggest so called “fuzzy matches”, where the source text is similar to the one in the TM, but not the same. These fuzzy matches can then be used as a starting point for the translation. In the problem of this thesis it is different. It is important that the translation is correct because the alignment is not used to increase the productivity of a translator, instead it is used to get a correct translation of a text before it reaches the translators.

Even if the correctness of a translation is not crucial for translators, it is still better with a correct translation. Therefore there have been attempts to make correct translations automatically from fuzzy matches. For example Biçici and Dymetman (2008) used a TM in combination with statistical machine translation to improve the fuzzy matches. They call it a dynamic translation memory. A more straight forward approach is to look for, dates, times, numerical expressions and other expressions that are language invariant or can easily be converted to the target language.
3 Analysis of Data

Only versions of service information modules in English, Swedish and German and where the source language was in one of them have been used for the analysis as described in 1.5. The database used had information from VCE with 58610 service information modules in a total of 69817 versions. Every version in one language is stored as one XML-document. No insertion was done as described in 2.1. Some metadata was added to the documents where the XML version, encoding and location of a XSLT-file were described. This was done so the files could be viewed correctly in a web browser.

Two things are needed to solve the problem of reducing translation: a previous translation and a change. These two things have been analyzed to get a better understanding of what the problems are.

Every analyzed module had Swedish or English as the source language and were in at least two versions. All the versions had the status “released” and had an English, Swedish and German translation. Some versions had duplicates in the source language and were not used. The statistics of the duplicates are presented in Table 3-1. These requirements resulted in 2643 modules in a total of 5961 versions. Modules with German as source language were not analyzed because there existed none with all the requirements. Versions with Swedish as the source language were overrepresented with 4707 versions.

<table>
<thead>
<tr>
<th>Source language</th>
<th>English duplicates</th>
<th>Swedish duplicates</th>
<th>German duplicates</th>
<th>Versions with duplicates</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>164</td>
<td>165</td>
<td>170</td>
<td>174</td>
</tr>
<tr>
<td>Swedish</td>
<td>865</td>
<td>884</td>
<td>851</td>
<td>921</td>
</tr>
</tbody>
</table>

Table 3-1: Number of duplicates

3.1 Translations

Two things were analyzed between the modules in source language and their translations. First the difference in structure of the XML and second the sentence alignments.

3.1.1 Structure

I have defined equal element structure of two XML-documents as: For every element in one document there exists an element in the other document of the same type, with the same number of elements before and with the same number of child elements.

The 5961 versions were run through a program that detected if the element structure between the source language and the translations were not equal. The result was split into four groups and is presented in Table 3-2, which shows how many documents that not had equal element structure. 49 versions were manually analyzed from each group. In the versions with Swedish as source language, the elements br and super were often added or removed with a translation, but with English as source language, the element pagebreak were more often added or removed.

<table>
<thead>
<tr>
<th>Source language</th>
<th>English translation</th>
<th>Swedish translation</th>
<th>German translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>-</td>
<td>49</td>
<td>61</td>
</tr>
<tr>
<td>Swedish</td>
<td>171</td>
<td>-</td>
<td>153</td>
</tr>
</tbody>
</table>

Table 3-2: Non-equal element structures
A translator should not change the layout of the documents, but there are some elements that the translator can add or remove. The most common case of elements that probably was changed by a translator was the element super, which denotes superscript. In 20 of the cases the element super had been added or removed. This had to do with different units, such as changing “dm$^3$” to “l”. But the most common reason for a non-equal element structure had to do with a feature called Limited Checkout. This feature enables a user to checkout a translation and change the content. This feature exists, so typos, small layout changes and other small things can be fixed, but the feature can be misused. In 26 of the cases; information had been added or removed.

3.1.2 Sentence Alignment

100 modules were randomly picked with English as source language and the same number were picked with Swedish as source language, with one version from each module, so a total of 200 versions were analyzed. This analysis was conducted to determine how the sentences are aligned between the source language and the different languages.

For every paragraph or group of text with more than one sentence a manual sentence alignment was performed and different beads were identified. The frequencies of the different beads are displayed in Table 3-3 and the percentage of each bead is displayed in Table 3-4. If the result is compared to other sentence aligned texts it is easy to notice that the 1-1 bead is more common in this result. This is probably due to a small number of sentences in each paragraph.

<table>
<thead>
<tr>
<th>Source-Target Language</th>
<th>0-1</th>
<th>1-0</th>
<th>1-1</th>
<th>1-2</th>
<th>2-1</th>
<th>1-3</th>
<th>3-1</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>English-German</td>
<td>0</td>
<td>2</td>
<td>292</td>
<td>6</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>304</td>
</tr>
<tr>
<td>English-Swedish</td>
<td>0</td>
<td>0</td>
<td>304</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>307</td>
</tr>
<tr>
<td>Swedish-English</td>
<td>0</td>
<td>0</td>
<td>405</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>412</td>
</tr>
<tr>
<td>Swedish-German</td>
<td>0</td>
<td>1</td>
<td>386</td>
<td>9</td>
<td>9</td>
<td>1</td>
<td>0</td>
<td>406</td>
</tr>
</tbody>
</table>

Table 3-3: Manual sentence alignment (frequency)

<table>
<thead>
<tr>
<th>Source-Target Language</th>
<th>0-1</th>
<th>1-0</th>
<th>1-1</th>
<th>1-2</th>
<th>2-1</th>
<th>1-3</th>
<th>3-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>English-German</td>
<td>0 %</td>
<td>0.66%</td>
<td>96.05%</td>
<td>1.97%</td>
<td>1.32%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>English-Swedish</td>
<td>0 %</td>
<td>0%</td>
<td>99.02%</td>
<td>0.65%</td>
<td>0.33%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Swedish-English</td>
<td>0 %</td>
<td>0%</td>
<td>98.30%</td>
<td>0.99%</td>
<td>0.74%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Swedish-German</td>
<td>0 %</td>
<td>0.25%</td>
<td>95.07%</td>
<td>2.22%</td>
<td>2.22%</td>
<td>0.25%</td>
<td>0%</td>
</tr>
</tbody>
</table>

Table 3-4: Manual sentence alignment (percentage)

3.2 Changes

100 modules were randomly picked with English as source language and the same number were picked with Swedish as source language. The changes between 2 consecutive versions from each module were inspected. This inspection was conducted to determine how much is typically changed and what is changed. Frequencies of some interesting properties of the changes were determined by manual inspection and are showed in Table 3-5. About half of the English versions and a third of the Swedish versions had change in only 1-5 places. Changes are typically small and therefore much can be done to reduce what has to be translated. Versions that only have change of numbers, language invariant names, links and structure do not need to be translated and they make up 15 % of the inspected versions. In only 10 % of the versions there were changes in other places in the translation. This means that the majority of the information that has already been translated is translated to the
The same thing. The reason for this is probably due to that translation firms use translation memories and reuse previous translations.

<table>
<thead>
<tr>
<th>Property</th>
<th>English</th>
<th>Swedish</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whole module changed</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>More than half changed</td>
<td>21</td>
<td>5</td>
</tr>
<tr>
<td>Change in 2-5 places</td>
<td>23</td>
<td>18</td>
</tr>
<tr>
<td>Change in 1 place</td>
<td>26</td>
<td>12</td>
</tr>
<tr>
<td>Only change of numbers</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>Only change of language invariant names</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Only links added</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>Only structural change</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>Changes in other places in the translations</td>
<td>13</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 3-5: Changes
4 Choice of Algorithms
There are good algorithms for sentence boundary detection and sentence alignment. It would be nice to choose many different algorithms for each problem, but it would complicate things and take a considerable amount of time. This thesis is not about comparing different algorithms for sentence boundary detection or sentence alignment. That has already been done by others. This thesis is about solving another problem, so one algorithm was selected per problem.

All the sentence boundary detection algorithms discussed in 2.2.4 show good and pretty equal results. The system developed by Riley (1989) shows very good results, but it is unclear what the results would be with a smaller training set. All the systems except the Punkt system need a training corpus with sentence boundaries labeled. The problem with such a training corpus observed in (Reynar & Ratnaparkhi, 1997), is that it must be a good match with the test corpus. The test data in this thesis consists of service information from Volvo Construction Equipment. This is a very specific genre of texts so the training corpus must also be trained by these texts. This means that a lot of sentence boundaries must be labeled. The time needed for constructing such a training corpus is not worth the while when there is an unsupervised alternative, the Punkt system that performs pretty well in comparison with the supervised alternatives and therefore it was chosen.

The algorithms discussed in 2.2.6 are good on different type of texts. The method by Sánchez-Villamil et al. (2006) and Zhu et al. (2011) are specialized for web pages (HTML). They both use structural information to help align sentences. There is a certain similarity with the XML-documents in AllAddin and HTML and the algorithms could probably easily be modified to fit the XML-documents. There is a difference in results with these methods compared to the rest of the evaluated ones. This is probably due to the fact that web pages are typically noisy with lots of 0-1 and 1-0 beads. It is hard to tell how well these methods would perform on the XML-documents. The performance of the classifier methods is not much better than the performance of the solely length-based methods. The classifier methods were not chosen, due to the time of extracting features for the classifiers and the little performance difference. The combination of simplicity and performance makes the length-based methods really good. One exception is the method by Moore (2002) that is more advanced, but is not necessarily better as showed by Singh and Husain (2005). The difference in performance between the method developed by Brown et al. (1991) and the method developed by Gale and Church (1991) is small, but the latter was anyhow chosen.

4.1 Sentence Boundary Detection Algorithm
The chosen algorithm for sentence boundary detection by Kiss and Strunk (2006) will here be explained in more detail. The algorithm consists of two phases: one training phase and one decision phase.

4.1.1 Training Phase
The first thing that is trained is a list of abbreviations. Abbreviations are very important because they are ambiguous as sentence boundaries. If there is a token with a final period and it is not an abbreviation or ellipsis it is nearly guaranteed to be a sentence boundary. A log-likelihood ratio test is used to detect abbreviations. By such a test two hypotheses are compared: a null hypothesis $H_0$ and an alternative hypothesis $H_a$. The hypotheses used are shown in Equation 2 and Equation 3, where
\( C(\ldots) \) signifies the frequency of some token or tokens in the corpus, \( N \) is the total number of tokens in the corpus and \( w \) is the token tested.

**Equation 2: \( H_0 \)**

\[
P(\cdot|w) = p_0 = \frac{C(\cdot)}{N}
\]

**Equation 3: \( H_a \)**

\[
P(\cdot|w) = p_a = 0.99
\]

\( H_a \) represents an arbitrary token and \( H_a \) represent an abbreviation where the probability of a final period is much higher. A binomial distribution is used to calculate the log-likelihoods ratio shown in Equation 5, where the values are specified in Equation 6. A high value indicates an abbreviation.

**Equation 4: Log-likelihood**

\[
\log L(p, k, n) = k \cdot \log p + (n - k) \log(1 - p)
\]

**Equation 5: Log-likelihood ratio for abbreviations**

\[
\log \lambda = -2 \cdot \log \frac{P_{\text{binom}}(H_0)}{P_{\text{binom}}(H_a)} = -2 \cdot \log \left( \frac{\binom{n}{k} p_o^k (1 - p_o)^{n-k}}{\binom{n}{k} p_a^k (1 - p_a)^{n-k}} \right) = 2(\log L(p_a, k, n) - \log L(p_o, k, n))
\]

**Equation 6: Abbreviation values**

\[
k = C(w), n = C(w)
\]

This log-likelihood ratio is not considered good enough by Kiss and Strunk (2006). Three different factors are multiplied with the ratio to get better detection. They come from three characteristic properties of abbreviations: they tend to be short, internal periods are common and they often occur with a final period. The first factor depends on the length of the token and is given in Equation 7. Characters except periods are counted in the length of a token.

**Equation 7: Length factor**

\[
F_{\text{length}}(w) = \frac{1}{e^{\text{length}(w)}}
\]

Internal periods are favored in the second factor shown in Equation 8.

**Equation 8: Internal period factor**

\[
F_{\text{periods}}(w) = \text{number of internal periods in } w + 1
\]

The last factor, given in Equation 9 penalizes long tokens without final periods.

**Equation 9: Penalty factor**

\[
F_{\text{penalty}}(w) = \frac{1}{\text{length}(w)^{C(w,r)}}
\]

A token is considered as an abbreviation if it gets a value larger than 0.3. The value was determined by manual experimentation.
I use the definition that a collocation is consecutive tokens that co-occur more often than would be expected by chance. Kiss and Strunk (2006) uses a broader definition to explain their algorithm, but I choose not to use it because it differs greatly from the common use of the word collocation in corpus linguistics. The next step learns pair of tokens that represent collocations between initials and their following tokens. Initials are a special case of abbreviations with one letter and a final period. There is also an optional part in this step where collocations between numbers and their following tokens are also learned. This extra part is used for languages such as German, where ordinal numbers written in digits ends with a final period explained in 2.2.2. Specific numbers are often not that frequent and are therefore seen as equal tokens. Log-likelihood ratios are used to detect collocations. The hypotheses used are taken from (Dunning, 1993) and are displayed in Equation 10 and Equation 11, where \( w_1 \) and \( w_2 \) are consecutive tokens.

**Equation 10: H_0**

\[
P(w_2|w_1) = p = P(w_2|\neg w_1)
\]

**Equation 11: H_a**

\[
P(w_2|w_1) = p_{a1} \neq p_{a2} = P(w_2|\neg w_1)
\]

The null hypothesis assumes that the probability of \( w_2 \) is independent on the preceding token. The alternative hypotheses states that there is a dependency between \( w_1 \) and \( w_2 \). This results in the log-likelihood ratio in Equation 12, where the values are specified in Equation 13.

**Equation 12: Log-likelihood ratio for collocations**

\[
\log \lambda = 2(\log L(p_{a1}, k_1, n_1) + \log L(p_{a2}, k_2, n_2) - \log L(p, k_1, n_1) - \log L(p, k_2, n_2))
\]

**Equation 13: Collocation values**

\[
p = \frac{C(w_2)}{N}, p_{a1} = \frac{C(w_1, w_2)}{C(w_1)}, p_{a2} = \frac{C(w_2) - C(w_1, w_2)}{N - C(w_1)},
\]

\[
k_1 = C(w_1, w_2), k_2 = C(w_2) - C(w_1, w_2), n_1 = C(w_1), n_2 = N - C(w_1)
\]

A problem with this test is that it can generate high values even if \( w_1 \) and \( w_2 \) do not form a collocation. This is the case when \( w_2 \) occurs less often after \( w_1 \) than expected. This is solved by adding a constraint. This leads to Equation 14, where the threshold is specified to 7.88 which represent a confidence of 99.5 %.

**Equation 14: Collocation heuristic**

\[
\langle w_1, w_2 \rangle \text{ is a collocation if } \log \lambda > \text{threshold and } p_{a1} > p
\]

The last step learns frequent sentence starters, in pretty much the same way as collocations are learned. Sure sentence boundaries are single token-final periods following a token that is not in the abbreviation list or is not an initial and in the case of German and languages alike is not a sequence of digits. Equation 14 is used to detect frequent sentence starters, but \( w_1 \) is here representing a sure sentence boundary and \( w_2 \) is the token after, the sentence starter. The threshold is set to the high value of 30 because of the uncertain information used.
4.1.2 Decision Phase

The second phase, the decision phase, is where decisions are made; if a period represents a sentence boundary or not. A heuristic that rely on an orthographic convention is used to decide some cases. The convention that a token preceding a sentence boundary is usually capitalized is used. All nouns in German and proper nouns in English are typically written capitalized even if they appear sentence-internal. There also exist names that are conventionally written in lowercase and a mathematical variable typically has the same case wherever it stands. A text could also be written in only upper- or lowercase, therefore capitalization information is only used cautiously. The pseudo code for the heuristic is shown in Figure 4-1.

Abbreviations with final periods and ellipses are classified the same way. If the orthographic heuristic decides in favor of a sentence boundary for the following token or the following token is a frequent sentence starter then they are considered to be sentence boundaries. Abbreviations with one letter are not classified this way and are considered to be initials.

Initials are classified in another way than abbreviations because there is a high probability of them being ordinary words. One example is the Swedish preposition “i” (“in”). Single letters can also be used in formulas and enumerations. This ambiguity of single letters leads to that they are hard to detect by the learning phase, therefore all single letters with a final period are considered to be initials. Collocation information from the learning phase is used because initials are often part of a complex name. A possible initial is not considered to be a sentence boundary if it forms a collocation with the following token, which is not a frequent sentence starter. If the orthographic heuristic decides against a sentence boundary for the following token the initial is also not considered to be a sentence boundary. The last thing that is considered is if the orthographic heuristic returns undecided and the possible initial always occurs with an uppercase first letter. The following token is then considered to be a proper name and the initial is therefore not considered to be a sentence boundary.

The last classification is used for languages where ordinal numbers written in digits have a final period, described in 2.2.2. Like for initials, if the number forms a collocation with the following token and the following token is not a frequent sentence starter, then the number is not considered to be a sentence boundary.

```
function DecideOrthographic(TOKEN) {
    if TOKEN has uppercase first letter and
    TOKEN occurs with lowercase first letter at least once and
    TOKEN never occurs with uppercase first letter {
        return sentenceBoundary
    }
    else if TOKEN has lowercase first letter and (TOKEN occurs with uppercase first letter at least once or
    TOKEN never occurs with lowercase first letter after a sure sentence boundary
    ) { return noSentenceBoundary
    }
    else { return undecided
    }
}
```

Figure 4-1: Pseudo code of the orthographic heuristic
sentence boundary. The same classification is made if the orthographic heuristic decides against a sentence boundary for the following token.

Every token with a final period that was not classified using the previous classifications is considered to be a sentence boundary.

4.2 Sentence Alignment Algorithm
The chosen algorithm for sentence alignment by Gale and Church (1991) will here be explained in more detail. Costs are calculated for all combinations of the beads: 0-1, 1-0, 1-1, 1-2, 2-1 and 2-2. All these beads sum up to a total cost that is minimized using a dynamic programming search.

4.2.1 The Cost Measure
The cost is an estimate of Equation 15, where \( \delta \) depends on \( l_1 \) and \( l_2 \), the total lengths in characters of the sentences under consideration. The log is used to get multiplication of probabilities when adding costs.

Equation 15: Wanted cost

\[
- \log P(\text{match}|\delta)
\]

A model is assumed where every character in one language, \( L_1 \), gives rise to a random number of characters in another language, \( L_2 \). The random variables are independent and are identically distributed according to a normal distribution with variance, \( s^2 \), and mean, \( c \). The variable \( \delta \) is defined according to Equation 16 with a standard normal distribution (when the two texts under consideration actually are translations of one another). The variance \( s^2 \) and the mean \( c \) are the variance and expected value respectively of the number of characters in \( L_2 \) per character in \( L_1 \).

Equation 16: Model

\[
\delta = \frac{l_2 - l_1 c}{\sqrt{l_1} s^2}
\]

Gale and Church (1991) determined the parameters \( c \) and \( s^2 \) empirically, using data from a trilingual corpus (English-German, English-French) of 15 economic reports issued by the Union Bank of Switzerland. The parameter \( c \) was simply determined by taking the number of characters in paragraphs of one language divided by the number of character in paragraphs in the other language. For English-German and English-French \( c \) was approximately determined to 1.1 and 1.06 respectively. The value of \( c \) was set to 1.0 for simplicity because the performance does not seem to be that sensitive to this parameter.

The model assumes that \( s^2 \) is proportional to the length, \( l_1 \). Therefore \( s^2 \) can be determined by taking the slope of robust regression of \( (l_1 - l_2)^2 \) and \( l_1 \). By doing this finding the average value of \( s^2 \). For English-German and English-French \( s^2 \) was approximately determined to 7.3 and 5.6 respectively. Like \( c \), this parameter is not that important and therefore set to 6.8.

Equation 15 cannot be calculated directly and is therefore estimated by the help of Bayes’ Theorem to Equation 17. The constant \( k \) can be ignored because it will be the same for all proposed matches.
Equation 17: Estimation using Bayes’ Theorem

\[-\log kP(\delta | \text{match})P(\text{match})\]

Another estimate used is shown in Equation 18, where \(P(|\delta|)\) is the probability that a random variable, \(z\), with a standard normal distribution, has magnitude of at least \(|\delta|\), shown in Equation 19.

Equation 18: Estimation of conditional probability

\[P(\delta | \text{match}) = 2(1 - P(|\delta|))\]

Equation 19: Cumulative distribution

\[P(|\delta|) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\delta} e^{-\frac{z^2}{2}} dz\]

The probability, \(P(\text{match})\), is the probability of a specific bead type. These probabilities were determined from the trilingual corpus.

A distance function, \(d\), is defined as the cost of matching sentences \(x_1\) and \(x_2\) in one language with sentences \(y_1\) and \(y_2\) in another language.

1. Let \(d(0, y_1, 0,0)\) be the cost of matching \(y_1\) with nothing (0-1),
2. \(d(x_1, 0, 0,0)\) be the cost of matching \(x_1\) with nothing (1-0),
3. \(d(x_1, y_1, 0,0)\) be the cost of matching \(x_1\) with \(y_1\) (1-1),
4. \(d(x_1, y_1, 0, y_2)\) be the cost of matching \(x_1\) with \(y_1\) and \(y_2\) (1-2),
5. \(d(x_1, y_1, x_2, 0)\) be the cost of matching \(x_1\) and \(x_2\) with \(y_1\) (2-1),
6. \(d(x_1, y_1, x_2, y_2)\) be the cost of matching \(x_1\) and \(x_2\) with \(y_1\) and \(y_2\) (2-2).

4.2.2 The Dynamic Programming Search

Let \(s_i, i = 1, ..., I\), be the sentences in source language and \(t_j, j = 1, ..., J\), be the translations of those sentences in target language. Let \(D(i, j)\) be the minimum distance/cost of matching \(s_i, ..., s_i\) with \(t_1, ..., t_j\) using function \(d\) from previous section. \(D(i, j)\) is calculated recursively according to Equation 20.

Equation 20: Total distance

\[D(0, 0) = 0,\]

\[D(i, j) = \min \begin{cases} D(i, j - 1) + d(0, t_j, 0,0) \\ D(i - 1, j) + d(s_i, 0, 0,0) \\ D(i - 1, j - 1) + d(s_i, t_j, 0,0) \\ D(i - 1, j - 2) + d(s_i, t_j, 0, t_{j-1}) \\ D(i - 2, j - 1) + d(s_i, t_j, s_{i-1}, 0) \\ D(i - 2, j - 2) + d(s_i, t_j, s_{i-1}, t_{j-1}) \end{cases}\]
5 Implementation and Results

There exist some implementations of the algorithms chosen, but by using them, integration to a whole application could be a problem. Making small modifications to get better result could also be a problem. These problems could surely be solved, but there is a limited amount of time for this thesis and nasty surprises are unwanted. By doing a new implementation instead of using existing ones, makes it possible to choose features from many different algorithms and combine them. It is also easier to analyze the result and answer questions like: why it failed or why it succeeded. Therefore no external libraries or code was used to implement solutions to the problem.

Two different implementations were made in an iterative way. The results from the first implementation were used to modify the first implementation into a better second implementation. Only the sentence boundary detection and the sentence alignment were changed between the two iterations.

The implementations were written in C# to make it easy to integrate the implementations into the AllAddin application, because AllAddin also is written in C#. The implementations are console applications and are using .NET Framework 4.0. The LINQ to XML was used to access and modify elements in the XML.

The different parts of the implementations are assembled in the same way with the same input and output. Figure 5-1 shows a data flow diagram of the implementations. $D_{s1}$, $D_{t1}$, $D_{s2}$ and $D_{t2}$ are the same as in 1.2. Paragraphs are extracted from the XML-documents from the database. These are divided into tokens that the sentence boundary detection implementation needs. The sentence boundary detection step also needs the original XML with tagged paragraphs. All the sentences that are detected are also tagged in the XML. The sentence aligner creates an alignment using the sentences. The last step translates what is possible to translate using the alignment and creates a new XML-document, $D_{s2/t2}$.

![Figure 5-1: Data flow diagram](image-url)
The two implementations are evaluated in three steps: sentence boundary detection, sentence alignment and correspondence. The last step evaluates the correspondence between sentences in \( D_s \) and \( D_t \) as a result of the first two steps. The results of the first step also affect the results of the second step, which leads to a dependence between the results.

5.1 Data
The XML-documents were extracted the same way as in chapter 3 with metadata added and saved on disk for faster and easier access.

5.2 Paragraphs
The sentence boundary algorithm described in 4.1 uses normal text for input, but XML is not normal text. It makes no sense to use XML directly for detection of sentence boundaries with this algorithm. Paragraphs were therefore extracted from the XML-documents. This was done by first extracting a list of elements that can have both parsed character data (PCDATA) and child elements, according to the Document Type Definition (DTD) described in 2.1. What this means is that the elements can have both normal text and child elements. Another list was also created that contained a list of all footnote elements. The footnote elements are special because they are used in paragraphs but the content is a paragraph itself. The contents of the elements in the first list are treated as paragraphs but without the content of any footnote element because it is treated as another paragraph. The contents of elements that are not in the lists that have PCDATA are also considered to be paragraphs.

The paragraph extraction step has two different outputs: the content of the paragraphs and the XML from input with paragraph-tags \(<\text{paragraph}>\) and \(</\text{paragraph}>\) surrounding all the paragraphs. The sentence boundary detection step uses the XML and the tokenizer uses the paragraphs as input.

5.3 Tokenization
The tokenization step splits paragraphs into tokens that can be used as input to the sentence boundary detection algorithm. Each paragraph is first split by possible XML-tags. The reason is that tags should not be used as tokens and this can also help the classification of a token. Figure 5-2 shows an example where a paragraph is split by a start-tag and an end-tag.

|This sentence has \textit{four} XML-tags. This sentence has none.| \rightarrow |This sentence has |, |four|, | XML-tags|, |. This sentence has none.|

Figure 5-2: Split by XML-tags

After the first split, another split is performed, but this time by the space character. This results in the actual tokens. For Figure 5-2 this will result in the tokens in Figure 5-3. The positions in the paragraphs of all periods, question marks and exclamation marks that end a token are saved in each such token. This creates a relation between the tokens and the paragraphs that is needed for the sentence boundary detection step to be able to specify the positions of sentence boundaries in the paragraphs.

|This|, |sentence|, |has|, |four|, |XML-tags|, |.|, |This|, |sentence|, |has|, |none.|

Figure 5-3: Split by space character
5.4 Translation Export
The translation export is the last step in the implementations. It creates a document, $D_{12/12}$, that is a modification of $D_{12}$, where all sentences that can be translated to the target language using the alignment are. This step was implemented to get viewable results of the whole implementation.

This step keeps track of which and how sentences can be translated, which makes it easy to, for example, modify this step so it tags all sentences that cannot be translated like how it is done in AllAddin described in 2.1 for modules.

5.5 First Implementation
The first implementation is mainly based on the algorithms described in 4.1 and 4.2. Some modifications were made to better fit in a complete solution for the problem of the thesis.

5.5.1 Sentence Boundary Detection
This sentence boundary detection implementation is based on the algorithm by Kiss and Strunk (2006), described in 4.1. The algorithm has two phases: one training phase and one decision phase.

Before the tokens can be used for training they need to be classified. There are four classes in the training phase: ellipsis, initial, ordinal and undecided. The classes ellipsis, initial and ordinal are the same as in the algorithm. The ordinal class is only used for languages where ordinal numbers written in digits have a final period, described in 2.2.2. The string for tokens classified as ordinals are replaced with “#ordinal#” so all ordinals will be seen as equal tokens. The last class, undecided is used for all other tokens.

Statistics of tokens is used in the training, for example, how many times a token ends with a period. A list of tokens with statistics is generated. Tokens with the same string except case and except existence of a final end-of-sentence punctuation mark (period, question mark and exclamation mark) are considered to be equal. Another list of pair of tokens is also generated with initials and ordinals and their following tokens, with statistics. This last list is used in the collocation step of the implementation.

The actual training can begin when the tokens are classified and the statistics are collected. The implementation of the training follows the algorithm. Export and import of abbreviations, collocations and frequent sentence starters to an XML-format are implemented. This gives the possibility to edit the different lists. The data flow diagram in Figure 5-4 shows what steps are needed to train, export and import.
In the implementation of the decision phase, tokens are classified into eight classes instead of four as in the training. The previous classes remain but four new classes are introduced: abbreviation at the end, abbreviation internally, abbreviation and sentence breaker. In the original algorithm there is only one type of abbreviations. With the possibility to edit the abbreviation list new features can be added to improve the performance. One feature was added so the position of an abbreviation could be specified when editing the abbreviation list. This feature is based on the assumption that some abbreviations always occur sentence internally and some at the end of a sentence. Tokens that are in the abbreviation list are classified into one of the three abbreviation classes. Periods are only considered in the original algorithm as sentence boundaries, but there are more characters that can end a sentence. Tokens that end with a question mark or exclamation mark are classified as sentence breakers. These two characters are less ambiguous as sentence boundaries than the period and are therefore classified directly. With the split by XML-tags described in 5.3 a lonely period can appear as a token. This was the case in Figure 5-3. These periods are also classified as sentence breakers.

The orthographic heuristic in Figure 4-1 needs some statistics about the token. As with the case of training, a list of tokens with statistics is generated.

In the algorithm, it is decided if abbreviations, ellipsis, initials and ordinals are sentence boundaries or not. The rest of the tokens with a final period are classified as sentence boundaries. The implementation makes the same decisions for the four common classes. Tokens classified as sentence breaker, abbreviation at the end and undecided with a final period are decided to be sentence boundaries. The rest of the tokens are not seen as sentence boundaries.

The decision phase has both the tokens from the tokenizer as input and XML-document with paragraphs tagged. The position information in the tokens is used to split up each paragraph into sentences that are surrounded with sentence-tags (\texttt{<sentence> and </sentence>}) as the paragraphs are. This results in an XML-document with paragraph-elements with sentence-elements.

5.5.2 Sentence Alignment
The sentence alignment implementation is based on the algorithm by Gale and Church (1991), described in 4.2.
This step has two XML-documents as input from the sentence boundary detection step: one in the source language; and one in the target language. Each paragraph-element in the source XML is paired with one paragraph-element in the target XML. The sentences of each pair are used in the matching.

The cost of matching sentences is calculated according to the algorithm with one exception. The integral in Equation 19 cannot be calculated directly so an approximation is used to calculate the probability. The approximation is described by Abramowitz and Stegun (1964; p. 932, equation 26.2.17). 0-1, 1-0, 1-1, 1-2, 2-1 and 2-2 are the bead types used by the original algorithm. The analysis in 3.1.2 shows that these are not enough because a 1-3-bead was found. To make a more dynamic implementation the distance function, \( d \) was replaced by a new function, \( d_2 \). This new function is defined as the cost of matching the total length of sentences in source language with the total length of sentences in target language using the corresponding bead type.

The more dynamic approach of \( d \) leads to a new definition of the minimum cost/distance \( D \) called \( D_2 \). Let \( s_i, i = 1, ..., I \), be the sentences in source language and \( t_j, i = 1, ..., J \), be the translations of those sentences in target language. Let \( D_2(i, j) \) be the minimum distance/cost of matching \( s_1, ..., s_i \) with \( t_1, ..., t_j \) using function, \( d_2 \) for all valid bead types. The calculation of \( D_2(i, j) \) is done in an iterative way. However the goal is to get an alignment not a cost. A new variable, \( B(i, j) \) is therefore introduced, which is the last bead in a path of beads resulting in \( D_2(i, j) \). Figure 5-5 shows the iterative algorithm for calculating both \( D_2(i, j) \) and \( B(i, j) \).

\[
\begin{align*}
B(0,0) & := (0,0) \\
D_2(0,0) & := 0 \\
\text{for sentences} & := 1 \text{ to } i + j \{ \\
& \text{foreach } ii \text{ and } jj, \text{ where } ii, jj > 0 \text{ and } ii + jj = \text{sentences and } ii \leq i \text{ and } jj \leq j \{ \\
& \quad D_2(ii,jj) := \text{INF} \\
& \quad \text{foreach valid bead, } (n,m), \text{ where } ii - n \geq 0 \text{ and } jj - m \geq 0 \{ \\
& \qquad \text{distance} := d_2(\text{length}(s_{ii-n},...s_{ii}), \text{length}(t_{jj-m},...t_{jj})), (n,m)) \\
& \qquad \text{totalDistance} := \text{distance} + D_2(ii-n, jj-m) \\
& \qquad \text{if } D_2(ii,jj) > \text{totalDistance} \{ \\
& \qquad \qquad D_2(ii,jj) := \text{totalDistance} \\
& \qquad \qquad B(ii,jj) := (n,m) \\
& \qquad \} \\
& \} \\
\}
\end{align*}
\]

Figure 5-5: Calculation of \( D_2(i,j) \) and \( B(i,j) \)

The beads that form the path from \( B(i, j) \) to \( B(0,0) \) are the beads that are used in the alignment. Figure 5-6 shows an example of the path of beads from \( (i, j) \) to \( (0,0) \), where \( i = 4 \) and \( j = 5 \). Every row corresponds to an iteration in the outer for-loop in Figure 5-5.
The sentences of a bead in one language are concatenated and then used to create a translation memory between the source and the target language. 0-1 and 1-0 beads are ignored. A text cannot map to more than one text because that would lead to ambiguity. All occurrences of texts that map to two or more different texts are therefore removed from the translation memory.

5.6 Results of First Implementation

The results of the first implementation were taken from the sentence boundary detection, the sentence alignment and the translation export. The first version of 100 modules with English as source language was run through the sentence boundary detection implementation and the sentence alignment implementation. To be able to do the translation export, the second version in the source language was also used. The modules were not the same modules in three different steps of evaluation. Only XML-documents in: English, German and Swedish were used.

For complete results see Appendix A.

5.6.1 Sentence Boundary Detection

All service information modules were used for training. The sentence boundary detection implementation was compared to a baseline to get more interesting results. A sentence boundary is detected by the baseline if a period is found with a following token with a first uppercase letter. The same precision, recall and F-measure are used as in 2.2.4 and the results are displayed in Table 5-7.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Language</th>
<th>Sentences</th>
<th>Boundaries</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Implementation</td>
<td>English</td>
<td>6264</td>
<td>237</td>
<td>93.03 %</td>
<td>95.78 %</td>
<td>94.39 %</td>
</tr>
<tr>
<td></td>
<td>German</td>
<td>6266</td>
<td>239</td>
<td>93.88 %</td>
<td>96.23 %</td>
<td>95.04 %</td>
</tr>
<tr>
<td></td>
<td>Swedish</td>
<td>6269</td>
<td>242</td>
<td>94.31 %</td>
<td>95.87 %</td>
<td>95.08 %</td>
</tr>
<tr>
<td>Baseline</td>
<td>English</td>
<td>6264</td>
<td>237</td>
<td>95.78 %</td>
<td>95.78 %</td>
<td>95.78 %</td>
</tr>
<tr>
<td></td>
<td>German</td>
<td>6266</td>
<td>239</td>
<td>87.45 %</td>
<td>96.23 %</td>
<td>91.63 %</td>
</tr>
<tr>
<td></td>
<td>Swedish</td>
<td>6269</td>
<td>242</td>
<td>95.08 %</td>
<td>95.87 %</td>
<td>95.47 %</td>
</tr>
</tbody>
</table>

Table 5-7: Results of first sentence boundary detection algorithm

The baseline is actually better with English and Swedish than the first implementation. Some common abbreviations were not found by the implementation. One example is the abbreviation “max” that exists in all three languages and is very common in the texts. The reason why these abbreviations were not found is that they were often written without a final period.
penalty factor in Equation 9 then gets very small which leads to a small final value for the abbreviation. If an abbreviation is not found it is treated as a sentence boundary, but it is more likely that it is not. The baseline is however much worse with the German language, this because all nouns in German are often written with a first upper case letter.

It is notable that both the algorithms did sometimes add a sentence boundary between brackets or in quotations. These are structures that should not be broken up.

5.6.2 Sentence Alignment
The valid bead types used in the implementation were: 0-1, 1-0, 1-1, 1-2, 2-1, 2-2, 1-3 and 3-1. The probabilities used for the different beads were taken from (Fattah, et al., 2006a). These values correspond pretty well to the values in the analysis, shown in Table 3-4. 107 modules were used, but seven modules were skipped due to unequal element structure of the XML-documents involved. The same precision, recall and F-measure are used as in 2.2.6 and the results are displayed in Table 5-8. Most of the paragraphs only have one sentence and that makes the problem trivial. In Table 5-9 only results from sentences in paragraphs with at least two sentences are displayed.

<table>
<thead>
<tr>
<th>Alignment</th>
<th>Sentences, Ds₁</th>
<th>Sentences, Ds₂</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>English-German</td>
<td>6350</td>
<td>6353</td>
<td>99.81 %</td>
<td>99.91 %</td>
<td>99.85 %</td>
</tr>
<tr>
<td>English-Swedish</td>
<td>6335</td>
<td>6343</td>
<td>99.90 %</td>
<td>99.95 %</td>
<td>99.92 %</td>
</tr>
</tbody>
</table>

Table 5-8: Results of first sentence alignment implementation

<table>
<thead>
<tr>
<th>Alignment</th>
<th>Sentences, Ds₁</th>
<th>Sentences, Ds₂</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>English-German</td>
<td>453</td>
<td>456</td>
<td>97.27 %</td>
<td>98.61 %</td>
<td>97.94 %</td>
</tr>
<tr>
<td>English-Swedish</td>
<td>444</td>
<td>452</td>
<td>98.62 %</td>
<td>99.31 %</td>
<td>98.96 %</td>
</tr>
</tbody>
</table>

Table 5-9: Results of first sentence alignment implementation (multi-sentence paragraphs)

Many of the mistakes of the sentence boundary detection implementation are repaired by the sentence alignment implementation by using for example the 1-2 and 2-1 bead types. Therefore these results are much better than for the sentence boundary detection implementation.

5.6.3 Correspondence
Like in the results of the sentence alignment, 107 modules were used and seven were skipped. Sentences are counted in Ds₂. Precision is here defined as the number of correctly translated sentences divided by the number of sentences translated. Recall is the number of correctly translated sentences divided by number of sentences. F-measure is the same as before shown in Equation 1. The recall is highly dependent on how many sentences that are the same in Ds₁ and Ds₂. Therefore another measure is also used. It is the number of sentences that are the same in Ds₁ and Ds₂ divided by the total number of sentences in Ds₂. The result is shown in Table 5-10.

<table>
<thead>
<tr>
<th>Correspondence</th>
<th>Sentences</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
<th>Same sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>English-German</td>
<td>5283</td>
<td>99.93 %</td>
<td>79.69 %</td>
<td>88.67 %</td>
<td>79.95 %</td>
</tr>
<tr>
<td>English-Swedish</td>
<td>5283</td>
<td>100.00 %</td>
<td>79.94 %</td>
<td>88.85 %</td>
<td>79.95 %</td>
</tr>
</tbody>
</table>

Table 5-10: Results of correspondence in first implementation

Of the English-German correspondences three were incorrect, therefore the precision of 99.93 %. One of the incorrect correspondence was due to a sentence in English was split up into four parts by the sentence boundary detection implementation. That sentence had one corresponding German sentence that was correctly identified. So for a correct alignment a 4-1 bead would have been
necessary, instead the sentence aligner used a 3-1 and 1-0 bead. The 1-0 bead was ignored by the aligner when creating the translation memory. This resulted in an incorrect correspondence. Another error occurred because two sentences had a correspondence to one German sentence, but the German sentence was split into two parts. This resulted in two 1-1 beads. One of the 1-1 beads was changed in $D_{12}$, which resulted in an incorrect correspondence. The last error occurred because one sentence in English and one sentence in German were split into two parts. A correct bead would be a 2-2 bead, but two 1-1 beads were used instead. One of the English parts was changed in $D_{12}$ and the 1-1 beads were not correct, which resulted in an incorrect correspondence.

5.7 Second Implementation

The second implementation is a modification of the first implementation. New features were added to fix some of the problems with the first implementation.

5.7.1 Sentence Boundary Detection

In the second implementation a new class is introduced in the decision phase, the non-sentence breaker class. All tokens with a final period in between brackets or quotation marks are classified as non-sentence breakers. The tokens with a final period and that represent the end of brackets or quotations are marked as sentence breakers.

The first implementation had a problem with common abbreviations that were often written without a final period and the main reason was the penalty factor in Equation 9. This factor cannot simply be removed because it fills a good purpose. The problem with it is that it uses the information of how many times a token does not have a final period $C(w|\neg \cdot)$, but not the information of how many times the token actually has a final period $C(w|\cdot)$. This causes problems especially if the abbreviation is common. The penalty factor was therefore redefined according to Equation 21.

Equation 21: New penalty factor

$$F_{\text{penalty}(w)} = \frac{1}{\text{length}(w)C(w|\cdot)}$$

5.7.2 Sentence Alignment

Two incorrect correspondences were in the result of the first implementation due to 1-0 beads that were ignored. Correct correspondences are more important than many correspondences. 0-1 beads and 1-0 beads are very rare and can cause a lot of problems. To be on the safe side all paragraphs that contain a 0-1 or a 1-0 bead are therefore not aligned in the second implementation.

5.8 Results of Second Implementation

As the results of the first implementation, the results of the second implementation were also taken from the sentence boundary detection, the sentence alignment and the translation export. 104 modules were used and four skipped due to difference in structure of the XML-documents. The first version of 100 random modules with Swedish as source language was run through the sentence boundary detection implementation and the sentence alignment implementation. To get the final translation the second version in the source language was also used. The modules that were used for the first implementation can be seen as training data to the second implementation because information from the result is used in the second implementation. That is why modules with Swedish
as source language were used in this case. The same modules were used for all three steps in English, German and Swedish.

For complete results see Appendix A. For an example of $D_{s1}$, $D_{t1}$ and $D_{s2}$ from a module with the corresponding result $D_{t2/s2}$ using the second implementation see Appendix B and Appendix C.

5.8.1 Sentence Boundary Detection
All service information modules were used for training. The most common abbreviations that were missed in the results of the first implementation were added to the abbreviation list. Many of them were actually already found by the new sentence boundary detection algorithm. There existed some tokens in the abbreviation list that were not abbreviations, these were removed. Some collocations were also added to the German collocation list.

The sentence boundary detection implementation is compared to the baseline described in 5.6.1. The result is displayed in Table 5-11 with the same measures as in 5.6.1.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Language</th>
<th>Sentences</th>
<th>Boundaries</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Second Implementation</td>
<td>English</td>
<td>5201</td>
<td>263</td>
<td>98.85 %</td>
<td>98.48 %</td>
<td>98.67 %</td>
</tr>
<tr>
<td></td>
<td>German</td>
<td>5208</td>
<td>270</td>
<td>96.72 %</td>
<td>98.52 %</td>
<td>97.61 %</td>
</tr>
<tr>
<td></td>
<td>Swedish</td>
<td>5203</td>
<td>265</td>
<td>99.24 %</td>
<td>98.49 %</td>
<td>98.86 %</td>
</tr>
<tr>
<td>Baseline</td>
<td>English</td>
<td>5201</td>
<td>263</td>
<td>98.47 %</td>
<td>98.10 %</td>
<td>98.29 %</td>
</tr>
<tr>
<td></td>
<td>German</td>
<td>5208</td>
<td>270</td>
<td>79.94 %</td>
<td>97.41 %</td>
<td>87.81 %</td>
</tr>
<tr>
<td></td>
<td>Swedish</td>
<td>5203</td>
<td>265</td>
<td>98.46 %</td>
<td>96.60 %</td>
<td>97.52 %</td>
</tr>
</tbody>
</table>

Table 5-11: Results of second sentence boundary detection implementation

The second implementation is better than the baseline algorithm. A better abbreviation list and the feature with brackets and quotations made a difference. The better German collocation list also helped the results and was one of the reasons why the very large difference in the results.

5.8.2 Sentence Alignment
The same beads with the same probabilities as in the evaluation of the first implementation were used. No errors were found in the two alignments: Swedish-English and Swedish-German.

5.8.3 Correspondence
The result is displayed in Table 5-12 with the same measures as in 5.6.3.

<table>
<thead>
<tr>
<th>Correspondence</th>
<th>Sentences</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
<th>Same sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Swedish-English</td>
<td>5760</td>
<td>100.00 %</td>
<td>80.17 %</td>
<td>89.00 %</td>
<td>80.43 %</td>
</tr>
<tr>
<td>Swedish-German</td>
<td>5760</td>
<td>100.00 %</td>
<td>79.90 %</td>
<td>88.82 %</td>
<td>80.43 %</td>
</tr>
</tbody>
</table>

Table 5-12: Results of correspondence in second implementation

There is a larger difference between recall and the proportion of same sentences compared to the first results. The reason for this is a combination of better implementation and more 2-1 beads which increases the probability of a larger difference.
6 Discussion
The result from the previous section and the generality of the solution will be discussed here.

6.1 Limitations
Evaluating the two implementations is a time consuming process. Only 100 XML-documents were used for each language, but the modifications made to the first implementations seem to have made a large difference. It is still hard to tell exactly how much difference the different modifications made to the result, because they were not tested individually. It is also unfair to compare the different implementations directly because they were tested on different data and the lists in the sentence boundary step were manually changed in the second evaluation, leading to an unsupervised approach versus a supervised approach.

English, German and Swedish are pretty closely related languages, but so are also many other languages. I see no reason why the results would differ with documents in other related languages. This is confirmed by the knowledge that the two algorithms involved have been used successfully before on many other related languages. The limitation of the results is still there with other languages that are not that closely related.

6.2 First Implementation
The results of the first implementation, even with the limitations of the evaluation, show that it is possible to reduce translation by using previous versions of a module. A recall of approximately 80 % leads to an 80 % reduction in translation if it was used in a real application. An unsupervised approach was used in the evaluation, meaning that no knowledge what so ever of the languages involved was required and still only three errors were made. This means that the implementation could probably be used directly without any modifications to other languages, especially if they are somewhat similar to the ones used in this thesis.

6.3 Second Implementation
The results of the second implementation were not significantly better than of the first implementation, but there was a quite large difference in the other two steps. If it is possible to change the lists used in the sentence boundary detection step, then it should be done. Not all abbreviations and collocations were found and some were not correct.

6.4 Further Development
The error rates of the implementations are very low, but it is hard to say anything about how they would perform on noisier texts with less 1-1 beads and texts with more ambiguous periods. Still the algorithms that the implementations are based on are not state of the art, so it is possible to perform well even on noisier texts with better algorithms. The precision on the tested texts were very good, so it is not possible to do so much better on that part for the tested texts.

The precision cannot get so much higher but there is still room on the other hand to get better recall. The same sentence results and the recall were pretty close, which means that nearly all sentences that were not changed were used in the new version. By using also changed sentences it is possible to get better recall. Instead of using the exact same sentences a fuzzy match can be used described in
2.3. The analysis of data in 3.2 showed that there exist some changes on languages invariant words. By also using sentences with only changes of the invariant words is one way to get better recall.

Some modules were skipped due to unequal element structure, with the risk of different information in source and target language. It possible to see this at two different perspectives, where one is to say that unequal structure is a problem made by the writers and it is not a problem of this solution. The other way is to see it as one of the problems that can be fixed by altering this solution. For example the two algorithms discussed in 2.2.5 by Sánchez-Villamil et al. (2006) and Zhu et al. (2011) do not need an equal structure to align sentences.

6.5 Other Types of Texts
The texts used are quite specific, service information modules from Volvo Construction Equipment in the AllAddin application. In the survey made by Lagoudaki (2006) translation professionals were asked what format the information they translated were stored in. 96 % said that normal text formats like for example Word. The respondents had also the possibility to choose more than one format and 26 % answered XML and HTML. The implementations used in this thesis can easily be modified to fit other XML-tags, the HTML format or other similar structural formats. If texts are stored without any tags and just normal texts it is also possible to use this solution by modifying the paragraph extractor and removing the splitting by tags.
7 Conclusion

I have presented a solution to reducing translation of changes in structural information. It uses a sentence boundary detection algorithm in combination with a sentence alignment algorithm. With the tested texts it was possible to reduce text that requires re-translation by about 80% with a low error rate. The reduced translation was proportional to the change, but it is still possible to reduce translation even more by for example using fuzzy matching. Similar information that is used in this solution can and is used by the translators to achieve higher productivity, but this thesis shows that it is possible to reduce translation even before the texts reaches the translators.

Two complete implementations were made. The first was tested in an unsupervised manner with no manual interference. The first sentence boundary detection implementation was compared to a simple baseline algorithm that performed better on English and Swedish texts. It showed that the complex algorithm used could be beaten by a much simpler algorithm. The final result was good and proved that it is possible to use an unsupervised approach. It is still preferable to manually edit the lists that are created by the sentence boundary detection algorithm, which helped the second implementation and resulted in better performance.

The information used was stored in the structural format, XML, but this solution also work for normal texts, by a small modification. It is also possible to use this solution on other structural formats that use similar structure.
References


### Appendix A  Results of Implementations

Below are the results of the implementations. The first four tables present the results from the first implementation and the other four present the results of the second implementation. C stands for correct and I for identified. All columns with zero in all cells are not presented in the tables.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Language</th>
<th>Sentences</th>
<th>C-Boundaries</th>
<th>I-Boundaries</th>
<th>Boundaries</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>First</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>implementation</td>
<td>English</td>
<td>6264</td>
<td>227</td>
<td>244</td>
<td>237</td>
<td>93.03</td>
<td>95.78</td>
<td>94.39</td>
</tr>
<tr>
<td></td>
<td>German</td>
<td>6266</td>
<td>230</td>
<td>245</td>
<td>239</td>
<td>93.88</td>
<td>96.23</td>
<td>95.04</td>
</tr>
<tr>
<td></td>
<td>Swedish</td>
<td>6269</td>
<td>232</td>
<td>246</td>
<td>242</td>
<td>94.31</td>
<td>95.87</td>
<td>95.08</td>
</tr>
<tr>
<td>Baseline</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>English</td>
<td>6264</td>
<td>227</td>
<td>237</td>
<td>237</td>
<td>95.78</td>
<td>95.78</td>
<td>95.78</td>
</tr>
<tr>
<td></td>
<td>German</td>
<td>6266</td>
<td>230</td>
<td>263</td>
<td>239</td>
<td>87.45</td>
<td>96.23</td>
<td>91.63</td>
</tr>
<tr>
<td></td>
<td>Swedish</td>
<td>6269</td>
<td>232</td>
<td>244</td>
<td>242</td>
<td>95.08</td>
<td>95.87</td>
<td>95.47</td>
</tr>
</tbody>
</table>

| Table 1: Results of first sentence boundary detection implementation |
|-----------------|----------|-----------|--------------|--------------|------------|-----------|--------|-----------|
| Selection       | Alignment | Sentences, $D_{c}$ | Sentences, $D_{i}$ | C-Beads | I-Beads | Beads | Precision | Recall | F-measure |
| All             | English-German | 6350  | 6353 | 6324 | 6336 | 6330 | 99.81 | 99.91 | 99.85 |
|                 | English-Swedish | 6335 | 6343 | 6320 | 6326 | 6323 | 99.90 | 99.95 | 99.92 |
| Multi-sentence paragraphs | English-German | 453 | 456 | 427 | 439 | 433 | 97.27 | 98.61 | 97.94 |
|                 | English-Swedish | 444 | 452 | 429 | 435 | 432 | 98.62 | 99.31 | 98.96 |

| Table 2: Results of first sentence alignment implementation |
|-----------------|----------|-----------|--------------|--------------|------------|-----------|--------|-----------|
| Selection       | Alignment | I-1-0 | C1-1 | I-1-1 | C1-2 | I-1-2 | C1-3 | I-3-1 | C1-4 | I-4-1 |
| All             | English-German | 2 | 6296 | 6304 | 6296 | 19 | 19 | 8 | 8 | 4 | 0 | 0 | 0 | 1 | 3 | 1 | 4 |
|                 | English-Swedish | 1 | 6299 | 6303 | 6299 | 12 | 12 | 5 | 5 | 5 | 2 | 3 | 3 | 1 | 2 | 1 | 2 |
| Multi-sentence paragraphs | English-German | 2 | 399 | 407 | 399 | 19 | 19 | 8 | 8 | 4 | 0 | 0 | 0 | 1 | 3 | 1 | 4 |
|                 | English-Swedish | 1 | 408 | 412 | 408 | 12 | 12 | 5 | 5 | 5 | 2 | 3 | 3 | 1 | 2 | 1 | 2 |

| Table 3: Beads of the first sentence alignment implementation |
|-----------------|----------|-----------|--------------|--------------|------------|-----------|--------|-----------|
| Correspondence  | C-Sentences | I-Sentences | Same sentences | Sentences | Precision | Recall | F-measure | Same sentences %|
| English-German  | 4210 | 4213 | 4224 | 5283 | 99.93 | 79.69 | 88.67 | 79.95 |
| English-Swedish | 4223 | 4223 | 4224 | 5283 | 100.00 | 79.94 | 88.85 | 79.95 |

| Table 4: Results of second sentence alignment implementation |
|-----------------|----------|-----------|--------------|--------------|------------|-----------|--------|-----------|
| Algorithm       | Language | Sentences | C-Boundaries | I-Boundaries | Boundaries | Precision | Recall | F-measure |
| Second          |          |           |              |              |            |           |        |           |
| implementation  | English  | 5201      | 259          | 262          | 263        | 98.85     | 98.48  | 98.67     |
|                 | German   | 5208      | 266          | 275          | 270        | 98.73     | 98.52  | 97.61     |
|                 | Swedish  | 5203      | 261          | 263          | 265        | 99.24     | 98.49  | 98.86     |
| Baseline        |          |           |              |              |            |           |        |           |
|                 | English  | 5201      | 258          | 262          | 263        | 98.47     | 98.10  | 98.29     |
|                 | German   | 5208      | 263          | 329          | 270        | 79.94     | 97.41  | 87.81     |
|                 | Swedish  | 5203      | 256          | 260          | 265        | 98.46     | 96.60  | 97.52     |

| Table 5: Results of second sentence boundary detection implementation |
|-----------------|----------|-----------|--------------|--------------|------------|-----------|--------|-----------|
| Selection       | Alignment | Sentences, $D_{c}$ | Sentences, $D_{i}$ | C-Beads | I-Beads | Beads | Precision | Recall | F-measure |
| All             | Swedish-English | 5203 | 5201 | 5193 | 5193 | 5193 | 100.00 | 100.00 | 100.00 |
|                 | Swedish-German | 5203 | 5208 | 5179 | 5179 | 5179 | 100.00 | 100.00 | 100.00 |
| Multi-sentence paragraphs | Swedish-English | 477 | 475 | 467 | 467 | 467 | 100.00 | 100.00 | 100.00 |
|                 | Swedish-German | 491 | 496 | 467 | 467 | 467 | 100.00 | 100.00 | 100.00 |

| Table 6: Results of second sentence alignment implementation |
|-----------------|----------|-----------|--------------|--------------|------------|-----------|--------|-----------|
| Selection       | Alignment | C1-1 | C1-2 | C1-3 | I1-1 | I1-2 | I1-3 | C2-1 | C2-2 | C2-3 | I2-1 | I2-2 | I2-3 | C3-1 | C3-2 | C3-3 | I3-1 | I3-2 | I3-3 |
| All             | Swedish-English | 5175 | 5175 | 5175 | 8 | 8 | 10 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|                 | Swedish-German | 5129 | 5129 | 5129 | 24 | 24 | 24 | 23 | 23 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| Multi-sentence paragraphs | Swedish-English | 449 | 449 | 449 | 8 | 8 | 10 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|                 | Swedish-German | 417 | 417 | 417 | 24 | 24 | 24 | 23 | 23 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |

| Table 7: Beads of the second sentence alignment implementation |
|-----------------|----------|-----------|--------------|--------------|------------|-----------|--------|-----------|
| Correspondence  | C-Sentences | I-Sentences | Same sentences | Sentences | Precision | Recall | F-measure | Same sentences %|
| Swedish-English | 4618 | 4618 | 4633 | 5760 | 100.00 | 80.17 | 89.00 | 80.43 |
| Swedish-German | 4602 | 4602 | 4633 | 5760 | 100.00 | 79.90 | 88.82 | 80.43 |

| Table 8: Results of correspondences in second implementation |
Axlar fram och bak

Till 70% av kylfläkts hastigheten

Om omgivningstemperaturen är 25 °C (77 °F) eller lägre påkallas kylning av laddluft

36 °C (98 °F)

Kylfläktvartal

Omgivningstemperatur +25 °C (77 °F)

> 0,5 m/s (9.9 °F)

> 0,3 m/s (5.2 °F)

> 0,2 m/s (3.6 °F)

< 0,1 m/s (9.9 °F)

Påkallar kylning

När omgivningstemperaturen är 25 °C (77 °F) eller lägre påkallas kylning av laddluft

Kylfläktvartal

Omgivningstemperatur +25 °C (77 °F)

Motormodul

> 85 °C (185 °F)

> 80 °C (212 °F)

> 75 °C (167 °F)

Kylfläktvartal

När omgivningstemperaturen är högre än 25 °C (77 °F) påkallas kylning av laddluft

70 °C (158 °F)

75 °C (237 °F)

80 °C (392 °F)
The signals from the different sensors vary linearly and command the fan speed depending on the temperature. The signal that gives the highest value is the one that controls the requested cooling fan speed. If the engine speed is below 800 rpm, there is no charge-air cooling. Normally, the fan's basic speed is 400 rpm. There is a function that reduces the fan speed if the ambient temperature is below 5 °C and no function requests more than 25% fan speed increase.

<table>
<thead>
<tr>
<th>Ambient temperature</th>
<th>Charge-air cooling requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 °C (41 °F)</td>
<td>No charge-air cooling</td>
</tr>
<tr>
<td>7 °C (45 °F)</td>
<td>Low charge-air cooling</td>
</tr>
<tr>
<td>9 °C (48 °F)</td>
<td>Medium charge-air cooling</td>
</tr>
<tr>
<td>11 °C (52 °F)</td>
<td>High charge-air cooling</td>
</tr>
<tr>
<td>13 °C (55 °F)</td>
<td>Full charge-air cooling</td>
</tr>
</tbody>
</table>

If the engine's coolant temperature is 95 °C or lower, the hydraulic transmission cannot request cooling. If the engine's coolant temperature is 85 °C or lower, there is a request for cooling of charge-air.

<table>
<thead>
<tr>
<th>Ambient temperature</th>
<th>Charge-air cooling requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>95 °C (203 °F)</td>
<td>No charge-air cooling</td>
</tr>
<tr>
<td>90 °C (194 °F)</td>
<td>Low charge-air cooling</td>
</tr>
<tr>
<td>85 °C (185 °F)</td>
<td>Medium charge-air cooling</td>
</tr>
<tr>
<td>80 °C (176 °F)</td>
<td>High charge-air cooling</td>
</tr>
<tr>
<td>75 °C (167 °F)</td>
<td>Full charge-air cooling</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ambient temperature</th>
<th>Charge-air cooling requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>70 °C (158 °F)</td>
<td>Low charge-air cooling</td>
</tr>
<tr>
<td>65 °C (149 °F)</td>
<td>Medium charge-air cooling</td>
</tr>
<tr>
<td>60 °C (140 °F)</td>
<td>High charge-air cooling</td>
</tr>
<tr>
<td>55 °C (131 °F)</td>
<td>Full charge-air cooling</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ambient temperature</th>
<th>Charge-air cooling requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>50 °C (122 °F)</td>
<td>No charge-air cooling</td>
</tr>
<tr>
<td>45 °C (113 °F)</td>
<td>Low charge-air cooling</td>
</tr>
<tr>
<td>40 °C (104 °F)</td>
<td>Medium charge-air cooling</td>
</tr>
<tr>
<td>35 °C (95 °F)</td>
<td>High charge-air cooling</td>
</tr>
<tr>
<td>30 °C (86 °F)</td>
<td>Full charge-air cooling</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ambient temperature</th>
<th>Charge-air cooling requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>25 °C (77 °F)</td>
<td>No charge-air cooling</td>
</tr>
<tr>
<td>20 °C (68 °F)</td>
<td>Low charge-air cooling</td>
</tr>
<tr>
<td>15 °C (59 °F)</td>
<td>Medium charge-air cooling</td>
</tr>
<tr>
<td>10 °C (50 °F)</td>
<td>High charge-air cooling</td>
</tr>
<tr>
<td>5 °C (41 °F)</td>
<td>Full charge-air cooling</td>
</tr>
</tbody>
</table>
Om temperaturerna i motorns kylsystem är 85 °C (185 °F) eller
morerows

När laddluften når omgivningstemperatur + 25 °C (77 °F) eller
morerows

När laddluften når omgivningstemperatur + 15 °C (59 °F)

morerows

40 °C (104 °F)

morerows

När laddluften når omgivningstemperatur + 15 °C (59 °F)

morerows

När laddluften når omgivningstemperatur + 25 °C (77 °F)

morerows

Max kylning

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

Max kylning

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)

morerows

0 °C (32 °F)
Laddluften startats.

När laddluften når omgivningstemperatur + 15 °C (59 °F) eller några komponenter i motorns kylsystem är mindre än 5 °C (41 °F) och ingen funktion begär mer än 2 % fläktvarvstallning.

Det finns en del laddluft som behövs för att kyla motorn, och detta kan påverka prestandan.

Det finns delar av motorn som kräver att luften är under 85 °C (185 °F) för att fungera optimalt. Den lägsta tillåtna temperaturen i motorns kylsystem är 85 °C (185 °F) eller lägre.

### Tabell: Kylsystem och funktionsnivåer

| Temperatur | Funktion | Kylsystem
|------------|----------|-----------
| Max 85 °C (185 °F) | Max Cooling | Motor, kylsystem
| 70 °C (158 °F) | High Cooling | Motor, kylsystem
| 60 °C (140 °F) | Normal Cooling | Motor, kylsystem
| 50 °C (122 °F) | Low Cooling | Motor, kylsystem
| 40 °C (104 °F) | Min Cooling | Motor, kylsystem
| 30 °C (86 °F) | No Cooling | Motor, kylsystem
| 20 °C (68 °F) | No Cooling | Motor, kylsystem
| 10 °C (50 °F) | No Cooling | Motor, kylsystem
| 0 °C (32 °F) | No Cooling | Motor, kylsystem
| -5 °C (23 °F) | No Cooling | Motor, kylsystem
| -10 °C (14 °F) | No Cooling | Motor, kylsystem
| -15 °C (5 °F) | No Cooling | Motor, kylsystem
| -20 °C (-4 °F) | No Cooling | Motor, kylsystem
| -25 °C (-13 °F) | No Cooling | Motor, kylsystem
| -30 °C (-22 °F) | No Cooling | Motor, kylsystem
| -35 °C (-31 °F) | No Cooling | Motor, kylsystem
| -40 °C (-40 °F) | No Cooling | Motor, kylsystem
| -45 °C (-49 °F) | No Cooling | Motor, kylsystem
| -50 °C (-58 °F) | No Cooling | Motor, kylsystem
| -55 °C (-67 °F) | No Cooling | Motor, kylsystem
| -60 °C (-76 °F) | No Cooling | Motor, kylsystem
| -65 °C (-85 °F) | No Cooling | Motor, kylsystem
| -70 °C (-94 °F) | No Cooling | Motor, kylsystem
| -75 °C (-106 °F) | No Cooling | Motor, kylsystem
| -80 °C (-112 °F) | No Cooling | Motor, kylsystem
| -85 °C (-122 °F) | No Cooling | Motor, kylsystem
| -90 °C (-130 °F) | No Cooling | Motor, kylsystem
| -95 °C (-138 °F) | No Cooling | Motor, kylsystem
| -100 °C (-148 °F) | No Cooling | Motor, kylsystem
| -105 °C (-156 °F) | No Cooling | Motor, kylsystem
| -110 °C (-164 °F) | No Cooling | Motor, kylsystem
| -115 °C (-172 °F) | No Cooling | Motor, kylsystem
| -120 °C (-180 °F) | No Cooling | Motor, kylsystem
| -125 °C (-188 °F) | No Cooling | Motor, kylsystem
| -130 °C (-196 °F) | No Cooling | Motor, kylsystem
| -135 °C (-204 °F) | No Cooling | Motor, kylsystem
| -140 °C (-212 °F) | No Cooling | Motor, kylsystem
| -145 °C (-220 °F) | No Cooling | Motor, kylsystem
| -150 °C (-228 °F) | No Cooling | Motor, kylsystem
| -155 °C (-236 °F) | No Cooling | Motor, kylsystem
| -160 °C (-244 °F) | No Cooling | Motor, kylsystem
| -165 °C (-252 °F) | No Cooling | Motor, kylsystem
| -170 °C (-260 °F) | No Cooling | Motor, kylsystem

*Observera: När temperaturen i kylsystemet är 85 °C (185 °F) eller lägre kan komponenterna inte påkalla kylning.*
Appendix C  Module Swedish-English

Below are the same documents as in Appendix B, but transformed using an XSLT-document to viewable HTML-documents.

### Signaler från de olika givarna varierar linjärt och påkallar fläktvarvtalet, beroende på temperaturen. Den signal som ger högst värde är den som styr det begärda kylfläktvarvtalet. Om motorvarvtalet understiger 800 rpm blir det ingen laddluftskylning. Normalt är basvarvtalet på fläkten 400 rpm. Det finns en funktion som sänker fläktvarvtalet om omgivningstemperaturen är mindre än 5 °C och ingen funktion begär mer än 2 % fläktvarvtaletssökning.

<table>
<thead>
<tr>
<th>System</th>
<th>Päkallas kylning</th>
<th>Max kylning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motorn</td>
<td>90 °C (194 °F)</td>
<td>100 °C (212 °F)</td>
</tr>
<tr>
<td>Hydrauliksystem</td>
<td>70 °C (158 °F)</td>
<td>95 °C (203 °F)</td>
</tr>
<tr>
<td>Axlar fram och bak</td>
<td>75 °C (167 °F)</td>
<td>85 °C (185 °F)</td>
</tr>
<tr>
<td>Hydrauliktransmission</td>
<td>90 °C (196 °F)</td>
<td>90 °C (194 °F)</td>
</tr>
</tbody>
</table>

**NOTE!**
Gäller om motorvattentemperaturen är 95 °C eller högre. Om motorvattentemperaturen är 85 °C eller lägre kan ej hydraultransmissionen påkalla kylning.

| Omgivningstemperatur är 25 °C (77 °F) eller lägre påkallas kylning av laddluft | 40 °C (104 °F) | 50 °C (122 °F) |
| Omgivningstemperatur er högre än 25 °C (77 °F) påkallas kylning av laddluft | Omgivningstemperatur +15 °C (59 °F) | Omgivningstemperatur +25 °C (77 °F) |
| HVAC                          | Från 0%         | Till 70% av kyfläkts hastigheten |

---

**Document 1: D11 (Swedish)**

The signals from the different sensors vary linearly and command fan speed depending on the temperature. The signal that gives the highest value is the one that controls the requested cooling fan speed. If the engine speed is below 800 rpm, there is no charge-air cooling. Normally, the fan's basic speed is 400 rpm. There is a function that reduces the fan speed if the ambient temperature is below 5 °C and no function requests more than 2% fan speed increase.

<table>
<thead>
<tr>
<th>System</th>
<th>Requests cooling</th>
<th>Max. cooling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Engine</td>
<td>90 °C (194 °F)</td>
<td>100 °C (212 °F)</td>
</tr>
<tr>
<td>Hydrauliksystem</td>
<td>70 °C (158 °F)</td>
<td>95 °C (203 °F)</td>
</tr>
<tr>
<td>Axlar, front and rear</td>
<td>75 °C (167 °F)</td>
<td>85 °C (185 °F)</td>
</tr>
<tr>
<td>Hydrauliktransmission</td>
<td>80 °C (176 °F)</td>
<td>90 °C (194 °F)</td>
</tr>
</tbody>
</table>

**NOTE!**
Applies if the engine's coolant temperature is 95 °C or higher. If the engine's coolant temperature is 85 °C or lower, the hydraulic transmission cannot request cooling.

| When the ambient temperature is 25 °C (77 °F) or lower, there is a request for cooling of charge-air | 40 °C (104 °F) | 50 °C (122 °F) |
| When the ambient temperature is higher than 25 °C (77 °F), there is a request for cooling of charge-air | Ambient temperature +15 °C (59 °F) | Ambient temperature +25 °C (77 °F) |
| HVAC                          | From 0%         | To 70% of cooling fan speed |

---

**Document 2: D11 (English)**
Signaleterna från de olika givarna varierar linjärt och påkallar fläktvarvtal, beroende på temperaturen. Den signal som ger högst värde är den som styr det begärda kyfflåtvarvtalet. Om motorvarvtalet understiger 800 r/min blir det ingen laddluftskylning. Normalt är basvarvtalet på fläkten 400 rpm. Det finns en funktion som sänker fläktvarvtalet om omgivningstemperaturen är mindre än 5 °C (41 °F) och ingen funktion begär mer än 2 % fläktvarvatssökning.

Laddluftens kylbehov ignoreras de två första minuterna sedan maskinen startats.

Om temperaturen i motorns kylsystem är 85 °C (185 °F) eller lägre kan hydraultransmissionen eller axlarna inte påkalla kylning.

<table>
<thead>
<tr>
<th>System</th>
<th>Påkallar kylning</th>
<th>Max kylning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motor, kylsystem</td>
<td>90 °C (194 °F)</td>
<td>100 °C (212 °F)</td>
</tr>
<tr>
<td>Hydraulsystem</td>
<td>70 °C (158 °F)</td>
<td>80 °C (176 °F)</td>
</tr>
<tr>
<td>Axlar fram och bak</td>
<td>75 °C (167 °F)</td>
<td>85 °C (185 °F)</td>
</tr>
<tr>
<td>Hydraultransmission</td>
<td>80 °C (176 °F)</td>
<td>90 °C (194 °F)</td>
</tr>
<tr>
<td>Motor, laddluft</td>
<td></td>
<td></td>
</tr>
<tr>
<td>När omgivningstemperature är 25 °C (77 °F) eller lägre</td>
<td>40 °C (104 °F)</td>
<td>50 °C (122 °F)</td>
</tr>
<tr>
<td>Motor, laddluft</td>
<td></td>
<td></td>
</tr>
<tr>
<td>När omgivningstemperature är högre än 25 °C (77 °F)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AC</td>
<td>Från 0%</td>
<td>Max 70% av fläktvarvtalet</td>
</tr>
</tbody>
</table>

**Document 3: Ds2 (Swedish)**

The signals from the different sensors vary linearly and command fan speed depending on the temperature. The signal that gives the highest value is the one that controls the requested cooling fan speed. If the engine speed is below 800 rpm, there is no charge-air cooling. Normally, the fan's basic speed is 400 rpm. The function that lowers the engine speed is activated if the engine temperature is below 5 °C (41 °F) and no function requests more than 2 % fan speed revision.

Laddluftens kylbehov ignoreras de två första minuterna sedan maskinen startats.

Om temperaturen i motorns kylsystem är 85 °C (185 °F) eller lägre kan hydraultransmissionen eller axlarna inte påkalla kylning.

<table>
<thead>
<tr>
<th>System</th>
<th>Requested cooling</th>
<th>Max. cooling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motor, kylsystem</td>
<td>90 °C (194 °F)</td>
<td>100 °C (212 °F)</td>
</tr>
<tr>
<td>Hydraulicsystem</td>
<td>70 °C (158 °F)</td>
<td>80 °C (176 °F)</td>
</tr>
<tr>
<td>Axles, front and rear</td>
<td>75 °C (167 °F)</td>
<td>85 °C (185 °F)</td>
</tr>
<tr>
<td>Hydraulics transmission</td>
<td>80 °C (176 °F)</td>
<td>90 °C (194 °F)</td>
</tr>
<tr>
<td>Motor, laddluft</td>
<td></td>
<td></td>
</tr>
<tr>
<td>När omgivningstemperature är 25 °C (77 °F) eller lägre</td>
<td>40 °C (104 °F)</td>
<td>50 °C (122 °F)</td>
</tr>
<tr>
<td>Motor, laddluft</td>
<td></td>
<td></td>
</tr>
<tr>
<td>När omgivningstemperature är högre än 25 °C (77 °F)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AC</td>
<td>Från 0%</td>
<td>Max 70% av fläktvarvtalet</td>
</tr>
</tbody>
</table>

**Document 4: Ds2/Es (Swedish/English)**
På svenska

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under en längre tid från publiceringsdatum under förutsättning att inga extra-ordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

For ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

In English

The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

© Daniel Resman