In-memory Translation Spotting Using Big Data Analysis

Author:
Jan Scherbaum

Supervisor:
Prof. Owen Conlan

Master in Computer Science

University of Dublin, Trinity College

Submitted to the University of Dublin, Trinity College, May, 2014

Declaration

I, Jan Scherbaum, declare that the following dissertation, except where otherwise stated, is entirely my own work; that it has not previously been submitted as an exercise for a degree, either in Trinity College Dublin, or in any other University; and that the library may lend or copy it or any part thereof on request.

Signature:

Date:

Summary

Localisation service providers look for ways to minimise the time needed for translations. Automated machine translation techniques do not provide quality of translations to make them usable in professional translation services. As a result of imperfections of automatic machine translation techniques, Translation memories are created. Translation memories are databases of previous translations which are collected in an attempt to reuse previously professionally translated segments of text.

Translation memory systems, however, pose a number of usability issues. Overwhelming the user with potentially large segments of text is one of the main problems. Translation spotting is a technique where given a segment of text, the corresponding segment is identified in the translated sentence. This technique would allow to either return the translation of the query itself, or to use it to highlight the most likely translation candidate in the original text. Translation spotting thus offers a solution to some of the usability issues of translation memory systems. Existing translation spotting techniques were using statistical translation models.

With the emerging trend of in-memory databases, big data analysis becomes possible in a very performant fashion. Additionally, many organisations such as the European Commission publish large numbers of translated documents as translation memories, which are freely available. This dissertation explores the extent to which in-memory computing can be used to support real-time translation spotting. The objectives of the dissertation are to design and implement an in-memory translation spotting system and evaluate it in terms of extensibility, effectiveness and efficiency.

Action research was used as a methodology for this project. The system was developed in an iterative manner. The results of each iteration were then analysed, reflected on and the knowledge gained was used as feedback into the following iterations.

An in-memory translation spotting framework was designed and implemented. The framework consists of tools and components necessary to build a translation spotting system. These components contain libraries for connection into the vendor specific database and tools for data ingestion. A number of algorithms to perform translation spotting were then proposed and implemented on top of the proposed framework. These algorithms mostly work by analysing the co-occurrence of patterns in order to identify translations of queries.

Both the translation memory framework and the translation spotting algorithms were then evaluated in terms of extensibility, effectiveness and efficiency. The system was proven to be easily extensible with further data. It was found, that the proposed single word algorithms yield up to 94% of effectiveness, scale very well and are able to perform in real-time. The proposed multi-word algorithms yield up to 76% effectiveness, even though their performance varies greatly based on the query and size of the analysed data set. A number of optimisations were made to make such analysis possible for multi-word queries. These implementation details mean that the effectiveness is limited by the quality of translation of the shortest ingested sentence containing the query. Additionally, the performance is decided by the length of the shortest sentence containing the query along with the size of ingested data. These results indicate that despite its limitations, the system could be used to bring value to both research regarding translation spotting techniques and modern translation memory systems.

A number of avenues for future work were then identified based on the limitations of the implemented system. Techniques for improving the performance of the multi word algorithms were suggested including the replacement of imperative features with set based logic, analysing possible ways to limit the number of potential translation candidates, limit the computational overhead by using categorisation tools and finally considering compression techniques to allow for pre-computation of translation candidates.

Acknowledgements

Firstly, I would like to thank my supervisor Prof. Owen Conlan for his support and guidance throughout this project.

Special thanks to Dr. Malte Kaufmann for supporting this dissertation and providing me with access to SAP expertise and technology.

Thanks also to Prof. Dave Lewis for pointing out the available data sets, and to Austin Devine for his technical support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Many organisations resolve to hire specialised companies - Localisation Service Providers (LSP), for translations of their products or documents [16]. These companies employ translators, who are in many cases paid per translated unit (a sentence or a paragraph of text). It is therefore understandable that these companies attempt to reuse existing translations as much as possible or to develop automated translation techniques. The results of widely available automatic machine translation techniques are still not satisfactory and generally require human input to fix syntactic or grammatical errors. Another way of LSPs trying to reduce the amount of labour needed for their jobs is to keep all previous translations - create translation memories (TM). These are documents divided into a number of translation units (TU), which are translated to a single or a number of other languages. Translation memories therefore create databases of previous translations aligned by translation units. By developing translation memory systems, which are able to search through the translation memories in order to find a match between currently translated piece of text and the same text that could have been translated in the past, LSPs are attempting to reuse their previous translations.

Organisations such as the European Commission [27] and other parliamentary bodies are required to produce transcripts of their proceedings or documents in all of the official languages, some of which are published and freely available. This generates a large corpus of aligned documents and sentences across a number of languages, with very high standards of translations. There are even efforts to generate translation memories from multilingual websites [7] [17], which result in larger scales of covered terminology and semantic areas of documents. These efforts to archive and generate large corpora, covering many fields of specialisation and levels of translation quality promote the use of translation memory systems. These growing translation memories may be difficult to search and analyse locally on the translators' laptops or PCs due to their enormous scale.

It is the efficiency and effectiveness of the translation tools to help translators with their jobs that is highly valued. Research [7] [8] even suggests that many translators use web search engines such as Google to search for specialised terms. It takes them around two minutes to identify a resource that contains the required term and offers a translation in the target language. Since the translators are paid per the amount that they translate, it is desirable that assistance in terms of locating previously translated term is fast and offers a range of high quality translation candidates. There are a number of online concordancers, which attempt to provide such functionality. These systems take an input in terms of the required term or phrase, the source and target language and return a number of translated sentences, within which the translator can attempt to identify the most relevant translation. The returned results are often the whole translation unit in both the source and the target language. It is then responsibility of the translator to process the information and identify the relevant terminology that is of value. There were a number of attempts to produce techniques for translation spotting [21] [25] [24], which based on the source and target translation units, would attempt to identify fragments of the text that have the highest probability of matching the translator's query. These techniques are a computationally heavy task and it might not be feasible to employ them at the scale of concordancers.

With the continuous development of technologies, such as large server grids, high performance computing (HPC) and cloud computing, the resources needed to search and perform analyses on such large scale corpora are becoming readily available. It is possible to perform the search and real-time analysis on larger and larger quantities of data. This improvement is additionally supported by the continuously increasing limits of main memory size in computer systems. Modern workstations and servers are currently able to support in the order of terabytes of main memory. This is therefore leading to development and increasing popularity of in-memory databases and platforms. In-memory databases combined with parallelism provided by multicore, multi-cpu and distributed systems offer huge performance increases. These systems are even offering the potential for performing big data analytical tasks. Such tasks could have previously taken days to execute on hard-drive based database systems. Since the main memory offers access times which are orders of magnitude smaller than hard drive access times, such tasks could now be performed within minutes, seconds or even real-time.

The development of such parallel, in-memory computing creates new ways of consumption and analysis of large-scale translation memories. Such architectures, leveraging modern information technology paradigms could lead to novel approaches of translations retrieval. The additional computing power could even provide potential for the development of different techniques of translation spotting or automatic machine translation. Large quantity of aligned documents across a number of languages could now be searched and analysed in real-time using the in-memory database technologies. This could lead to improving user experience of translation memory systems, while being able to perform even more computationally heavy analytics and thus potentially improve the quality of translation candidates.

1.2 Research Question

The research question of this dissertation is to explore the extent to which in-memory computing can be used to support real-time translation spotting.

Translation memories (TMs) are large databases of aligned sentences corresponding to previously translated documents. Translation memories are made up of modular pieces of text called translation units (TUs). These units are then given to translators for translations into a number of languages creating translation unit variants (TUVs). Translation spotting is a technique, which attempts to find a translation of a query using aligned sentences. If the query is contained in one of the aligned sentences, the technique attempts to identify translation of the query in the target language.

In-memory database is a type of database management engine. In-memory databases use main memory as their main storage, unlike traditional databases, which use hard disk to store data. In-memory databases use the disk only for back up and persistence layer purposes. As such, they retrieve data and analyse it from the main memory which leads into potential performance increases compared to traditional database systems.

State-of-the-art systems and techniques will be **examined** and **analysed** in the areas of translation memory systems, translation spotting techniques, and in-memory databases and platforms.

A framework will be **implemented** using an in-memory database to allow the storage and efficient retrieval of relevant translation units from a large number of translation memories. The system will be designed in such a way, that translation spotting can be performed runtime. It is important that the data does not need to be pre-processed prior to its ingestion and the whole system does not need to be retrained when additional data is ingested. Based on this framework, techniques will be developed to perform real-time translation spotting by analysing co-occurence of terms.

The proposed framework will be **evaluated** in terms of:

- **Extensibility** - how easily can the system be extended with further data?
- **Effectiveness** - how accurate are the translation candidates proposed by the system? Precision will be calculated for every algorithm implemented in terms of percentage of correct results.
- **Efficiency** - Can the system perform in real-time even when adding more data? The performance will be measured in terms of the time needed to perform the operation on a given data set size and to transfer the result set to the client.

This work will also **identify** potential avenues for future research and potential improvements of the proposed algorithms and the framework.

1.3 Methodology

Action Research [6] was used as a research methodology for this project. The author was interested in the way the in-memory computing concept could be used to enhance the modern translation tools. The state-of-the-art review showed that translation memory systems are becoming increasingly popular and their use and distribution are being actively researched. Namely, the translation spotting concept shows a great potential for improving user experience for these types of systems. Such processing, but performed in parallel and in-memory presented a great potential for improvement. The author and his supervisor discussed the initial theories, which led into the design of first experiments. The framework was developed in short cycles in an agile development manner. After each step, the framework was tested and the author reflected back on the results. These reflections were used in deciding the next steps of the research. In some cases they led to other theories, which, when implemented, in fact improved the efficiency of the framework, while others helped to shape the underlying understanding of the problem space.

1.4 Thesis Overview

This chapter has described the motivation behind this work, defined the research question and challenges that are addressed by the thesis, and briefly described the methodology which was used during the research.

Chapter 2 will describe the start-of-the-art of the translation memory systems as currently used by the translation and localisation industry, identify the strengths and weaknesses of these systems, and will relate this work to ongoing research in the area of translation spotting. It will then focus on the latest shifts in computing paradigms that lead to the increasing popularity of in-memory computing systems. It will consider currently available platforms and will identify the in-memory database which will be used for the implementation of the translation spotting system developed as a part of this dissertation.

Chapter 3 will describe the design and implementation of the proposed framework. The high level description of the translation spotting system will be presented first. The administrative layer and the in-memory database sub-systems will then be described in detail, in top-down manner. The blocking issues and coding challenges, which were mostly encountered during the implementation of the translation spotting logic, will be analysed in terms of their impact on the solution design and performance of the implemented framework.

Chapter 4 will focus on how the system was evaluated in terms of test framework, tests and test cases and the corpus used for the project. The evaluation chapter will then critically evaluate the framework by presenting a number of experiments which were performed. The system was tested in two phases, both of which will be presented. In each phase, all single word and multi word algorithms will be evaluated. The evaluation will take into account the performance of each algorithm and its quality of translations with respect to changing size of the data set.

Chapter 5 will summarise the motivation and objectives of the disssertation. The achievements will be described mainly in terms of successful effectiveness measurements, well scalable single word algorithms and easy extensibility of the system. The feasibility of using in-memory platforms to support the translation memory systems will be confirmed. Limitations such as the poor scalability of the multi word algorithms will be described. These limitations will then indicate avenues for future research, which mostly contain potential ways of optimising the multi word translation spotting algorithms.

Chapter 2

State-of-the-art Review

2.1 Introduction

This chapter identifies the state-of-the-art products and research in the areas of both the translation memory systems and the in-memory database systems.

To begin with, the reader will be presented with the idea of modern translation memory systems in Section 2.2. The features of these systems are listed along with their advantages and disadvantages. A number of research projects are then described to show how academic research has been actively trying to improve these systems.

Section 2.3 explains the relevance and importance of in-memory databases along with commercially available products that could be used for an implementation of a translation memory system.

2.2 Translation Memory Systems

There are a number of widely used translation memory systems. These systems differ in functionality offered to the users as well as the internal mechanisms for performing such functionality.

In most cases, translation memory systems offer a series of tools for creation, management and use of translation memories. These systems often allow users to create local or shared databases of translation memories by importing documents in various formats. Some tools even work as plug-ins for popular office suite applications such Microsoft Office or OpenOffice. These tools then perform various techniques to segment the texts into translation units, index the text and transform the documents into an internal format for storage in the specified translation memory database.

Upon translating a document, these tools then allow for different types of search functionalities among translation memories already existing in the database. If a match is found, the system offers its users potential to reuse existing previously translated texts.

In addition to the basic functionality of the translation memory system, the latest versions of the most popular systems additionally offer features such as cloud-based translation memory database, user defined terminology dictionaries, machine translation plug-ins and interoperability with other types of systems.

2.2.1 Commercially Available Systems

There are a large number of translation memory tools available. Based on the translation memory survey [8], the most widely used translation systems are: TRADOS, WordFast and Déjà Vu.

The three of the most widely used tools were analysed based on their online presentations. It was found that these systems offer great variety of tools for increasing the productivity of translators. Many of these functionalities overlap, while others seem to be specific to given systems.

SDL Trados Studio

SDL Trados Studio [3] is the latest translation memory system produced by SDL. It is a successor of the SDL Trados and SDL Trados 2006, both of which appeared in the first three most widely used translation memory products in the survey.

SDL Trados is offered in a number of versions, each of which offers different types of functionality and is targeted at different types of translators. The fees for the software range from €99 yearly subscriptions to €2595 for a perpetual licence (at the time of writing this dissertation).

The basic features of SDL Trados Studio include the ability to use number of languages, translate a single file at a time, unlimited size of a translation memory, batch tasks of file preparations and creations of packages.

The more advanced features include creation of aligned translation memories, manual annotation and tagging of translated text with user defined tags, real-time preview of the document being translated, auto-suggest of written text and analysis of context of translation memories.

Many of these features are very vaguely described in the web presentation of the product and it is unclear what these features do and to what extent they improve the productivity of translators.

Wordfast Translation Studio

Wordfast [5] is a producer of translation memory software. They offer a number of translation studio products, each of which is designed using different software development paradigms. They offer Wf Classic which is an MS Word plugin, Wf Pro which is a standalone application, Wordfast Anywhere which is a web application, and Wf Server which is built to run on a centralised server.

The Wf Classic and Wf Pro products are priced at €400 for a three year licence (at the time of writing this dissertation). The web based tool is free to use with limited functionality under certain terms and conditions.

The Wf Server product acts as central translation memory database, which can be connected to from the Wf Classic, Wf Pro and Wordfast Anywhere tools. It allows for sharing the central corpus across the whole organisation, while offering security in terms of users' permissions.

The Wf Classic and Wf Pro promise a number of features including user defined macros, terminology management in the form of user defined glossaries, interoperability with external tools such as dictionaries or machine translation engines, quality assurance features, batch processing or sentence alignment modules.

Similarly to SDL Trados Studio, Wordfast claims that their products greatly improve the productivity of translators.

Déjà Vu X3

Déjà Vu X3 [1] is another suite of applications, which were identified among the most widely used by translators. Similarly to the other tools, Déjà Vu X3 is also offered in a number of varieties, which are targeted at different types of users and use cases. It is offered as three different products DVX3 FREE, DVX3 Professional and DVX3 Workgroup.

The DVX3 FREE product is available for free, but provides very limited functionality. The DVX3 Professional and DVX3 Workgroup are then priced at €420 and €1490 respectively.

Both the DVX3 Professional and DVX3 Workgroup provide similar sets of features, with the Workgroup version providing tools for collaboration and real-time sharing. Both versions provide features such as inline formatting, spell checking, multiple file projects, number of supported formats, multilingual projects, and databases among others. The AutoWrite feature uses analysis of existing translation memories in order to provide autocomplete functionality. A particularly interesting feature for this dissertation is a function called DeepMiner. The company claims that the feature uses data mining functionality to analyse existing translation memories, which are then used for retrieving the most relevant translations. Further details on the feature are not provided. It is not clear how DeepMiner feature works and what sort of value it brings, but it is the first mention of using data mining techniques on existing records in order to filter and improve retrieved records.

Advantages Offered by Translation Memory Tools

Based on the claims of vendors of translation memory software [3] [5] [1] and on the outcomes of the translation memory survey [8] there seem to be a number of advantages in using translation memory tools.

The basic advantages comprise of the functionality that these systems were built for, namely:

- Searching for a given sentence in previously translated texts

- Retrieval of previous translations based on the matched sentences
- Allow translators to reuse previous translations, especially in highly repetitive texts (such as technical documents and financial or marketing content)

Translation memory systems often offer tools not only for reusing previous translations, but also to build translation memories. These tools often take a document and process it using techniques such as segmentation. Segmentation divides the document into translation units (sentences or paragraphs), which are then to be translated into one or more languages. Such an automatic segmentation saves time to divide the document into smaller pieces, but also promotes the alignment of sentences as the document is translated.

The modern translation memory tools also claim to offer many other, more advanced features to improve productivity of translators such as:

- User defined macros and dictionaries
- Interoperability with other tools such as spell checkers and machine translation tools
- Quality assurance tools
- Integration with text editors and office suite applications

Disadvantages of Translation Memory Tools

Despite many of the mentioned advantages of using translation memory systems, even the most modern tools have a number of disadvantages.

Translation memory systems employ the use of fuzzy or exact matching in order to identify the most relevant translation unit, which offers the required translation. The translation memory system then retrieves the original language sentence along with its translation in the relevant language, which the translator can reuse. In the current translation memory systems, the problem has been arising with the fact that the translation memory system would be trying to search for the whole sentence in order to retrieve its translation. Even in the case of highly repetitive texts, an occurrence of the whole translation units could be quite rare [21]. In fact, in the translation memory survey it was found that the reason why translators do not use translation memory systems is the low frequency of occurrence of such translation units [8].

When the translation memory system finds a translation unit, which matches the query exactly, the whole translated sentence is returned. However, there is not much information on how the documents are segmented when they enter the process of becoming translation memories. Is the document being segmented based on sentences? Is the document being segmented based on paragraphs? Such questions lead to the division of documents into translation units. In many aligned corpora such as the translation memories produced by the European Commission [27], the translation units can take up to whole paragraphs of text. Returning a whole section of text could potentially overwhelm the user. It would take additional amount of time in order for the translator to identify the relevant translation within larger segments of text. This poses a usability issue on translation memory systems.

In addition to the success of matching and usability issues regarding the returned results, the translation memory systems were identified to be relatively expensive. Even though free versions of the systems exist, these versions are often very limited in functionality. It is possible that such versions of the software could bring value to amateur translators, for whom such software is an improvement compared to manual translations performed using dictionaries. For experienced and professional translators, the features offered by the more advanced versions of the software such as the spell checking, collaboration features or the functionality within corporate networks would be more than desirable. The licences for such versions of the available translation memory tools could be expensive. It was identified that the professional versions of the translation memory systems could cost anywhere between €420 and €2595 per licence.

The disadvantages of using commercially available translation memory systems, such as the high cost and the usability issues, led into academic research attempting to resolve these problems. Free to use online concordancers have been built in order to generate translation memories to overcome high prices of translation memory systems. In addition to online concordancers, there has been an active research in the areas of effective performamce of translation spotting and consequent improvement of usability of such systems.

2.2.2 Translation Spotting

Translation Spotting is a technique, which attempts to find a segment of a sentence, which corresponds to the literal translation of the corresponding segment in an aligned sentence [24] [28].

Many techniques have been explored on how to align segmented sentences. During the generation of translation memories, the text is first segmented and then aligned. The segmentation divides the text into natural chunks of text such as sentences. It is then important to match the corresponding segments of both the source language and translated language texts. These techniques often use fuzzy matching [19], syntax trees, thesauri, neural networks or part of speech tagging [21].

During the ARCADE project, which researched methods of performing such alignments, word level alignment techniques were also considered [28]. The alignment on the word or phrase level has been defined as translation spotting.

It has been proposed to perform matching of previously translated sentences based on its segments. In the case where the exact match has not been found, using smaller segments of the sentence could yield more matches [25]. Such an approach should improve the issues related to poor frequency of matching a sentence. A translation of a sentence would be found based on a part of the currently translated sentence rather than the full one. These segments could be decided by the translators. It is possible that if the full sentence has not been found, the translator may still look for a certain part of it. TransSearch [17] or WeBiText [7] are examples of such a system. They offer such a functionality as online systems, which are free to use and offer aligned content collected from online resources. It allows the translators, students, linguists and professional writers to search for a query, which is contained within previously translated sentence. The user can then see and reuse the translation of the given query.

Even though such an approach would help to solve the issue of rare matches, the fact that the whole translated sentences could overwhelm the translator was still a problem. Therefore, experiments on using translation spotting to identify the segment of the translated sentence corresponding to the user's query [25] [24] were carried out. The techniques proposed for performing translation spotting in such systems used statistical translation models along with a number of general natural language processing tasks. It was acknowledged that the performance and precision of these techniques are only as good as those of the translation models used.

Additionally, these statistical translation models offer language specific functionality. If the system was to be used in a truly multi-lingual environment such as the European Union, it would be necessary to provide these engines for possibly all official languages pairs.

2.3 In-memory Database Systems

Conventional relational database systems were designed to store data on the disk. Disks are non-volatile storage media, which were always known to have large capacities. The capacities of disks were much higher than those of the main memory. It is however a well known fact, that the access times of disks are orders of magnitude slower than those of main memory modules. This issue was often addressed by increasing sizes of caches and by optimising the techniques and data-structures for use on disks. Based on experimentations [15], it was found that even despite these optimisations, the main memory reads are still around six times faster for sequential access and around million times faster for random access.

An in-memory or main memory database is a concept, which has been known for a very long time [11]. There were many thoughts on how to overcome issues related to insecurity of data stored in the volatile storage such as logging and recovery techniques. In-memory database would therefore still use disks, but mainly as a persistent data storage to avoid loss of data.

One of the main reasons why in-memory databases were known, but not popular was their limit on the size of data that could be supported. Main memory sizes were very limited both in terms of manufacturing large modules and operating systems support.

The recent advances on both software and hardware meant that the interest in in-memory databases re-appeared. The latest top of the range server systems support up to 12TB [2] of main memory, while the latest operating systems are designed to support up to 1PB [4] of main memory. These limits mean that in-memory databases could be realistically used today to store and process very large quantities of data. With their performance advantages over the disk based database systems, in-memory databases could be used for both enterprise applications and also for big data analysis tasks.

2.3.1 Commercially Available In-memory Databases

Even in early 1990s, there were a number implemented and available in-memory databases [11]. At the time of writing this document, the concept of in-memory computing was becoming increasingly popular. Many database system vendors started introducing in-memory add-ons or optimisations into their database engines. There were only a small number of pure in-memory database systems, which were gaining on popularity.

SAP announced the release of SAP HANA in 2010 and was soon followed by Oracle Exalytics. Later IBM released an in-memory optimisation to their DB2 called IBM Blu Acceleration. When analysing these products, it was found that in general similar features were offered.

Column-store

Most of in-memory database engines that are currently in the market promote the use of columnar storage. Columnar storage of data is a type of internal representation of the tables within the database. Conventional relational database engines would store the data organised by rows (row store). The column store on the other hand stores the data organised by columns.

The column store organisation of data leads to much faster OLAP (Online Analytical Processing) tasks [20]. In analytical tasks, there are often aggregations performed on a single column. Using the column store means that such analysis can be carried out by accessing data items, which are stored in a sequential manner. The column store is therefore designed in a cache-conscious manner and can yield great performance improvements for OLAP tasks.

The reason for conventional database system using the row-store was better performance in OLTP (Online Transactional Processing) tasks [10]. In such tasks, it is often the case that new rows are inserted, deleted or many attributes are changed in a single record. All of the described in-memory databases also support OLTP tasks. The choice to use either row store or column store is often given to database developers. The tables which are expected to be used exclusively for OLTP tasks can be declared to use row storage, while other tables can be declared to use column store along with a number of OLTP optimisations.

Compression

Compression was another feature which was observed to be heavily used in modern in-memory databases and platforms [20] [13] [14]. Often it was explicitly mentioned that dictionary compression is the method for compressing data. This means, that non-integer cells will be translated into integer values. A dictionary is therefore maintained, which keeps relations between the integer attributes and their original values. This approach can yield space optimisations on columns, the values of which are often repetitive (such as names or addresses).

If the space required to store the data is smaller, the operations on the large quantities of such data are likely be faster. The number of records, which can be transferred into the processor from main memory in a single read, is higher, thus minimising memory read latencies.

Parallelisation

The increasing level of potential parallelism is another hardware advance that the computer science and engineering community has seen in recent years. When the single-core CPUs reached its limits in terms of increasing frequencies and hardware optimisations, it became apparent that the future of computing is shifting towards parallelism. The trend has therefore become to build low powered processors on lower frequencies, which are however equipped with a number of fully functional cores. The modern server systems often consist of a number of processors, each of which offers a number of cores.

In-memory databases seem to be perfect examples of how software architecture paradigm shifts to leverage the parallelisation potential introduced by multi-cpu and multi-core systems. Most of the widely used and commercially available database systems claim to be designed to perform high degrees of parallelism on all computations [13] [9] [10] [14].

Vendor Specific Features

Oracle Exalytics A marketing approach specific to Oracle Exalytics [13] [18] is that the software is tailored to run on hardware designed and built by Oracle itself. Oracle claims that running the Exalytics software on their own hardware ensures top compatibility and hardware specific levels of optimisations.

There does not seem to be much information available on the architecture and feature sets available by the Exalytics software. Based on the available information, the Exalytics software consists of four main components. The OBIEE In-Memory Accelerator is a component built exclusively for optimising business intelligence tables and infrastructure. The Essbase In-Memory Accelerator then seems to be the actual in-memory database module of the system. It's responsibilities include parallelisation, distribution and storage level operations. Besides the standard SQL queries, the Essbase module enables the developer to use performance enhanced MDX queries. The In-Memory Data Caching component then takes care of a number of caching operations in order to speed up executions of queries. The Oracle BI Publisher Accelerator then consists of a number of tools, which allow the enterprise clients to generate rich reports on the data and simple user interfaces. Oracle also claims that the exalytics in-memory machine provides the ideal platform for Business Analytics Consolidation.

SAP HANA The main selling point specific to SAP HANA is the fact that it was designed as a complete platform for data management and modern business applications [9]. To do that, the architecture of SAP HANA offers a number of internal components. These components are designed to provide a number of optimisations, but also a large number of domain specific languages [10] to provide application expressiveness. Such an expressiveness allows the developers to build any sort of business application directly in SAP HANA.

In order to satisfy the needs of both OLAP and OLTP tasks, SAP HANA provides means of storing data organised in either rows or columns. To do that efficiently, it provides two data store engines - row store and column store. The column store engine is the recommended way of storing data within HANA and as such it provides a number of optimisations to make OLTP tasks efficient even when using the columnar approach [23]. It keeps a number of delta storages for OLTP tasks, which are read or write optimised. These deltas are then merged to the main column-store engine using intelligent techniques for determining when the right time to perform the delta merge would be.

In conjunction with data store engines, SAP HANA provides a number of domain specific engines. The graph engine allows for representing data as linked graphs. It allows for traversal of data as graphs and to perform graph operations and calculations on data stored in the graph engine. SAP HANA also provides a dedicated text engine, which was built to store unstructured text, perform analysis on it and even allow for linking of structured data with such an unstructured text. Spatial engine then offers means of storing geospatial information and of performing spatial computations on such data. The XS engine is a built-in HTTP server, which allows for exposing the business logic defined within the SAP HANA application through REST APIs. It allows for server-side JavaScript processing of requests and responses or definition of oData services.

Besides providing a number of native computational models and engines, SAP HANA also offers a number of domain specific languages. It allows for the use of standard SQL. Similarly to Oracle Exalytics it also provides the support of MDX. There are a number of view types, which allow for definition of data flows through the application and different levels of logic expressiveness directly within those views. Should the developer need additional features such as imperative logic, SQLScript language provides imperative extensions to SQL. R is also supported within the database to carry out highly statistical tasks and computations.

On top of the very rich feature set, the appliance also comes packaged with HTML5 based user interface framework called SAPUI5. SAPUI5 allows for development of fast and lightweight web applications native to SAP HANA.

Other In-memory Databases There are a number of other in-memory databases commercially available. Many of these products have only very recently been released. It can therefore be expected that these technologies are not as matured as Oracle Exalytics or SAP HANA. Additionally, there is a lack of information published on their features or internal architectures. These in-memory databases will therefore not be considered by this thesis.

2.3.2 Advantages of In-memory Databases

Based on the state-of-the-art review, it seems that in-memory database systems offer a number of advantages, while previously identified disadvantages have already been overcome.

The main advantages relevant to this work include the following:

- Most in-memory databases offer optimisations for both OLAP and OLTP tasks rather than just OLTP as in disk based databases
- Compression techniques allow to minimise the space needed for repetitive values
- Automatic parallelisation techniques used in main memory databases improve the performance of in-memory applications without the need for explicit development of concurrency within these applications
- Using main memory as the main data storage offers potential for performance improvements

2.4 Summary

This chapter summarised the state-of-the-art products and research in areas of both translation memory systems and in-memory databases.

Some of the commercially available and most widely used translation memory systems were identified. For each of these systems, pricing information and productivity features were described in order to gain a high level idea of the value brought to translators. With such knowledge, the attempt was made to identify the advantages and disadvantages of these systems. The literature was then consulted to find ways of how academic research is trying to solve these problems by performing sentence and word level alignments (translation spotting). It was found that there were a number of techniques proposed for performing translation spotting mainly with the use of statistical translation models.

The history of in-memory databases was then studied and discussed. It was found that the advantages of using in-memory databases were known for a long time. It was the case that up until recently it was not very feasible to use them for many real-world tasks due to hardware and software limits. With recent advances in both hardware and software, in-memory databases started receiving attention. Modern computer systems allow to store very large quantities of data in main memory and the analytical tasks on such data became much faster. The availability of commercial in-memory databases was then discussed. The most common and relevant features of modern in-memory systems were described along with features specific to two leading in-memory database systems - Oracle Exalytics and SAP HANA.

SAP HANA was chosen as the in-memory database of choice for this project. The proposed translation spotting system will be designed and implemented using SAP HANA.

Chapter 3

Design and Implementation

3.1 Introduction

This chapter describes both the design and implementation of the proposed translation spotting techniques. First, influences on the system from the state of the art are described. The high level design and implementation overviews are then presented. The further sections then drill down into the design and implementation of each of the components of the system in a top-down manner. In each case, the design of the component is presented first, followed by the implementation details, and where needed, code snippets for each of the subcomponents and algorithms.

3.2 Influences from the State of the Art

The implemented system will provide the functionality listed as basic advantages of translation memory systems in Section 2.2.1. Namely, it will provide means of querying the system for either a single word or a multi word query. The system will perform search for the given query in previously translated sentences and allow for returning their translations, thus allowing the translator to reuse previous translations.

The identified disadvantages of translation memory systems will be addressed. Firstly, the matching rate will be improved by allowing the translator to specify the query, which will then be matched based on a subsegment of any previously translated sentence. The usability issues related to the length of the result will be addressed by performing translation spotting. The user interface will then be able to highlight the most likely translation candidate of the query in the original text.

The more advanced productivity features of modern translation memory systems are not in the scope of this project. These features include integration with text editors, quality assurance tools, user defined macros or dictionaries and interoperability with other systems.

In order to avoid the needs for pre-processing of the data and retraining the system, all analyses will happen runtime. In order to achieve that the system performs in real-time, all advantages of modern in-memory databases listed in Section 2.3.2 will be used. The

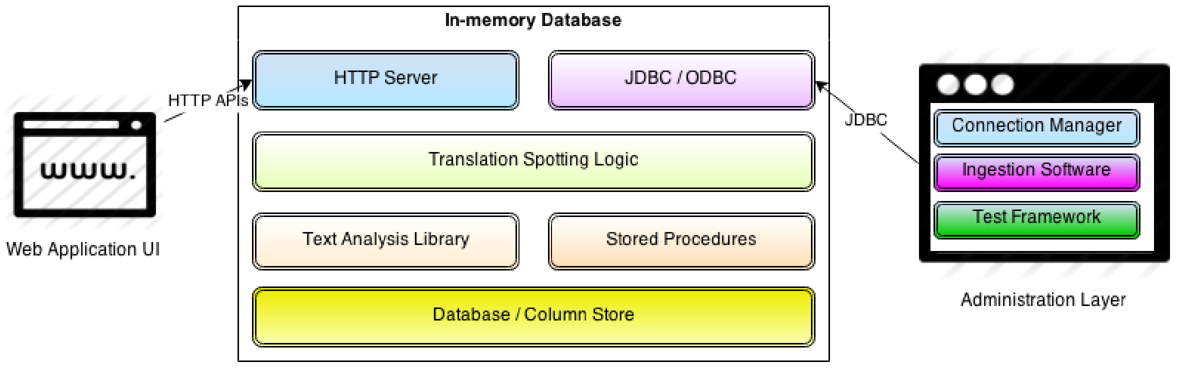

Figure 3.1: High level design of the system

tables will be using column store data representation in order to optimise the system both for the OLAP and OLTP tasks. In regards of the implemented system, OLAP tasks are the actual translation spotting techniques, while OLTP tasks include the ingestion of data. The dictionary compression will be used implicitly by the database engine itself. The queries will be designed in such a way as to promote the automatic parallelisation of the computations (where possible), which will help to reach real-time performance of the system. Furthermore, the use of an in-memory database for the implementation of such system will make such Big Data analyses possible in the first place.

3.3 High Level System Overview

3.3.1 Design

The proposed system can be divided into three separate subsystems (see Figure 3.1):

1. User Interface
2. Administrative Layer
3. In-memory Database System

User Interface

The user interface could be designed for specific purposes. An organisation using the system might prefer to develop a native desktop application, a mobile app, or a lightweight web-application.

Based on the latest trends and paradigm shifts in software development, it would make sense that the user interface for the system is built as a web application. Based on the chosen technologies, the user interface might be cross-platform and even usable on mobile devices. This would allow the translators to use the system in many settings, such as working from home or while travelling. If hosted on a cloud platform, the system could even be accessible and used from mobile devices or available to the general public.

A sample user interface has been built as a proof of concept. The user interface was built as a web application, using the HTML5 framework packaged with SAP HANA called SAPUI5. The web application asynchronously communicates with the APIs exposed by the HTTP server as a part of the database system.

Administrative Layer

The administrative layer is a software package, responsible for ingestion and management of data as well as for testing the system. As part of this dissertation, these components are designed to run on a laptop or a workstation of the system administrator. The connection manager is a general component, which is used by other parts of the administrative layer. The ingestion module provides applications for importing both data in terms of translation memories as well as stop words for a particular language. The administrative layer could be implemented using various technologies, the features of which should support connectivity to the database and provide functionality to parse the stop words form a simple text file and a number of provided XML based translation memory formats.

In-memory Database System

The main part of the project is the actual translation spotting system. This is a component, which as a part of the proposed design resides on the database subsystem. Once the data is imported, it performs the text analysis on the data and is able to perform the translation spotting. The functionality of the system can be exposed to the front-end applications via HTTP-based APIs or through database connectors.

To be able to execute the algorithms, which are described in this project, the text analysis library should provide features to perform tokenization of text in a number of languages, perform stemming and optionally part of speech analysis. The analysis should be performed upon importing the translation memories and could be done either in the database itself, or as a part of the administrative system in the ingestion phase applications.

3.3.2 Implementation

Adminitrative Layer

The administration layer was implemented in Java. Java offers the features required for the administrative layer such a number of built-in tools for parsing and processing XML based documents. The data set used in this dissertation offers translation memories in one of the standard XML based formats called Translation Memory eXchange (TMX). Additionally, connecting to relational database engines is easy using widely accessible JDBC and ODBC libraries.

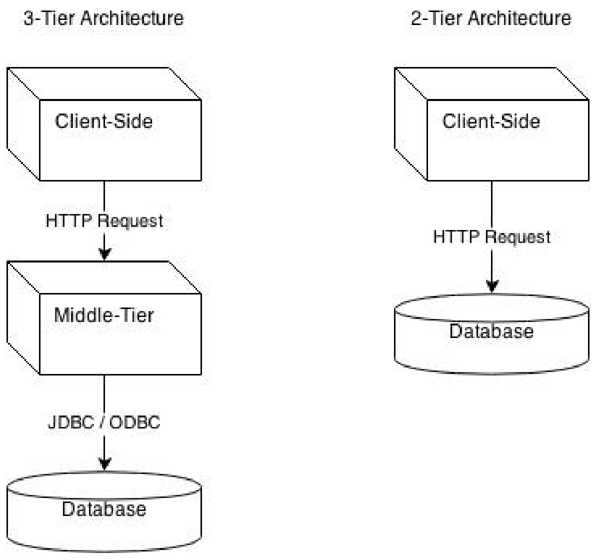

Figure 3.2: Comparison between 3-tier and 2-tier architecture

In-memory Database System

As an in-memory database engine for this project, SAP HANA was chosen. HANA is a column-store, in-memory database and platform. Column-store database engines are well known for their performance in highly analytical (OLAP) tasks [20] [23], which, combined with the in-memory storage of the data, promises sufficient performance for real-time translation spotting.

The in-memory database system is represented solely by an SAP HANA instance. The database schema uses the column-store engine to store the ingested stop words and translation memories. HANA itself provides a text analysis library, which performs tokenization, stemming, normalisation and part-of-speech analysis for a large number of languages. The logic of the translation spotting is then defined directly within the database engine itself in the form of stored procedures. These procedures are then made available for consumption by the front-end application via the built-in HTTP server called the XS Engine.

SAP HANA promotes the use of 2-tier systems (see Figure 3.2). In general, with enterprise applications it was common to design large-scale systems using the 3-tier architecture. Such systems would consist of user interface running at the client-side, a middle layer (written in Java or other language), and the underlying database management system. The client side was merely designed to contact the middle layer with queries resulting from user interaction with the user interface components. The middle layer would then contact the database system, retrieve the data needed for the given query, analyse it and return the result to the user interface. Such moving of potentially large quantities of data between the database and the middle layer could pose a network bandwidth overhead, potentially slowing down the execution of the query. To minimise such overhead, SAP HANA offers a number of engines to provide the application programmer with expressiveness needed to write their business logic directly within the database. Such functionality was used within this project, namely the text analysis library and SQLScript stored procedures.

The text analysis library provides means of performing many natural language processing tasks. It gives the user a chance to specify the desired level of functionality when creating a fulltext index on a column in a table. The text analysis library will then perform language specific tasks such as tokenization or part-of-speech analysis. Such processing can be done either synchronously or asynchronously upon inserting data into the table. The library then creates a text analysis table exclusive to the given text analysis index. This functionality is used in all of the translation spotting algorithms described in this dissertation. The text analysis library satisfies the text analysis functional requirements set out by the design of the system.

3.4 Administrative Layer

3.4.1 Design

The administrative layer is a system, which is not directly related to the end user. It should contain applications that include mechanisms for importing further corpora into the system, managing and importing stop words, and a test framework. The subsystem is designed as a suite of applications to run locally on a laptop or a workstation of the system administrator. It should contain a number of supporting tools and applications. The administrative layer should:

1. Define single **support tools** module defined for connecting to the specific database engine and linking the administrative data structures with database schema.

2. Offer a way to allow the administrators of the system to easily develop and add **parsers** for different formats of translations memories.

3. Provide the **ingestion phase framework**, which will actually import a given data set into the database.

4. Provide a **test framework** to allow the evaluation of implemented algorithms and monitoring of the system.

Support Tools

The support tools component defines a number of helper classes that can be used throughout the application.

The connection manager is a support tool used for defining connection details to a database engine instance used in the current implementation. It should then take care of ensuring that there is an active connection to the database engine and distribute it among other components.

The data access objects are then used to bridge the gap between the internal data structure of the administrative layer and the database engine. It could be used to retrieve data from the database, but also to translate the internal administrative layer objects into structures that will be compatible with the database engine for importing data.

Parsers

To satisfy the extensibility property, which was identified as a crucial non-functional requirement of a successful system, the administrative framework should provide a number of modular parsing mechanisms. Being able to plug in a parser for a new format of translation memories or even being able to determine the type of a file and selecting the correct parser runtime would be an asset to the system. Each parser should extract a translation memory from a file and produce its representation in an in-memory data structure provided by the framework. The in-memory representation of the translation memory can then be translated into SQL statements by an implementation specific data access object (DAO).

Ingestion Phase Framework

The ingestion phase applications will then allow the administrators to specify a single file or a directory with translation memories and using the parser components retrieve an in-memory representation of the translations, which can then be imported into the database system using the common connection facilities in the administrative layer.

Test Framework

The test framework is a part of the system, which allows the developers of the system to evaluate their implementation and possibly help administrators of the system to monitor the performance and precision of the system on the given corpus at any point in time. To do that, the framework should provide logging facilities and the ability to automatically execute defined tests for particular parts of the system, each of which can have a desirable number of test cases. The administrators could therefore develop a whole suite of tests, each testing a different part of the system to ensure that it can perform under required time and precision constraints. Automatic execution of tests could indicate to the administrators of the system that importing a new data set to the system changed the precision of translation spotting or produced too high computational overhead. An implementation of a test framework for a translation spotting system should offer the following features:

1. allow the user to define and reuse a number of test cases
2. allow each test case to accept a number of possible answers and be able to verify or refute a potential translation
3. allow the user to test for single word or n-gram queries with an acceptable level fuzziness in the n-gram matching

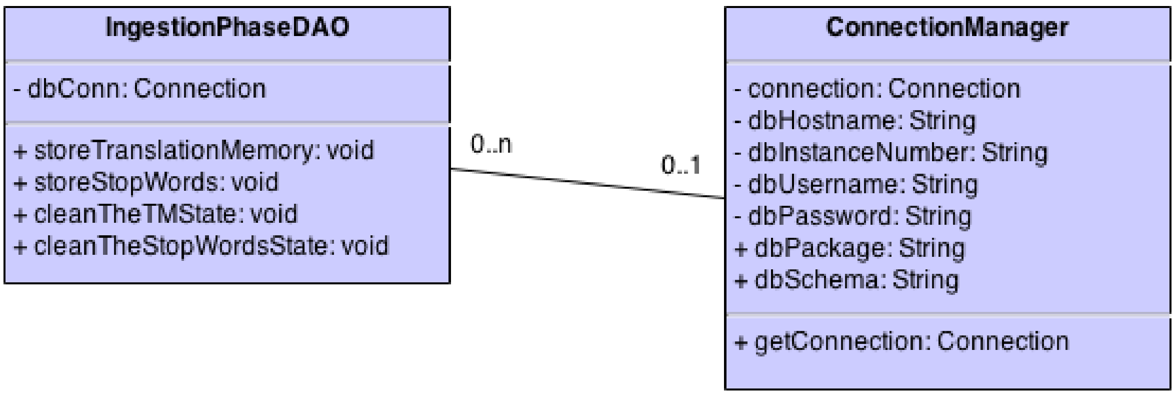

Figure 3.3: Support tools class diagram

4. allow the user to create a test with a number of test cases associated with it
5. report the results of the tests by writing the performance and precision information to a log file
6. allow the test case developer to debug the test cases by reporting the refused solutions

A test framework offering these features may be used to support evaluation of the implemented system.

3.4.2 Implementation

The administrative layer was implemented in Java. Java was the language of choice as it provides easy to use utilities for XML parsing (SAX and DOM). Additionally, the connection to the SAP HANA database is programmatically possible using the JDBC driver provided as a part of the SAP HANA Client. Java is also a widely used, cross-platform object oriented language, which allows easy and modular development of parsing extensions to the system.

Support Tools

Support tools (see Figure 3.3) are classes to be used internally within the administrative framework. The purpose of these classes is to modularise the common functionality that is expected to be used throughout the administrative framework. In the current implementation, support tools consist of a connection manager class and ingestion phase data access object (DAO).

Connection Manager The connection manager is a class, which ensures that there is an active connection to the configured SAP HANA instance and that there is only a single instance of the connection object existing at any point in time. Any application or class as a part of the administrative layer then uses the connection manager to get the connection object that facilitates communication with the backend system. The connection manager therefore checks that there is an existing active connection every time a connection object is requested by any part of the administrative system. If there was no connection to the database created at that stage or if the connection has been closed or reset for some reason, the connection manager takes care of opening a new connection, which is then passed back to the requester. The new connection is opened using the SAP HANA JDBC driver, which comes packaged with the database client software and is correctly included in the build path of the project. This is done in a single public, static method called getConnection.

The current implementation of the connection manager also stores database connection details, such as the address of the database, instance number, username, and password. Based on the deployment frequency and security considerations, these credentials could be taken from the user upon starting any application within the component, or could be stored in a secure storage. The connection manager fulfills the design requirement of providing a single module, specifying the access to the in-memory database which will be used throughout the administrative layer system.

Ingestion Phase DAO Closely tied to the connection manager is the data access object. This is a class, which is used for translating existing in-memory objects into the SQL statements. These statements are then sent via the JDBC connection acquired from the connection manager to the database engine. It is a component providing simple APIs for the communication with the back-end to the rest of the system.

The DAO allows other components and applications in the system to clean the state of the database by deleting all of the stop words, translation memories and their corresponding TU and TUV records. This functionality might not be useful to an administrator of the system, but was used often while developing the ingestion phase software. The corresponding in-memory database system can be cleaned by calling the cleanTheTMState and cleanTheStopWordsState methods on an existing DAO instance.

Besides cleaning the state of the in-memory database system, the ingestion phase DAO was built to allow for easier ingestion of existing translation memories into the database system. The storeTranslationMemory method takes an in-memory representation of a translation memory and translates it into a number of SQL statements. These statements are then executed using the JDBC connection object. To support the extensibility requirement of the system, the method uses batch insert statements for translation unit variants (the translation unit string translated into a number of languages). This means that the translation memory record is created, followed by a single batch statement of its translation units, each of which also has a single batch insert statement inserting all of its variants at once. Such an approach was taken to minimise the network delay produced by large numbers of round-trips, should each of the inserts be sent as a separate statement.

Ingestion Phase Framework

The ingestion phase software is a component, which is solely responsible for parsing and importing data into the SAP HANA database. Easy extensibility of the corpus by new docu-

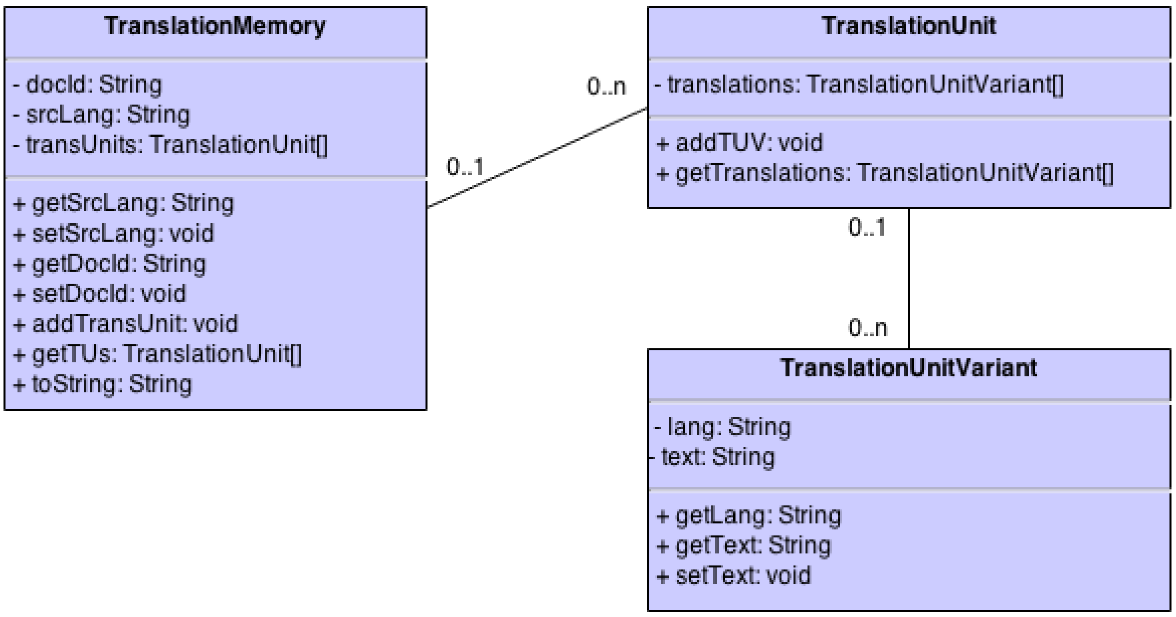

Figure 3.4: Design of the ingestion phase TM representation

ments or documents from a different source is a desired feature of the system. The ingestion phase mechanism has therefore been designed in such a way to allow for import of a number of widely accessible formats of translation memories into the analytical system. It provides an interface describing a parser with an API that the system expects to use. As a part of the implementation, a parser for the widely used TMX format has been provided. The administrator of the system should be able to supply additional parsers for importing translation memories that are in a non-standard format or in a format that has not previously been used. The strategy design pattern could be especially useful to perform choice of the desired parser runtime based on the extension of the translation memory file or based on an optional user input. This allows the parsers written for each of the formats to be simply "plugged" into the system.

While stop words are represented simply as an array of strings, the module also includes a collection of java classes that are used as in-memory data structures for storing the parsed translation memories (see Figure 3.4). The parsing module creates the in-memory representation of a single translation memory. The structure of these classes is designed closely based on the TMX format, which is one of the standard formats for the exchange of translation memories. Each file therefore represents a single document (translation memory), which is divided into a number of separate sentences (or paragraphs) called translation units. Translation unit is the smallest piece of a translation job for a translator. Each translation unit then has a number of translation unit variants, which are the translations of the translation unit into desired languages. Each translation unit variant contains information about the language that it represents and the text of the translation.

The import applications use all of the aforementioned components, combining their functionality to perform ingestion of data specified by the user. The applications for importing translation memories (user interfaces which are not in scope of the project) simply open a

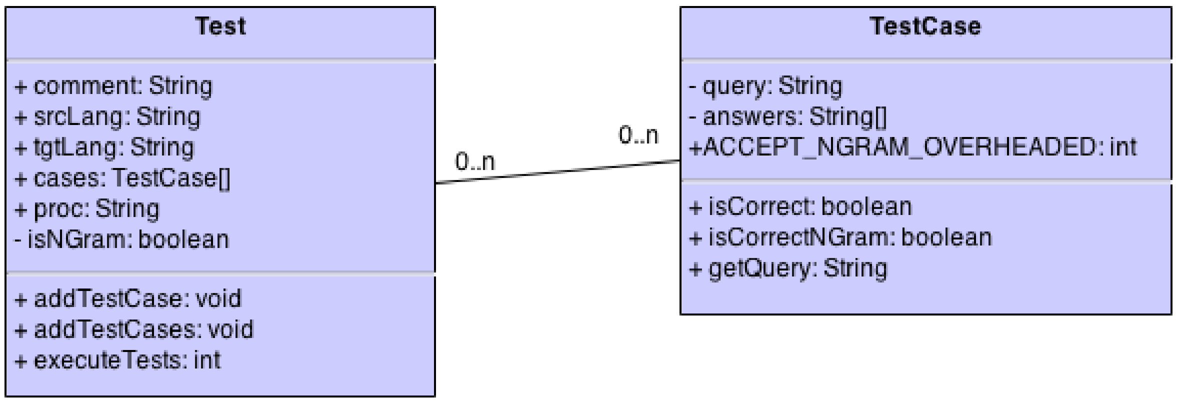

Figure 3.5: Object oriented design of the test framework

file (or a folder) specified by the user. For each file in the specified folder, reference to the file (or the current file in case of the folder) is then passed to the parser, which returns the in-memory representation of the contents. As the system is extended with additional corpora of documents, the application should be rewritten to inspect the type of the file and choose the correct parser provided in the system. The obtained translation memory object is then passed into the IngestionPhaseDAO, which translates the structure into corresponding SQL statements, which are then imported into the in-memory database system. The application for importing stop words works in a similar fashion. It opens a single file specified by the user, uses a standard stop word parser (stop words are represented by a text file storing the stop words for a single language delimited by a newline character), which returns an array of strings. These stop words which are then passed to the IngestionPhaseDAO object to be translated into a number of SQL insert statements, which are sent to the database system.

Test Framework

The test framework has been developed as a part of the administrative layer. It is not semantically related to the translation spotting, rather it provides a way of evaluating and monitoring the implemented translation spotting system.

The test framework is implemented as a collection of classes (see Figure 3.5), which allow the developers or system administrators to define a number of tests. Each of these tests could test different aspects of the system such as measuring the performance and precision of different algorithms provided by the system under the currently imported corpus.

The test case class represents a single value to be tested against a number of possible acceptable answers. A single test case allows to test for either a single word with an exact match or an n-gram with a certain level of fuzziness. This is implemented by the means of providing two different methods isCorrect and isCorrectNGram. The isCorrect method should be called on tests of the single word algorithms, which require exact (case insensitive) verification of a potential result, i.e. the result has to match one of the defined allowable answers. The isCorrectNGram then uses a predefined constant for acceptable overhead. This method verifies the result, if it is a substring of any acceptable answer and does not have a word overhead of more than the predefined number of words. It should be the case, that if the desired translation is produced, with an addition of two or three words, it will still be useful for the translator.

The test class represents a single test. A single test can be thought of testing a single procedure or a piece of functionality and passing a number of different values (test cases) to it as an input. When calling the executeTests method, the class iterates over the list of specified test cases, calling the tested procedure for every single one. It then uses the functionality provided by the TestCase class to determine whether the answer obtained from the tested translation spotting procedure is an acceptable result. For each test case the method keeps a number of records - the time needed to execute the query, whether it was correct or not and the number of cases executed up to that point. The test object also takes in a reference to a log file, to which it appends test information at the start of the test execution, such as description of the test, source and target languages and the current time of the test execution. If a test case fails, the wrong and the expected values are also appended to the log to allow for debugging of the test cases. At the end of the test execution, the method logs the test results in terms of percentage success rate, number of successful attempts and the total number of attempts and average query execution time.

As a part of the evaluation of the translation spotting system, a test application has been created for each of the separate algorithms described in the following sections. Each of these applications then creates two test objects calling the same procedure in the translation spotting system - one for translation from a given language to another and vice versa. The application then adds predefined test cases (which might be shared with other tests), opens a log file which is then used by the Test and executes both tests.

3.5 In-memory Database System

3.5.1 Design

In order for an in-memory database system to be able to perform translation spotting as described in this dissertation, it should:

1. Store the translation memories in a **predefined schema**
2. Be able to perform the required **text analysis** tasks on imported data
3. Define **application logic** within the database engine

Database Schema

The in-memory database engine of choice should allow the user to define a certain database schema. The schema of the database defines the structure, in which the imported data will be stored.

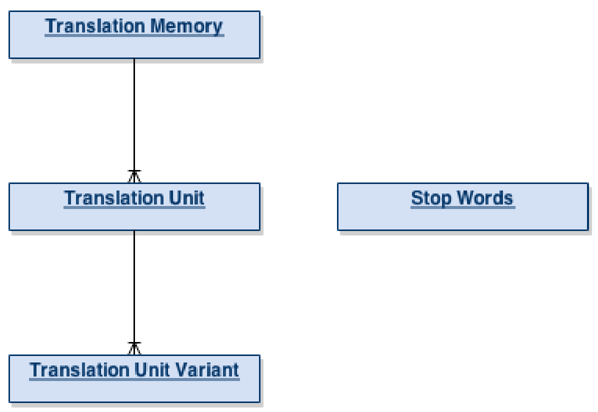

Figure 3.6: Overview of the proposed database schema design

The database schema of the proposed system (see Figure 3.6) is again closely modelled on the TMX format and the object oriented data structure that was used for parsing the documents. The top level entity is a translation memory, which is the representation of a single document. Each translation memory may have a number of translation units. Translation unit is an atomic segment of text separated out from the document for translation. Translation unit could be a title, sentence or a whole paragraph. Each translation unit then may have a number of language variants. In each of the variants, the text of the translation unit is either in the original language or translated into another language. Stop words entity is then independent entity, which simply stores the stop words for any single language.

Text Analysis

In the proposed system, the text analysis tasks are carried out on all data imported into the system asynchronously upon insertion. This means that the ingestion applications do not get stalled by the text analysis during the import and yet, once analysed, the data is ready to be used. The text analysis tasks currently used in the proposed algorithms comprise of tokenization and stemming, with potential to extend the algorithms to perform the analyses with regards to part-of-speech types.

The tokenization, also known as segmentation, is a technique, in which the text is divided into its elements [22]. In the case of an English sentence, the input is divided into separate words. Stemming is a technique, in which a word's stem is identified. Stems are base dictionary forms.

Part-of-speech tagging means annotating each word within a sentence with the role it plays in that sentence. Based on the rules of any given language, a word can be a verb, subject, object, preposition. Part-of-speech tagging then annotates each word in a sentence with these roles.

Once the text analysis tasks are finished, the text library should be able to store the result of the analysis in the database itself for real-time processing. These records should also be annotated with the link to the original translation memory and translation unit to ensure that the algorithms will be able to create relations between the obtained information and the original unstructured text.

Application Logic

The reason of using an in-memory database to perform big data analysis is that the data stored in main memory can be accessed and processed much faster than in disk-based database systems. An average access time for a main memory read is orders of magnitude faster than a read from the disk. This offers a great potential to speed up analytical tasks. To avoid introducing additional latency in transferring the data from the database system to a processing engine, it is crucial that the application/business logic is defined directly within the database system.

It depends on the database system of choice how the application logic can be defined in the database engine. Most in-memory databases should however offer some means of relating data through views or defining imperative logic through SQL extensions as stored procedures.

Development of novel techniques for real-time translation spotting through big data analysis was one of the main goals of the project. As such, it was found important that the whole corpus of available data is analysed in order to make proper use of the data available to the system. While the existing techniques of translation spotting use word alignment techniques and statistical translation models in an attempt to perform translation spotting on a pair of translated sentences, this project will attempt to spot and analyse common patterns on the whole corpus of text. To achieve such analysis, the proposed algorithms will attempt to identify co-occurrences of words or phrases across translation memories that include the required query.

3.5.2 Implementation

The in-memory database chosen for the implementation of the system is SAP HANA. HANA is an in-memory database, but also offers a number of components, which make it usable as a whole platform. HANA offers text analysis library, which includes all of the functionality that the design of the system requires. The features provided by the text analysis library include tokenization, stemming, but also additionally features such as part-of-speech tagging or sentiment analysis features.

SAP HANA also offers a number of engines and languages to allow developers to define the business logic of applications directly within the core of the database engine. One can define many types of views, but SAP HANA also offers imperative extensions to SQL, which allow for the creation of the user defined functions and stored procedures.

The logic, which is defined within the core of the database engine can then be exposed via JDBC connections or through REST APIs using the built-in XS engine. The JDBC connections allow applications written in high level languages such as Java to execute any SQL queries - this functionality is described in section 3.4.2 dealing with the database connectivity of the administrative layer. The XS engine is a type of HTTP server, which is closely tied to the database. It allows the definition of APIs in server-side JavaScript, which can then query or manipulate the database through SQL queries or by calling relevant stored procedures. At the time of implementation, HANA offered XSJS or XSODATA, where XSJS allows for programmatic execution and retrieval of data in any desired format, while XSODATA was designed to link the APIs to the database with the use of OpenData protocol.

Database Schema

In SAP HANA, on a single database instance, there can be a number of database schemas. Database schema is an equivalent to databases or projects in other database systems. A schema defines a namespace. Each schema can then have a number of tables, indexes, views and procedures.

There are a number of ways of defining database objects such as schemas and tables - they can be defined dynamically using SQL queries, or they can be defined using so called design time artifacts. These are definition files, which once committed to the database repository and activated, are actually created in the database. This makes them easily transferrable to other systems, which is the reason why it is the best practise to use design time artifacts where possible.

Schema Creating a schema within SAP HANA is a simple task. A file TRANSSPOTTING2.hdbschema was created (see Figure 3.7) After activating the file into the database repository, a schema called TRANSSPOTTING2 is created within the database.

```
schema_name="TRANSSPOTTING2";
```

Figure 3.7: Database schema definition

Tables The tables within the database were built to satisfy the database schema design (see Figure 3.6). Each translation memory (representing a single document) is separated into a number of translation units (usually at a sentence level). Each translation memory is then further separated into a number of translation unit variants (TUVs), each of which represents the translation unit translated to a given language.

The tables are created by defining hdbtable files. Each file creates a table, name of which corresponds to the name of the file. Within the definition, properties of the table are defined such as the schema to which the table belongs, type of storage (row store or column store), columns definitions and primary key. The code snippet in Figure 3.8 is an example of the table definition, namely the TRANSLATION_UNIT_VARIANT.hdbtable. This definition file creates a table called TRANSLATION_UNIT_VARIANT. The table belongs to the TRANSSPOTTING2 schema. It has five columns with the specified types, optionality and comments. The table will use columnar storage as its internal data representation.

```
table.schemaName = "TRANSSPOTTING2";
table.tableType = COLUMNSTORE;
table.columns = [
    {name = "ID"; sqlType = INTEGER; nullable = false; comment = "ID of
  the translation unit variant";},
    {name = "TUID"; sqlType = INTEGER; nullable = false; comment = "Id
  of the translation unit that this variant belongs to";},
    {name = "lang"; sqlType = VARCHAR; length = 10; nullable = false;
  comment = "Language of this variant";},
    {name = "text"; sqlType = NCLOB; nullable = false; comment = "Text
  of this translation";},
    {name = "ACTIVE"; sqlType = INTEGER; defaultValue = "1";}
];
table.primaryKey.pkcolumns = ["ID"];
```

Figure 3.8: Table Definition

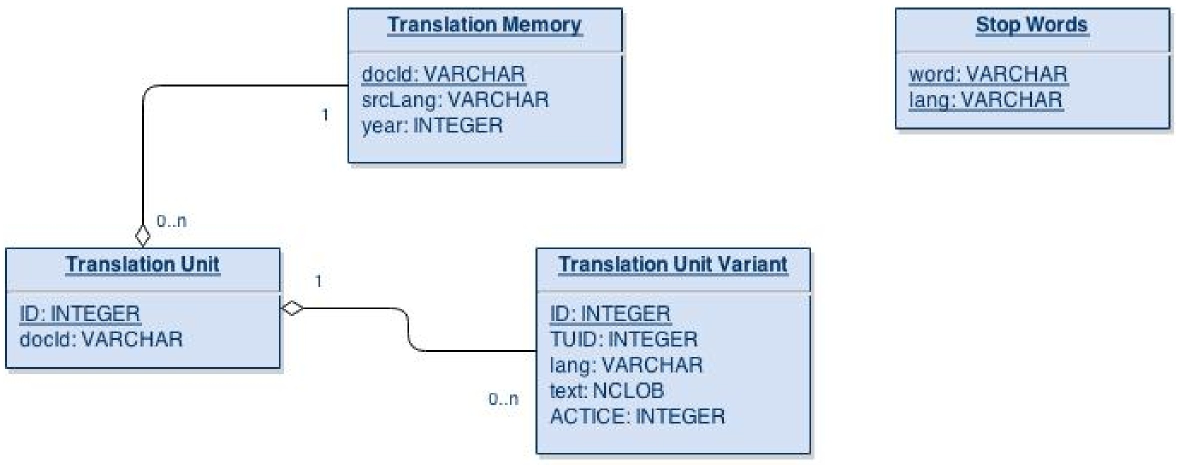

Figure 3.9: Tables implemented in the system

Column-store is the recommended way of storing data within SAP HANA. While row-store is well suited to transactional operations (OLTP), in which data is often read in chunks of whole rows and is often updated, the column-store should offer better performance in highly analytical (OLAP) tasks [20] [23]. Since this system is being built to perform translation spotting using big data analysis, all tables within the system will be using the HANA's column-store engine. The rest of the tables (see Figure 3.9) are implemented in similar fashion.

Privileges The installation of SAP HANA instance provides a user called SYSTEM by default. This user was used for the purposes of development and testing. By default, users do not have any privileges on newly activated content. Once tables have been developed, it is important to grant SQL privileges to the user that would be used for further development. The following SQL statement (Figure 3.10) grants the SELECT privilege on TRANSSPOTTING2 schema to the user SYSTEM. Should the system be used in production, it would be desirable to create further users. Based on business rules of the system, users would be assigned roles. Based on the roles and subroles, different types of privileges would be assigned to the newly created users. The required privileges were stored in a SQL file and executed manually.

```
call SYS_REPO.GRANT_SCHEMA_PRIVILEGE_ON_ACTIVATED_CONTENT('select',
    'TRANSSPOTTING2', 'SYSTEM');
```

Figure 3.10: Grant Select Privelege to System User

Text Analysis

To utilise the functionality built into SAP HANA, namely the text engine, there is a fulltext index created for the texts of the TUVs with the text analysis library enabled. This means that upon the ingestion of data, the database engine indexes the data asynchronously as expected. It also additionally performs text analysis on the data, which includes tokenization, stemming, and part-of-speech analysis. For storing all of the analysis outputs, another table is created. The text analysis functionality itself offers great benefit and greatly simplifies the implementation of translation spotting.

Definition The functionality of the text analysis library is declared dynamically through the use of SQL.

The full-text index is created on the TRANSLATION_UNIT_VARIANT table, namely the text column (see Figure 3.11). The index includes fuzzy search capabilities and turns on the text analysis library with the full linguistic analysis configuration. The text analysis library is then provided the name of the column, which specifies the language - this feature is especially useful when dealing with a multiple language data set such as the one published by the European Commission.

To ensure that entries in the table produced by the text analysis library correspond to the state of the translation memories imported even after update or deletion, a foreign key with the cascade option is defined. The foreign key is linking the ID of an entry in the output table to the ID of the unit variant it corresponds to. If the given variant is deleted, the referential engine within the database will take care of deleting any corresponding entries in the text analysis output table.

```
CREATE FULLTEXT INDEX TUV_INDEX ON "TRANSSPOTTING2"."sap.tcd.
    dissertation2.tables::TRANSLATION_UNIT_VARIANT" ("text")
     FUZZY SEARCH INDEX ON
     TEXT ANALYSIS ON
     CONFIGURATION 'LINGANALYSIS_FULL'
     LANGUAGE COLUMN "lang"
     ASYNC;

ALTER TABLE "TRANSSPOTTING2"."$TA_TUV_INDEX" ADD CONSTRAINT ALTER_COMMAND
      FOREIGN KEY("ID") REFERENCES "TRANSSPOTTING2"."sap.tcd.dissertation2.
    tables::TRANSLATION_UNIT_VARIANT"("ID") ON DELETE CASCADE;
```

Figure 3.11: Text Analysis Index Definition

Output Representation The output of each fulltext index with the option of text analysis enabled produces a new table in the system, which represents the output of the text analysis on that table and attribute. The table representing the output of the text analysis is structured in a way to provide as much information as possible per token in the text. The entries in the table can be identified by matching the on the ID value, which corresponds to the ID of the text column being analysed. Each word in the text is then annotated with information including the index of the word within the text, the token itself, language of the token, its part-of-speech type, normalised version of the word (without special characters and diacritics), the stem of word (if available), and the number of the sentence and paragraph that the token belongs to. Additionally, there is metadata associated with each token such as the timestamp of when the text analysis was performed and its character offset within the sentence.

Stored Procedures

Stored procedures are used as a technique to represent the application logic directly within the database engine. As a part of SAP HANA, stored procedures can be defined using languages called SQLScript or R. R is a language, which can be used for statistical calculations within procedures. The described implementation is using the SQLScript language, which is a superset of standard SQL and allows for definition of declarative, but also imperative logic within the database engine.

As described in the application logic design section (3.5.1) of this chapter, to ensure the use of the whole available corpus, it is proposed to perform analysis of phrase co-occurrence across the corpus. Based on the output of the text analysis library, it was clear that the implementation of such mechanism for single word queries will be relatively straightforward. The text analysis library produces single word tokens, the occurrences of which can then be counted in the matching translation units. It was becoming clear, that the logic of the n-gram or multi-word queries will need additional calculations in order to generate the possible substrings of matched text, which would then be evaluated.

Due to such additional operations required for the n-gram queries it was decided that these use cases will be implemented as separate algorithms. The iterative implementation started with the most basic procedure for the single word translation spotting, the result of which was evaluated and fed into the design and implementation of the next iterations. The following sections describe the algorithms and their implementation for both the single word and n-gram algorithms.

Single Word Translation Spotting Single word algorithms are relatively simple to understand and perform well. These algorithms generally don't require the use of any of the imperative constructs of SQLScript. When using mostly set based logic, the HANA engine is able to automatically perform optimisations of the queries and parallelisation of any required processing.

The most basic algorithm that was initially implemented is the simple co-occurrence algorithm.

Simple Co-occurrence Spotting The simple co-occurrence spotting algorithm was the first and most naive algorithm that was implemented as a part of the system.

First, the algorithm uses the fulltext index and its text search capabilities to identify the translation units, which contain the relevant translation (see Figure 3.12). Identifying the relevant translation units is done by executing SQL full join query, where TRANSLATION_UNIT_VARIANT table is joined on itself based on the translation unit ID. This generates a number of combinations between the translations of the given translation units. These combinations are then filtered out using the WHERE statement. The condition of the filter is that source language text contains the queried text, the language of source table is the language specified as the source language parameter of the procedure, while the language of the target table is target language. In simpler terms, the snippet of code generates a list of

```
TUs = SELECT tgt.ID FROM
  "TRANSSPOTTING2"."sap.tcd.dissertation2.tables::
    TRANSLATION_UNIT_VARIANT" as src
 FULL JOIN "TRANSSPOTTING2"."sap.tcd.dissertation2.tables::
    TRANSLATION_UNIT_VARIANT" as tgt
 ON src.TUID = tgt.TUID
 WHERE src."lang" LIKE :srcLang
 AND tgt."lang" = :targetLang AND contains(src."text", :inQuery);
```

Figure 3.12: Simple Co-occurrence Spotting - Part 1

translation unit variant IDs, which contain the translation of the specified text in the target language. The inQuery is a parameter, which specifies the searched text, the srcLang is a parameter specifying the source language and the targetLang specifies the target language.

Once translation unit variants which are known to contain the required translation are obtained, the output of the text analysis library is analysed with respect to these variants. The analysis attempts to spot the words, which are occurring across the identified sentences most frequently. The code in Figure 3.13 selects each word from the text analysis output. It

```
res = SELECT IFNULL(TA_STEM, TA_TOKEN) as "token", COUNT(*) as "score"
    FROM "TRANSSPOTTING2"."$TA_TUV_INDEX2" WHERE TA_TYPE != 'punctuation'
    AND ID IN(SELECT ID FROM :TUs) GROUP BY IFNULL(TA_STEM, TA_TOKEN)
    ORDER BY "score" DESC;
```

Figure 3.13: Simple Co-occurrence Spotting - Part 2

attempts to prioritise the stems of the words. Since these are single word queries, it makes sense that a stem of the query is also a valid translation and an experienced translator should be able to change the form of the word to obtain syntactically and grammatically correct translation in the given context. If the stem is not available (possibly due to imperfections in the stemming functionality of the text library), the token itself is used. These words are then counted, with using the SQL count function and grouping by either the stem or the token. To ensure, that this operation only takes the desired translation unit variants into account, the

IDs of the text analysis output are filtered to only count the IDs, which were found to contain the translation. Finally the entries are ordered by the count in descending order, which acts as a score for the translation candidate.

The filter on the TA_TYPE column of the text analysis output table was introduced as a fix for an issue discovered during the phase one of the evaluation (see Section 4.2.2). It was found that after an update of SAP HANA to a newer version, the punctuation token type was a new feature. This caused issues in the results of most implemented algorithms and the decision was made to introduce a fix. This filter therefore filters out all tokens, which are of type punctuation. The fix was also implemented across all of the following algorithms.

Single Occurrence Limit Spotting During manual testing and evaluation of the simple co-occurrence spotting algorithm, it was found that in almost all cases the required results were present in the result set, but not in the top positions. If the approach is to be usable and bring value to translators, it is desirable that the correct translation is the one with the highest score. It was found that in most cases, the most common stop words in the given language would take up the highest score positions in the result set.

It is understandable that stop words in any given language are the most frequently used terms in any text. It was therefore attempted to limit the number of stop words that will be counted in the score of the translation candidate. To limit the occurrences of stop words it was decided to limit the maximum added score of any single word per translation unit variant to one. This means, that if a single word occurs twice, it will not be counted twice in the score, but only once.

The occurrence limitation was implemented by using the DISTINCT functionality of SQL. The algorithm therefore only selects one stem or token per translation unit variant ID. As such, when counting the number of occurrences across the whole corpus, a single translation candidate will only be counted once per sentence. This functionality is implemented

```
resFilter = SELECT DISTINCT IFNULL(TA_STEM, TA_TOKEN) as "token", ID
  FROM "TRANSSPOTTING2"."$TA_TUV_INDEX2"
  WHERE ID TA_TYPE != 'punctuation' AND IN(SELECT ID FROM :TUs);
res = SELECT "token", COUNT(*) as "score"
  FROM :resFilter GROUP BY "token"
  ORDER BY "score" DESC;
```

Figure 3.14: Single Occurrence Limit Spotting

by performing one additional query to filter out any duplicate words in any one sentence (see Figure 3.14). Only the output of this query is then fed into another query, which similarly to the simple occurrence algorithm uses the count functionality to assign a score to each of translation candidates.

It was found, that this approach decreases the impact of stop words on the result set, but does not fully eliminate it.

Stop Words Co-occurrence Spotting As an approach to eliminate the impact of stop words, it was considered to identify and filter-out any stop words from the result set completely. It could be the case that the user's query is a stop word itself, in which case, if the corresponding translation is also a stop word, this algorithm could yield wrong results. This has been taken into account and the algorithm has been designed with two assumptions. Firstly, the algorithm assumes that there are lists of stop words for each language in the translation memories. Secondly, there is an assumption that if a word is marked as a stop word in a given language, its translations will be marked as stop words in the other languages.

The algorithm starts off as the previous algorithms by identifying the translation unit variants in the target language that do contain the relevant translation. The rest of the analysis is based on whether the query is a stop word or not. A query is therefore executed to retrieve

```
SELECT count(*) INTO countStop
  FROM "TRANSSPOTTING2"."sap.tcd.dissertation2.tables::STOP_WORDS"
  WHERE "word" = :inQuery;
```

Figure 3.15: Stop Words Co-occurrence Spotting - Part 1

the number of stop words from the STOP_WORDS table (see Figure 3.15), which are same as the query. The count of such stop words is stored in an integer variable. Based on whether

```
 if(:countStop > 0)
THEN
   res = SELECT IFNULL(TA_STEM, TA_TOKEN) as "token", COUNT(*) as "score"
  FROM "TRANSSPOTTING2"."$TA_TUV_INDEX2"
  WHERE TA_TYPE != 'punctuation' AND ID IN(SELECT ID FROM :TUs)
  GROUP BY IFNULL(TA_STEM, TA_TOKEN) ORDER BY "score" DESC;
ELSE
   stopWord = SELECT "word" from
   "TRANSSPOTTING2"."sap.tcd.dissertation2.tables::STOP_WORDS" ;
   resTmp = SELECT IFNULL(TA_STEM, TA_TOKEN) as "token", COUNT(*) as "
     score"
  FROM "TRANSSPOTTING2"."$TA_TUV_INDEX2"
  WHERE TA_TYPE != 'punctuation' AND ID IN (SELECT ID FROM :TUs)
  GROUP BY IFNULL(TA_STEM, TA_TOKEN) ORDER BY "score" DESC;
   res= SELECT * FROM :resTmp WHERE ("token" NOT IN (SELECT "word" from :
     stopWord));
END IF;
```

Figure 3.16: Stop Words Co-occurrence Spotting - Part 2

the count of stop words is greater than zero or not, a decision is made on the type of algorithm that will be used (see Figure 3.16). If there were stop words, which matched the query, the simple co-occurrence algorithm is executed and the results of that analysis are returned. In case of the number of stop words being zero, meaning that the current query is very likely not a stop word, a slight modification of the co-occurrence algorithm is executed. First, all of the stop words are stored in a local result set. The matching and ranking of co-occurrences is then carried out in a similar fashion as it happens in the simple co-occurrence algorithm, but the result is only stored in a local result set. A final query is then executed, which filters the retrieved stop words from the ranked translation candidates and returns the result.

Based on manual preliminary testing, this algorithm was found to be effectively filtering the stop words out of the ranked list of translation candidates.

Stop Words & Single Occurrence Spotting Both the single occurrence limit and stop words filtering algorithms showed improved results. It was the case, that both performed better in different scenarios. It was therefore worth trying to combine the two algorithms. If the query is a stop word, then using the single occurrence algorithm as a fallback option should improve the results by limiting the scoring of other stop words, which could affect the overall position of the correct translation. In the case, where the query is not marked as stop word, the precision of the translation spotting could be improved if the query really is stop word, which is missing in the stop words lists.

The implementation corresponds to the stop words filtering algorithm, with the addition of the extra filter query. The filter query uses the DISTINCT functionality of SQL to filter out duplicates of tokens within a single sentence. This filtering mechanism is applied both in the case of the query being a stop word and not.

This algorithm was found to be by far superior in terms of its precision, but was found to be less performant than the other algorithms described.

N-Gram Translation Spotting Algorithms The n-gram algorithm implementation was identified to be different from the aforementioned single word techniques. The implementation of the single word translation spotting was made possible with little development overhead due to the text analysis library. The output of the text analysis allowed for analysing co-occurrences of single words across large quantities of unstructured text. This was possible simply by combining a number built-in SQL features such as joins and count. It was considered to use the single word algorithms to translate multi word queries word by word. This approach would however not be realistic. Different languages have different sentence structures and word orderings. Additionally, the linguistic techniques such as declension would result in grammatically wrong translations. It was therefore decided to use similar approach to the single word queries and analyse co-occurrences of multi word phrases. At the time of the implementation of the project, there seemed to be no functionality built-in into the query language or the text analysis library, which would allow for matching of substrings across unstructured text.

A number of potential ways of implementing such functionality were considered. The abilities of different types of views available in SAP HANA offer great potential when dealing with analytical and computational tasks on data and defining data flows within the database. These views however didn't offer potential for combination of tokens from the text analysis output table to generate all of the possible substrings of the identified translation unit variants. It was therefore decided to resolve to using the imperative features of the SQLScript language. Since the text analysis produces all of the tokens from the analysed text, and their location within the original sentence, it is possible to imperatively build all of the substrings of a given sentence.

Using such imperative functionality prevents the optimisation components of the database engine to perform automatic parallelisation of the queries and could lead to performance issues. It is therefore unfeasible to perform such an operation on all of the identified translation units. Instead, the proposed n-gram algorithms suggest to identify the relevant translation units and only select one from which translation candidates will be generated. Since the corresponding source language variants contain the searched queries, it is assumed that all of the identified target language variants will contain the desired translation. All of the substrings of the selected sentence will therefore be generated and for all of them it should be a matter of employing the same techniques of analysing co-occurrence that were successfully used in the single word algorithms.

Two algorithms for multi-word translation spotting were proposed, implemented and evaluated. The first one is a simple Co-occurence N-Gram Spotting, which is inspired by the simplest of the single word algorithms. The second algorithm is the Co-occurence with Dictionary Spotting, which is using the existing single word translation spotting procedures to improve the scoring mechanism by identifying common keywords in both the query and the suggested translation candidates.

Co-occurrence N-Gram Spotting The first and simplest algorithm is the Co-Occurence N-Gram spotting algorithm. As previously described, this algorithm selects a single sentence from the identified translation unit variants and uses that sentence to build a list of all potential translation candidates. This is done by generating a list of all substrings of the sentence, which are then assigned scores based on the co-occurrence in the identified sentences and the differences in their lengths and the length of the query.

First, the algorithm executes the same query as in previous algorithms to identify the list of translation unit variant IDs, which do contain the relevant translation.

The algorithm then calls another procedure, which is responsible for counting the number of words in the query. The number of words is going to be used when calculating the score of potential translation candidates. The numWords procedure is very simple (see 3.17), it uses a while loop to iterate over all of the tokens, counting each word until the remaining input is an empty string. The spotting algorithm retrieves the count from output parameter

```
while(:input <> '') DO
   cnt := :cnt + 1;
   input := SUBSTR_AFTER(:input, ' ');
END WHILE;

res := :cnt;
```

Figure 3.17: Number of Words Procedure

of the count procedure, which will later be used for scoring of translation candidates (see Figure 3.18). The algorithm then chooses a (semi)random sentence from the identified set of sentences that are known to contain the translation of the query (see Figure 3.19). Since all of the sentences are assumed to contain the translation, the selection decision should not

```
call "TRANSSPOTTING2"."sap.tcd.dissertation2.procedures::numOfWords"(:
    inQuery, :input_length);
```

Figure 3.18: Co-occurrence N-Gram Spotting - Part 1

affect the functionality of the algorithm. The length of the selected sentence in terms of the number of words will affect the performance of the system. The longer the sentence, the more substrings there are, the more potential translation candidates will be identified and the more computational effort is needed to identify co-occurrences of these candidates across the potentially large subset of the corpus. To minimise such effort, the shortest sentence in terms of the number of words is selected using the ORDER BY functionality, selecting the top row. The number of words in the selected sentence is represented by the count of words and is stored in a local variable. The tokens that represent the words of the sentence are

```
-- select (semi)random row with a translation
orderedCandidates = SELECT COUNT(*) AS CNT, ID FROM "TRANSSPOTTING2"."
    $TA_TUV_INDEX2" WHERE TA_TYPE != 'punctuation' AND ID IN(SELECT ID
    FROM :TUs) GROUP BY ID ORDER BY CNT;

SELECT TOP 1 ID INTO id FROM :orderedCandidates;
SELECT TOP 1 CNT INTO length FROM :orderedCandidates;
```

Figure 3.19: Co-occurrence N-Gram Spotting - Part 2

then selected from the text analysis output table into a local result set (see Figure 3.20). The result set is then converted into the array type, ordering the tokens by their position within the original sentence. The translation candidates are then generated using nested for loops

```
toks = SELECT TA_TOKEN, TA_COUNTER FROM "TRANSSPOTTING2"."$TA_TUV_INDEX2"
     WHERE TA_TYPE != 'punctuation' AND ID = :id;

-- convert the tokens of the selected translation into an array
tokens := ARRAY_AGG(:toks.TA_TOKEN ORDER BY TA_COUNTER);
```

Figure 3.20: Co-occurrence N-Gram Spotting - Part 3

(see Figure 3.21). The outer for loop index i represents the starting position of the currently generated substring, while the inner for loop index j represents the ending position of the current substring in the original sentence. At each iteration of the outer loop, the current sentence variable is initialised to an empty string, which is then built-up to contain up to the rest of the sentence. In each iteration of the inner loop then, the current sentence is evaluated using the scoring formula based on its length compared to the query and its co-occurrence across the identified translation units. Each of these evaluated sentences along with their corresponding scores is then stored in parallel arrays. There have been a number of experiments in order to identify the ideal formula of scoring based on the number of co-occurrences and lengths of the original query and the current sentence. Both the current sentence and its score are stored in corresponding indices of two parallel arrays.

```
FOR i IN 1 .. :length DO
  DECLARE currSentence VARCHAR(2048) := '';
        FOR j IN :i .. :length DO
    SELECT CONCAT(CONCAT(:currSentence, ' ') , :tokens[:j]) INTO
    currSentence FROM DUMMY;
     call "TRANSSPOTTING2"."sap.tcd.dissertation2.procedures::numOfWords"
    (:currSentence, :curr_length);
    SELECT COUNT(*)*(1000-(ABS(:input_length - :curr_length)+1)*10) INTO
    score FROM "TRANSSPOTTING2"."sap.tcd.dissertation2.tables::
    TRANSLATION_UNIT_VARIANT" WHERE ID IN(SELECT ID FROM :TUs) AND
    CONTAINS("text", :currSentence);
     sentences[:id] := :currSentence;
     scores[:id] := :score;
     id := :id + 1;
        END FOR;
END FOR;
```

Figure 3.21: Co-occurrence N-Gram Spotting - Part 4

The formula, which was empirically found to perform the best, is $score = o*(1000 - |(ql - cl) + 1| \times 10)$ where o is the number of occurrences of the current candidate across the number identified TUVs, ql is the number of words within the query and cl is the number of words of the current translation candidate.

This formula ensures, that the score increases with growing number of co-occurrences, while decreasing with larger differences in length between the query and the current sentence. The two parallel arrays representing the translation candidates and their corresponding scores are then combined into a table representation (see Figure 3.22). The table is then sorted in the decreasing order of their scores.

```
res1 = UNNEST (:sentences, :scores) AS ("token", "score");
res = SELECT * FROM :res1 ORDER BY "score" DESC;
```

Figure 3.22: Co-occurrence N-Gram Spotting - Part 5

During the manual evaluation of the algorithm, it was found that the algorithm suffers problems, which are similar to the problems encountered in the simple single word co-occurrence spotting. The main issues are caused by the stop words. It is the case, that combinations of stop words would occur possibly across all of the identified sentences. N-Grams such as "of the" or "and the" would occur many times and their occurrences would beat the negative score produced by the length difference. Stop words can be a valid part of a multi-word query; filtering them out of the translation candidates was not a feasible solution. It was desirable to find means of identifying translation candidates, which are more relevant to the query.

Co-occurrence with Dictionary Spotting The problems with stop words in the n-gram translation spotting were partially solved with the use of dictionary techniques. The theory behind this approach was, that if it was possible to spot common keywords across sentences, it would make scoring of translation candidates more precise. For example if words “European” and “Commission” appeared in both the query and a given translation candidate, and the translation candidate occurs across a number of identified sentences, while the difference in length is not major, it is very likely that the given candidate is the right translation.

Since not having to include language specific mechanisms in translation spotting systems is a desirable property, the algorithm should not rely on language specific dictionaries. The single word translation spotting algorithms showed reasonable efficiency. It was therefore decided to use these single word procedures instead of importing language specific dictionaries.

This algorithm therefore uses a single word translation spotting procedure in order to translate words (namely keywords) into the target language. To perform the operation, another procedure was created (see Figure 3.23). It uses similar pattern to the one used in the word count procedure. A while loop is used to iterate over the words in the query. Additionally, for each word encountered, the procedure checks if it is a stop word. If it is not a stop word, the Stop word & Single Occurrence procedure is called, trying to translate the current word from the source to the target language. The top result of the single word spotting is then stored in an array containing the translated keywords. The array is converted into a table result set and returned to the output parameter of the procedure. Once the spotting procedure

```
while(:input <> '') DO
  toSpot := SUBSTR_BEFORE(:input, ' ');
        if(:toSpot = '')
        THEN
            toSpot := :input;
        END IF;

        SELECT count(*) INTO isStop FROM  "TRANSSPOTTING2"."sap.tcd.
    dissertation2.tables::STOP_WORDS" WHERE "word" = :toSpot;
        if(:isStop = 0) THEN
            call "TRANSSPOTTING2"."sap.tcd.dissertation2.procedures/
    spotSingleOccurenceSW"(:tgtLang, :srcLang, :toSpot,spot);
            SELECT COUNT(*) INTO numSpot FROM :spot;
            IF(:numSpot > 0) THEN
                SELECT TOP 1 "token" INTO spott FROM :spot;
                words[:cnt] := :spott;
                cnt := :cnt + 1;
            END IF;
        END IF;
        input := SUBSTR_AFTER(:input, ' ');

END WHILE;

res = UNNEST(:words) AS ("WORD");
```

Figure 3.23: Split Words Dictionary Procedure

obtained the translated words of the query, it calls the numOfWords procedure to determine the size of the query (see Figure 3.24). Similarly to all the other procedures, the relevant translation unit variant IDs are retrieved and the shortest sentence is selected. Nested for loops are used to generate the list of substrings of the selected shortest sentence. The scoring

```
-- Attempt to translate each word in the sentence
call "TRANSSPOTTING2"."sap.tcd.dissertation2.procedures/splitWords"(:
    inQuery, :targetLang, :srcLang, dict);
call "TRANSSPOTTING2"."sap.tcd.dissertation2.procedures::numOfWords"(:
    inQuery, :input_length);
```

Figure 3.24: Co-occurrence with Dictionary Spotting - Part 1

technique, which is performed on each of these translation candidates then differs slightly from the previous algorithm. Firstly, the stem of the word that the inner loops is currently looking at is retrieved from the translation analysis output table (see Figure 3.25). The stem is not related to any particular sentences in the text analysis table, instead the most commonly appearing stem for the given token is used. If a stem for the current token exists, the most

```
stem := :tokens[:j];
stem_res = SELECT TA_STEM, COUNT(*) AS STEMCNT FROM "TRANSSPOTTING2"."
    $TA_TUV_INDEX2" WHERE TA_TYPE != 'punctuation' AND TA_TOKEN LIKE :stem
     AND TA_LANGUAGE = lower(:targetLang) AND TA_STEM IS NOT NULL GROUP BY
     TA_STEM ORDER BY STEMCNT DESC;
```

Figure 3.25: Co-occurrence with Dictionary Spotting - Part 2

commonly appearing stem is converted into a local string variable. A query then examines the occurrences of the stem in the dictionary entries produced when translating the words of the input query (see Figure 3.26). If any matches are found, the score is incremented by one to ensure that each match is not counted more than once. This means the the current translation candidate includes a word with the same meaning as a word, which was spotted in the query. The algorithm also takes into account the number of occurrences of the cur-

```
SELECT COUNT(*) INTO matchCnt FROM :stem_res;
IF(matchCnt > 0) THEN
        SELECT TOP 1 TA_STEM INTO stem FROM :stem_res;
END IF;
SELECT COUNT(*) INTO matchCnt  FROM :dict WHERE WORD LIKE :stem OR WORD
    LIKE :tokens[:j];

IF(:matchCnt > 0) THEN
        sc := :sc + 1;
END IF;
```

Figure 3.26: Co-occurrence with Dictionary Spotting - Part 3

rent translation candidate across the identified set of translation unit variants. The count of matches of the translation candidate is then also stored in a local variable (see Figure 3.27). Once having all the components of the score, the sentences array is updated with the current translation candidate and the score is stored in the corresponding position of the scores array. Similarly to the previous algorithm, a number of scoring formulae were empirically evaluated by observing the precision of the algorithm based on a number of queries and the positions of the desirable translation in the result set. After careful consideration, score in this algorithm is being calculated based on the number of matched keywords, the number of

```
cooccur := 0;
IF((:j -:i)+1 >= 2) THEN
        SELECT COUNT(*) INTO cooccur FROM "TRANSSPOTTING2"."sap.tcd.
    dissertation2.tables::TRANSLATION_UNIT_VARIANT" WHERE ID IN(SELECT ID
    FROM :TUs) AND CONTAINS("text", :currSentence);
END IF;
```

Figure 3.27: Co-occurrence with Dictionary Spotting - Part 4

occurrences in the identified translation variants as well as the difference in length between the input query and the translation candidate.

The formula is: $score = m^2 + \sqrt(n) - (|(ql - cl)|)$ where m is the amount of matched keywords, n is the number of matched occurrences of the translation candidate across the identified TUVs, while the ql and cl are the query and translation candidate lengths respectively.

This ensures that the matched keywords have the highest weighting in the score, while also taking the number of occurrences and differences in length into account.

This algorithm was found to be the least efficient in terms of performance. This can be explained by the fact that the single word spotting procedures can be called as many times as there are words in the source query. The number of analytical operations for identifying co-occurrence across the corpus is dependent on the shortest identified TUV. The algorithm also delivers the highest precision in n-gram translation spotting compared to other proposed algorithms.

3.6 Summary

This chapter described the design and implementation of all components of the proposed translation spotting system. The whole system was first described along with high level implementation details. The design and implementation of each component was then described in more detail.

A brief suggestion on how a user interface could be built using different software paradigms was given along with a short description of the user interface built for the implemented system.

The design and implementation of the administrative layer was presented. No major problems or findings were observed during the development of the administrative layer. Its components are designed and built to be easily extensible to support different file formats and even different database engines. The test framework was designed to allow for automatic testing of the system and reporting of the results. The test framework implemented as part of this dissertation was used throughout the evaluation of the system.

The translation spotting logic was designed to use as much functionality provided by SAP HANA as possible. The text engine of the database provided features, which became crucial for the implementation of the translation spotting logic such as tokenization or stemming. Additionally, the translation spotting procedures were implemented using the expressive power of SQLScript, thus providing all of the application logic directly within the in-memory database. The translation spotting algorithms were implemented in two phases. First, a number of single word algorithms were proposed, which were later extended to support multi-word queries.

The single word algorithms were designed to be simple. The text analysis library output provides the tokens, which are linked to the unstructured text of the TUVs. No complex logic was therefore needed to implement the single word procedures. The results of each of the procedures were analysed and reflected on, which led to the development of more advanced algorithms. The Simple Co-occurrence Spotting has shown that stop words will pose a problem. The Single Occurrence Limit Spotting algorithm was therefore developed in an attempt to limit the effect of stop words on the analysis. The effect of stop words was successfully scaled down even though it did not solve the problem fully. Stop Words Co-occurrence Spotting was then designed to explicitly filter out stop words where possible. This yielded improved precision. The Stop Words & Single Occurrence Spotting was the last and most effective algorithm, which is a combination of the Single Occurrence Limit Spotting and the Stop Words Co-occurrence Spotting algorithms.

Using the single word algorithms was considered to perform word-by-word translations of multi word queries. It was decided that due to different sentence structures in each language, such an approach would not be feasible. Additionally, even though the precision of the single word algorithms was promising, it was not flawless. If used for every word of the sentence, the imprecisions would add up producing results, which might not make sense. It was decided to use an approach similar to the single word algorithms - attempt to identify co-occurrences of phrases. The simple Co-occurrence N-Gram Spotting was implemented, which used the length differences between the query and each translation candidate along with its co-occurrence across the identified sentences as the method for ranking the candidates. It was found that the focus was shifted more towards the candidates, which contained stop words. It would not be a solution to simply filter out stop words from the multi word translations just like the single word algorithms do. It was considered to use dictionaries in order to identify the most relevant candidates. The language specific dictionaries were however, not available and thus the Co-occurrence with Dictionary Spotting used the single word algorithms in order to identify candidates, which contained most keywords common in both the query and the translation candidate.

In conclusion, a useful framework for translation spotting systems was implemented using an in-memory database. A number of translation spotting algorithms were then considered for both single and multi word queries offering differing levels of precisions and execution times.

Chapter 4

Evaluation

4.1 Introduction

This chapter will walk the reader through the testing and evaluation of the implemented system. Section 4.1.1 will describe the functionality of the test framework. Section 4.1.2 will then provide the background information on the data that was used for the implementation and evaluation of the system. Section 4.1.3 will explain how the test cases used in the experiments were selected. The description of different tests is then given section 4.1.4.

All the different tests and their results are then presented and interpreted in section 4.2 under the categories of:

- **Extensibility** - how easily can the system be extended with further data?
- **Effectiveness** - how accurate are the translation candidates proposed by the system? Precision will be calculated for every algorithm implemented in terms of percentage of correct results.
- **Efficiency** - Can the system perform in real-time even when adding more data? The performance will be measured in terms of the time needed to perform the operation on a given data set size and to transfer the result set to the client.

Finally, the Discussion of the Results section (4.3) of this chapter will summarise findings and interpret them. Additionally, an attempt will be made to identify real world use cases, in which a system such as the one developed in this work could be used.

4.1.1 Test Framework

Test framework has been designed and implemented to allow the developers and administrators to perform automatic testing of the system, namely of its precision and performance on the current data set. As mentioned in the Design and Implementation chapter, the test framework is available as a part of the administrative subsystem.

The framework allows for testing of stored procedures in the system and as such, it can be used to evaluate each of the implemented translation spotting algorithms. Such test functionality can be defined through the implementation of tests. Each test is related to a single stored procedure with a number of parameters. A number of test definitions were therefore combined in test applications. Each application defines two tests, each of which evaluates the same stored procedure / translation spotting algorithm in translating text from one language to another and vice versa.

Each test then contains a list of test cases. Each test case contains a word, which represents the translation query and a number of acceptable answers. The test case class also includes the functionality for verifying an answer. This is used with the top result from the procedure. If the result is a correct translation of the query within the test case, the result is verified and accepted. The test case includes two ways of verifying a potential result. One is used in tests which evaluate single word procedures, while the other is used when testing the n-gram translation spotting procedure.

It is important to distinguish between the single word and the n-gram result verification. While the single word translation spotting procedures attempt to return the most likely translation candidate as the stem of the word, the n-gram procedures attempt to find the most likely candidate as it appears across the TUVs. In single word queries, it is very likely the case, that the translator is only looking for a specialised term either as a reminder or to produce a consistent translation with the most of other translations in the corpus. A qualified translator should therefore be able to change the word into a form, which makes its placement in the target sentence syntactically and grammatically correct. When it comes to n-gram queries, it could be the case that the returned phrase would be hard to understand if all of the words were changed to their stemmed version. It is therefore desirable that the single word queries are verified against a number of acceptable answers, where the result has to match (in case insensitive manner) exactly. For n-gram queries on the other hand, it is desirable to match an acceptable answer on a part of the result. In other words, the translation candidate should still be valid and useful to the translator if the acceptable answer is contained within the result, with a certain level of overhead. If there are additional prepositions or conjunctions before the start or after the end of the acceptable answer, the translator will be able to identify the required part without being overwhelmed by the whole paragraph of text. The acceptable overhead of a translation was defined to three words during the execution of tests as part of the evaluation of the system. Each Test is defined to be either single word, or n-gram based on the procedure it is evaluating. It then automatically makes a decision, which verification method is going to be called in each of the test cases.

The application / deployment specific test framework should consist of a number of test applications. Each of these applications contains a number of tests, each of which in turn contains a number of test cases. The test applications could potentially decide which tests will report their outcomes to different files, which could be used by automatic monitoring and reporting tools.

As a part of the evaluation of the system, a test application was defined for each of the implemented procedures.

4.1.2 Test Data

A data set used for the implementation of the system was produced by the European Commission. This data set contains translation memories of legal documents and proceedings transcripts, which were published and translated between 2004 and 2012. In 2011 statistics on the corpus [27], there were over 38 million translation units in the 22 official EU languages.

During the development and evaluation of the system, translation units in two languages were imported into the database - Czech and English. This decision was made as the author of the work is a native Czech speaker with business proficiency in English. The manual development of test cases between the two languages therefore did not require the use of dictionaries or translation tools, which could produce inconsistencies or mistakes in the test cases.

The development of the system initially started with the documents from year 2004, imported into the system and further data was imported additionally during the evaluation to ensure that the evaluation metrics are collected for data sets of different sizes.

4.1.3 Test Cases

There were four sets of test cases defined for the evaluation of the system. These include the 100 cases defined for single word queries from Czech to English, 100 cases for single word queries from English to Czech, 100 cases for n-gram queries from English to Czech and finally 100 cases for n-gram queries from Czech to English.

Single word queries were selected manually from existing EU documents. The words selected were mainly (though not exclusively) keywords, which could appear across documents such as the ones imported into the database. These single word test cases have the structure of the searched query, and a number of acceptable translations. An example of single word test cases from English to Czech are:

Search Query	**Acceptable Answers**
transfer	převod, převést
understanding	porozumění

Examples of single word test cases from Czech to English are:

Search Query	**Acceptable Answers**
PROHLÁŠENÍ	declaration
spravedlnost	justice

Some of the test cases are uppercase. Even though the matching mechanism is case insensitive, these words were copied directly from the documents and left the way they appeared in these documents. Such an approach could be taken if the system was used as a part of an automated translation mechanism. Certain acceptable answers also have a vertical bar character at the end. It was found during debugging of the test cases, that as a side effect of stemming in the text analysis library, some produced stems have “|” character appended to the end. This seems to be an issue with the implementation of the text analysis library of SAP HANA, as no formal mention about such feature was found in the documentation.

Based on the statistics collected from online concordancers [7], most of the searched queries are between 2 and 3 words in length. The n-gram test cases take this fact into account. Even though not all of the cases are between 2 and 3 words, most of them are. Similarly to single word queries, the n-gram queries were designed in such a way, as to contain mostly keyword phrases. It did not seem plausible that qualified translators would need to use translation spotting systems in order to identify common phrases made up purely out of stop words. Examples of English to Czech n-gram test cases are:

Search Query	**Acceptable Answers**
constitutional requirements	Ústavněprávní požadavky
European Economic Area	Evropský hospodářský prostor

Examples of n-gram Czech to English test cases are:

Search Query	**Acceptable Answers**
elektronických komunikačních sítích	electronic communication networks
ochraně osobních údajů	data protection

A full list of the test cases used during the evaluation of the system can be found in Appendices A to D of this dissertation.

4.1.4 Tests

A number of tests were defined and executed on the system containing different sizes of data. Each of these tests corresponded to testing a single feature or algorithm of the system, such as a test for the Simple Co-occurrence Spotting algorithm. The results of these tests are described and interpreted in the Experiments section (4.2).

The tests always contained the same test cases for both single word, or n-gram procedures, while testing the system with different portions of the data set imported into the system.

4.2 Experiments

The evaluation of the system was carried out in two phases (except for extensibility measurements). The initial evaluation has shown irregularities in the results, which were caused by migration to a different version of SAP HANA between the initial implementation and the first test executions. The translation spotting system was moved by SAP staff to a different physical server within SAP’s infrastructure due to maintenance reasons. While moving the system, the version of HANA was also updated. New features in the text analysis library in the new version of HANA caused a number of issues, which were observed during the phase

Measure / Year	2004	2005	2006	2007	2008	2009	2010	2011	2012
Time taken	2 776s	3 856s	5 591s	3 947s	4 566s	3 332s	2 968s	4 391s	8 040s
No. of documents	2 155	3 405	3 293	2 656	2 560	1 906	1 651	2 449	3 765
Number of TUs	195 179	333 421	451 809	320 286	430 619	276 173	274 794	322 377	538 949

Table 4.1: Extensibility of the system measurements

one evaluation. Minor changes were then made to the algorithms to minimise the effect that the new features have on the results. In phase two, the system was evaluated again with the implemented changes.

4.2.1 Extensibility

Extensibility of the system has been measured mainly as the amount of time required to import data into the system (see Table 4.1). The evaluations of the algorithms were carried out for a number of different data set sizes. The European Commission corpus has been ingested into the system in chunks of documents grouped by the years they were published in. During the ingestion of data, time taken for the ingestion along with a number of documents was collected.

It is important to note, that the ingestion happens in batch statements per document. The data has been ingested from a laptop at home connection speeds through SAP's corporate VPN (connections are routed from Ireland to the VPN proxy in Germany and back to Ireland). At the time of ingestion, the average round-trip time was measured to be approximately 100-150ms, this time is therefore included in time needed for the ingestion of every single document. It is expected that if ingestion was carried out geographically locally to the server, the execution would perform faster.

4.2.2 Phase One Evaluation

Phase one evaluation was the first official testing of the system. The test cases were executed based on two different sizes of the data set and the test results were analysed. Based on the analysis, it was found that the results show what appeared to be an error in the implementation, which has not previously been observed. The problem was analysed and minor changes in the implementation were made, leading to phase two of the system evaluation.

This section will describe the tests executed and the test results observed during the initial phase of the evaluation. It should be noted that all execution times presented in this section include the network latency produced by the communication through SAP's Germany based VPN, which at the time of the test execution was on average approximately 100ms.

Single Word Translation Spotting

Simple Co-occurrence Spotting The Simple Co-occurrence Spotting algorithm is the easiest and performance-wise the most efficient. The tests have shown that precision of the algorithm is not great and this algorithm would very likely not be very usable in real-world use

Test / Year	**2004**	**2004-2005**
Czech to English	200ms	320ms
English to Czech	220ms	390ms

Table 4.2: Phase One - Efficiency of Simple Co-occurrence Spotting

Test / Year	**2004**	**2004-2005**
Czech to English	1%	1%
English to Czech	4%	4%

Table 4.3: Phase One - Effectiveness of Simple Co-occurrence Spotting

cases.

Efficiency Table 4.2 represents the average time taken for the execution of a test case based on the size of the data set. These results show that the Simple Co-Occurrence algorithm performs in real-time (in sub-second times) and promises to be scalable. The scalability of this algorithm is promised by the simplicity of the algorithm, which meant that no imperative constructs needed to be used in its implementation. With the use of set based logic, SAP HANA is able to perform automatic optimisations of the queries and parallelise many of the computations.

Effectiveness Table 4.3 represents the effectiveness of the algorithm as a percentage of successfully spotted translations. The analysis of the test results showed a number of interesting facts, which led to development of more advanced techniques.

In the Czech to English test, only 1% of test cases yielded correct result. Looking at the log files, it became obvious that stop words do cause the low effectiveness of the algorithm - 98% of test cases returned the result “the”, while 1% returned “a” in both data sets.

The first notice of the aforementioned irregularities in the test results was shown in the results of the English to Czech test. Even though 4% effectiveness is slightly better than the one of the Czech to English test, most of the failed test cases returned the result of “,” and a small number of prepositions “v”. Prepositions would be expected in the result set, as these are stop words, which were already identified to produce poor results. The comma character was however never observed in the manual testing previously. This indicated that there may have been an error introduced into the implementation. At the stage of analysing the outputs of this experiment, this irregularity did not seem very relevant.

Single Occurrence Limit Spotting The Single Occurrence Limit Spotting algorithm is closely modelled on the Simple Co-occurrence Spotting algorithm, but adds extra logic in an attempt to minimise the effect that the stop words have on the effectiveness. This is done by limiting the score of an occurrence of a given translation candidate to one per a single translation unit. The logic behind this is that stop words are probably very often going to appear more than once in a sentence, while it is likely that a searched keyword might only

Test / Year	2004	2004-2005
Czech to English	230ms	390ms
English to Czech	230ms	440ms

Table 4.4: Phase One - Efficiency of Single Occurrence Limit Spotting

Test / Year	2004	2004-2005
Czech to English	19%	20%
English to Czech	49%	46%

Table 4.5: Phase One - Effectiveness of Single Occurrence Limit Spotting

appear once. It would then be case that the unwanted stop words in form of prepositions or articles would be weighted more than the required phrase. Limiting their co-occurrence per translation unit is therefore an attempt to limit the scope of the problem.

Efficiency As depicted in Table 4.4, the performance of the Single Occurrence Limit Spotting algorithm is still very promising. It is slightly less performant than the Simple Co-occurrence algorithm, which is due to the extra filtering of duplicate entries. This algorithm is also promising to be well scalable due to the lack of imperative features in its implementation.

Effectiveness The percentage precision of this algorithm (see Table 4.5) is showing improvements compared to the previous algorithm. Both Czech to English and English to Czech tests increased in precision. English to Czech precision seems to be significantly better than the one of the Czech to English test. When analysing the results of the tests, the failed test cases indicate what the reason behind this trend is. Most of the English to Czech failed test cases still yielded the result of “the”. The article “the” is very common in English, in fact more common than any single stop word used in Czech. Based on the analytical tools provided by SAP HANA, the word “the” is the most common (non punctuation) word in the corpora of 2004-2005 documents, accounting for 6.9% of all words. The Czech to English failed results are still mainly punctuation.

Stop Words Co-occurrence Spotting The Stop Words Co-occurrence Spotting was developed as a solution to all of the stop words problems, which were mentioned in previous sections. The algorithm uses language specific stop word lists, which were imported to the system. The stop words are then filtered out based on their occurrence in the stop words lists. To solve the irregularities observed with the punctuations in the results, the most common punctuation characters were added to the stop word lists.

Efficiency The performance of the Stop Words Co-occurrence Spotting algorithm (see Table 4.6) is again slightly worse than performance achieved by the less complex algorithms. The stop words filtering adds extra computational overhead. Additionally, a small amount of imperative logic is used to determine whether the query is a stop word itself in the source

Test / Year	2004	2004-2005
Czech to English	370ms	700ms
English to Czech	400ms	760ms

Table 4.6: Phase One - Efficiency of Stop Words Co-occurrence Spotting

Test / Year	2004	2004-2005
Czech to English	73%	77%
English to Czech	78%	84%

Table 4.7: Phase One - Effectiveness of Stop Words Co-occurrence Spotting

language. In a case where it is, it would not be desirable to filter out stop words from the translation candidates, since one of them could potentially be the correct translation. In the case, where the query is a stop word, the Single Occurrence Spotting algorithm is used instead. This imperative feature is negatively reflected on the scalability of the algorithm. The increase in execution times based on the size of the data set is more drastic than in the previous tests.

Effectiveness The effectiveness of this algorithm (see Table 4.7) was expected to beat the precisions observed in other single word algorithms since the main problem previously observed (i.e. stop words) is being explicitly filtered out. That indeed turned out to be the case. The effectiveness is similar in translating from Czech to English and English to Czech without such drastic gaps that were previously observed.

This algorithm also shows a visible improvement in the effectiveness with increased size of the data set. An improvement in effectiveness with increased corpus size was an expected property in such a translation spotting system as there should be more co-occurrences of the matched as a pattern across the identified translation units.

Stop Words & Single Occurrence Spotting Once observing an improvement in quality of spotted translations in both the Single Occurrence Limit Spotting and the Stop Words Co-occurrence Spotting algorithms, it seemed feasible to combine the functionality of both algorithms in an attempt to increase the effectiveness even further.

Efficiency Since both approaches of limiting the occurrences and filtering out the stop words added an extra complexity into the algorithm, it was to be expected that the performance of the combination algorithm (see Table 4.8) will also be hit. With two years of imported documents in the system, the execution of this algorithm exceeds 1 second.

Test / Year	2004	2004-2005
Czech to English	530ms	940ms
English to Czech	630ms	1.1s

Table 4.8: Phase One - Efficiency of Stop Words & Single Occurrence Spotting

Test / Year	**2004**	**2004-2005**
Czech to English	88%	87%
English to Czech	87%	88%

Table 4.9: Phase One - Effectiveness of Stop Words & Single Occurrence Spotting

Effectiveness Despite the lower performance of this algorithm, its precision (see Table 4.9) is the best one yet observed. There are no significant differences between translation from one language to the other or vice versa and there seems to be no major difference in the effectiveness with different sizes of the data set. This could mean potential improvements in performance by only using subset of available data without the loss of precision.

N-Gram Translation Spotting Algorithms

Even though the observed irregularities in terms of the punctuation characters in the results were successfully hidden the single word algorithms by marking them as stop words, the problem has appeared again when analysing the output of the multi-word algorithms. As described in the Design & Implementation chapter, filtering stop words out of the multi-word results did not seem feasible. Filtering out articles or prepositions out of the running text could make the text hard to understand and could not bring any value to the translator.

The scoring mechanisms for the n-gram algorithms work on a number of techniques, mainly the difference in lengths between the translation candidate and the source query, and the number of co-occurrences matched across the identified translation units. If a punctuation character such as a comma was counted as a word, this could negatively affect the scoring of the translation candidate. Similarly, if brackets were identified as a token (or a word), the possible substrings would contain “ (“ with additional text appended or prepended. Since a bracketed expression in the text could be local to only a small number of translation units, the co-occurrence count would also be affected. Thus, the n-gram algorithms with the presence of punctuation tokens in the output of the text analysis were problematic.

It was identified, that there was no error in the implementation of the algorithms as such. Instead, the database instance was moved from one physical machine to another, and while doing so the version of the database was updated. The punctuation tokens were a feature of the new version of the text analysis library, which had not been encountered during the initial development stages.

The following sections describe the test results with the presence of punctuation marks in the text analysis output before a fix was implemented in the phase two of the evaluation.

Co-occurrence N-Gram Spotting Translating the n-gram queries word by word using the implemented single word translation spotting algorithms would very likely not result in a usable solution. Different languages have different word orderings and sentence structuring. The sentences, which would be produced by simply translating a multi-word query word by word would very likely be syntactically and grammatically incorrect. Furthermore, some

Test / Year	**2004**	**2004-2005**
Czech to English	6.3s	22s
English to Czech	8.7s	26s

Table 4.10: Phase One - Efficiency of Co-occurrence N-Gram Spotting

Test / Year	**2004**	**2004-2005**
Czech to English	19%	25%
English to Czech	8%	4%

Table 4.11: Phase One - Effectiveness of Co-occurrence N-Gram Spotting

languages use declension to change the form of words to indicate plurality and gender, while others do not. The test cases used in the evaluation of this project would be a perfect example - Czech uses declension, while English does not.

The Co-occurrence N-Gram Spotting algorithm is therefore an n-gram equivalent of the single word Simple Co-occurrence Spotting algorithm. It attempts to identify the whole phrases from the translated sentences, which correspond to translations of the whole query. As such, it is promising that the word orderings and declension of these results could be correct. The implementation is complex as it requires to generate all possible substrings out of an identified translation units - i.e. all of the possible translation candidates. Once the set of all possible answers is generated, the answers are then scored based on the differences in length between the query and translation candidate as well as the degree of co-occurrence across the identified translation units.

Efficiency Similarly to some of the more advanced single word algorithms, this algorithm makes use of the available imperative constructs within the database engine. During the initial development, it was seen as the only option to generate all of the possible substrings and therefore a number of for and while loops are used in the implementation. This fact makes it impossible for the optimisation tools to parallelise the computation. Analysing the results, it makes it clear that the n-gram spotting will not be possible to carry out in times that could be considered real-time.

Interesting observation in the results is, that the execution times largely vary based on the query. The execution times range from sub-second times for some queries, while reaching the order of minutes for others. The positive fact is, that most of the queries are executed reasonably fast, while small number of queries take extremely long, which affects the statistical measures such as the average figures listed in Table 4.10.

Such performance differences are caused by the optimisation of always generating the possible translation candidates from sentences, which contain the least number of words. The shortest sentences will also yield the smallest possible number of its substrings. The smaller the number is, the smaller computational overhead related to both generating the translation candidates as well as calculating scores for them.

Test / Year	**2004**	**2004-2005**
Czech to English	11s	25s
English to Czech	15s	33s

Table 4.12: Phase One - Efficiency of Co-occurrence with Dictionary Spotting

Effectiveness In order to verify a result as a correct translation, the full translation of the query needs to be included in the result. There is a small allowable overhead in terms of additional word (for this evaluation 3), which is tolerated to allow for cases where there is a stop word returned at the beginning or the end of the translation. Such a small number of additional words should not overwhelm the translator, who should be able to quickly identify the relevant part of the result.

In the failed test cases in this experiment, it was found that often a part of the actual translation is returned with a missing word. In cases, where the translator is only using the tool for a reminder, such results could also be useful.

In terms of the result verification mechanism developed in the test framework, the effectiveness results of this algorithm (see Table 4.11) are not particularly good.

In addition to the observation regarding the partially present translations, the punctuation would further distort the translation candidates. The top candidate for a two word query, could contain the second word and a full stop instead of the two translated words.

Co-occurrence with Dictionary Spotting The Co-occurrence with Dictionary Spotting algorithm was implemented after seeing the poor effectiveness results of the Co-occurrence N-Gram Spotting algorithm. As previously mentioned, it was observed that often the required translation was partially present in the top translation candidate. By analysing the trends in these results, it became apparent that taking the length differences between the query and the translation candidate into account is a good idea. It often shifts the focus of the translation candidate towards one side (mostly the side which contained a number of stop words). This meant, that the translation candidate, which contains part of the translation along a number of stop words at the beginning or the end of the string would gain more focus in the result set.

The use of dictionary (in this case the single word spotting procedures) to identify keywords in both the query and their corresponding translations in the translation candidate seemed to promise that the focus of the translation candidate would shift towards the right direction.

Efficiency In addition to performing massive computation of generating all of the possible substrings and then ranking them as translation candidates, this algorithm performs an analysis on the query. It tokenizes the query and on each of the words, it calls the Stop Words & Single Occurrence Spotting algorithm in an attempt to identify the translations of keywords in the query.

Test / Year	**2004**	**2004-2005**
Czech to English	69%	58%
English to Czech	62%	38%

Table 4.13: Phase One - Effectiveness of Co-occurrence with Dictionary Spotting

The performance of this version of the n-gram translation spotting (see Table 4.12) is therefore not very impressive and does not seem to scale very well.

Similarly to the Co-occurrence N-Gram Spotting algorithm, the execution times for different queries vary greatly based on the shortest sentence containing the query.

Effectiveness Based on the metrics presented in Table 4.13, the Co-occurrence with Dictionary Spotting algorithm improved the effectiveness of the multi-word translation spotting. Based on the manual testing during the development, it was expected that the effectiveness would be better.

Upon looking closer at the test results and the failed test cases, it became apparent that the punctuation tokens produced by the new version of the text analysis library posed a problem which needed to be addressed.

Phase One Summary

The analysis of results from phase one of the evaluation showed a number of interesting observations.

Firstly, the punctuation marks appearing as translation candidates across all single word procedures and as words in multi word results indicated an error in the implementation. After closer analysis it turned out to be a side effect produced by a new feature in the text analysis library of SAP HANA. The effects this had on the results of the tests were considered too radical and the decision was made to implement a fix for this issue.

It is interesting to see the trend of increasing execution times with the increasingly complex logic of the single word procedures. The times almost tripled in the Stop Words & Single Occurrence Spotting compared to the Simple Co-occurrence Spotting. Both techniques, the filtering of stop words and limiting of the translation candidate occurrence to one per translation unit add extra computation to the algorithm. The opposite trend can be observed when it comes to effectiveness. With the additional logic, the effectiveness of the Stop Words & Single Occurrence Spotting reach to up 88% compared to the 4% effectiveness of the Simple Co-occurrence Spotting algorithm.

Another interesting observation made in the test results of the single word algorithms is the difference in effectiveness between the Czech to English and English to Czech tests of the Single Occurrence Limit Spotting. This algorithm is attempting to reduce the effect that the stop words have on the results, but does not completely eliminate it. There is a 30% difference in effectiveness between the two tests. After analysing the failed test cases it became obvious that the definite article "the" in English is the problem. The token "the" was responsible for vast majority of the failed test cases, while the wrong candidates in Czech were more equally distributed among different stop words. The article "the" is the most often used word in the English texts, thus having such a serious effect on the effectiveness of this algorithm.

The effectiveness has improved greatly in the Co-occurrence with Dictionary Spotting algorithm compared to the Co-occurrence N-Gram Spotting, improving from 25% to 69%. It was expected to observe massive differences in performance of the multi word algorithms compared to that of the single word algorithms. The multi word algorithms produced average execution times of up to 33s. It should be noted that individual execution times vary greatly. Some queries were observed to perform in sub-second intervals, while others would take very long to execute. This can be explained by the fact that the translation candidates are generated out of the shortest identified sentence. For certain queries, the shortest sentence can still be very long. This produces a computational overhead of generating all possible substrings and additional overhead of calculating the degree of co-occurrence for each.

4.2.3 Phase Two Evaluation

After executing tests in the phase one of the evaluation, it became apparent that it will not be sufficient to simply hide the problems produced by the punctuation tokens in the output of the text analysis library. A fix was therefore implemented across all of the implemented procedures. In every procedure, when retrieving any data from the output table of the text analysis library, the algorithm filters out all tokens, which are of type punctuation. This means that for the purposes of the pattern co-occurrence analysis, such entries will simply be ignored. The solution therefore simulates the functionality, which was provided by the text analysis library before the database engine was upgraded.

This section will present the results of the same tests as those presented in phase one, except that phase two was executed after implementing the aforementioned fix.

Similarly to phase one evaluation, it should be noted that the reported times of execution include the network latency produced by using the SAP's Germany based VPN, which at the time of execution of the tests were around 100-150ms.

Single Word Translation Spotting

The single word algorithms have been evaluated extensively during the phase two evaluation. Most of these algorithms proved to scale relatively well and therefore it was possible to run the tests on nine different sizes of the data set.

Simple Co-occurrence Spotting

Efficiency It was previously mentioned that the Simple Co-occurrence Spotting algorithm out of all of the implemented algorithms has the greatest potential for scalability due

Test / Year	2004	’04-’05	’04-’06	’04-’07	’04-’08	’04-’09	’04-’10	’04-’11	’04-’12
Czech to English	230ms	390ms	530ms	530ms	630ms	690ms	770ms	840ms	1.04s
English to Czech	260ms	430ms	600ms	610ms	740ms	820ms	920ms	1.03s	1.22s

Table 4.14: Phase Two - Efficiency of Simple Co-occurrence Spotting

Test / Year	2004	’04-’05	’04-’06	’04-’07	’04-’08	’04-’09	’04-’10	’04-’11	’04-’12
Czech to English	1%	1%	1%	1%	1%	1%	1%	1%	1%
English to Czech	20%	20%	18%	17%	18%	20%	22%	22%	20%

Table 4.15: Phase Two - Effectiveness of Simple Co-occurrence Spotting

to its simplicity. The performance measurements presented in Table 4.14 prove that it really scales well even while performing analysis on very large data sets, which contain whole years of documents published by European Commission, the algorithm is still able to remain subsecond execution times (in most cases).

Effectiveness It is important to note the difference in effectiveness of the English to Czech queries between the results of phase one and phase two (see Table 4.15) of the evaluation. The effectiveness achieved in the previous test was up to 4%, while the same algorithm on the same data set size was able to produce up to 22% effectiveness after implementing the punctuation token type filter into the algorithm.

An interesting fact is, that similar increase in efficiency is not observed in the Czech to English test cases. When analysing the distribution of tokens in the English texts, it was found that the token “the” appears more often than the most frequent punctuation mark. The stop words in Czech texts were more equally distributed.

For Czech to English tests this algorithm does not bring much value even after fixing the issue with the punctuation token types.

Single Occurrence Limit Spotting

Efficiency The Single Occurrence Limit Spotting algorithm performs slightly worse than the Simple Co-Occurrence Spotting. The measures collected over a variety of data set sizes (see Table 4.16) show, that the scalability makes the algorithm usable even with massive amounts of data imported into the system.

This algorithm is still considered to scale well and perform the analysis in real-time.

Effectiveness Similar observation regarding the distribution of stop words is apparent in the Single Occurrence Limit Spotting algorithm. While the English to Czech cases yield precision of over 60%, the Czech to English cases remain at around 20% (see Table 4.17).

Test / Year	2004	’04-’05	’04-’06	’04-’07	’04-’08	’04-’09	’04-’10	’04-’11	’04-’12
Czech to English	260ms	410ms	570ms	560ms	680ms	750ms	830ms	910ms	1.12s
English to Czech	280ms	460ms	660ms	680ms	820ms	880ms	1.0s	1.08s	1.34s

Table 4.16: Phase Two - Efficiency of Single Occurrence Limit Spotting

Test / Year	2004	'04-'05	'04-'06	'04-'07	'04-'08	'04-'09	'04-'10	'04-'11	'04-'12
Czech to English	18%	24%	23%	23%	25%	22%	21%	25%	25%
English to Czech	63%	67%	62%	62%	64%	65%	64%	64%	65%

Table 4.17: Phase Two - Effectiveness of Single Occurrence Limit Spotting

Test / Year	2004	'04-'05	'04-'06	'04-'07	'04-'08	'04-'09	'04-'10	'04-'11	'04-'12
Czech to English	400ms	720ms	1.15s	1.45s	1.82s	2.07s	2.4s	6.37s	3.32s
English to Czech	420ms	780ms	1.27s	1.59s	2.0s	2.27s	2.64s	2.96s	3.65s

Table 4.18: Phase Two - Efficiency of Stop Words Co-occurrence Spotting

Upon analysing the failed test cases, it became obvious that the article "the" is still causing the issue, while in the English to Czech there were a number of prepositions, which happen to appear often in Czech texts.

The English to Czech precision has increased from values around 45% to values around 65% after filtering out the punctuation characters from the tokens. This improvement moves this algorithm slightly closer to bringing value to a translator.

Stop Words Co-occurrence Spotting

Efficiency The efficiency of the Stop Words Co-occurrence algorithm (see Table 4.18) is again slightly worse than the efficiency of the Single Occurrence Limit Spotting algorithm.

The execution times also seem to grow more rapidly with the increasing data set. This could indicate than with much larger corpora, this algorithm could pose performance problems.

In the metrics collected in this evaluation, the times of execution, which are in the order of seconds rather than milliseconds are still acceptable and should be usable for development of applications, which would use such services.

The fact that this algorithm scales slightly worse than the previous ones can be explained by the additional computational logic, which filters out stop words, but also the imperative logic which dynamically chooses the algorithm to be executed based on the fact whether the query is a stop word itself.

Interesting value was observed in the Czech to English test under the years of documents between 2004 and 2011. 6.37s as the average execution time of a test case seems to be an outlier and does not fit well into the other observed measurements. This could be due to temporary network problems (such as a lossy connection) or possibly due an internal administrative task happening within the database engine (such as delta store merge).

Test / Year	2004	'04-'05	'04-'06	'04-'07	'04-'08	'04-'09	'04-'10	'04-'11	'04-'12
Czech to English	72%	78%	86%	86%	88%	89%	87%	87%	85%
English to Czech	78%	84%	86%	86%	87%	83%	83%	82%	81%

Table 4.19: Phase Two - Effectiveness of Stop Words Co-occurrence Spotting

Test / Year	2004	’04-’05	’04-’06	’04-’07	’04-’08	’04-’09	’04-’10	’04-’11	’04-’12
Czech to English	550ms	980ms	1.48s	1.68s	2.26s	2.54s	2.66s	2.98s	3.52s
English to Czech	630ms	1.16s	1.75s	2.05s	2.72s	3.05s	3.23s	3.6s	4.27s

Table 4.20: Phase Two - Efficiency of Stop Words & Single Occurrence Spotting

Test / Year	2004	’04-’05	’04-’06	’04-’07	’04-’08	’04-’09	’04-’10	’04-’11	’04-’12
Czech to English	88%	87%	92%	92%	93%	93%	94%	94%	93%
English to Czech	85%	87%	87%	88%	91%	91%	91%	91%	91%

Table 4.21: Phase Two - Effectiveness of Stop Words & Single Occurrence Spotting

Effectiveness Despite the lower performance of this algorithm, the effectiveness (see Table 4.19) reaches the levels, which could be imagined to be useful in computer assisted translation systems. The precision levels reach similar qualities both ways of translations (due to filtering out “the”) and these values reach up to 89% success rate.

Stop Words & Single Occurrence Spotting

Efficiency The efficiency of the Stop Words & Single Occurrence Spotting algorithm (see Table 4.20) is similar to the efficiency achieved by the Stop Words filtering algorithm. The logic for dynamic selection of algorithm based on the query does seem to be the reason for slightly decreased scalability of the algorithm.

It should be noted that despite the metrics showing worse performance, the test executions on this algorithms seemed fast enough for a human perception and should be acceptable waiting times for a response of an application, which is communicating with a remote system.

Effectiveness The effectiveness of the Stop Words & Single Occurrence Spotting algorithm was identified as the best so far observed even in the phase one of the evaluation. The measures presented in Table 4.21 do support that statement. The effectiveness does seem to be very stable over the varying sizes of the data set and perform very well even with small quantities of data.

Additionally, the algorithm’s precision seems to be growing with the increasing size of the data set, which was a property expected to be observed in such a type of translation spotting system.

N-Gram Translation Spotting Algorithms

Even though nine different data set sizes were imported into the system for evaluation of the algorithms, the N-Gram translation spotting techniques have only been evaluated under four different sizes of the data.

Based on the design and implementation of these algorithms, which required the use of many imperative constructs, it was expected that the N-Gram spotting algorithms would not

Test / Year	**2004**	**2004-2005**	**2004-2006**	**2004-2007**
Czech to English	6.75s	18s	36s	48s
English to Czech	7.8s	20s	42s	68s

Table 4.22: Phase Two - Efficiency of Co-occurrence N-Gram Spotting

Test / Year	**2004**	**2004-2005**	**2004-2006**	**2004-2007**
Czech to English	22%	29%	24%	19%
English to Czech	11%	8%	8%	8%

Table 4.23: Phase Two - Effectiveness of Co-occurrence N-Gram Spotting

scale very well. This expectation was confirmed when the test results were analysed in the phase one evaluation.

During the execution of tests on increasing size of ingested data on the N-Gram algorithms, it became apparent that the execution of such a number of test cases with decreasing performance is becoming infeasible. The N-Gram translation spotting algorithms have therefore only been evaluated on four different sizes of the ingested data.

Co-occurrence N-Gram Spotting

Efficiency The poor performance of the Co-occurrence N-Gram Spotting algorithm (see Table 4.22) can be associated with the massive computational tasks which are being executed. Another important reason for the poor performance is the lack of parallelism due to the methods employed in the implementation. The implementation uses large number of imperative constructs in SAP HANA, which prevent the optimisation mechanisms to automatically parallelise the computation.

Slightly positive observation regarding the performance of the N-Gram algorithms is that the increase in performance could be directly related to the increase of the ingested data size. The number of translation unit has doubled between 2004 and 2005, it almost doubled between 2005 and 2006 and then increased by factor of approximately 1.5 between 2006 and 2007. Increases in execution time of the algorithm are of similar factors, which means that the algorithm performs under a linear time complexity.

Effectiveness The effectiveness of this algorithm has been identified to be poor in the phase one of the evaluation. The precision measures presented in Table 4.23 are similar to those identified in the phase one.

It has been found that many of the results failed because of not containing the required translation fully. In general, the focus of the result candidate would shift towards the translation candidates which contained more stop words and would be more likely to generate higher score due to co-occurrence.

Even though the precision results are not very promising, the manual inspection of the failed test cases has shown, that most of the returned results could be useful to the translator to a certain extent.

Test / Year	2004	2004-2005	2004-2006	2004-2007
Czech to English	11s	21s	37s	50s
English to Czech	16.9s	26s	50s	74s

Table 4.24: Phase Two - Efficiency of Co-occurrence with Dictionary Spotting

Test / Year	2004	2004-2005	2004-2006	2004-2007
Czech to English	74%	70%	70%	70%
English to Czech	76%	75%	71%	69%

Table 4.25: Phase Two - Effectiveness of Co-occurrence with Dictionary Spotting

Co-occurrence with Dictionary Spotting

Efficiency The Co-occurrence with Dictionary Spotting uses the Stop Word & Single Occurrence single word algorithm in addition to all of the imperative logic that was used in the Co-occurrence N-Gram Spotting algorithm. It would therefore be expected that the performance of this algorithm (see Table 4.24) would be slightly worse. This is indeed the case.

Similarly to the Co-occurrence N-Gram Spotting algorithm, based on the observations of the performance and the size of the ingested data, it was found that this algorithm performs in linear time complexity with respect to data size.

Effectiveness An interesting fact about the effectiveness (see Table 4.25) is, that it does not seem to change significantly with the changing scale of the ingested data. The scoring mechanism takes three measures into account, namely the difference of length between the translation candidate and the query, the number of keywords matched between them and the degree of co-occurrence of the translation candidate in the identified TUs.

The co-occurrence factor is therefore only a part of the scoring mechanism. Additionally, the translation candidates are all generated out of the single shortest sentence. This ensures that the computational overhead of generating the substrings and subsequently calculating ranks for them is kept to a minimum. The fact that all translation candidates are generated from a single sentence means, that the quality of the translations are always dictated by the quality of the translation of the shortest sentence. If it is the case that the shortest sentence does not translate literally into the target language, it might be impossible to find the perfect translation of the query. One way in which ingesting further data could help the effectiveness is if the new shortest sentence is part of the newly ingested data and its translation is of a higher quality. Similarly, it can be the case that the newly added translation of the shortest sentence could be of bad quality, resulting in a poorer result. Such a trend can be observed in the precision of the algorithm between the test on documents published 2004-2005 and 2004-2006. The documents published in 2006 contained shorter sentences, which contained some of the queries. It seems that the translations of these new sentences might not contain the literal translation of the query and this results into the drop of effectiveness.

4.3 Discussion of the Results

A number of algorithms were implemented and empirically evaluated for both the single word and n-gram queries. The results were analysed and an attempt was made to interpret the messages they portray.

4.3.1 Single Word Algorithms

Stop Words

Stop words were found to pose major issues when attempting to perform co-occurrence analysis on large unstructured texts. Both ingested languages - English and Czech had large numbers of common stop words, which kept affecting the results of the analyses.

In the Simple Co-occurrence Spotting and Single occurrence Limit algorithms, the results showed a clear gap between the effectiveness of translating from Czech to English and English to Czech. When analysing the test results, it was found that in all cases, the English article "the" would make it to the top of the results as most frequently occurring word. Many of the failed cases in the translations from English to Czech returned the conjunction "a" (and).

The analytical tools in SAP HANA were then used to verify, that these words really occur so often. It was found that "the" was indeed the most occurring non-punctuation token and accounts for 7.1% of all tokens in the ingested data. This explains why the algorithms, which do not filter out stop words perform poorly on the translations from Czech to English.

Similar trend was observed in translations from English to Czech before limiting the punctuation characters from the queries retrieving data from the text analysis output table. In Czech, the comma character was acting as the most frequently occurring stop word. A major improvement of the effectiveness was observed in such translations after implementing the fix to remove the punctuation tokens from the analysis.

Effectiveness with Respect to Data Size

During the design and the implementation stage, it was expected that the effectiveness of the algorithms would increase with the increasing size of the ingested data. The results do not offer any conclusive proof that such a property would hold.

In the single word spotting algorithms, which do not filter out stop words, the effectiveness rises and drops when importing different data into the system. The different stop words distributions in these chunks of the corpus could play a role. It was not determined precisely why there is no significant difference in efficiency with growing size of the data.

In the stop words filtering algorithms there is some overall growth in the effectiveness observed with the growing size of the data set. The precision of the Stop Words Co-occurrence spotting algorithm for example grows from 72% and 78% (for Czech to English and English to Czech respectively) to 89% and 83% with 6 years of documents ingested into the system.

The effectiveness then drops again to 85% and 81% as additional 3 years of documents were ingested. Even though this growth is slightly more significant, the tests should be executed on larger samples of translations to prove a statistically significant difference. It is possible that a number of test cases which kept failing in all of the algorithms were poorly chosen. Among a larger number of executed tests these test cases' effects would not be as significant and a real improvement of the precision with growing data size could be observed.

Scalability

The scalability of the different algorithms varies greatly. This is due to the differences in complexity and computational overhead in these algorithms. Some of the implementations required the use of imperative rather than set based logic within SQLScript, which meant the database engine might not be able to parallelise the computations performed on the data well enough or not at all.

The size of the data has grown from 195 179 TUs in 2004 to 3 143 607 TUs with the years of 2004 to 2012 ingested. That is approximately 16 times larger data set. The simple algorithms have proven to scale quite well - the execution time needed for the Simple Co-occurrence Spotting algorithm for example has risen just over 4 times despite executing on 16 times larger data set. The execution time of the Stop Words & Single Occurrence Spotting algorithm still only increases 6 times despite the 16 fold increase of data size.

This proves than SAP HANA is indeed able to parallelise some of the operations despite the imperative logic. It is not entirely known how optimisations techniques of SAP HANA work internally. Based on the analysis of the results, it seems that the database engine is able to optimise certain parts of a procedure even though it contains parts, which are using imperative constructs.

Potential Uses

The proposed algorithms produced different levels of precision and performance. This suggests, that based on the use cases, some of these algorithms could be used in different environments, offering more or less value to the users.

The Simple Co-occurrence Spotting and Single Occurrence Limit Spotting algorithms were implemented mainly as a proof of concept algorithms and led into developing further and more advanced algorithms. Their efficiency is however very promising. While these algorithms may not bring much value in computer assisted translation software packages, let alone automatic machine translations, their outputs could be used for analyses of existing translations. Their outputs have led into interesting insights into distributions of words between Czech and English. It was interesting to observe that the very same algorithm can yield up to 20% difference in effectiveness between performing translation spotting from one language to the other and vice versa. It is not entirely clear how such insights could help in other research, but if there was any significance in the results, these algorithms were clearly able to report it.

The results obtained from the Stop Words Co-occurrence Spotting and Stop Words & Single Occurrence Limit Spotting algorithms were more promising towards the use in actual computer assisted translation or translation memory systems. One can imagine how effectiveness of up to 94% could bring certain amount of value to the translators. Despite the limitation of only being able to translate single words queries, these algorithms could be used as dictionaries or as thesaurus software. These algorithms could also be used in more advanced pieces of software, just as they were used in the N-Gram Co-occurrence with Dictionary algorithm in this dissertation.

4.3.2 Multi-word Algorithms

The main difference between the single world and multi world translation spotting algorithms is the technique used in the implementation. The tokens representing single words are generated and provided as the output of the text analysis library. The single word algorithms then only have to rank the translation candidates. The translation candidates however had to be generated programmatically in the multi word algorithms and only then ranked. This resembles the NP complete [12] problems, where it may be fast to verify a solution, but there is no simple way to generate it in the first place. These problems then might have to generate all of the possible solutions and use the verification techniques to find the right one.

It was considered to use the single word algorithms to translate the multi word queries word by word. It was decided that such an algorithm would not provide much value to the translator. Firstly, there are different sentence structures in different languages, meaning that the resulting translation may not make much sense. Secondly, certain languages use linguistic techniques such as declension, shaping the words to express plurality or gender. Finally, the single word algorithms do not offer perfect effectiveness. Even a small percentage of imprecision would add up when using it to translate a number of words in a sentence.

Effectiveness Based on the Shortest Sentence

Even though certain values did indicate the property of increasing effectiveness with increasing size of the ingested data in the single word results, this is apparently not the case in the multi-word algorithms. If anything, the effectiveness values seemed to be dropping with increasing size of the data in n-gram queries. The fact that generating all of the possible substrings from all of the identified sentences was infeasible even in the world of in-memory computing led to a different approach. Only one sentence was to be selected, out of which the substrings were generated and subsequently ranked based on the number of measures. This sentence could be selected randomly, as there are no clear indicators of which one could contain the most relevant translation or translation of the highest quality. Since the heavy use of imperative logic meant poor performance and scalability of these algorithms, it was found that selecting the shortest sentence offers an optimisation. This means that the quality of the translation candidates produced by these n-gram algorithms is affected by the quality of the translation of the shortest sentence. If the translated sentence does not contain the literal translation of the query (possibly due to context of the sentence), the translation candidates generated from it will not contain the literal translation either. The effectiveness of this algorithm is therefore very unlikely to grow with the increasing size of the ingested data.

Performance

The fact that the multi word algorithms are similar to known NP complete algorithms does indicate that it will not be computationally efficient. The verification of a potential solution is done in a polynomial time. The occurrence of translation candidate would be counted across all identified sentences. The time needed for the verification is therefore linear with respect to the number of sentences in which the query appears. If it was possible to parallelise the solution generation well enough, the algorithm would have potential to scale much better.

Parallelisation was identified to be one of the common features of modern in-memory databases. The use of imperative constructs within the stored procedures limits the automatic optimisation, which could be done in any in-memory database. This means that the generation of potential results is very likely performed in sequential manner.

A number of optimisation techniques could be attempted such as limiting the size of potential solutions set. This could be done by analysing the position of the query within the selected sentences and only generating translation candidates from the similar position.

The main optimisation would be to rewrite the algorithm in such a way as to limit the use of imperative constructs in the multi word procedures. This would allow for automatic parallelisation of the solution generation.

Potential Uses

In the Design & Implementation chapter, it was suggested that a system like the one implemented could provide a user interface designed as a web application. The execution time measures of the multi-word algorithms were not very promising due to the poor scalability of their implementations. The measures obtained with the limited size of the data such as one or two years of documents could indicate that there could be a realistic real-world use even for these algorithms. Performing the analysis on larger sizes of the data does not yield any visible improvements. It could therefore be the case that only a subset of the data could be selected to perform the n-gram analysis on. Such an optimisation could bring the execution times into the order of seconds. It could be reasonable to make the user wait for a couple of seconds. The translation system could even retrieve a number of matching translations just as the online concordancers do and at the same time, asynchronously perform the translation spotting analysis. The results of such analysis could then be used for annotating or highlighting segments of the already retrieved texts. Additionally, the choice of the algorithm to be executed could be dynamically decided based on the user's query or the load of the system. It could always be the case that the user only needs to translate a single word query, while needing to see the context of the translation. In this case, a number of the original and translated sentences could be returned, while executing one of the fast single word algorithms. Only when needed, the user interface or additional logic on the database side could decide whether a multi-word translation spotting algorithm has to be executed.

Chapter 5

Conclusion

Section 5.1 of this chapter summarises the motivation behind this work. The objectives of the dissertation are then reiterated in Section 5.2. The reader is presented with the achievements of the project in Section 5.3 followed by list of limitations in Section 5.4. Finally, Section 5.5 outlines the potential avenues for future research.

5.1 Motivation

Many international companies and multi-lingual organisations have to translate their documents and products into a number of languages. These organisations often hire localisation service providers to perform such translations. The LSPs then in turn hire translators and pay them per unit of translated text. The drive for cost reductions led towards exploring options such as automated machine translation techniques and reusing previous translations. Machine translation engines are constantly evolving, but still do not produce results which could be directly used in the world of professional localisation. Means of storing and reusing previous translations were developed by building translation memories and translation memory systems.

The translation memory systems often help to improve the productivity of translators. This is done by searching the translation memories to find sentences which match the currently translated one or those, which are similar. The previous translations of these sentences are then returned to the translator, who can make use of them.

It was found that modern translation memory systems suffer with a number of problems, which limit their value proposition to translators. Firstly, the attempts to match the whole sentence often fail. There is an infinite number of different sentences which could occur in a single language. The probability of a single sentence being repeated is very small. The translation memory systems attempt to resolve such problem by performing fuzzy matching, which can often yield a match on the sentence, which might not be useful. The solution to the problem was offered by academic research suggesting that translators could select queries of any length, which the translation system would look for. Second problem is regarding the usability of such systems. Since translation units retrieved by the translation memory systems could be large segments of text, the translators could become overwhelmed with the translations. They would then have to manually crawl through the text to find the part relevant to them. It was suggested that this problem is solved by performing translation spotting in order to suggest the relevant piece of translated text only.

Previously explored means of translation spotting would work on a single pair of aligned sentences. Techniques such as the use of statistical translation models, syntactic trees, glossaries, and part of speech analysis were suggested.

Recent development in both software and hardware meant that in-memory databases became a realistic technology. The large capacities of main memory available in modern computer systems and theoretical capacities supported by operating system vendors mean that even very large data can be stored and processed by such databases. The in-memory computing yields significant performance improvements compared to disk based database systems. The performance promised by such databases means that very large quantities of data could be analysed very quickly.

This dissertation considers the use of in-memory databases to analyse co-occurring patterns across large quantities of aligned sentences in order to perform translation spotting.

5.2 Dissertation Objectives

The research question of this dissertation was to explore the extent to which in-memory computing can be used to support real-time translation spotting.

A framework for an in-memory system was to be designed and implemented using an in-memory database. A number of techniques of performing translation spotting using the implemented framework were to be suggested.

The implemented system was to be evaluated in terms of extensibility, effectiveness and efficiency.

5.3 Achievements

The objectives set out at the start of the project were successfully achieved. SAP HANA was selected as the in-memory database of choice for this project and an in-memory framework was designed and developed for translation spotting. A number of translation spotting algorithms were proposed and implemented using the framework.

The translation spotting framework in terms of both the database schema and the administrative layer software was built and actively used throughout all stages of the project. The schema was proven to be well designed without posing additional implementation or performance difficulties. The administrative framework was built in such a manner as to make the ingestion of data easy, fast and allow for support of additional data formats. All data used during the initial implementation of the system and during the evaluation were ingested with the use of the administrative framework. The administrative framework was also designed to provide a number of testing tools, which were used heavily during the evaluation of the system. Testing was carried out in an automated manner allowing for empirically evaluating the system under many different conditions and collecting precise metrics related to executed tests.

A number of single word translation spotting algorithms were proposed as the functionality using the translation spotting framework. These algorithms were developed in an iterative way, where the results of each algorithm were analysed and observations made were fed into the design of the subsequent algorithms. Such observations included the effects that generic stop words have on effectiveness of co-occurrence translation spotting, and led to the development of ways to limit such effects.

To make the translation spotting useful to potential use in translation memory systems or online translation tools, the functionality was then extended to provide support for multi word queries. The development methodology was again iterative. A very simple, naive algorithm was developed. Latter algorithms were developed using similar concept, but were enhanced with further logic in order to avoid the problems which were encountered during manual testing of the simple algorithm.

The evaluation of the system was carried out in two phases. The phase one of the evaluation has shown a previously unseen issue related to migration of the system to a new version of the database engine. Once the results were interpreted, it became clear that most of the proposed algorithms were affected by the change and the collected metrics do not reflect the actual potential of the system. A fix of the issue was therefore implemented and the second phase of the evaluation tested the newest versions of the algorithms. A number of differences were observed in the test results between the phase one and the phase two of the evaluation. The empirical evaluation of the system has shown a number of interesting facts:

- analysing co-occurrence of patterns across large number of aligned sentences is a valid way of performing translation spotting (the precision of the single word algorithms reached up to 94%, while the multi word algorithm would yield up to 76% precision)
- the single word algorithms have proven to be well scalable and perform in real-time fashion
- the efficiency of the of the multi word algorithms is hardly predictable and depends greatly on both the query and the matched translation units (some queries would perform in sub-second fashion, while others would take in the order of minutes to execute)
- the translation spotting system is easily extensible with further data (based on the size of ingested data, a year of published documents could take between 30 and 75 minutes to be ingested and ready to use)

The results of the empirical evaluation of the system are considered a success. The results show how co-occurrence translation spotting could enhance the modern translation memory systems or give rise to new generation of translation tools. Furthermore, the measurements show how such big data analyses are not only possible but also performant with the use of in-memory databases. Certain implementation techniques used in the system pose a number of limitations on how the implemented system could be used in practice.

5.4 Limitations

The translation spotting algorithms which were implemented as a part of this project are divided into two categories - the single word and multi word algorithms. The single word algorithms show very high levels of efficiency and effectiveness. It would make sense to dynamically choose the algorithm to be executed based on the query - only execute single word algorithms for single word queries and the multi word algorithms for multi word queries. Both the single word and the multi word algorithms have a number of limitations.

Firstly, the single word translation spotting algorithms assume that a single word query in the source language would translate into a single word in the target language. This is not always the case. An example of a translation, where a single word is translated into a multi word phrase is the word “citizen” in English translated into Czech “státní příslušník”. Such cases would not be handled in the current implementation of the single word algorithms, even though they would be handled well by the multi word algorithms.

Secondly, the multi word algorithms have proven to be extremely computationally challenging even for in-memory computing. In the ideal scenario, it would be possible to generate all possible substring of all possible sentences which are identified to contain the translation translation of the query, and then rank them based on co-occurrence. This is a massive computational task and thus the multi word algorithms consider only the translation candidates generated from the shortest sentence. This poses a limitation on the quality of the translations. The translation candidates would be only as good as the quality of the shortest translated sentence. Furthermore, the performance of the multi word algorithm is greatly dependent on the length of the sentences in which the given query occurs. In cases, where the query occurs in very large sentences, which could in the original text occupy a whole paragraph, the number of potential translation candidates can be very large. The computation of generating all of the potential candidates is itself a challenging task and then all of these candidates have to be ranked against the identified list of sentences. There could even be a very large number of these matched sentences. While the execution times of multi word procedures could be in sub-second intervals for certain queries, other queries could take in the orders of tens of seconds or even minutes to execute. This poses a usability issue for the multi word algorithms.

These limitations lay the groundwork for potential further research.

5.5 Future Research

There are many potential areas of future research. The limitations mentioned in this chapter provide a basic outline of what would be the most pressing improvements.

5.5.1 Replacement of Imperative Logic

It was previously mentioned that the use of imperative logic within the stored procedures of SAP HANA prevents a number of automatic optimisation techniques. The most important is the fact that automatic parallelisation techniques will not work within imperative constructs. Any computation, which is therefore happening within the imperative features of SQLScript, will very likely not be parallelised.

The imperative features such as if statements, for and while loops should be replaced by set based logic with the use of SQL where possible. The places where such replacements could be performed should be analysed across the number of implemented procedures.

In cases, where it would not be possible to simply exchange the imperative constructs for SQL operations, the definitions of user defined functions should be considered. If it was possible to encapsulate the imperative logic in a user defined function, these functions could be used simply in SQL statements and it would be more likely that the optimisation techniques built into SAP HANA would be able to significantly improve the performance.

The imperative logic is used heavily mainly in the implementation of multi word algorithms, but also appears in some of the single word procedures. It would be desirable to analyse the potential of reducing the amount of imperative constructs used in the implementation and thus optimise the performance of the system.

5.5.2 "Smart" Optimisations of the Multi Word Algorithms

The behaviour of the multi word translation spotting techniques should be analysed further in an attempt to identify additional optimisations. The initial optimisation was to choose the shortest sentence from which the translation candidates are generated. This yielded great improvement in performance in some queries.

Similar means of optimisations should be explored. An example could be a way to limit the number of translation candidates which are generated from a sentence. It could be worth considering only generating the substrings from the words, which are positioned similarly to the position of the query in the original language sentence.

The single word algorithms are used in the dictionary approach in order to increase the effectiveness of the multi word translation spotting. Similar technique could be used to identify the position range of words from the sentence, which are used to generate translation candidates. If the keywords identified in the original sentence only appear between the words x and y of the sentence, it is possible that these positions could indicate potential position of the translation.

5.5.3 Use of Compression for Pre-generation of Translation Candidates

If the translation candidates could be pre-computed possibly at the time of ingestion of the data, the n-gram translation spotting would become as simple, efficient and possibly even as effective as the single word translation spotting algorithms.

The problem with pre-computing all of the possible substrings from all of the ingested sentences is clear - the memory requirements for such a system would be huge and such approach would be infeasible.

It could be explored how much memory overhead would be generated if a compression technique was used. Many of the commercially available in-memory database systems use compression techniques on columns to limit the size of data and speed up the analytical tasks. If the potential substrings were stored as sequences of integers with the use of dictionary compression techniques, the overhead produced by storing pre-computed translation candidates would not be so massive.

Analysing the cost of the compressed pre-computed translation candidates could provide insights into how feasible such technique would be.

5.5.4 Annotation of Documents with Thematic Categories

The existing document categorisation tools such as the JRC JEX [26] could be used to categorise documents and annotate the ingested translation memories with thematic categories. These categories could then be used to filter down the data selected for the co-occurrence analysis based on the categories of the query.

The use of these filters could yield potential improvements in both the effectiveness and the efficiency. In regards to the effectiveness, it could be the case that the co-occurrence of terms in documents, which, are semantically related to the query could produce more relevant results. Additionally, the amount of data to be analysed would be limited, which especially in cases where the translation memory database is very large, could have positive effects on the performance.

The effects of annotating the documents with thematic categories should be explored further.

Appendix A

Czech to English Single Word Test Cases

Table A.1: Czech to English single word test cases

Search Query	Acceptable Answers
Článek	Article
ESVO	EFTA
PROHLÁŠENÍ	declaration
spravedlnost	justice
mechanismu	mechanism
finančního	financial
norského	Norwegian
oblasti	area
porozumění	Understanding
Memoranda	Memorandum
Zásady	principle
postupy	procedure
EHP	EEA
pravidla	rule
údajů	data
Přehled	overview, review, summary
zprávy	report
Tabulky	table
států	states, state
členských	member
odborníci	expert
dříve	previously, early
Společenství	Společenství, Community
přijatého	adopt, receive

Continued on next page

Table A.1 – continued from previous page

Search Query	Acceptable Answers
seznam	list
význam	importance
návrhy	proposal
označil	identify
předpisů	regulation
právních	law, legal
oznámení	notification
očekávaných	expected
rozhodnutí	decision
Statistického	statistical
databáze	database
číslo	number
Podvýbory	subcommittee
Stálý	standing
Připomínky	comment
přijaly	adopt
instituce	institution
Nástroje	instrument, tool
programy	program
Poradního	advisory
protokolů	protocol
příloh	Annexes, annex
Konsolidované	consolidated
tajemník	secretary
Předseda	president
související	relate
ZVEŘEJNĚNÍ	publication
URČENÉ	intend
června	June
účinnosti	effectiveness, effect, efficiency
přezkoumá	review
reprodukovat	reproduce
osoby	person
omezuje	limit
rejstříku	registry, register
vypracován	draw

Continued on next page

Table A.1 – continued from previous page

Search Query	Acceptable Answers
doručit	deliver
datum	date
obsah	content
referenční	reference
občanům	citizen
způsobem	manner
Odkazy	reference
žádosti	application, request
základě	basis
požadovaný	required, require
žadateli	applicant
povinnost	obligation
splnit	meet
zdarma	free
elektronické	electronic
přístup	access
A4	A4
formátu	format
stran	party, parties
pořízení	acquisition
kopie	copy
náklady	cost
zaslání	send
zaslat	send
skutečné	actual, actually
poplatek	fee, charge
služební	staff, Staff
Označení	indication, designation
uvedením	state, provide
písemně	write, written
zamítnuta	reject
částečně	partially, partly, part
úplně	complete, full
zamítnutí	refusal
neprodleně	immediately
odpověď	reply

Continued on next page

Table A.1 – continued from previous page

Search Query	Acceptable Answers
pracovních	working
poučen	inform
přiměřené	reasonable
neformálně	informally

Appendix B

English to Czech Single Word Test Cases

Table B.1: English to Czech single word test cases

Search Query	**Acceptable Answers**
rectification	oprava, výmaz
effect	účinek
diplomatic	diplomatický
decision	rozhodnutí|
pursuant	podle, soulad
transfer	převod, převést
amended	pozměněný, měnit
published	zveřejnět
turnover	obrat
processing	zpracování|
regard	ohled
indicated	uvedený
mechanism	mechanismus
financial	finanční
Norwegian	norský
understanding	porozumění|
guidelines	pokyn
procedures	postup
rules	pravidlo, nařízení|
EEA	EHP
press	tisk, stiskací
states	stát
experts	odborník
EFTA	ESVO

Continued on next page

Table B.1 – continued from previous page

Search Query	Acceptable Answers
relevant	příslušný
already	již
acquis	acquis
marked	označit, označený, označeníl
EC	ES
adopted	přijmout
list	seznam
sheet	LIST, list
fact	skutečnost
numbers	číslo, počet
requirements	požadavek, nařízeníl
constitutional	ústavný, ústavněprávní, ústavní
committee	výbor
joint	smíšený, společný
subcommittee	podvýbor, výbor
legislation	předpis
proposals	návrh
notifications	oznámeníl
awaited	očekávaný
office	úřad
statistical	statistický
produced	vyrábět, výrobek, vyrobený, vyráběný
statistics	statistika, statistik
programme	program, Program
working	pracovní
chair	předseda, předsedat, předsednictvo, předsednictví
reports	zpráva
agreement	dohoda
annual	roční, výroční
documents	dokument, doklad
information	informace, údaj
conclusions	závěra, závěr
consultative	poradní, poradný
parliamentary	parlamentní
states	stát
standing	stálý

Continued on next page

Table B.1 – continued from previous page

Search Query	Acceptable Answers
groups	skupina
comment	vyjádřit, připomínka
opinions	stanovisko
version	verze, znění|
consolidated	konsolidovaný
resolutions	rezoluce, usnesení|
institutions	instituce, orgán
annexes	příloha
surveillance	kontrolní
court	dvůr, soudní
original	původní, originál
internet	internet, internetový
public	veřejný, veřejnost
available	dispozice, dostupný
general	generální, obecný, všeobecný
secretary	tajemník
Brussels	Brusel
article	článek
year	rok
review	přezkum
subject	podléhat, vztahovat, předmět
October	říjen
journal	věstník
supplement	dodatek
section	oddíl
repealed	zrušovat, zrušit
released	uvolnit
right	právo
limit	lhůta, limit
existing	stávající
access	přístup
citizens	občan
assistance	pomoc, podpora
secretariat	sekretariát
interests	zájem
protection	ochrana

Continued on next page

Table B.1 – continued from previous page

Search Query	Acceptable Answers
undermine	narušit, rozhodnutíl
manner	způsob
register	rejstřík
received	obdržet, obdržený

Appendix C

Czech to English Multi Word Test Cases

Table C.1: Czech to English Multi word test cases

Search Query	Acceptable Answers
byla provedena	being carried out
ohledem na dohodu	regard to the Agreement
Evropském hospodářském prostoru	European Economic Area
Spojenými státy americkými	United States of America
SMÍŠENÝ VÝBOR EHP	EEA JOINT COMMITTEE
Slovenské republiky	Slovak Republic
Ústavní požadavky nebyly oznámeny	No constitutional requirements indicated
Technické předpisy	Technical regulations
kosmetických prostředků	cosmetic products
islandském a norském jazyce	Icelandic and Norwegian
vnitřního trhu	internal market
členskými státy	Member States
SPOLEČNÉ PROHLÁŠENÍ	JOINT DECLARATION
volnému pohybu zboží	free movement of goods
Finanční služby	Financial services
Evropského parlamentu	European Parliament
cenných papírů	securities
informací o místě volajícího	caller location information
elektronických komunikačních sítích	electronic communication networks
ochraně osobních údajů	data protection, Protection of personal data
dopravní politiky	transport policy
norského finančního mechanismu	Norwegian Financial Mechanism
Statistického úřadu	Statistical Office

Continued on next page

Table C.1 – continued from previous page

Search Query	Acceptable Answers
Výroční zpráva	Annual report
Usnesení Poradního výboru	Consultative Committee resolutions
Kontrolního úřadu	Surveillance Authority
Konsolidované znění	Consolidated Version
autorských práv	copyright
přístup k dokumentům	Access to documents
referenční číslo	Reference number
Seznam dokumentů	List of documents
elektronické formě	computerised form, electronic form
Stálý výbor států ESVO	STANDING COMMITTEE OF THE EFTA
15 pracovních dnů	15 working days
zbývající části dokumentu	being carried out
veřejný zájem	public interests, Public interest
třetích osob	third parties, third party
duševního vlastnictví	IPR, Intellectual property
fyzické nebo právnické osoby	natural or legal person
veřejnou bezpečnost	public security
Generální tajemník	Secretary General
finančního řízení	financial management
záruky nezávislosti	guarantees of independence
vodního hospodářství	Water Management
letecké služby	Air Navigation Services
hospodářské soutěže	competition
Bezpečnost letectví	Aviation Safety
Spojené království Velké Británie a Severního Irska	United Kingdom of Great Britain and Northern Ireland
fixního kapitálu	fixed capital
dopravní politika	transport policy
internetových stránkách	website
rezervačních systémů	reservation systems
společných pravidlech	common rules
protiprávní podpora	unlawful aid, illegal aid
veřejných zakázek	PROCUREMENT
nízkou hustotou obyvatelstva	low density population, low population density

Continued on next page

Table C.1 – continued from previous page

Search Query	Acceptable Answers
anglickém jazyce	English
s ručením omezeným	limited company, Limited Liability Company
Ministerstvo spravedlnosti	Ministry of Justice
Soudním dvoru	Court
úrokové sazby	interest rate
Ohlašovací povinnost	notification duty, Notification requirement, REPORTING OBLIGATIONS
hospodářských činností	economic activities
přírodními katastrofami	natural disasters
smluvními stranami	contracting parties
dceřinou společností	subsidiary
Evropskou komisí	European Commission
Průzkum trhu	market survey, Market research
licenční poplatek	licence fee
investiční projekty	INVESTMENT PROJECTS
dosud nezveřejněno	Not yet published
úpadkového řízení	insolvency proceedings
podpory na restrukturalizaci	restructuring aid
regionálního rozvoje	Regional Development
Oblast působnosti	Scope
nezbytné minimum	minimum necessary
platební neschopnosti	insolvent, insolvency
vratná pomoc	reversible assistance
Statistická klasifikace	Statistical classification
roční obrat	Annual turnover
státní intervence	State interventions
Zjednodušený postup	Simplified procedure, Simplified approach
celková výše	overall amount
skutečné platby	actual payments, Paid actually
daňové ztráty	tax losses
přesná čísla	precise figures
národní měně	national currency
vzdělávání a zaměstnanost	training and employment
Konec platnosti	Expiry

Continued on next page

Table C.1 – continued from previous page

Search Query	Acceptable Answers
malým a středním podnikům	SME
Doba trvání	Duration
Celkový rozpočet	Total Budget
OZNAMOVACÍ FORMULÁŘ	NOTIFICATION FORM
dodatečné informace	Additional information
bloková výjimka	block exemption
ekonomických činností	economic activities, Economic activity
výzkum a vývoj	Research and development, RESEARCH DEVELOPMENT
audiovizuálních děl	audiovisual work
železničního vozového parku	railway rolling stock, railway stock, rolling stock
pracovní faktor	labour factor

Appendix D

English to Czech Multi Word Test Cases

Table D.1: English to Czech Multi word test cases

Search Query	Acceptable Answers
in order to guarantee	Zaručit
constitutional requirements	Ústavněprávní požadavky
European Economic Area	Evropský hospodářský prostor
amended by the Protocol	
EEA Joint Committee	SMÍŠENÝ VÝBOR EHP
transitional periods	přechodného období, Přechodná období
European Health Insurance Card	evropského průkazu zdravotního pojištění, Evropský průkaz zdravotního pojištění
incorporated into the Agreement	začlenění do Dohody
replacement and discontinuance	náhradě a zrušení
Icelandic and Norwegian languages	islandském a norském jazyce, jazyce islandském a norském
Official Journal of the European Union	Úředním věstníku Evropské, věstníku Evropské unie
No constitutional requirements indicated	Ústavněprávní požadavky neuvedeny
Decision of the EEA Joint Committee	Rozhodnutí Smíšeného výboru EHP
Technical regulations	Technické předpisy
attesting the conformity	ověřování shody
wood flooring	dřevěné podlahy, podlahy, Dřevěná podlahovina
industrial products	průmyslové zpracování
specific fields outside the four freedoms	některých oblastech mimo čtyři svobody
Press release	Tiskové zprávy

Continued on next page

Table D.1 – continued from previous page

Search Query	Acceptable Answers
Contracting Parties	smluvních stran, SMLUVNÍ STRANY
terms and conditions	Základní podmínky
Republic of Lithuania	Litevské republiky, LITEVSKÁ REPUBLIKA
Republic of Malta	Maltě, Malta
EEA Enlargement Agreement	dohodou o rozšíření EHP, Dohoda o rozšíření EHP
internal market	Vnitřní trh
provisional basis	prozatímně
Republic of Slovenia	Slovinské republiky, Republiky Slovinsko, REPUBLIKA SLOVINSKO
remain unaffected	zůstat nedotčeny, není dotčeno
provisional application	prozatímní provádění, Prozatímní uplatňování
professional qualifications	odbornou způsobilost, odborné způsobilosti, odborných kvalifikací, odborné kvalifikaci, Odborná kvalifikace, ODBORNÉ KVALIFIKACE
Mutual recognition	vzájemné uznávání, Vzájemné uznání
Transitional arrangements	Přechodná opatření, Přechodná ustanovení
Technical regulations, standards, testing and certification	Technické předpisy normy zkoušení a certifikace
cosmetic products	kosmetické prostředky
free movement of goods	Volný pohyb zboží
EFTA STATES	STÁTŮ ESVO, Státy ESVO
JOINT DECLARATION	SPOLEČNÉ PROHLÁŠENÍ
medical devices	přístrojů vyvinutých pro lékařské účely, Zdravotnické prostředky
Constitutional requirements indicated	Ústavněprávní požadavky uvedeny
processing of caller location information	zpracovávání informací o místě volajícího
electronic communication networks	elektronických komunikačních sítích, elektronické komunikační sítě
emergency call services	služeb tísňového volání
heavy goods vehicles	těžká nákladní vozidla, těžkými nákladními vozidly, nákladních vozidel

Continued on next page

Table D.1 – continued from previous page

Search Query	Acceptable Answers
repeals and replaces	zrušuje a nahrazuje
Community-wide turnover	obrat v rámci celého Společenství, obratu ve Společenství
common market	společným trhem
first sentence	První věta, první pododstavec
Community residents	rezidentů Společenství
gross premiums	hrubé pojistné
credit institutions	úvěrových institucí, úvěrové instituce
interest income	úrokové výnosy
commissions receivable	splatné provize
operating income	provozní příjem, Provozní výsledky
insurance contracts	Pojistné smlouvy
surveillance authorities	KONTROLNÍMI ÚŘADY
effective competition	účinná hospodářská soutěž, účinné hospodářské soutěže
EC Advisory Committee	PORADNÍ VÝBOR ES
Court of Justice of the European Communities	SOUDNÍ DVŮR EVROPSKÝCH SPOLEČENSTVÍ
Public security	veřejná bezpečnost, veřejnou bezpečnost
supply information	doplňující informace, podávat informace
PROFESSIONAL SECRECY	SLUŽEBNÍ TAJEMSTVÍ, Profesní tajemství
official languages	úředních jazycích, úředních jazyků
OTHER PROCEDURAL QUESTIONS	DALŠÍ PROCESNÍ OTÁZKY
after their finalization	po jejich dokončení
Common Procurement Vocabulary	společném slovníku pro veřejné zakázky, společný slovník pro veřejné zakázky
waste electrical and electronic equipment	odpadních elektrických a elektronických zařízeních, odpadní elektrická a elektronická zařízení
survey characteristics	ukazatelem zjišťování, Ukazatele zjišťování
United Kingdom	Británie
statistics on income and living conditions	statistice Společenství v oblasti příjmů a životních podmínek
consumer policy	spotřebitelské politiky
	Continued on next page

Table D.1 – continued from previous page

Search Query	Acceptable Answers
veterinary checks	veterinární kontroly
swine fever	mor prasat
Veterinary and phytosanitary matters	Veterinární a rostlinolékařské předpisy
domestic animals	chovného skotu, Domácí zvířata
salmonella for consignments	salmonelózy u zásilek, salmonely na zásilky
narcotic drugs and psychotropic substances	omamných a psychotropních látek, omamnými a psychotropními látkami
blood components	lidské krve
marine radio communication equipment	námořní radiokomunikační zařízení
stabilisation of financial instruments	stabilizace finančních nástrojů
Fact Sheet	Přehled údajů
European Parliament	Evropský parlament
financial collateral arrangements	dohodách o finančním zajištění
Prevention of Pollution from Ships	zabránění znečišťování z lodí, znečištění ropnými látkami z lodí
information society	INFORMAČNÍ SPOLEČNOSTI, Informační společnost
self-employed workers	pracovníky nebo osoby samostatně výdělečně
equal treatment for men and women	rovného zacházení s muži a ženami
vocational training	odborného vzdělávání, Odborné vzdělávání
professional careers	profesní kariéře
young people	mladí lidé, MLADÝCH LIDÍ
financial year	Finanční rok
cultural matters	kulturních otázkách
transport by rail	železniční přepravy, Železniční přeprava, ŽELEZNIČNÍ DOPRAVA
text of adaptation	Znění úpravy
shall be deleted	se zrušuje
Review clause	Doložka o přezkumu
CONCERNING THE COOPERATION	O SPOLUPRÁCI
GENERAL PRINCIPLES	OBECNÉ ZÁSADY
INITIAL PHASE	POČÁTEČNÍ FÁZE
30 working days	30 pracovních dnů
MAKE OBSERVATIONS	PŘEDKLÁDAT PŘIPOMÍNKY

Bibliography

[1] Discover Déjà Vu X3. `http://www.atril.com/content/discover-déjà-vu-x3`. Accessed: 10/05/2014.

[2] IBM x86 enterprise servers. `http://www-03.ibm.com/systems/x/hardware/enterprise/`. Accessed: 10/05/2014.

[3] SDL Trados Studio. `http://www.translationzone.com/products/sdl-trados-studio/`. Accessed: 10/05/2014.

[4] SUSE Linux Enterprise Server for System z. `https://www.suse.com/products/systemz/technical-information/`. Accessed: 10/05/2014.

[5] Wordfast PRO. `http://www.wordfast.com/products_wordfast_pro.html`. Accessed: 10/05/2014.

[6] AVISON, D. E., LAU, F., MYERS, M. D., AND NIELSEN, P. A. Action research. *Communications of the ACM 42*, 1 (Jan. 1999), 94–97.

[7] DÉSILETS, A., FARLEY, B., STOJANOVIC, M., URDININEA, F., AND AD, A. B. Using WeBiText to Search Multilingual Web Sites Using WeBiText to Search Multilingual Web Sites. *NRC Publications Archive* (2010).

[8] ELINA LAGOUDAKI. Key findings of the TM Survey 2006 carried out during July and August 2006.

[9] FÄRBER, F., CHA, S. K., PRIMSCH, J., BORNHÖVD, C., SIGG, S., AND LEHNER, W. SAP HANA Database - Data Management for Modern Business Applications. *ACM SIGMOD Record 40*, 4 (2011), 45–51.

[10] FRANZ, F., MAY, N., AND LEHNER, W. The SAP HANA Database – An Architecture Overview. *IEEE Data Eng. Bull. 35*, 1 (2012), 1–6.

[11] GARCIA-MOLINA, H., AND SALEM, K. Main memory database systems: An overview. *IEEE Transactions on Knowledge and Data Engineering 4*, 6 (1992), 509–516.

[12] GAREY, M., AND JOHNSON, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. 1979.

[13] GLIGOR, G., AND TEODORU, S. Oracle Exalytics: Engineered for Speed-of-Thought Analytics. 3–8.

[14] IBM. BLU Acceleration changes the game, 2013.

[15] JACOBS, A. The Pathologies of Big Data. *Communications of the ACM 52*, 8 (2009), 36–44.

[16] LEWIS, D., AND CONNOR, A. O. On Using Linked Data for Language Resource Sharing in the Long Tail of the Localisation Market Resource Sharing via Linked Data Language Data Resource Sharing in the.

[17] MACKLOVITCH, E., SIMARD, M., AND LANGLAIS, P. TransSearch : A Free Translation Memory on the World Wide Web. *LREC* (2000).

[18] MURTHY, V., DESHPANDE, P., LEE, A., GRANHOLM, D., AND CHEUNG, S. ORACLE EXALYTICS IN-MEMORY MACHINE : A BRIEF INTRODUCTION, 2014.

[19] NEVADO, F., CASACUBERTA, F., AND Ý, J. L. Translation memories enrichment by statistical bilingual segmentation. *LREC* (2004), 335–338.

[20] PLATTNER, H. A common database approach for OLTP and OLAP using an in-memory column database. *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data.* (2009), 1–2.

[21] RAPP, R. A Part-of-Speech-Based Search Algorithm for Translation Memories. *LREC* (2002), 466–472.

[22] SAP. *SAP HANA Developer Guide*. 2013.

[23] SIKKA, V., FÄRBER, F., LEHNER, W., AND PEH, T. Efficient Transaction Processing in SAP HANA Database – The End of a Column Store Myth. *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data* (2012), 731–741.

[24] SIMARD, M. Translation spotting for translation memories. *Proceedings of the HLT-NAACL 2003 Workshop on Building and using parallel texts: data driven machine translation and beyond 3* (2003), 65–72.

[25] SIMARD, M., AND LANGLAIS, P. Sub-sentential Exploitation of Translation Memories. *Machine Translation Summit VIII* (2001).

[26] STEINBERGER, R., EBRAHIM, M., AND TURCHI, M. JRC EuroVoc Indexer JEX – A freely available multi-label categorisation tool. 798–805.

[27] STEINBERGER, R., EISELE, A., KLOCEK, S., PILOS, S., AND SCHLÜTER, P. DGT-TM : A freely Available Translation Memory in 22 Languages. *arXiv preprint arXiv* (2013).

[28] VÉRONIS, J., AND LANGLAIS, P. Evaluation of parallel text alignment systems. *Parallel text processing. Springer Netherlands* (2000), 369–388.