Master Thesis

Using Machine Learning Methods for Evaluating the Quality of Technical Documents

Author:	Michael Luckert Moritz Schaefer-Kehnert
Supervisor:	Prof. Dr. Welf Löwe
Examiner:	Prof. Dr. Andreas Kerren
Reader:	Dr. Ola Petersson
Semester:	HT2015
Subject:	Computer Science
Course:	15HT - 5DV50E/4DV50E

Abstract

In the context of an increasingly networked world, the availability of high quality translations is critical for success in the context of the growing international competition. Large international companies as well as medium sized companies are required to provide well translated, high quality technical documentation for their customers not only to be successful in the market but also to meet legal regulations and to avoid lawsuits.

Therefore, this thesis focuses on the evaluation of translation quality, specifically concerning technical documentation, and answers two central questions:

- How can the translation quality of technical documents be evaluated, given the original document is available?
- How can the translation quality of technical documents be evaluated, given the original document is not available?

These questions are answered using state-of-the-art machine learning algorithms and translation evaluation metrics in the context of a knowledge discovery process. The evaluations are done on a sentence level and recombined on a document level by binarily classifying sentences as *automated translation* and *professional translation*. The research is based on a database containing $22,327$ sentences and 32 translation evaluation attributes, which are used for optimizations of five different machine learning approaches. An optimization process consisting of $795,000$ evaluations shows a prediction accuracy of up to 72.24% for the binary classification. Based on the developed sentence-based classification systems, documents are classified using recombination of the affiliated sentences and a framework for rating document quality is introduced. Therefore, the taken approach successfully creates a classification and evaluation system.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Being able to overcome language barriers is a topic of interest since the beginning of the 17th century. Since then, devices and ideas have been developed to ease the understanding between people who speak different languages, such as mechanical dictionaries or universal languages. Due to highly internationalized companies and the overall globalization, the ability to automatically translate texts from one language to another without human assistance is a subject that has been addressed for roughly 60 years and has gained more interest throughout the last decade. To be successful in an international market, companies must provide well translated documentation for their products. As complex products often have more than one user group, e.g. administrators, users and developers, the number of different documents for a single product can be very high. Especially for export oriented companies, it is hard to find professional translators with an appropriate technical background to create properly translated technical documentation for reasonable costs. Therefore, machine translation solutions obtain increasing interest affected firms.

It is of great interest, to provide automated, high quality translations of texts to ensure equal access to information regardless of their source language. In particular, accurate translations of technical documentation is a high priority for companies, since they are essential for a smooth workflow, customer satisfactions and describe the handling, functionality as well as security features of products. In this field, misunderstandings can be very severe. In addition, cooperation among companies can suffer from misunderstandings that are caused by bad translations.

Since natural language processing is a very complex issue, the output generated by machine translation software still needs to be approved in order to ensure the required quality. Thereby, the simple use of software to translate business documents moves the problem from the creation of the document to the evaluation and correction, but does not solve it. Consequently, the evaluation of translated technical documentation is an important step for companies to reduce time and costs as well as to create an effective way of translating critical documents. Additionally, this ensures a certain level of quality. The difficulty in evaluating translation quality is due to the subjective nature and different aspects concerning the term quality, such as grammatical correctness, style improvements or semantic correctness.

Having access to computerized systems that perform correct translations of any sentence is still visionary, especially due to the problem of translating the meaning of a sentence to the system. Concerning this problem, it is essential to be able to rank the quality of a given translation - otherwise it is not possible to ensure that a document has been well translated. A focus on technical documentation is especially interesting, due to their high quantity in every product selling company which further increases the motivation of automatically translating these types of documents. Nowadays, companies deal with the given translation problem of technical documents by outsourcing this task to external translators. Since the person requesting these translations does not necessarily speak the translated language, it is important to ensure that the work has been done properly and professionally by a human as ordered and not by an automated translation system.

Based on this background, the aim of this thesis is to select and implement a machine learning process that produces an algorithm, which is able to detect whether documents have been translated by humans or computerized systems. This algorithm builds the basic structure for an approach to evaluate these documents.

1.2 Related Work

As mentioned above, the idea of automatically rating machine translations is not new. One of the basic methods discussed in the literature is the “round-trip translation”, which works by translating a text fragment into a foreign language and back. Afterwards, the original text and the newly generated text are compared [1]. On this basis, the “bilingual evaluation understudy” (BLEU) algorithm was developed and presented by Kishore Papineni et al. in 2002. BLEU defines document quality as a strong correlation between machine translations and the work of professional human translators. It is based on the idea that in addition to word-accuracy, the length of a translated text is important. According to Papineni et al., human translations have the tendency to be of higher quality and shorter than computerized translations [2]. This idea was further developed in the United States forming the *NIST algorithm for machine translation evaluation* by the National Institute of Standards and Technology. This algorithm weights matching words according to their frequency in the respective reference translation [3]. A second evolution of the BLEU metric, is the *Metric for Evaluation of Translation with Explicit Ordering*, called Meteor, which was developed by Lavie et al. in 2005. The main difference is Meteor’s ability to detect synonyms of words which results in potentially less erroneous translations [4]. Furthermore, Kulesza and Shieber (2004) propose the use of Support Vector Machines for classifying machine translations on a sentence level [5]. Extending on sentence level evaluation, there has been additional research on substituting the use

of human-produced reference translations, which often come with high resource requirements for the evaluation of machine translation systems. Popović et al. propose the use of lexicon probabilities [6], while Gamon et al. suggest the use of pseudo references replacing the commonly used human reference translations with multiple automated translation systems as references and combining calculated perplexity scores with a machine learning classifier to evaluate sentence quality [7]. Finally, Albrecht and Hwa successfully use regression learning in combination with pseudo references [8, 9]. As shown above, a lot of research has been done on the topic of machine translation evaluation.
However, the focus on a specific domain of documents in order to gain implicit additional knowledge by using machine learning techniques is not sufficiently addressed and neither is the comparison of different machine learning approaches in order to classify whether documents have been translated professionally or automatically. This work sets out to answer these questions.

1.3 Purpose and Research Question

In this thesis, a machine learning technique will be used in a knowledge discovery process to classify documents by their translation type (*professional translation*, *automated translation*). Further, an approach on how to evaluate the quality of translated technical documents will be proposed. Concerning this issue, we address two main research questions:

- How can the translation quality of technical documents be evaluated, given the original document is available?
- How can the translation quality of technical documents be evaluated, given the original document is not available?

1.4 Approach and Methodology

This work focuses on using machine learning methods and algorithms in order to evaluate translations of technical documentation. There are two different problems that will be solved within this thesis. First, translations of technical documents will be classified and evaluated with the machine learning algorithm having access to the original document. In the second attempt, an algorithm will be optimized on the same task without having knowledge of the original.

The planned procedure for our master thesis is the following:
Based on research on existing methods and metrics, an iterative knowledge discovery process will be started to answer the given research questions. This process includes the determination of quality criteria for translated documents, the implementation of needed

metrics and algorithms as well as the optimization of the machine learning approaches to solve the given task optimally. It is important to note that this process is of iterative nature, since the criteria and attributes as well as their impact on translation quality and classification possibilities will be determined by evaluating the algorithms' results using a database of technical documents and their translations. The used data set will range from automated translations of technical documents using computerized translation systems to manual and professional translations. Furthermore, during this iterative process, the methods and algorithms used will be continually changed and optimized to achieve the best possible results. Finally, the process and results will be critically reviewed, evaluated and compared to one another. The limits of automated translations with the current state of the art will be pointed out and a prospect for possible further developments and studies on this topic will be given.

1.5 Scope and Limitation

Due to the fixed time frame, some limitations have to be set on this research to ensure the work can be finished in time.

- The document classification and evaluation will focus on syntactic aspects of technical documentation, while the semantic parts will be left out.
- This work focuses specifically on technical documents. This focus has the potential to implicitly generate knowledge during the machine learning process, due to a smaller sized vocabulary compared to having no limitations on text domains. Other document domains will not be included in this thesis.
- Since the examined technical documents did not provide multiple professional translations, there will be no human references used for an evaluation of technical documents. Instead, pseudo references, as described in section 3.2 will be used to circumvent the lack of human translations.
- This work focuses on evaluations based on the results of machine learning approaches. Other techniques, such as the creation of a new metric comparable to the BLEU or Meteor metric will not be taken into account.
- This work will focus on translations between German and English.

1.6 Target group

First, this work is especially interesting for researches in the area of machine translation and machine translation evaluation by using combinations of different machine translation metrics and machine learning approaches. As mentioned in 1.1 the private interest group for this research are international companies, especially export oriented ones. This is due to the fact that finding technically versed translators on a limited budget is clearly problematic. Second, this work is of interest for all kinds of customers, since it can lead to long-term improvement of available information for certain products. This can be of additional interest for customers and companies that are active in lesser populated languages.

1.7 Outline

The following chapter 2 will address the theoretical background concerning this work, focusing on the knowledge discovery process, machine learning approaches, machine translation and technical documentation. After describing the taken methodology in chapter 3, the empirical data and the respective results of the experiments will be shown in chapter 4. Chapter 5 will discuss the results, critically review the taken approaches and methods as well as examine the validity and reliability of presented results. Finally, chapter 6 will summarize the work and give an outlook for possible future research and expansion on this topic.

Chapter 2

Theoretical Background

This section gives an overview of the relevant theoretical foundations. In the context of the experiment's framework, the following questions are answered:

- What is the data knowledge discovery in databases process?
- What are the advantages of data mining and which algorithms are promising in the field of machine translation evaluation?
- How does machine translation work and which different approaches are available?
- Which metrics are commonly used to measure machine translation quality?
- What characterizes technical documentation?

To answer these questions, the first section deals with the *knowledge discovery in databases* process, followed by a definition of data mining as well as a detailed elaboration on machine learning in general and relevant algorithms. Afterwards, the current state of the art in the field of machine translation and its quality evaluation is presented. The section is closed by characterizing technical documentation and outlining the special characteristics of these kind of documents.

2.1 Knowledge Discovery in Databases

The knowledge discovery in database process (KDD) describes the overall step of discovering useful knowledge from data. Although there exist numerous definitions of the KDD process [10, 11, 12], most of them agree on the essential parts. Fayyad et al. define the KDD as an interactive and iterative process. They outline nine main steps [10, p.42 et seq.]:

1. Identify the goal of the process and gather prior needed knowledge about the application domain.
2. Choose an appropriate data set to extract knowledge from.

3. Preprocess the data. This includes removing noise or harmful data records and deciding on certain settings, such as how missing attribute values are handled in the data set.

4. Reduce the data to a representable format, for instance, removing not helpful variables or parameters in terms of the task's objective.

5. Decide on a data mining approach for the defined objective of the KDD process.

6. After deciding on a general data mining approach, the next step is to choose the data mining algorithm. It is important to note that this choice often depends on the end user's preference, for instance, whether an understandable format or the maximum amount of prediction quality is favored.

7. This is the main data mining step. It consists of applying the algorithm to the preprocessed data set. The algorithm then searches for valuable knowledge within the data.

8. Interpret the patterns found by the algorithm and possibly return to one of the previous steps in order to readjust the setup of the KDD process.

9. The final step of the knowledge discovery in databases is using the interpreted results for further actions, for instance, using them for further research or applying a system to a real-world scenario.

As mentioned in step 8, the KDD process can consist of many iterations and loops. For instance, after interpreting the results of an algorithm, one might conclude that the chosen algorithm was a wrong choice and go back to step 5 or after reducing the data to a representable format in step 4 that the preprocessing has been done wrongly and return to step 3.

2.2 Data Mining

Data mining is often used as a synonym for the KDD. In fact, data mining represents the part of the KDD-process, in which the appropriate approach and algorithm is chosen and applied to the data set [10, p.39]. Therefore, it is a central part of the knowledge discovery in databases process [13, p.2]. Data mining consists of taking any form of data and applying analysis algorithms to it in order to reveal patterns or models within the data set and using these structures to classify the data into different classes (labels). It comprises a number of research fields such as database systems, statistics and pattern recognition. Data mining tasks are differentiated depending on the knowledge the algorithm has about the existing classes in the data set [14, p.51]:

- **Supervised learning** includes every task in which the algorithm has access to input and output values. Input values are defined as the external information that the algorithm is allowed to use, such as attribute values and meta data, while output values, are the specific labels of the class attribute. This means that the structure of the data is already known and the goal of these programs is to assign new data to the correct classes.
- In contrast to supervised learning, **unsupervised learning** includes all tasks that have no access to output values and therefore try to find structures within the data by creating classes on their own.

Since the proposed task is a binary classification problem, concerning the two known classes *professional translation* and *automated translation*, this work will focus on supervised learning methods.

Furthermore, data mining can be differentiated into two main objectives: verification and discovery. While verification tries to prove the user's hypothesis, discovery looks for yet unknown patterns within the data. The discovery step splits up into description, where the system finds patterns in order to present the data in an understandable format and prediction, where the system tries to predict the future outcomes of data from patterns. The subgroup prediction can further be distinguished into classification and regression tasks. While classification tasks have fixed labels and each data record has one of these labels as its class attribute value, regression tasks have continuous values as output [13]. This work focuses on algorithms that try to predict whether a given technical documentation has been translated professionally or automatically. Therefore, the problem consists of two fixed labels (*professional translation* and *automated translation*) and belongs to the *discovery-prediction* sector of data mining. Figure 2.1 shows a general view of the data mining taxonomy.

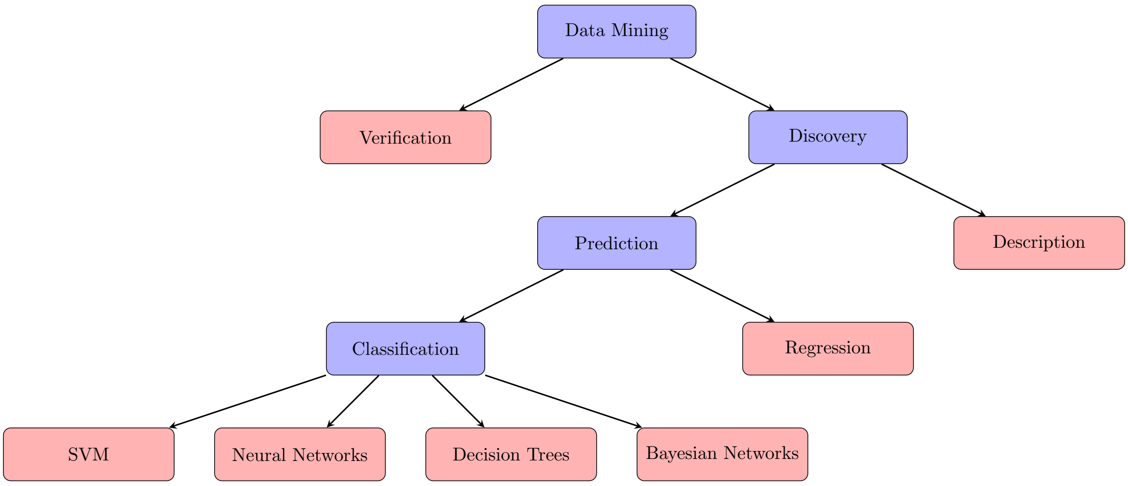

Figure 2.1: Data mining taxonomy based on [13].

2.3 Machine Learning

In context of the data mining step, it is important to choose the correct approach for tackling the task appropriately. This is often done using machine learning methods. A major difference between humans and computers has been for a long time that a human beings tend to automatically improve their way of tackling a problem. Humans learn from previous mistakes and try to solve them by correcting them or looking for new approaches to address the problem. Traditional computer programs do not look at the outcome of their tasks and are therefore unable to improve their behavior. The field of machine learning addresses this exact problem and involves the creation of computer programs that are able to learn and therefore improve their performances by gathering more data and experience. The first scientist to create a self-learning program was A. Samuel in 1952, who created a program that became better at playing the game checkers with the number of games played [15, p.881]. In 1967, the first pattern recognition program was able to detect patterns in data by comparing new data to known data and finding similarities between them. Since the 1990's machine learning is used in data mining areas, adaptive software systems as well as text and language learning fields. As an example: A computer program that gathers data concerning the customers of an e-commerce shop and creates better personalized advertisements out of these pieces of information has the ability to acquire new knowledge and comes close to being artificial intelligence.

Furthermore, machine learning systems are normally classified by their underlying learning strategies, which are often identified by the amount of inference the computer program is able to perform [16, p.84 et seq.]:

- **Rote Learning** describes the strategy that all traditional computer programs use. They do not perform any kind of inference and all their knowledge has to be directly implemented by the programmer, since the application is not able to draw any conclusions or transformations from the given information.

- **Learning from Instruction** encompasses all computer programs that are able to transform information from the given input language to an internal language. Although the knowledge on how to effectively perform this transformation is still given by the programmer, this requires little forms of inference from the side of the computer program. Therefore, this defines a separate level of learning system compared to rote learning.

- In contrast to Learning from Instruction, **Learning by Analogy** tries to develop new skills that are almost similar to existing skills and therefore easy to adopt, by performing transformations on known information. This system requires the ability of creating mutations and combinations of a dynamic knowledge set. It creates new functionalities, which were unknown to the original computer program and therefore requires a lot of inference.

- **Learning from Examples** is nowadays one of the most commonly used learning strategies as it provides the most flexibility and enables computer programs to develop completely unknown skills or find unknown structures and patterns in data [17]. Learning from examples is a technique that is often used in classification and data mining tasks to predict the class label of new data entries based on a dynamic set of known examples. In this work, the proposed research questions will be tackled with strategies and algorithms that belong to this category.

Following, the most common machine learning systems will be briefly described:

2.3.1 Decision Tree

A Decision Tree is a classification technique that focuses on an easily understandable representation form and is one of the most common learning methods. Decision Trees use data sets that consist of attribute vectors, which in turn contain a set of classification attributes describing the vector and a class attribute assigning the data entry to a certain class. A Decision Tree is built by iteratively splitting the data set on the attribute that separates the data as well as possible into the different existing classes until a certain stop criterion is reached. The representation form enables users to get a quick overview of the data, since Decision Trees can easily be visualized in a tree structured format, which is easy to understand for humans.

One of the first algorithms concerning Decision Tree training were the Iterative Dichotomiser 3 (ID3) and its successor the C4.5 algorithm, both developed by Ross Quinlan in 1986 and 1993 [18, 19]. These algorithms formed the basis for many further developments. Decision trees are directed trees, which are used as a decision support tool. They represent decision rules and illustrate successive decisions.
In Decision Trees, nodes can be separated into the root node, inner nodes, and end nodes, also called leafs. The root node represents the start of the decision support process and has no incoming edges. The inner nodes have exactly one incoming edge and have at least two outgoing edges. They contain a test based on an attribute of the data set [20, p.769]. For instance, such a test might ask: “Is the customer older than 35 for the attribute age?”. Leaf nodes consist of an answer to the decision problem, which is mostly represented by a class prediction. As an example, a decision problem might be the question whether a customer in an online shop will make a purchase or not, with the class predictions being *yes* and *no*. Leaf nodes have no outgoing and exactly one incoming edge. Edges represent the decision taken from the previous node.

Given a node n, all following nodes that are separated by exactly one edge to n are called children of n, while n is called parent of all its child nodes. Figure 2.2 shows an example of a Decision Tree. For instance, a data record, having the attributes $cold, polarBear$

would be passed down to the left subtree, since his temperature attribute is cold and then down to the leaf "North Pole" being classified with the corresponding label.

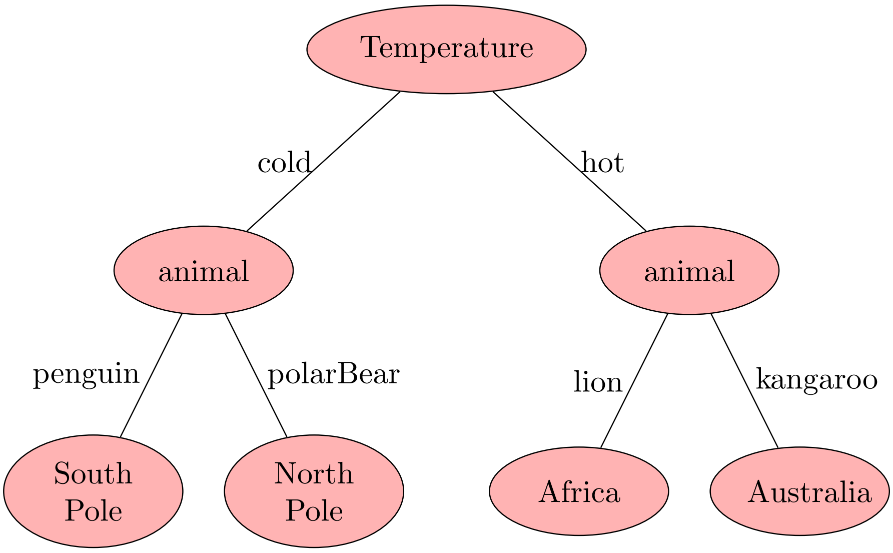

Figure 2.2: Example of a Decision Tree.

Training a Decision Tree is a common data mining method that is mainly used for classification purposes. Its goal is to predict the value of a target attribute, based on a number of input attributes. Training a Decision Tree in a supervised scenario is done by using a training set to find patterns within the data and build the Decision Tree. Afterwards, a set of previously unseen examples can be used to predict their target attribute's value. The training set contains data records in the form of:

$$\left(\vec{x}, Y\right) = \left(x_1, x_2, x_3, ..., x_n, Y\right)$$

with Y being the target attribute value and $\vec{x}$ being a vector containing n input values, where n is the number of attributes in the data set [20, p.765 et seq.].

In order to train a Decision Tree and thereby create a classifier, a training set is needed containing a target attribute, input attributes, a split criterion and a stop criterion. At a given node, the split criterion calculates a value for all attributes. This value represents a measure of the amount of information that is gained by splitting the node using this attribute. Afterwards, the best value from all attributes is taken and the node is split into the different outcomes of the respective attribute. At this point, the process of finding the best split among the attributes is applied recursively to all generated sub trees until a stop criterion is reached.

Common stop criteria are:

- The maximum height of the tree has been reached.
- The number of records in the node is less than the allowed minimum.

- The best split criterion does not overcome a certain threshold in terms of gained information.

If the splitting attribute is of numeric type there is no possibility to split the records into all outcomes of the attribute. This is one of the main upgrades of the C4.5 Decision Tree compared to the ID3. The C4.5 is additionally able to calculate the best splitting points for numeric attributes as well and split them by using *greater than or equal* and *smaller than* operators.

Training a Decision Tree with this automated process can result in large Decision Trees with sections of very little power in terms of classification. Additionally, trees tend to be overfitted, which means that they fit the training instances too closely. This results in bad performances when these trees are applied to unseen data. Therefore, a technique called *pruning* has been developed. Its objective is to remove the less or non-productive parts from the Decision Tree, such as parts based on noisy or erroneous data or parts that are overfitted. This often results in further improvements in terms of accuracy and shrinks down the tree size. This process is especially important, due to the fact that every real-world data set contains erroneous or noisy data.

Computation Time

The time complexity of the original tree-growing algorithm that considers only nominal attributes, is $O\left(m * n^2\right)$ with m being the size of the training data set and n being the number of attributes. The most time consuming part of the tree-growing algorithm is the calculation of the gained information for each attribute [21, p.500 et seq.]. In order to calculate the information gain the values of the corresponding attribute for all data records in the current training subset is needed. In worst case scenarios the union of all subsets at all levels of the Decision Tree is the same size as the original training set. Therefore, the complexity for computing the information gains for each level of the tree is already $O\left(m * n\right)$. Since the number of levels of the tree is n for the worst case the overall complexity for training a Decision Tree is $O\left(m * n^2\right)$.

After training a Decision Tree, the following process is to use the tree in order to predict the class labels for unseen data records. To do so, the record is passed down from the root node to a leaf testing the corresponding attribute at each node and following the edges to the appropriate leaf.

```
**Algorithm** *Decision Tree Training Process*
training set = S;
attribute set: A;
target Attribute = C;
split criterion = sC;
stop criterion = stop;
*Grow*$(S, A, C, sC, stop)$
**if** $stop(S) =$ **false**
  **then**
        **for** all $a_i \in A$
            **do** find $a_i$ with the best $sc(S)$;
        label current Node with $a$;
        **for** all values $v_i \in a$
            **do** label outgoing edge with $v_i$
               $Ssub$ = $S$ where $a = v_i$;
               create subNode = *Grow*$(Ssub, A, C, sC, stop)$;
  **else** currentNode = leaf;
        label currentNode with $c_i$ where $c_i$ is most common value of $C \in S$;
```

Figure 2.3: Pseudo code of training a Decision Tree [13, p.131].

Figure 2.3 shows the process of training a Decision Tree in pseudo code, not taking numeric attributes into account. The algorithm starts by testing whether the stop criterion has been reached or not. If so, the current Node is labeled with the most common value of all existing class labels for the training set. If the stop criterion is not true, the algorithm calculates the split value for all attributes and labels the node with the attribute corresponding to the best split value. Afterwards, it splits the node into multiple nodes, one for each value of the chosen attribute. The algorithm calls the same process recursively for all training subsets, containing all data records with the corresponding value of the chosen attribute.

2.3.2 Artificial Neural Networks

Singh and Chauhan define Artificial Neural Networks as "a mathematical model that is based on biological neural networks and therefore is an emulation of a biological neural system" [22, p.37 et seq.]. Compared to conventional algorithms, neural networks can solve problems that are rather complex, on a substantially easier level in terms of algorithm complexity. Therefore, the main reason to use Artificial Neural Networks is their simple structure and self-organizing nature which allows them to address a wide range of problems without any further interference by the programmer. Example given, a neural network could be trained on customer behavior data in an online shop and predict whether the person will make a purchase or not.

An Artificial Neural Network consists of nodes, also called neurons, weighted connections between these neurons that can be adapted during the learning process of the network and an activation function that defines the output value of each node depending on its input values. Every neural network consists of different layers. The input layer receives information from external sources, such as attribute values of the corresponding data entry, the output layer produces the output of the network and hidden layers connect the input and the output layer with one another. The input value of each node in every layer is calculated by the sum of all incoming nodes multiplied with the respective weight of the interconnection between the nodes [23, p. 165]. Furthermore, neural networks can be divided into two main types [22, p.38 et seq.]:

- **Feedforward Networks** are defined as all networks that do not gain feedback from the network itself. This means that the input data flows in one direction, from the input nodes through 0 to n hidden nodes to the output nodes. There is no information given backwards to readapt the system.

- **Recurrent Networks** are defined as all networks that contain a feedback option and therefore are able to reuse data from later stages for the learning process in earlier stages.

The output value of each node is calculated by using all input values on a predefined function that is the same for every node in the network. The most commonly used function is the sigmoid function(o_j), which is defined as follows [23, p.167]:

$$o_j = \frac{1}{1 + e^{-i_j}}$$

where i_j is the sum of the input nodes of j.
According to Erb, the two main advantages of this function are its normalization to values between 0 and 1 and its nonlinear nature, which results in easier rapid network learning and prevention of overload and domination effects. A domination effect occurs, when a single or a few attributes get a very high influence on the predicted target attribute, rendering other attributes meaningless and therefore dominating them [23, p.167].
Figure 2.4 shows an Feedforward Neural Network with three input nodes in the input layer, a single hidden layer and two output nodes.

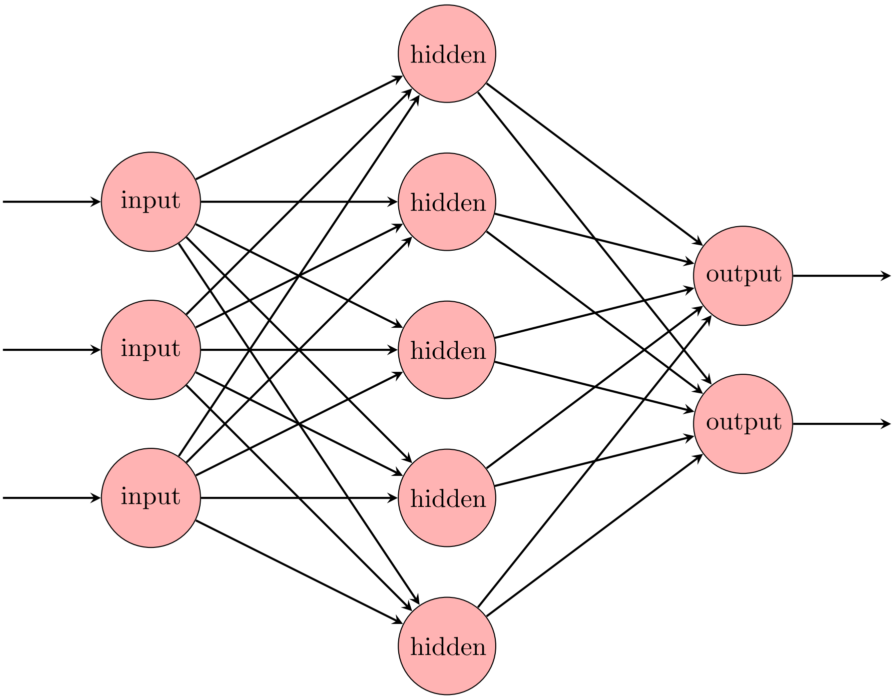

Figure 2.4: Example of an Artificial Neural Network.

In a supervised learning scenario, Artificial Neural Networks can be trained using the backpropagation algorithm, which readjusts the weights of the interconnections in the neural network based on local error rates.

Backpropagation

Backpropagation in neural networks describes the process of using a local error of the network to readjust the weights of the interconnections backwards through the neural net. Explicitly, this means that after a prediction for a set of input values has been made, the actual output value is compared to the prediction value and an error is calculated. This error is then used to readjust the weights of the connections starting at the edges that are directly connected to the output nodes of the network and then proceeding further into it. In order to train a neural network, it is important to understand the main parameters that can be used to optimize the learning process:

- The **learning rate** specifies how fast the learning process is performed. The parameter's value lies between 0 and 1 and is multiplied with the local error for every output value. Therefore, a learning rate of 0 would result in no adaptation at all. The correct setting for the learning rate is crucial to the success of the learning process. If the value is set too high the weights can oscillate and complicate the finding of the optimal values. However, if the value is set too low, found errors will not have enough weight to push the network into a new optimization and the weights can get stuck in local maxima [23, p.167]. In order to find the correct settings, a decay parameter can be added. This parameter ensures a high learning rate value in the earlier cycles of the training process to avoid getting stuck in local maxima and forces its reduction during the learning process to avoid oscillation.

- Another important parameter for neural networks is called **momentum**. It is used to smooth out the optimization process by using a fraction of the last weight change and adding it to the new weight change.
- The **minimal error** is a stop criterion for the learning process similar to the stop criterion for Decision Trees described in subsection 2.3.1. Once the combined error of the network falls below this threshold, the learning process is stopped.

Combining these parameters, the formula for computing a new weight for a connection is the following[23, p.167]:

$$W = l * \epsilon + m * W_p$$

where

- W is the new weight change.
- l is the learning rate.
- ϵ is the minimal error.
- m is the momentum.
- W_p is the weight change of the previous cycle.

2.3.3 Bayesian Networks

Bayesian networks consist of nodes and directed connections between these nodes that symbolize dependencies between them. They are probabilistic directed acyclic graphical models. Each node represents an attribute of interest for the given task, such as pollution values in cities for the likelihood estimation of developing lung cancer. The most basic Bayesian network is called *Naive Bayes* and the reason for it being called Naive is that this network assumes that there are no dependencies between attributes. This is almost never the case in practical data mining tasks and therefore this method tends to achieve worse results than more detailed algorithms. Normal Bayesian networks use known data to estimate the dependencies between attributes and the class label and use this information to calculate probabilities of possible different outcomes of future events. It automatically applies the Bayes' theorem to complex problems and is therefore able to gain knowledge about the state of attributes and their dependencies.

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

where

- A and B are events.
- $P(X)$ is the probability that event X occurs.
- $P(X|Y)$ is the conditional probability that event X occurs if event Y is known to be true [24, p.30-31].

Each node of the graph is labeled with a probability distribution that defines the effects of a parent node on the child node [25, p.166 et seq.].

2.3.4 Instance-Based Learning (kNN)

Instance-based learning describes the process of solving problems based on solutions for similar already known problems, also known as *nearest neighbor learning*. Every instance-based learning system requires a set of parameters:

- A distance function that measure the similarity between problems or data entries. This is needed to measure which are the closest *neighbors* to the new problem.
- A number of neighbors that are considered when addressing the new problem
- A weighting function that enables further quantification of found neighbors to increase prediction and learning quality
- An evaluation method that describes a function on how to use the found neighbors to solve the given problem.

Instance-based learning methods are part of the lazy learning methods, which means that no computation is performed on the data before a query is given to the system. These methods stand in contrast to eager learning methods such as Decision Trees, which try to structure the data before receiving queries [15, p.549-550].

2.3.5 Support Vector Machines

Support Vector Machines belong to the area of supervised learning methods and therefore need labeled, known data to classify new unseen data. The basic approach to classify the data, starts by trying to create a function that splits the data points into the corresponding labels with *(a)* the least possible amount of errors or *(b)* with the largest possible margin. This is due to the fact that larger empty areas next to the splitting function result in fewer errors, because the labels are better distinguished from one another.
Figure 2.5 demonstrates that a data set may very well be separable by multiple functions without any errors. Therefore, the margin around a separating function is being used as an additional parameter to evaluate the quality of the separation. In this case the separation A is the better one, since it distinguishes the two classes in a more precise manner.

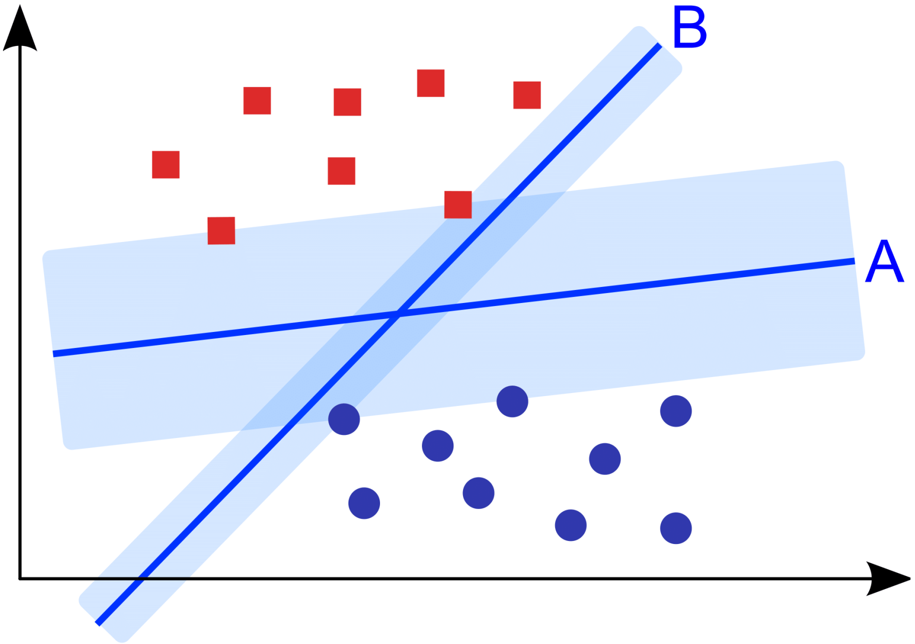

Figure 2.5: Visualization of a Support Vector Machine splitting a data set into two classes, by using two different linear separations resulting in differently sized margins around the splitting functions [26].

Formally, Support Vector Machines create one or multiple hyperplanes in an n-dimensional space. The first attempt in the process of splitting the data is always, to try to linearly separate the data into the corresponding labels. Example given, for a task of predicting the probability of a purchase of a customer in an online shop uses a data set with n data points, where each data point consists of a label $y \in \{purchase, nopurchase\}$ and an attribute vector $\vec{x}$ containing the data values for that specific session. The Support Vector Machine now tries to find a function that separates all data points $(\vec{x}, y)$ with $y = yes$ from all data points of the form $(\vec{x}, y)$ with $y = no$. If the data is completely separable in linear fashion, the resulting function can be used to classify future events. Steinwart and Christmann mention two main concerns, concerning this approach [27, p.14]:

- The data may not be well linearly separable or not linearly separable at all, which is often the case for real world data. In the example given above, it can happen for example that two customers are acting completely similar in an online shop, with only one of them making a purchase. This would result in not separable data, due to the same attribute vector having different labels.

- The second concern is the possibility of overfitting the SVM. To avoid this, data has to be preprocessed to identify noise and accept some misclassifications. Otherwise, the accuracy values of the SVM will be flawed and result in more erroneous classification for future events.

The first problem can be resolved by using the kernel trick, mapping the n-dimensional input data to a higher dimensional space, where the data can be separated linearly.

2.3.6 Evaluation of Machine Learning

Another very important part in machine learning, is the problem of how a computer program notices which of its results were appropriate and which contained mistakes. An example where this poses no problems to the algorithm would be a computer program that tries to predict whether a customer in an e-commerce shop will perform a purchase or not. The data entry will afterwards be logged with the given information whether the customer bought articles or not and can then be used to evaluate the performance of the algorithm. More difficult scenarios come up in research areas with limited or no access to real-world data, such as the evaluation of document translations. This requires an additional human effort to rank given translations into classes to be able to compare the computer program results in the end. As described in subsection 2.3.1 the evaluation of classification tasks is normally done by splitting the data set into a training data set and a test data set. The machine learning algorithm is then trained on the first one, while the test data set is used to calculate performance indicators in order to evaluate the quality of the algorithm. A common problem for machine learning algorithms lies in the access to limited test and training data. Therefore, overfitting can be a serious problem when evaluating these programs. In order to address this problem, a common approach is, to use an *X-Fold Cross Validation*. Cross Validation describes the process of splitting the whole data set into X parts and using each one of them sequentially as the test data set while combining the others to the training data. Afterwards, the performance indicators are averaged over all validation processes. There is no perfect indicator for every subject concerning evaluation of machine learning algorithms, since everyone has its flaws and advantages. The most important factors for evaluating the performance of a machine learning program are the following:

- The **misclassification rate** describes the relative amount of falsely classified data in a data set. If $\hat{y_i}$ is the prediction for the data point i and y_i is the actual label, the misclassification is defined as

  $$misc_n = \frac{1}{n} * \sum_i (y_i \neq \hat{y_i})$$

  The main problem for misclassification is that its results depend highly on the amount of labels or the data distribution among the class labels. Example given, achieving a misclassification rate of 0.03 may look very promising without further context, but in an example where 97% of the data set are labeled with class a and 3% with class b it is not hard to achieve. The same applies for differences in the amount of classes available. A misclassification rate of 20% or lower symbolizes a clearly better machine learning system for a data set of three classes than for one with two. To avoid this issue, benchmarking is used.

- **Benchmarking** describes the process of comparing the value of an indicator to a reference value, to strengthen its statement. Example given, for a binary classification task in a supervised learning scenario the benchmarking algorithm could be a classifier that always predicts the most common class. The misclassification rate could then be described in reference to the benchmark. A misclassification rate of 20% in this case, would result for equal class distribution in a relative improvement compared to the benchmark of 30 percentage points.

- The **precision value**, also called positive prediction value, is defined as the relative amount of correctly as true classified instances among all as true classified instances [28, p.2]. This can be illustrated by using the e-commerce example mentioned earlier. A precision value of 1 signifies that every as customer classified with the label *purchase*, truly makes a purchase. However, it is important to note that this has no influence on the amount of as customer classified with the label *no purchase* that make purchases as well.

- The **recall value** also called sensitivity is defined as the relative amount of as true classified instances among all true instances [28, p.2]. For the proposed example, this means that a recall value of 1 signifies that every single purchasing customer has been labeled with *purchase*. It is important to note that a recall value of 1 can be easily achieved by classifying every instance of the data set with *purchase*.

- The **F-Measure** aims to combine the statements of recall and precision by using the harmonic mean between the two:

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

- One of the best approaches to illustrate the performance of machine learning programs is the **confusion matrix**, also called contingency table, which distinguishes between true positive, false positive, true negative and false negative predictions [28, p.1 et seq.]:

		Actual Value		
		p	**n**	**total**
Prediction Outcome	**p′**	True Positive	False Negative	P′
	n′	False Positive	True Negative	N′
	total	P	N	

Table 2.1: Example of a confusion matrix.

The main disadvantage of a confusion matrix is that it requires human interpretation. In this work, machine learning methods will be used in the context of machine translation problems, which will be explained in section 2.4.

2.4 Machine Translation

The subject of overcoming language barriers by using devices has a long history starting in the 17th century. In that time the idea for a universal language emerged to ease understanding among people from all over the world. Examples in that time, consisted of symbols and logical principles. The automation of the translation task however came up as a topic in the middle of the 20th century and the first machine translation (MT) conferences for knowledge and research exchange in this specific field started in 1952. Today, machine translation is defined as

> “the translation from one natural language (source language) to another language (target language) using computerized systems, with or without human assistance” [29, 30].

According to Hutchins and Somers, machine translation includes the areas of Human-Aided Machine Translation and Machine-Aided Human Translation, which are defined as the production of translations by systems with assistance by human translators. However, computerized systems that provide dictionaries or other top level assistance tools to human translators are not included [29, p.3]. With the current state of the art, machine translation is still far from being able to perfectly translate any given text from any source language to any target language. Nevertheless, after more than five decades of development on this topic, computerized systems are able to create “raw” translations, which are normally still revised by humans to validate them. Additionally, these translation machines are often focused on a specific domain, vocabulary or text type to further improve their translation quality [29, p.1 et seq.]. Today, there are two different approaches to machine translations, which will be explained below.

2.4.1 Rule-Based Machine Translation

Rule-based translation covers all translation processes that are based on rules concerning syntactic, semantic and direct word aspects of the text. Using rule-based translation is always a trade-off between complexity and quality of the translations. This is, because one of the main advantages of rule-based systems is the missing quality ceiling for translations. In theory, every error can be corrected by implementing a rule for that specific case and since there is no limit to the amount of rules used, every error can be corrected. This is a theoretical point of view, since the amount of possible word or sentence combinations and their different meanings is generally too large. Therefore, a certain amount of errors has to be expected when practically applied. According to Hutchins and Somers, there are three types of rule-based translations:

- **Direct Translation** was the first type of machine translation and it describes all systems that are developed for a single specific translation direction. This means that the systems is only capable of translating texts from a specific source to one specific target language, which is usually done using a word-based translation dictionary with no attempt of understanding the context or meaning of the sentence or text. There is very little morphological analysis performed, such as identification of word ending and conjugation [29, p.72 et seq.]. Figure 2.6 shows the general process of the direct machine translation type.

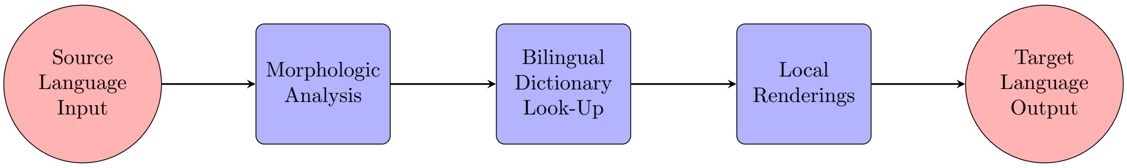

Figure 2.6: Direct translation process[29, p.72].

- Due to the grave lack of syntax and context understanding of the direct translation method, new approaches were taken to tackle the challenge of machine translation. The second type of rule-based translations is called **Transfer translation**, because it uses a transfer step to analyze source as well as target language and identifies word relations and possible meanings to create higher quality translations. A common approach to find out dependencies and correlations of words is the *parts of speech tagging* method, which labels every word with its corresponding part of speech, such as verb, noun, adjective or adverb. Today, transfer-based approaches are the most commonly used method, due to its clear advantages over direct translation and their comparably simple structure. One main disadvantage, the language dependency, of the transfer structure is shown in Figure 2.7. In order to support multiple languages and bidirectional tasks, a large amount of transfer processes is needed. Specifically, in order to add the $n + 1st$ language to a system, an additional $2 * n$ transfer steps are needed [29, p.75-76]. This led to the development of the third type of rule-based translations: Interlingua translation.

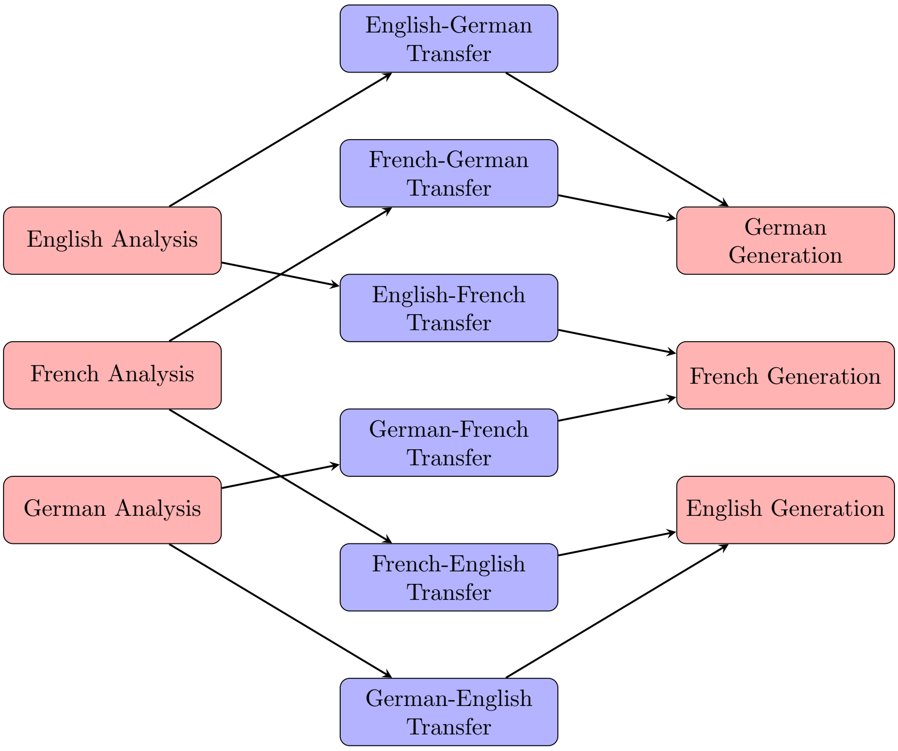

Figure 2.7: Visualization of the language and direction translation dependency of the transfer step in the Transfer translation process [29, p.76].

- **Interlingua Translation** was a second approach to solve the lack of syntactical and semantical understanding of the direct translation approach. The main difference to the transfer translation type is the use of an international auxiliary language as a separate step in the translation process. The use of such an auxiliary language that is easily compatible with a very wide range of different natural languages would reduce the amount of needed translation steps drastically, since the needed transformation steps are a transformation from each source language into the auxiliary language and vice versa. In conclusion, supporting a total of n languages, results in the need of $2n$ translation steps. The addition of a language would add two additional steps, an analysis step to produce a projection from the new language to the auxiliary language and a generation step to create sentences in the newly added language. This stands in contrast to $2n$ additional steps in the transfer translation process. Figure 2.8 visualizes an Interlingua translation process for two languages. The main problem concerning this approach, lies in the fact that the creation of such an auxiliary language is not trivial and the support of every existing language is close to impossible [29, p.74]. Furthermore, since the generation of the sentence in the target language is strictly independent from the original source sentence, it is not possible to optimize the source text analysis according to the target language. This imposes a hard to solve requirement to the auxiliary language, because it has to provide every possible analyzable aspect of the source text, since it might be needed for the generation step. To provide every piece of information, the auxiliary language has to understand the meaning of the sentence. This transforms the translation problem to a natural language processing task on semantic level, which is still in its early stages of research [29, p.118].

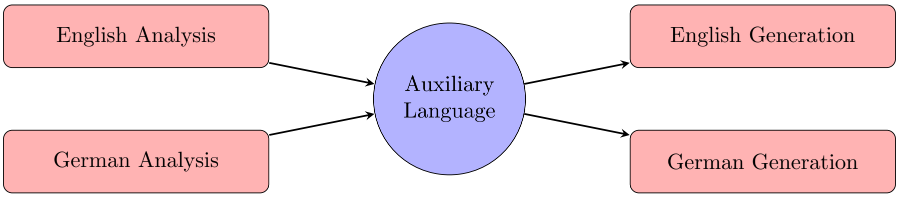

Figure 2.8: Interlingua translation process for two supported languages.

A proper way to visualize the different approaches to rule-based machine translation is the use of the Vauquois Pyramid as shown in Figure 2.9. The translation process starts at the bottom left corner of the triangle. Depending on the translation approach a certain amount of analysis is performed, before the transfer step is carried out to generate the output sentence by using the target language analysis. Since the direct translation uses close to no analysis at all, it transfers directly from the source language to the target language. The Interlingua approach on the other hand, performs as much analysis as possible to transform the input text into a neutral independent form and use this form to generate the output text.

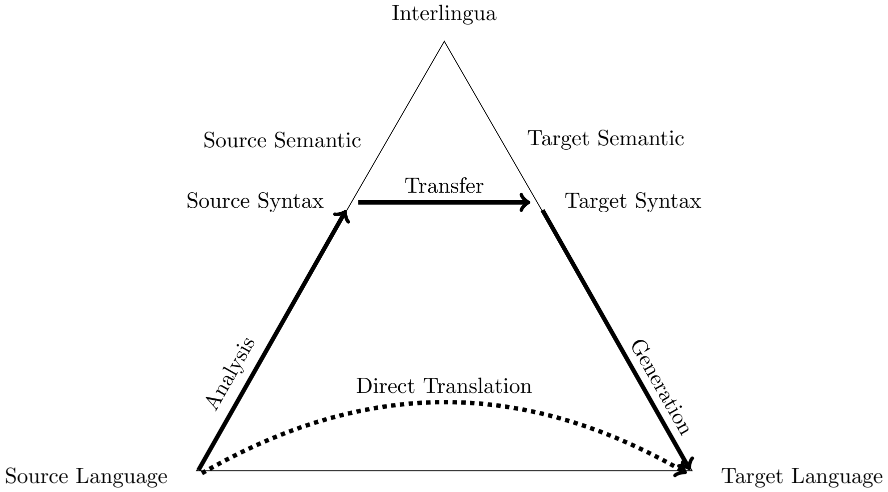

Figure 2.9: Visualization of the amount of analysis performed in a rule-based machine translation process.

2.4.2 Example-Based Machine Translation

The second large area of machine translation types is called example-based machine translation (EBMT). It was first proposed by Nagao in 1981 (published 1984) [31]. There are many types of machine translation approaches that fit in the area of example-based translation, but the main defining part of this field, is the use of a database containing already translated sentences or text parts. The translation process for EBMT systems consists of comparing fragments of new, untranslated text parts to the database, matching them with corresponding similar text fragments and combining these to a newly translated text [32, p.116]. The matching of new data points to the most similar ones in the database is mostly done by calculating distance measures between the fragments and choosing the "closest" among the examples. These matching algorithms will be further described in section 2.5. In order to point out the similarities between traditional (rule-based) machine translation and EBMT, Somers proposes an adapted version of the vauquois triangle as shown in Figure 2.10. The matching part replaces the analysis step, the selection of the corresponding text parts in the target language replaces the transfer step and instead of generating a new sentence based on the analysis results, these fragments are combined to the translated sentence [32, p.117].

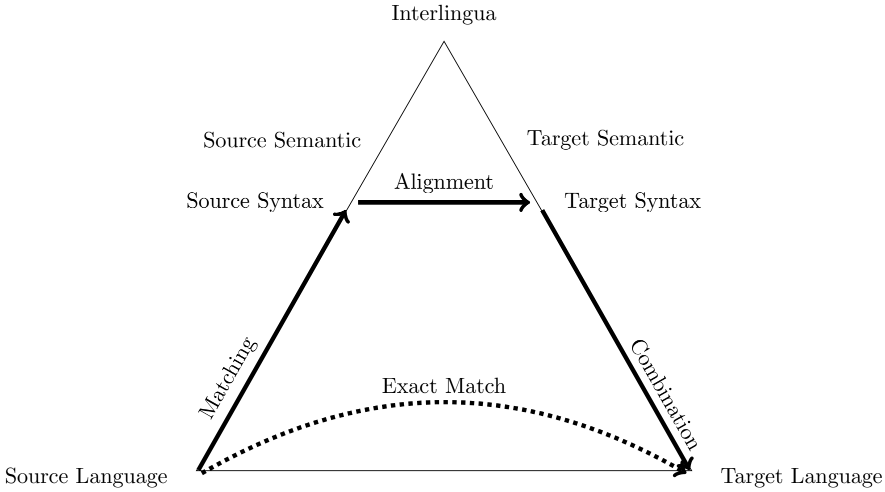

Figure 2.10: Adapted version of the Vauquois Triangle [32, p.117].

The main challenge that EBMT systems face, is the need of a suitable database containing a bilingual example set. Somers notes that choosing a sublanguage corpus that is specified on a certain domain for example, can greatly benefit the translation quality. Furthermore, the central points of concern about the example set are the text fragment length and the data set size. Nirenburg et al. state that the choice on the example length is always a trade-off. The longer the data point, the lower the probability of finding a match, thus rendering the data point useless, but the shorter the entry the higher the risk of falsely assigning it to not actually matching fragments [33, p.48]. The most commonly used fragment length is the sentence level. This is not necessarily due to that being the best length, but rather because sentence endings are easy to determine and split a text on. The size of the example set is a more straightforward problem. It is generally agreed that more examples lead to better results, although, as mentioned above the focus on a specific domain can still be advantageous even compared to a bigger example set. Furthermore, it is assumed that at some point the benefit by adding new examples to the data set stagnates [32, p.119].

Statistical Approach

The statistical machine learning approach is part of the EBMT, since it uses a data set of already translated fragments. It was first proposed by Peter Brown at the second TMI conference at Carnegie Mellon University, where he presented the “purely statistical” approach for translating text from a source language to a target language. The difference to other EBMT methods lies in the fact that statistical approaches try to find patterns in the data set, calculate probabilities and make intelligent conclusions based on the results [32, p.126].

2.5 Evaluation of Machine Translation

With the emergence of the field of machine translation, the problem of evaluating MT systems appeared as well. Without a proper approach to determine the quality of a translation machine, there was no point for further research. The first evaluation attempts were human-based, manual metrics. Dorr et al. note that the first two most common approaches were the evaluation of *fluency* and *adequacy*. Fluency is measured by a person fluent in the target language and measures whether the sentence is fluently readable with no regard to the translation correctness. Adequacy on the other hand, evaluates whether the essential information has been translated with no regard to grammatical correctness or fluency. Both metrics are normally measured on a scale of five to seven points [34, p.748]. Although it has been proven multiple times that these metrics are neither sufficient nor of high correlation, human evaluation is still being used as a benchmark or baseline to rate other automated translation metrics [35, 36]. There were mainly two problems with human judgement on fluency and adequacy. The first one was the high amount of human assistance needed for evaluations of translation, which made the point of automatically translating texts meaningless. Second, the evaluation of a text could differ a lot between two different evaluating persons. In order to solve these problems, the development of mathematical and automated metrics began at the end of the 20$^{\text{th}}$ century. These automated processes normally consist of comparing the translation output to a set of reference translations and calculating distance or similarity measures between them [34, p.815]. Following, we give an overview over the currently existing algorithms for the most important translation evaluations.

2.5.1 Round-Trip Translation

The round-trip translation approach (RTT) was one of the first techniques used to evaluate the quality of machine translation systems. In order to evaluate the performance of a translation program, it was given a text fragment, which was first translated into another language and then back into the original one. The results could easily be compared to the original text fragment. However, according to Somers, 2005, it is widely agreed that RTT translation is an inappropriate approach to evaluate the quality of a sentence. This is due to the fact that round-trip translation requires two well working translation systems (from the original to the foreign language and vice versa) and as a result normally produces a high amount of errors [1]. Additionally, if there would be no errors in the resulting document, it could still mean that errors occurred during the first half of the round-trip translation process, which got corrected on the way back. Therefore, research has developed multiple additional metrics to evaluate translation quality, which will be discussed in the following subsections.

2.5.2 Word Error Rate

The word error rate (WER) was one of the first metrics used to evaluate translations and is still the standard technique for Automatic Speech Recognition evaluation [34, p.816]. The key aspect of the word error rate is the use of the Levenshtein distance as a measurement for the similarity between two sentences. This is done by identifying, which words cannot be found in the new translation (*deletions(D)*), which words cannot be found in the reference translation (*insertions(I)*), which words have been substituted by another (*substitutions(S)*) and which words match each other. All these edits are calculated in reference to an existing professional translation of the same text fragment, which is called reference translation. The word error rate is then computed by the sum of the *substitutions*, the *insertions* and the *deletions* divided by the word count of the given text fragment *(N)*:

$$WER = \frac{S + I * D}{N}$$

As proposed by Nießen in 2000, the word error rate can be extended to be used with multiple reference translations (MWER) by calculating all individual word error rates and using the closest reference as a result [34, p.817]. However, this is an approach that only aims to include WER in multi-reference tasks, since the calculated result does not gain any additional information of the use of multiple references. The main shortcoming of the word error rate for text translation lies in the fact that there can be multiple correct translations of the same sentence that neither use the same word order nor even the same words, which would both result in worse WER scores.

2.5.3 Translation Error Rate

The translation error rate, also called translation edit rate (TER), was developed in 2005 by the GALE research program. It was used to calculate the number of edits needed to convert an automated translation into a correct text fragment concerning its fluency and adequacy. In order to normalize the TER metric, the number of edits is put into context of the length of the reference text, or, for multiple references, the average length of the references. The resulting formula for the TER score is defined as follows [37, p.3]:

$$TER = \frac{number\ of\ edits}{average\ number\ of\ words\ in\ the\ references}$$

It is important to note that punctuation marks are considered as full words for the TER score and the following actions are considered edits:

- Inserting of words that are missing in the translated text fragment.
- Deleting words that do not exist in the reference text.
- Substituting of words in the translated text.

- Shifting 1 to n sequential words for 0 to n steps in any direction in the fragment.

An example for the TER score is given below:

Reference:	*Tsai Ing-wen has been elected Taiwan's first female president*
Translation Candidate:	*Taiwan's first feminine president Tsai has been elected*

Although the hypothesis has almost the exact same meaning as the reference text and is correct in its fluency, the TER score will not result in a value of 0. In order to convert the translation candidate into an exact match, the following steps are needed:

1. Insert the word *Ing-wen* behind the word *Tsai*.
2. Substitute the word *feminine* with the word *female*.
3. Shift the word sequence *Taiwan's first female president* four times to the right.

According to the formula presented, this would result in a TER score of $\frac{3}{9} = 33,33\%$ (1 insertion, 1 substitution, 1 Shift). This illustrates the main weakness of the automated TER score. Since it has no access to the semantic meaning of the translation, the calculated result has to be interpreted with respect to this aspect. As proven by Shapira and Storer, the calculation of minimal edit distances is an NP-Complete problem and is normally approximated by using greedy search or dynamic programming approaches [38]. In order to overcome the lack of semantic recognition by the TER score, an extension has been proposed by Snover and Dorr called HTER (human-annotated translation edit rate) [37]. This approach includes the creation of a *targeted reference* that has been created by a human annotator.

2.5.4 BLEU

The bilingual evaluation understudy algorithm (BLEU) was presented in 2002 by Papineni et al. It aims to establish a fast and inexpensive method for evaluating the quality of machine translations without active human interaction. Papineni et al. define quality as the closeness of a candidate text to professionally, human translated references. BLEU achieves high correlations between automatic and human text ratings and therefore became the basic algorithm to rate and improve machine translation systems. The algorithm is language independent and mostly used to rate complete documents due to shortcomings on sentence level evaluations [2, p.311 et seq.].
The BLEU metric is based on a precision measurement, counting the amount of similar words in a candidate and a human translated reference translation. Afterwards, the matches are divided by the total amount of words used in the candidate translation.
The following example shows a candidate sentence that produces a score of $5/6$, which would indicate a good translation.

- Candidate 1: The bird sits on the tree.
- Reference 1: On the tree sits a bird.
- Reference 2: A bird sits on the tree.

To address the problem, that low quality candidate sentences, such as *(the the the the the the)* would produce a higher score (in this case $6/6$), Papineni et al. introduced the Modified Unigram Precision, which limits the score of a single word by considering a reference word exhausted after a matching candidate word is identified. Therefore, the Modified Unigram Precision of *(the the the the the the)* is $1/6$.
To take the fluency of a candidate translation into account, the modified precision is calculated on blocks of words, also called n-grams, where n is the amount of sequential words in the text fragment block. Papineni et al. show high correlations between automated ratings on blocks of 3, 4 or 5 words and human ratings.
To take the different lengths of candidate and reference translations into account and ensure that very short candidates do not produce comparably high scores, a brevity penalty is added to the Modified N-gram Precision. As candidate translations that are longer than the references are already penalized by the Modified N-gram Precision, the implicit penalty factor for long sentences is 1. Shorter translations are penalized by a brevity penalty (*BP*). The complete formula for the BLEU metric on a corpus level is [2, p.311 et seq.]:

$$BLEU = BP * exp(\sum_{i=1}^{n} w_i \; log \; p_i)$$

and

$$BP = \begin{cases} 1 & \text{if } c > r \\ exp(1 - \frac{r}{c}) & \text{if } c \leq r \end{cases}$$

where

- w_i are the positive weights for each used n-gram setting, which is computed by using geometric mean of co-occurrences over n,
- n is the maximal n-gram length,
- i is the n-gram, block length,
- p_i is the modified n-gram precision,
- c is the total length of the candidate translation corpus,
- and r is the total length of the best matching reference corpus.

The presented BLEU metric combines fast computations and language independency with high human judgement correlations and is therefore widely used for machine translation evaluations. However, the algorithm ignores several language specific characteristics such as synonym handling and is therefore most productive on a document level. Additionally, the importance and information value of the rated text is not taken into account and common words and phrases are overrated. Variations such as the NIST metric or Meteor score are attempting to address these shortcomings and will be discussed in the following subsections.

2.5.5 NIST

The NIST metric is named after the US National Institute of Standards and Technology. It extends the basic BLEU metric and takes some shortcomings into account. It introduces n-gram weighting according to their information value and therefore relatively reduces the weight of common n-grams [3, p.3]. It also substitutes the geometric mean of co-occurrences over N used in the BLEU algorithm (N being the number of words in the given text) by using an arithmetic average of n-gram counts. The NIST formula is composed as follows [3, p.141]:

$$NIST = \sum_{n=1}^{5} \left(\frac{\sum \Big(Info(co\text{-}occuring\ n\text{-}grams) \Big)}{Number\ of\ n\text{-}grams\ in\ candidate} \right) * p$$

where p is a penalty factor, which is calculated as the following [3, p.140]:

$$p = exp\Bigg(\beta * \log^2 \Big(min\Big(\frac{Scored\ Candidate\ Words}{Average\ Number\ of\ Words\ in\ Reference}, 1 \Big) \Big) \Bigg)$$

and $Info(x)$ is the individual n-gram information weight for the n-gram x, defined as follows [3, p.140]:

$$Info\Big(w_1, w_2, \ldots, w_n \Big) = \log_2 \left(\frac{number\ of\ occurrences\ of\ w_1, \ldots, w_{n-1}}{number\ of\ occurrences\ of\ w_1, \ldots, w_n} \right)$$

The NIST metric is an adjustment to the earlier described BLEU score and also focuses on document rating rather than sentence level evaluation. Compared to BLEU, NIST allows more precise translation evaluations and prefers rare n-grams to common ones, based on the idea that rarer n-grams contain more information than common ones. While NIST achieves better scoring results as BLEU, when it comes to similarities to human ratings of translations, it requires multiple references to calculate the n-gram information weight.

2.5.6 METEOR

Meteor (Metric for Evaluation of Translation with Explicit Ordering) is another BLEU-based algorithm for evaluating machine translations. The metric was designed to fix common problems of the BLEU algorithm and achieves better results on a sentence level.

Meteor is based on the harmonic mean of unigram precision and recall, with recall being weighted higher than precision [39]. First, the algorithm creates an alignment between two sentences, the candidate and a reference sentence. The alignment is a unigram-based mapping and tries to bind each candidate unigram to a matching reference unigram. Every unigram in the candidate can map a maximum of one reference word. Meteor considers four different kinds of matches:

- **Exact:** The matching words are identical, they share the same surface.
- **Stem:** The aligned words share a word stem and are therefore matching.
- **Synonym:** The word meanings are equal; they are synonyms of each other.
- **Paraphrase:** Entire paraphrases are considered matching, if they share the same meaning.

For the final alignment the number of covered words across the sentences is maximized, while the number of chunks is minimized. Chunks are defined as chains of matches that are contiguous and identically ordered in the compared sentences. The initial setup is transformed into a formula using the harmonic mean, which is calculated using unigram precision and recall. The unigram precision P is calculated as:

$$P = \frac{m}{w_t}$$

where m represents the number of matches and w_t the number of words in the candidate text.
The unigram recall R is computed as:

$$R = \frac{m}{w_r}$$

Where m remains unchanged and w_r is the number of words in the reference.
Both scores are combined to the harmonic mean in the following fashion:

$$F - mean = \frac{P * R}{\alpha * P + (1 - \alpha) * R}$$

In order to penalize short sentences and multiple chunks, Meteor introduces a penalty p, computed as:

$$p = \gamma * \left(\frac{c}{u_m}^{\beta}\right)$$

where c is the number of chunks and u_m is the number of mapped unigrams.
The final Meteor score is composed of $F - mean$ and the penalty p in the following manner:

$$Meteor = F - mean * (1 - p)$$

To tune the Meteor results, Denkowski and Alon Lavie suggest to weight unigrams according to their concurrency in the target language and recommend a tuning of the α,β and γ parameter. Common values are [39]

- $\alpha = 0.9$
- $\beta = 3$
- $\gamma = 0.5$

The main advantage of the Meteor metric is the use of paraphrase tables and synonyms and therefore the ability to recognize similarities on smaller text bodies with a higher precision. However, due to its large use of external data, such as paraphrase and synonym tables as well as the extensive method of finding matches in general, the metric requires not only language specific preparations but also extensive computing time to calculate compared to the BLEU score. Additionally, following the suggestions to optimize the alpha, beta and gamma individually would increase the effort even further. Still, besides the high preparation and calculation effort, Meteor is currently acknowledged to be the best metric for sentence level evaluation.
In this work, machine learning methods are used in the context of machine translation problems with a focus on the specific domain of technical documentation. These will be described in section 2.6.

2.6 Technical Documentation

Technical documentation is a generic term for every kind of product-related document, aiming to reveal information about a product or service. Technical documents are grouped into internal and external documentation. Internal documentation is understood to represent technical drawings, part lists, work lists, work instructions and others. It is fundamental for the development, construction and maintenance of products. External documentation, includes data sheets, spare parts catalogs and manuals, address existing customers and are partially used to acquire them. External documentation also verifies product specifications to government authorities [40, p.2 et seq.]. The different kinds and several functions require technical documentation to fulfill certain requirements, such as [40, p.37 et seq.]:

- The audience has to be known and appropriately addressed.
- The linguistic style has to focus on the understanding of the described matter.
- The documentation has to be complete and structured according to the users' needs.
- Laws and standards have to be met.
- An adequate balance between pictures and text has to be found.

- The document should be appealing and as short as possible.

All technical documents emphasize understandability as a key aspect of their structure. The central goal of technical documentation is that end users as well as employees of different company departments can understand it without lots of additional research. This point ensures that texts are mostly written in a clear, simple and concise way and do not contain too many abbreviations or internal corporate terminology. Furthermore, technical documents are normally reviewed multiple times before being published either to ensure their quality by using proof readers or to ensure their understandability through audience analysis [40, p.58 et seq.]. Nevertheless, the analysis of technical documentation comes with many challenges for machine translation systems, since the used language will be highly technical and not every abbreviation can be avoided.

To ensure compliance with high quality requirements and to be legally safeguarded, several product specific standards must be met to achieve compliance with relevant laws and avoid possible compensations claims. Within the European Community, multiple binding directives are adopted into national law and are often complemented by common industry standards [40, p.5 et seq.]. Concluding, the extensive requirements of technical documentation is expected to offer high quality text passages in a complex environment and therefore build a solid but challenging ground for machine translation systems.

Chapter 3

Method

In this chapter we address the different steps taken to perform the creation of a machine translation evaluation system with focus on technical documentation. The following questions are answered over the course of this chapter:

- How can new knowledge be discovered?
- What goals and limitations influence the chosen method?
- What method provides the possibility to answer the research questions and how is it composed?
- What data attributes are required to apply the method?
- How can discovered knowledge be further applied?

In order to answer these questions, the structure of this chapter reflects an adaption of the earlier presented knowledge discovery process. The performed process is illustrated in the following figure: The first step of the KDD process is discussed in section 3.1. It defines the data mining task and outlines the restriction it is performed under. Afterwards, the choice of an appropriate data set and the preparation of the raw, gathered data is presented. Section 3.3 describes the phase of choosing the most suitable attributes for the given task, calculating and gathering the values for each data entry and preprocessing the data by finding outliers, noise or erroneous data. Section 3.4 describes the choice of a data mining approach, the considered algorithms and techniques as well as the actual data mining. The interpretation of the calculated results is described in section 3.5. This section also considers how the algorithm results were analyzed, which measures were used to compare different result to one another and how the algorithm parameters were optimized. The final KDD-step consists of using the results for future research or other actions. As further described in section 3.1, the evaluation of documents is done using results of the classification task. Therefore, the taken approach for the evaluation of technical documentation is described in section 3.6. Additionally, this section deals with the

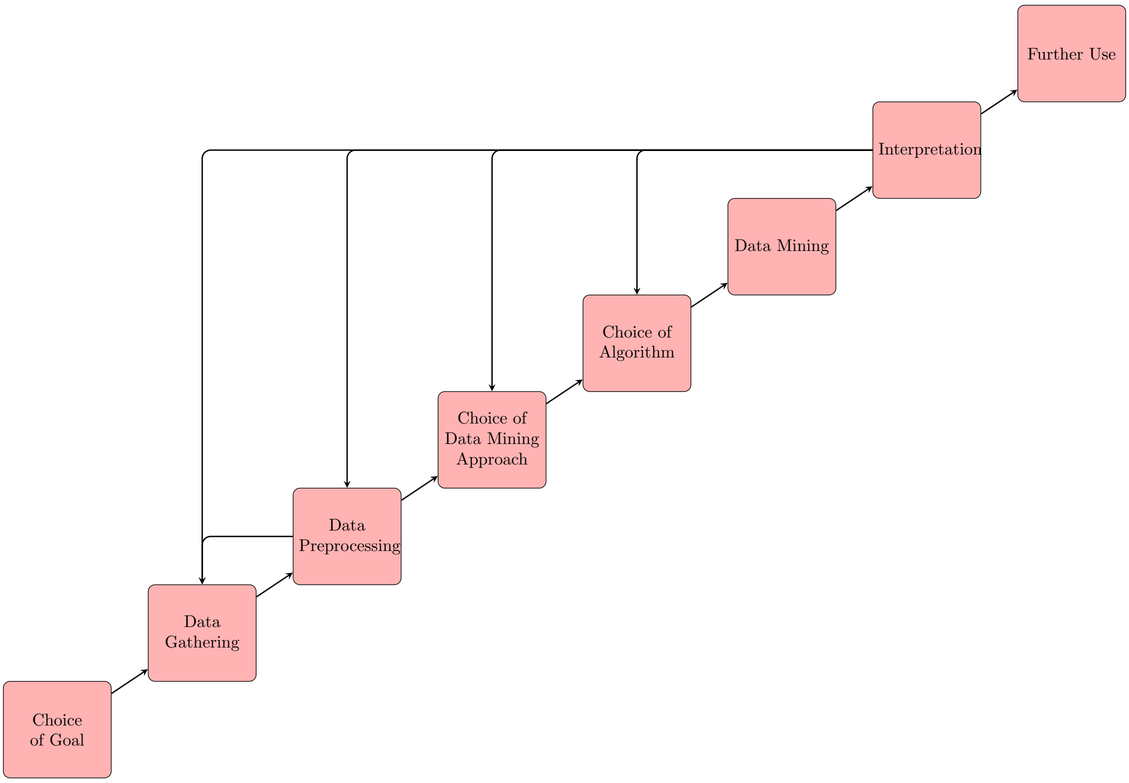

Figure 3.1: Adapted knowledge discovery process, containing the taken steps from gathering the data to interpreting the results and using them for further tasks.

recombination of classified sentences to documents in order to gain additional information concerning the classification quality on a document level of the presented algorithms. Due to the characteristics of the used data sets, not all initial KDD steps are required. For example, reducing the data and replacing missing values, a common step in the theoretical KDD-process, are unnecessary as missing values and unneeded attributes do not occur. Given the iterative nature of knowledge discovery processes, our work is split into two main loops:

- The first loop includes the KDD-Steps 2 and 3. In contrast to common knowledge discovery processes, the data set was created and suitable attributes were chosen manually. This resulted in a high dependency between the preprocessing step and the data gathering step, because the optimization of the data set by removing erroneous data entries or improving the overall data point quality, can change the suitability of chosen attributes or increase the required amount of data. Furthermore, with the configuration of a data set comes the decision about sizes of each data entry. After choosing suitably looking attributes, the length might have to be readjusted in order to fit the attributes more precisely. This creates the need for an iterative process between the choice or creation of an appropriate data set, containing the translated text fragments, and the choice of attributes.

- The second loop includes the KDD-steps $2-8$, which allows the readjustment of the data set, the preprocessing, the attribute choice or the machine learning algorithm. This loop is even more essential to our work, since the main goal is the optimization of prediction qualities. In order to do so, we use the presented process iteratively, testing multiple algorithms to compare them to one another. Furthermore, the algorithms' results can indicate a badly chosen database and therefore create the need for readjustments in the steps 2 and 3.

3.1 Identification of the Data Mining Goal

The predefined goal of this thesis was to evaluate the quality of technical documents and their translations using machine learning methods with a focus on detecting the difference between automated translations and professional translations done by humans. As mentioned in 1.3, the thesis addresses two research questions:

1. How can the translation quality of technical documents be evaluated, given the original document is available?
2. How can the translation quality of technical documents be evaluated, given the original document is not available?

To answer these questions, they were broken down into these practical working steps:

1. Provide a machine learning algorithm with optimal prediction quality for identifying professional and automated translations of technical documents with and without access to the original document.
2. Introduce a proper framework for ranking the quality of technical documents and fragments of technical documents regardless of their translation type.

This splitting was chosen, because the first steps including the data creation and preprocessing are rather similar among the two research questions. Therefore, it is more efficient to build a setup for providing a machine learning algorithm that solves the given data mining task with and without knowledge of the original document at once. The main difference between the two variants lies in the choice of attributes, with substantially more variables being allowed for the setup with knowledge. The creation of a new framework for ranking the quality of technical documents will then be built on the outcome of our first working step. Therefore, the knowledge discovery process focuses on the first working step and the second one is solved by using the achieved results.
An important limitation, as mentioned in section 1.5, is that this work focuses on the evaluation of syntax and does not take the semantic parts of a text into account, since this would go beyond the purpose of this thesis.

3.2 Translation Data

In order to evaluate machine translations and start a new knowledge discovery process, it is indispensable to gather a sufficient amount of raw data and to assign a basic data structure. Given the setup of the present task, the following questions will be answered in this section:

- What kind of data is required?
- What structure fulfills the requirements for the present study?
- Where does the data come from?
- Which data needs to be generated?
- How can data be gained and how is it possible to achieve a sufficient quality?
- How many data sets are required to make creditable statements?

Classic data mining tasks require the following kinds of data: A training data set, used to train the algorithm on the problem specific data and a test data set to evaluate the algorithm's quality. As mentioned in section 3.1, the given task is aiming for a binary classification of technical documents with the classes being *automated* and *professional* translation. Therefore, during the data gathering step it was required to generate at least a professional, human translation and an automated translation for each German text of the database. Due to the fact that many of the presented metrics require a reference translation in order to evaluate a candidate text fragment, it is necessary to provide an additional translation that can be treated as a reference to compare the candidates to. In conclusion, the final translation data set contains the following parts for each entry:

- The original, German text.
- A professional or automated translation as a candidate to be evaluated.
- One or multiple reference translations of the same original text, used to calculate scores. Classic machine translation evaluation algorithms often require high quality reference texts. The references are then used by the algorithms, such as BLEU or Meteor, to calculate quality scores.

Consequently, the documents to gather should already be available in an original German version and one or more matching professional translated English versions, since the generation of additional professional translations of texts would go beyond the purpose of this work.

The first attempts of gathering data showed a lack of freely accessible data to perform evaluations on a document level. Due to the discrepancy between the small amount of available documents and the high amount of required data for effective data mining, the documents were broken down into sentences and the data set was built on this smaller logical unit, the sentence level. This extends the purpose of this work, to provide an additional classification system that is able to binary classify sentences into the two labels mentioned above. By recombining sentences into the original documents, evaluation of document quality is still possible. This break down enlarges the data set significantly, but creates the additional challenge of gathering sentence by sentence translations of the documents.[1]

Text Extraction and Sentence Splitting

After thorough research, it was found out that an appropriate number of freely accessible technical documents, including a professional translation, are only available in PDF format. Due to PDF being a visualization focused document format, extracting single sentences from it is not a straight forward task. Following, the steps performed to extract and match sentences from a German document and its English translation are presented in detail.

First, the available PDF files are converted into plain text files. Due to the characteristics of the PDF format, the resulting text file is highly erroneous and not logically ordered, such as footnotes being integrated into sentences and sentences or words being split and separated. A helpful characteristic of the used documents were easily distinguishable paragraphs.

Therefore, the plain text files were divided into their respective paragraphs, where paragraphs are defined as a set of multiple consecutive text containing lines without interrupting empty lines. In order to find and remove paragraphs that do not contain actual sentences, research showed that the existing paragraphs could be filtered by three requirements. They needed to consist of at least 15 characters, three blanks and one punctuation mark. Otherwise, they contained most likely not actual text but rather not usable text, such as footnote entries, headlines and figure captions.

After ensuring that the remaining paragraphs contained nothing but usable content, they were divided into single sentences using part of speech tagging algorithms. Determining the parts of speech tags in a sentence, allows punctuation independent sentence splitting, which is very beneficial in the context of technical documentation. This is because of their technical nature, resulting in multiple technical punctuations within the documents, which makes the splitting of sentences at punctuation marks ineffective.

Afterwards, the extracted sentences were analyzed further, in order to remove final chunks and to ensure, that the extracted lines actually form complete sentences. Therefore, the sentences were filtered on having more than two blanks, to end with a punctuation and to

[1] The used documents were a set of manuals from VMWare Inc., concerning their product VSphere, a virtual machine for x86-64-based hardware and their operating systems. The technical documents are accessible via: https://pubs.vmware.com/vsphere-51/index.jsp?topic=%2Fcom.vmware.vsphere.doc%2FGUID-1B959D6B-41CA-4E23-A7DB-E9165D5A0E80.html, last accessed: January 19, 2016

contain at least seven characters. Further requirements, such as verb existence and capitalized sentence starts proved to be ineffective, due to the processed documents containing multiple lists and trademarks that interfered with those rules.
To finalize the raw data gathering, the extracted sentences needed to be associated with the proper translation. However, it was not possible to set up a sentence extraction and splitting process that guaranteed to build the exact same number of extracted sentences the German and English version. Therefore, the German sentences were translated by automatic translation systems into English ones, which were in turn used to build a process that detects whether two translated sentences could result from the same original document. This allowed a sorting of the data set, matching the respective sentences to one another and removing sentences with no match on either side.
Finally, for every German sentence three independent machine translations were generated to create a sufficient amount of automated translations and resolve the problem of a missing professional reference translation. Since it was not possible to create an additional professional translation for the given data set, we used two approaches to address this problem:

- First, one of the automated translation sets was used as a reference. Most translation evaluation metrics expect a high quality reference to compare the sentence to. Example given, the BLEU score calculates a similarity value, with a higher value meaning a higher similarity to the given reference. This means that if the reference is not of high quality, the similarity of low quality sentences can actually be higher to the reference than the one of high quality sentences. Due to the fact that the first research question tackles the problem of binary classification and not of evaluating the quality of the given texts, the reference does not need to be of high quality. The expected result would be that by using an automated translation as reference, automated candidate texts will achieve higher similarity scores than the professional ones.

- The second approach is based on the findings by Albrecht and Hwa from 2007. They state that the combination of multiple pseudo references can result in a high quality reference, similar to a human-created translation. Hereby, a pseudo reference is a reference that has not been created by a human, in this case a machine translation program [8, p.298 et seq.]. Therefore, two automated translations were used in combination to form the reference translation. In this case, the expected results were higher similarity scores for professional translations than for automated translations, since the combination was expected to generate a reference of higher quality.

Furthermore, to create additional attributes that do not need any knowledge of the original document, a round-trip translation approach was used to create a second version of the candidate sentence, which will be referred to as round-trip reference. This was done by translating the given candidate sentence into a foreign language and back to English. By doing so, the sentence was clearly altered, since round-trip translations are generally producing erroneous data as mentioned in 2.5.1. But this allowed the creation of a reference text fragment that could be used as a comparison for the candidate translation without knowledge of the original document. Therefore, the algorithms that normally require a reference translation could be calculated for the research question that does not allow the use of the original document as well. In order to create a sufficient amount of automated translations the following machine translation systems were used:

- SDL's freetranslations.com[2]
- Google's GoogleTranslate[3]
- Microsoft's Bing Translator[4]

To reduce possible dependencies from chosen automatic translators, the further steps use nine different kinds of data combinations, where each machine translation is once used as candidate with the other two each being used individually and in combination as a reference translation. However, due to the need of a round-trip reference, a third of the data sets, containing the Freetranslation system as a candidate, was removed in the early stages of the optimization process. The following figure illustrates a data set, which will be referred to as the preprocessed translation set:

German Sentence	Candidate Sentence	Reference 1	Reference 2
German Sentence 1	Prof. Trans.1	Auto. Trans. 1 Google Translate	Auto. Trans. 1 Freetranslation
German Sentence 1	Auto. Trans.1 Bing Translator	Auto. Trans. 1 Google Translate	Auto. Trans. 1 Freetranslation
German Sentence 2	Prof. Trans.2	Auto. Trans. 2 Google Translate	Auto. Trans. 2 Freetranslation
German Sentence 2	Auto. Trans.2 Bing Translator	Auto. Trans. 2 Google Translate	Auto. Trans. 2 Freetranslation
...	...	...	...

Table 3.1: General setup of a data set.

[2] accessible via: http://www.freetranslation.com/de/

[3] accessible via: https://translate.google.com/

[4] accessible via: https://www.bing.com/translator

3.3 Choice of Attributes

This section deals with the selection, creation and preprocessing of the attributes in the data set. Thereby, the following questions will be answered:

- Which attributes where chosen for the given task and what is their main information contribution?
- How can these attributes deal with multiple reference translations?
- What were the main problems encountered during this phase?

The choice of attributes is an essential part of knowledge discovery. The aim of an attribute is to generate some form of knowledge concerning each data entry. Normally, an attribute is a variable which has a value for each data entry of the data set and can be calculated or accessed in some form. Furthermore, the chosen attribute type is an important factor, since not every algorithm can deal with any type of attribute. The two main types are nominal and numerical attributes. Nominal attributes consist of a fixed set of normally not sortable values, while numeric ones are of continuous and infinite nature. To gain additional knowledge of the used metrics, such as the *Meteor score*, the intermediate steps of the calculation process were added as well. Following, the chosen attributes will be shortly described:

- **Modified Unigram BLEU Score**
  As mentioned in subsection 2.5.4, the bilingual evaluation understudy algorithm is a fast and inexpensive method for evaluating the quality of machine translations in a fully automated manner, by calculating a value between 0 and 1 depending on "closeness" of reference and candidate. The unigram BLEU score uses *1-grams*, meaning that no sequences of words are taken into consideration, but only single words for calculating the modified BLEU score.

- **Modified Bigram and Trigram BLEU Score**
  In order examine the difference between single word BLEU scores and word sequence BLEU scores for the given data set, the BLEU scores for *2-grams* and *3-grams* were calculated and added as attributes.

- **Meteor Score**
  As described in subsection 2.5.6, the Meteor score is a version of the BLEU score that has been optimized to work on sentence level.

- **Meteor Recall**
  The Meteor recall score describes the amount of similar unigrams in candidate and reference translation, relative to the unigram count in the candidate fragment.

- **Meteor Precision**
  The Meteor precision score describes the amount of similar unigrams in candidate and reference translation, relative to the unigram count in the reference fragment.

- **Meteor Chunks**
  The Meteor Chunk score counts the minimally needed amount of chunks for each sentence.

- **Meteor Fragmentation Penalty**
  The fragmentation penalty calculates penalties to give more weight to longer *n-gram matches*. This means that a sentence with large amounts of longer *n-grams* in candidate and reference translation, requires fewer chunks, which in turn results in a lower penalty score.

- **Meteor F1**
  The F1 score calculates the harmonic mean between precision and recall.

- **Meteor Mean**
  The Meteor Mean score computes the harmonic mean between precision and recall, with recall being weighted nine times more than precision.

- **Meteor Total Matches**
  The total matches score counts the total matches found between reference and candidate translation on a unigram basis.

- **Reference Length**
  The reference length score compares the candidate translation length to the reference translation length, by generating the difference between two text fragments and storing it as the attribute value. This results in negative values for shorter candidates and in positive ones for longer candidates.

- **Parts of Speech Score**
  The parts of speech score is a boolean value for describing whether the candidate translation matches a given minimal pattern of needed parts of speech tags to form a grammatically correct English sentence.

- **Flesch Reading Ease**
  The Flesch reading ease algorithm calculates a score that measure the readability of sentences and documents.

- **TER Score**
  As described in subsection 2.5.3, the Translation Edit Rate measures the amount of needed edits to transform the candidate text to the reference translation, relative to the text length.

- **TER Total Edits**
  The intermediate step of calculating the total edit count was implemented to generate information on the amount of edits with no regard to the reference length. By not using a relative score, the lengths of candidate and reference translation are taken into account, which results in additional knowledge for the machine learning algorithm.

- **Used References**
  As described in section 3.2, for parts of the experiment, multiple reference translations were used to generate a more standardized reference version. Most metrics, such as the Meteor and BLEU score are able to cope with multiple reference translations, by calculating scores for all references and choosing the best score. We monitored, which reference translation was used more often during the calculation of all metrics to generate additional knowledge that could ease the classification process for the given machine learning algorithm.

- **Mistake Count**
  A style word and grammar checking system was used to analyze the candidate text fragments on their word and grammatical correctness (*language tool*). The program counts all found mistakes and separates them into different categories. In addition to the total mistake count, the individual categories were added, resulting in an additional 94 attributes.

One of the main problems that appeared during this phase, was finding an appropriate number of attributes that fit the requirement of having no knowledge of the original document. The removal of the original document as a resource, transforms the problem from a machine translation task to a text quality analysis task, since there is no more information available to the algorithms that the given text is a translation of any kind. Therefore, the field of machine translation is not directly applicable for solving this part of the research question. The main properties for good text quality are grammatical correctness, word correctness, semantic correctness and style. As mentioned in section 1.5, the semantical correctness of a sentence is going beyond the purposes of this thesis and therefore cannot be taken into account. This leaves the part of the research question without knowledge of the original document with three types of problems to address:

- The grammatical correctness of a sentence or text can be validated or evaluated. This is normally done by rule-based programs that check the sentence on violations of these rules and report the corresponding errors.

- The word correctness is harder to evaluate than the grammatical correctness for the given task. The focus on technical documentation results in some properties of the documents that complicate these evaluations. First, the used language is of highly technical nature, resulting in many unknown words for common dictionaries and second, technical documentation often focuses on a specific product in a company and therefore contain many proper nouns that would not even be detected by dictionaries that are specialized for technical documents.

- The style of a sentence is in general not applicable for detecting false sentences, however metrics, such as the readability algorithm can still help to generate information about the document quality.

As described above, the possibilities for document quality evaluation without knowledge of the original text, are limited. In contrast to that, the performance and prediction quality of a machine learning algorithm highly depend on the available attributes to train and test on. This is a problem for solving this part of the research question that could endanger its success. In order to address this problem, a round-trip reference was created as mentioned in section 3.2. This allowed the additional creation of the attributes mentioned above without usage of the original document, resulting in an additional 14 attributes that could be used by the machine learning algorithms.

For the preprocessing step of the attributes, the main concern for the presented task is the detection of outliers among the attribute values. A value is considered to be an outlier, if it deviates by a significant margin from the average of its nearest values. For the given task, the outliers were calculated using the Class Outlier Factors (COF). The COF is calculated by ranking each instance of the data set using the following formula [41]:

$$COF = PCL(T, K) - norm(deviation(T)) + norm(kDist(T))$$

- $PCL(T, K)$ is the probability of the class label of the instance T with respect to the class labels of its K nearest neighbors.

- $norm(Deviation(T))$ and $norm(KDist(T))$ are the normalized values of $Deviation(T)$ and $KDist(T)$ respectively and their values fall in the range $[0 - 1]$.

- $Deviation(T)$ is how much the instance T deviates from instances of the same class. It is computed by summing the distances between the instance T and every instance belonging to the same class.

- $KDist(T)$ is the summation of the distance between the instance T and its K nearest neighbors.

Other important parts of the preprocessing step are the handling of missing values, the removal of duplicates, normalization and the detection of correlating attributes. Since the data set has been created and the attributes were chosen and calculated manually, there are no missing values in the data set after calculating the metrics for the preprocessed

translations. Therefore, missing value treatment is not necessary for the given task.
The main benefit of normalization is comparability, which is mainly a concern for machine translation metrics, such as BLEU, Meteor and TER, which are all normalized by default. However, in order to perform outlier detection, it is important to have comparability among all existing attributes and therefore a $range\ transformation$ is performed for the given data set, mapping the data to values between 0 and 1.

3.4 Specification of a Data Mining Approach

This section deals with the KDD-steps $5 - 7$, which includes the choice of a data mining approach, the choice of an algorithm and the actual data mining step. The following questions will be answered in this section:

- Which and how many data mining techniques were considered suitable for the given task?
- How were the algorithms optimized for the data set?
- What were the main problems encountered during this phase?

Choosing the most suitable machine learning technique comes with a trade-off between the different advantages and disadvantages that each approach possesses. Therefore, multiple approaches were tested for the given task, the results analyzed and the algorithms' parameters optimized. The following techniques were considered suitable for the given task:

- **Decision Trees** As mentioned in subsection 2.3.1, the main advantage of Decision Trees is the understandability of the resulting model and the analysis results, since the relevance of the respective attributes is easily accessible by humans. Additionally, the robustness of Decision Trees when it comes to outliers and missing values is a huge benefit for data mining tasks. In contrast to that, the main disadvantages of Decision Trees are their unreliability and their challenging setup to be used efficiently. Unreliability means that the splitting attribute in a high-level node can change by only adding a few additional data entries, changing the structure of the whole tree and the corresponding predictions. Furthermore, the setup to use a Decision Tree effectively is rather complex due to the amount of parameters to adapt the machine learning process.
  Used Algorithm: C4.5 (Quinlan)[5]
- **Artificial Neural Networks** The main advantages of neural networks are their ability to implicitly detect nonlinear relationships between attributes and the detection of interactions between input variables due to their hidden layer structure, resulting

[5] Similar to an implementation accessible via: https://github.com/scottjulian/C4.5

in generally high prediction accuracies. These have to be put in contrast to their high demand of computational resources, resulting in high computation times and their proneness to overfitting the model [42].
Used Algorithm: Feedforward Neural Network with Backpropagation

- **Bayesian Network** For the Bayesian Network, the Naive Bayes algorithm will be used. As explained in subsection 2.3.3, the Naive Bayes algorithm is rarely relevant in terms of prediction quality, due to its approach of assuming independency of all input attributes. Nevertheless, since the required resources are very low it has been included in the examined algorithms.
Used Algorithm: Naive Bayes

- **Instance-Based Learning** The k-NN algorithm is a lazy learner, which means that no learning or computation is done during the learning phase of the algorithm. Instead, the computation of the k nearest neighbors is done, when the corresponding query is made. The main advantages are highly correlated with the lazy nature of the algorithm, being that the training set is easily scalable, since there is no learning process involved and it is very robust to outliers by classifying new instances based on a distance metric to its most similar data entries. The main disadvantages are the high computation time during the classification phase, especially for large data sets and the sensitivity towards meaningless attributes and uneven class distributions.
Used Algorithm: k-Nearest Neighbor

- **Support Vector Machines** Support Vector Machines are flexible due to their ability of mapping data to a higher dimensional space for enabling an easier separation between the labels (2.3.5) and are defined by the use of quadratic programming which avoids local minima in the learning process. In contrast to this, Support Vector Machines require an in-depth understanding of the kernel techniques and the learning and test process to be used efficiently. Furthermore, the setting of parameters, such as the kernel type and SVM type can result in substantially deviating classifications.
Used Algorithm: mySVM (Rüping)[6]

For each of the approaches mentioned above, one particular implementation of an algorithm was chosen and used to predict the outcomes of the given data set. An important aspect, when it comes to building a classifier, is the tuning of the different parameters for every algorithm. The parameters were optimized using a combination of grid-based evaluation and evolutionary evaluation. First, a tournament-based, evolutionary optimization was performed, by building the respective algorithm a certain amount of times, while evolutionary adapting its parameters, until a certain number of generations had been created or no improval in terms of prediction quality was monitored for more than a fixed amount of generations. The used prediction quality to validate the performance of the algorithm

[6] accessible via: http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/index.html

was prediction accuracy. After determining the general area, where the best possible values for each parameter lie, had been found, a grid-based optimization followed, testing multiple values in range of the evolutionary result to affirm the optimal value had been found. (E.g., if the maximal depth of the Decision Tree was optimized towards a value of 25, the grid-based approach tested an additional 10 values between 10 and 40 to further optimize the Decision Tree.)
Since there were nine available data sets, this process was done for all of them to examine the significance of the different reference translations in correlation with the used candidates. The results of these experiments will be presented in section 4.2. There were several challenges that appeared during this phase. First, the computation times for algorithms, such as neural networks that require a lot of CPU and RAM resources, posed a problem for the given optimization setup. This is, because the optimization process builds many models, which in turn require a full recalculation of the classification approach for every new parameter setting. This challenge could partly be solved by allocating more resources in terms of time and CPU to the respective algorithm, but still set a certain amount of limitations to the given setup. Second, due to the fact that multiple algorithms, such as neural networks can only handle a certain type of variables (in this case numeric attributes), some transformation steps had to be included in order to convert nominal to numerical attributes and vice versa.

3.5 Interpretation

In order to determine, if and which iterative loops have to be done after performing the data mining step, certain metrics are needed to evaluate the results of the different algorithms. These metrics have been described in subsection 2.3.6 and were used to determine the needed adjustments in the preprocessing and data mining steps. The first and most significant metric is the misclassification rate. Since the label distribution is exactly even for the given data set, a low misclassification rate or respectively a high accuracy rate can be treated as an algorithm result of very high quality. To improve the validity of algorithm accuracy, the results were put into context with the results of a dummy classifier as a corresponding benchmark. The dummy classifier always predicts the most common class for the whole data set, resulting in a classification accuracy of 50%. By calculating the difference between the algorithms' results and the dummy classifier, an accuracy gain was calculated to generate a more explicit insight on the additionally gained knowledge by using the algorithms. Although the class distribution is equal for the given data set, it is still of great interest to determine, if an algorithm is considerably better at detecting automated translations or is considerably better at detecting professional translations. This can be done by monitoring the achieved precision and recall of each algorithm. Only calculating the precision is not sufficient, since, although precision gives the amount of correctly classified instances for a single instance relative to all instances classified with that label, a high precision could also be achieved, by only classifying the respective label in very specific scenarios and thereby missing out on many instances that actually belong to that label. In order to counteract that, the recall can be taken into account in combination with the precision score. Since the recall measures the correctly classified instances relative to all instances that actually belong to that class, the strategy described above would receive a substantially lower recall score. In order to combine both values, the F-Measure is used, calculating the harmonic mean between the two scores, resulting in a score to evaluate the ability of an algorithm for classifying a certain class. To take both classes into account, the F-Measure was calculated for both labels and the most promising results regarding F-Measure and Accuracy-Gain were further examined.

3.6 Further Use of the Results

The last step of every knowledge discovery process is the application of the obtained results for further use. In the proposed task, this is especially important, since the main focus of this thesis was the classification of document level translations and additionally the evaluation of translations. This section explains, how the KDD-process results are used to perform document level classification and how to create a framework for the evaluation of translated sentences and documents.

3.6.1 Document-Based Analysis

In order to predict original documents, the single sentences have to be recombined to their respective technical documents. This results in a professional and an automated version for each technical document. In order to classify a document with a certain label, all sentences belonging to that specific document were used as holdout, while combining the remaining documents to the training data set for the algorithm. (E.g., a professionally translated document called *installation setup* is held back as the holdout set and the remaining documents are combined to build the training set for the algorithm.) The algorithm then classifies each sentence of the holdout set with a certain label and the document is classified with the most common label among its sentences. This process is repeated for every version of every technical document and the misclassification rate is calculated by putting the amount of correctly classified documents in relation to all documents.

Due to the small amount of available documents, a second step was taken to prove the results' validity. We created an additional database of technical documents, by randomly combining sentences to fictive documents. Figure 3.2 illustrates the process of creating additional fictive documents.

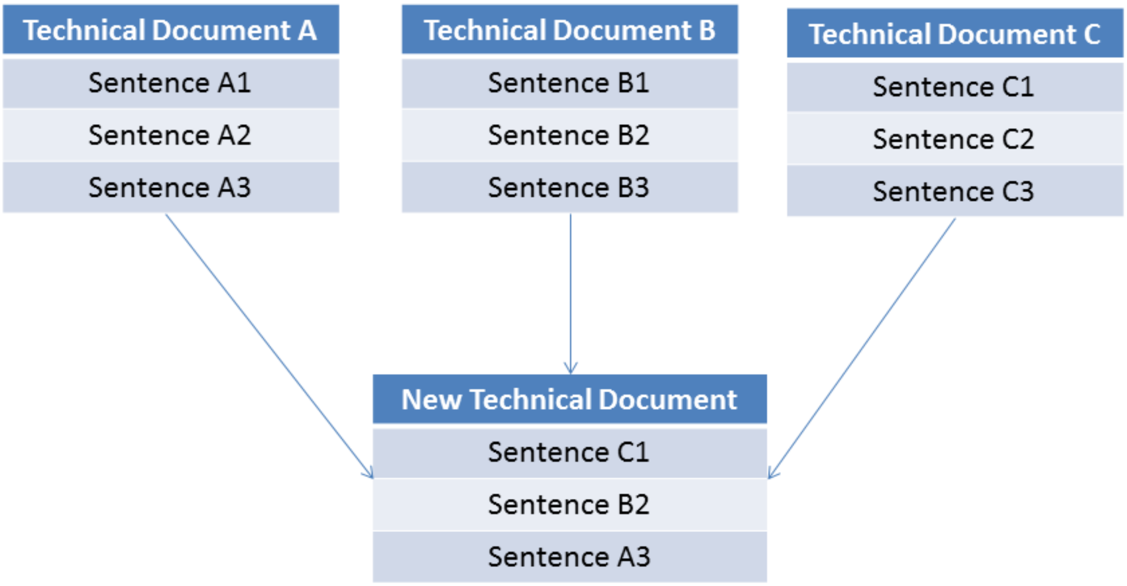

Figure 3.2: Technical document creation.

Additionally, this approach allowed the examination of the required size of technical documents. By creating documents of substantially differing sizes, it was possible to determine a minimal length of technical documents in order to have a high probability of classifying them correctly. Furthermore, this approach allowed the detection of the needed distribution rate of the document's sentences depending on the document's size in order to classify them with a certain label.

3.6.2 Proposal of an Evaluation Framework

The second practical working step to answer the given research questions, was the proposal of a framework for evaluating technical documentation with no regard to their type of translation. By using the results from the knowledge discovery process, it was possible to generate classes that allow the subdivision of sentences and therefore documents, according to their quality of translation. The three used properties of the classification process for the evaluation were the predicted label for the two best performing algorithms and the amount of mistakes monitored for the sentence. To get a more appropriate representation of the predicted label and the monitored mistakes, the mistake count was split down into a more fine-grained attribute, while the class predictions for the two algorithms remained unchanged. The resulting 94 categories were analyzed concerning their impact on the quality decrease of a sentence. Each category was weighted depending on their severeness, giving high impact mistakes additional weight and low impact ones less or even no weight if the mistake was not relevant for the quality of a sentence.

The prediction of a sentence being professionally translated was treated as a high indicator for a high quality sentence, due to the fact that in general, a human translation is of higher quality than an automated translation.

Therefore, the three resulting properties for the quality evaluation of a sentence were the two data mining predictions and the weighted score of its mistakes. A sentence was labeled with a quality class depending on the values of the three mentioned attributes. The exact classes will be shown in section 4.2.

Chapter 4

Results / Empirical data

This chapter will give an overview over the achieved results, the used data and the experiment process to solve the given research questions. The following questions will be answered in detail, with section 4.1 focusing on the setup of the data set and the experiments and section 4.2 presenting the achieved results:

- How many sentences and documents were used for training, testing and analyzing the algorithms?
- Which tools were used to calculate the metrics, create the data set and perform the experiments?
- Which algorithm parameters were the most influential?
- Which algorithm optimizations were done?
- What were the achieved results on sentence level analysis?
- What was the setup of the best performing algorithm on sentence level?
- What were the achieved results on document level?

4.1 Empirical Data

The following section gives the numbers to the approach described in chapter 3. Table 4.1 gives an overview over the amount of sentences, attributes and documents used for the given tasks.

Number of Automated Translation Systems	3
Number of Technical Documents	14
Original Number of Sentences in all Documents	$30,000$
Number of Sentences used per Translation System for the Data Sets	$22,327$
Approximate Number of Sentences in the Data Sets	$44,654$
Total Number of Data Sets	9
Number of Attributes needing a Reference Translation	14
Number of Attributes needing no Reference Translation	18
Total Number of Attributes	32
Maximum Amount of Available Attribute Values	$1,428,928$
Number of Created Fictitious Documents	$19,190$

Table 4.1: Statistics of used data sets.

The sentence extraction of technical documents resulted in 30000 lines containing text fragments. Since sentence extraction is not a straight forward task, 8000 of these lines had not been extracted correctly, resulting in erroneous text fragments that did not form valid sentences. Therefore, each final data set that was used to train and test the machine learning algorithms, consisted of 22327 sentences per translation system. To ensure an even distribution between the labels *professional translation* and *automated translation*, each data set was combined out of two sets of sentences, one being translated professionally and the other automatically. This resulted in each data set containing a total amount of 44654 sentences. These sentences were extracted out of 14 technical manuals regarding the handling of a virtual machine program.
To further examine the algorithm's validity on different sizes of technical documents, the manufactured documents were varied in sizes of 5 to 3000 sentences per documentation. It is important to note that the creation of larger sized documents was more challenging, due to the limited amount of sentences and the needed amount of training data to generate a meaningful classification model. Furthermore, the information gained by the smaller sized documents is more valuable, since the needed amount of sentences in order to correctly classify a document, was expected to range between 60 and 300 sentences.
Table 4.2 shows the nine used data sets, created by using the three machine translation systems as candidates and references respectively and creating a round-trip translation for six of the nine data sets using the Freetranslation program.

Candidate	Reference1	Reference2	Round-Trip Reference
Google Translate	Bing	—	Google RTT via Freetranslation
Google Translate	Freetranslation	—	Google RTT via Freetranslation
Google Translate	Bing	Freetranslation	Google RTT via Freetranslation
Bing Translator	Google	—	Bing RTT via Freetranslation
Bing Translator	Freetranslation	—	Bing RTT via Freetranslation
Bing Translator	Google	Freetranslation	Bing RTT via Freetranslation
Freetranslation	Google	—	—
Freetranslation	Bing	—	—
Freetranslation	Google	Bing	—

Table 4.2: Used combinations of references and candidates.

The attribute calculation and the generation of the database in combination with the setup from section 3.2, was done in Java, using Eclipse as a development environment. The described metrics were integrated using open-source packages, such as the *Stanford NLP Package* and the *Language Tool Core Package*. Most of the metrics were adapted to fit the requirements of the given machine learning task and calculated to form the final database used to train and validate the machine learning algorithm on.

Before the optimization of each algorithm was performed, the data set was further prepared to convert it into an optimal setting for the algorithm to train on. These steps including the optimization process were done using the RapidMiner software and are shown in detail in figure 4.1.

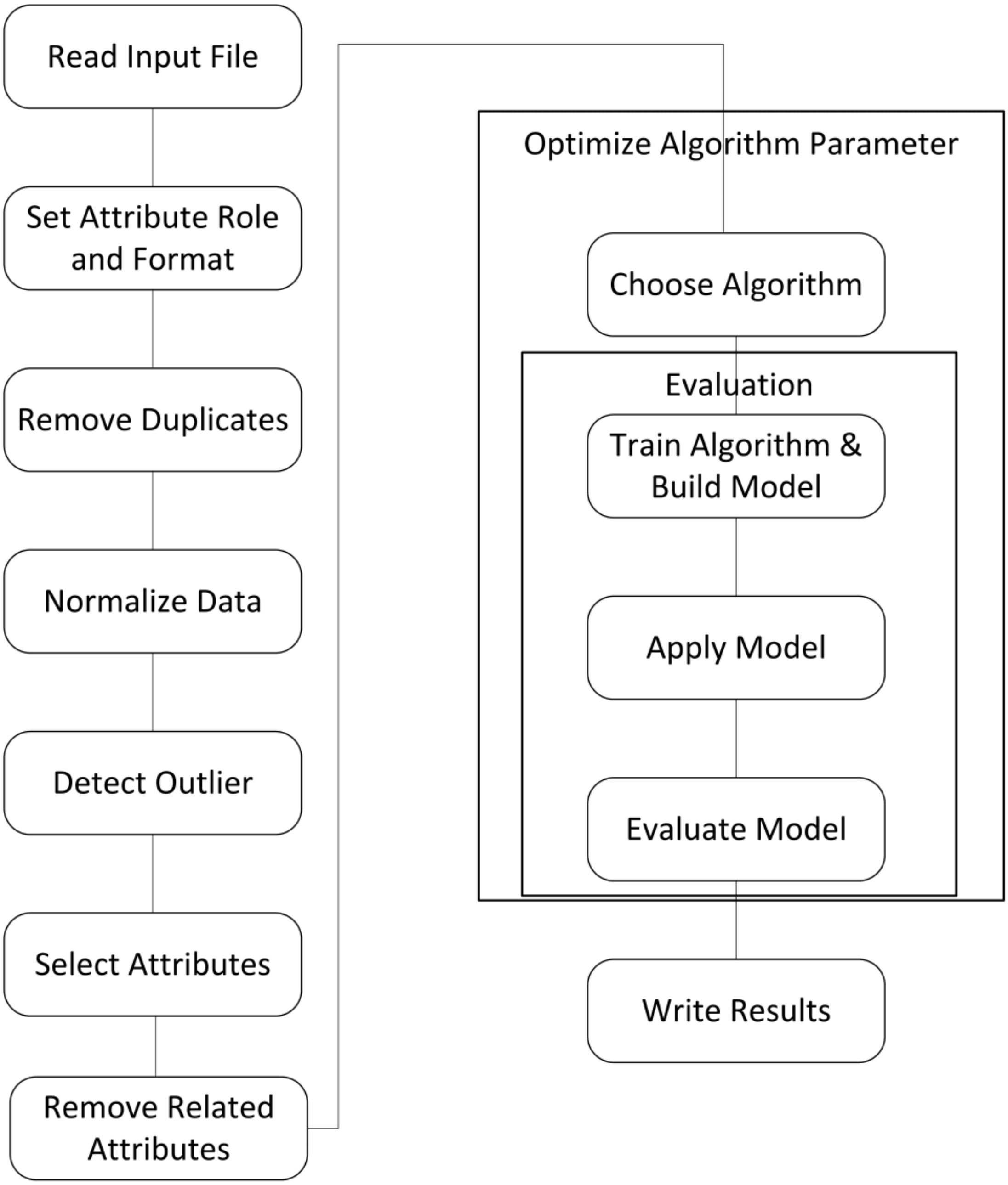

Figure 4.1: Preprocessing of the data set and optimization of the machine learning algorithm using RapidMiner.

After the input data set has been read and the attribute types have been set, the first preprocessing step, *Remove Duplicates* is run. This step removes all but one of similar entries in the data set, which is important to limit the influence of multiple identical data entries. Next up, the data is normalized using the *range transformation*. This maps all attributes to the same value range, which allows for an easier comparison between them and enables the next step, the detection of outliers. Outliers are detected using class outlier factors as described in section 3.3 by comparing data entries to one another using the euclidean distance metric. The 5% most deviating data entries are identified as outlying values and are not considered for the machine learning task. Furthermore, depending on the research question, the attributes that are allowed for the given task are selected, which includes the removal of 14 attributes for the second research question. The final preprocessing step, to further decrease computation times and remove redundancies among the attributes, is the identification of highly correlating attributes and their removal if the correlation is higher than 90%.

To evaluate the influence of the steps mentioned above, the optimization process was performed for different setups by enabling and disabling certain steps, example given, *Remove Duplicates = true, Normalize Data = false, Detect Outlier = true, Remove Related Attributes = false*. An optimization iteration consists of setting the algorithm's parameters, building a model, applying the holdout set to the built model and evaluating the results. The holdout set was randomly selected as 30% of the data set. As mentioned in 3.4, this process was integrated into an automated setup, to perform multiple evolutionary-based optimization processes. The first step for each algorithm consisted of a minor optimization step to examine the influence of different data preparation setups and the general suitability of the algorithm for the given task. The experiments were performed on all nine data sets and it turned out that data preparation was useful for every setup and therefore normalization, duplicate removal, outlier detection and correlated attribute removal were applied for the next optimization steps. Furthermore, Support Vector Machines and the Naive Bayes classifier were removed from the further optimization process, since SVM achieved substantially worse results than the remaining algorithms and Naive Bayes had no further optimization possibilities due to the lack of adaptable parameters. The next step consisted of a more in depth optimization, by generating a larger amount of models per algorithm and testing bigger ranges of parameter settings. Table 4.3 gives an overview of the generated models for both research questions and the corresponding optimizations.

	Research Question 1	Research Question 2	**Total**
Built Decision Trees	$480,000$	$300,000$	$780,000$
Built Neural Networks	$6,000$	$2,000$	$8,000$
Built k-Nearest Neighbors	$3,500$	$1,500$	$5,000$
Built Support Vector Machines	$2,500$	$1,000$	$3,500$
Total Number of Optimizations	$492,000$	$304,500$	$796,500$

Table 4.3: Overview over the created models and optimizations.

4.2 Results

This section will present the results of the experiments. Subsection 4.2.1 will deal with the results concerning the algorithm having access to the original documents, while 4.2.2 will show the achievements for the same task with the algorithm having no access to the documents. Each subsection will present the best and most notable results for the tested algorithms and the resulting document-based analysis.

4.2.1 Research Question 1

The first research question focuses on the evaluation of translations with the knowledge of the original document. Following, the best results for each tested setup will be presented, ordered by the used algorithm. The exact ranges used for optimizing the models are listed in appendix A.

In order to provide a brief overview, table 4.4 shows an aggregation of the afterwards more detailed presented results and illustrates the general performance of the used algorithms.

Algorithm	Accuracy	Standard Deviation
Decision Tree	68.69%	0.014
Artificial Neural Network	**70.33%**	0.014
k-Nearest Neighbor	69.74%	**0.009**
Naive Bayes	65.84%	0.019
Support Vector Machines	61.31%	0.017

Table 4.4: Averages and standard deviations of the tested algorithms.

Table 4.5 shows the results with the highest accuracies for sentence predictions using Decision Trees including the respective F1-scores. For each candidate-reference combination shown in the table, $50,000$ Decision Trees were built and evaluated. The used data included all available attributes in normalized fashion, reduced by up to 5% by an outlier detection and further due to the removal of duplicates.

Decision Tree					
Candidate	Reference 1	Reference 2	Accuracy	F1-Automated	F1-Professional
Google	Bing	—	69.09%	0.724	0.649
Google	Freetranslation	—	67.91%	0.706	0.646
Google	Bing	Freetranslation	**70.48%**	0.744	0.651
Bing	Google	—	70.18%	0.706	0.698
Bing	Freetranslation	—	68.75%	0.692	0.683
Bing	Google	Freetranslation	70.23%	0.715	0.689
Freetranslation	Bing	—	66.22%	0.689	0.629
Freetranslation	Google	—	67.85%	0.687	0.669
Freetranslation	Bing	Google	67.52%	0.676	0.675

Table 4.5: Overview over the best Decision Tree results for the respective candidate-reference combinations.

The following table presents the highest results for artificial Neural Networks. Each candidate-reference combination was optimized 150 times. The data preparation is similar to the previous Decision Tree optimization. Each lineup contains all available attributes. The attributes are normalized and the data set is reduced by up to 5% outliers. Duplicates are not considered.

Artificial Neural Network					
Candidate	Reference 1	Reference 2	Accuracy	F1-Automated	F1-Professional
Google	Bing	—	69.93%	0.731	0.659
Google	Freetranslation	—	68.08%	0.709	0.647
Google	Bing	Freetranslation	70.93%	0.744	0.663
Bing	Google	—	**72.24%**	0.729	0.715
Bing	Freetranslation	—	69.14%	0.696	0.687
Bing	Google	Freetranslation	71.67%	0.723	0.711

Table 4.6: Overview over the best Artificial Neural Network results for the respective candidate-reference combinations.

The results for the used k-Nearest Neighbor algorithm is shown in table 4.7. For each candidate-reference combination, 50 models were built and evaluated. The results represent the respective models with the highest achieved accuracies. The data preparation is similar to the earlier mentioned setups and the data set is reduced by up to 5% outlier and due to the removal of duplicates, while the attributes are normalized to perform the outlier detection.

k-Nearest Neighbor					
Candidate	Reference 1	Reference 2	Accuracy	F1-Automated	F1-Professional
Google	Bing	—	68.41%	0.692	0.683
Google	Freetranslation	—	68.71%	0.706	0.698
Google	Bing	Freetranslation	**71.21%**	0.715	0.689
Bing	Google	—	70.23%	0.712	0.692
Bing	Freetranslation	—	69.55%	0.701	0.689
Bing	Google	Freetranslation	70.33%	0.718	0.694

Table 4.7: Overview over the best k-Nearest Neighbor results for the respective candidate-reference combinations.

The following table displays the results for evaluations using the Naive Bayes algorithm. Given the nature of the algorithm, the results are final for the used data set and are not further optimizable. The data set is similar to the earlier results: All available attributes are normalized and used. The data set is reduced by up to 5% outliers and does not contain duplicates.

Naive Bayes					
Candidate	Reference 1	Reference 2	Accuracy	F1-Automated	F1-Professional
Google	Bing	—	67.76%	0.697	0.655
Google	Freetranslation	—	66.08%	0.681	0.638
Google	Bing	Freetranslation	**68.76%**	0.717	0.651
Bing	Google	—	63.21%	0.671	0.582
Bing	Freetranslation	—	65.15%	0.689	0.604
Bing	Google	Freetranslation	64.10%	0.688	0.577

Table 4.8: Overview over the Naive Bayes results for the respective candidate-reference combinations.

The highest accuracies achieved for Support Vector Machines are shown in table 4.9. The used data set and attributes are similar to the earlier shown sentence-based algorithms. Support Vector Machines have been optimized 50 times.

Support Vector Machines					
Candidate	Reference 1	Reference 2	Accuracy	F1-Automated	F1-Professional
Google	Bing	—	**63.25%**	0.605	0.656
Google	Freetranslation	—	60.00%	0.578	0.619
Google	Bing	Freetranslation	62.56%	0.629	0.622
Bing	Google	—	59.42%	0.639	0.535
Bing	Freetranslation	—	59.47%	0.571	0.616
Bing	Google	Freetranslation	63.14%	0.664	0.591

Table 4.9: Overview over the best Support Vector Machine results for the respective candidate-reference combinations.

Recall and Precision		
	Candidate Google	Candidate Bing
Average Recall Automated	79.90%	72.46%
Average Recall Professional	58.73%	68.06%
Recall Difference	21.18%	4.40%
Average Precision Professional	74.28%	71.11%
Average Precision Automated	66.31%	69.51%
Precision Difference	7.97%	1.60%

Table 4.10: Overview over the most notable results for the first research question.

Table 4.10 shows additional notable results that appeared during the optimization process of the first research question and will be further discussed in subsection 5.1.1.

Document-Based Results

To evaluate not only sentences but entire documents, the sentence-based approach is applied on the document level. As mentioned in section 4.1, the complete data set was extracted from 14 technical documents. The classification results for these 14 original documents have been created by an individually optimized Decision Tree and are shown in the following table.

	Professionally Translated Documents		**Automatically Translated Documents**	
Document Length	% Prediction Professional	Predicted Class	% Prediction Automated	Predicted Class
213	65.73%	1	62.05%	0
360	66.94%	1	68.89%	0
722	67.87%	1	75, 67%	0
790	65.32%	1	72.93%	0
903	71.10%	1	73.71%	0
1081	66.05%	1	72.72%	0
1175	66.30%	1	75.13%	0
1387	59.63%	1	74.53%	0
1607	72.56%	1	66.15%	0
1973	69.54%	1	71.91%	0
2461	63.71%	1	66.74%	0
3065	59.38%	1	72.15%	0
3076	69.15%	1	71.80%	0
3514	73.76%	1	67.69%	0

Table 4.11: Classification of the original documents with knowledge of the German translation.

The more generalized approach, based on randomly manufactured documents used for the evaluations, support the initial findings. The evaluation uses a separately optimized Decision Tree model for each document length. Every used model represents the optimal result out of a set of 729 tested Decision Trees. The more detailed results are presented in table 4.12.

Document Evaluations						
Document Length	Number of Documents	Class	Average % Automated	Average % Professional	Misclassifications	% Misclassification
5	5000	1	32.55%	67.45%	969	19.38%
10	2500	1	31.85%	68.15%	153	6.12%
20	1250	1	31.34%	68.66%	23	1.84%
50	500	1	31.88%	68.12%	4	0.80%
100	250	1	31.78%	68.22%	0	0.00%
250	50	1	31.62%	68.38%	0	0.00%
500	25	1	31.75%	68.25%	0	0.00%
1000	10	1	31.30%	68.70%	0	0.00%
3000	10	1	31.12%	68.88%	0	0.00%
5	5000	0	71.04%	28.96%	1345	26.90%
10	2500	0	69.92%	30.08%	404	16.16%
20	1250	0	69.98%	30.02%	63	5.04%
50	500	0	70.56%	29.44%	1	0.20%
100	250	0	71.01%	28.99%	0	0.00%
250	50	0	70.20%	29.80%	0	0.00%
500	25	0	70.70%	29.30%	0	0.00%
1000	10	0	70.45%	29.55%	0	0.00%
3000	10	0	69.57%	30.43%	0	0.00%

Table 4.12: Document classification results considering all attributes for research question 1.

4.2.2 Research Question 2

While the previous subsection presents the results related to research question 1, the results for research question 2 are shown in the following section.

Each data-set used to compute the shown results in this subsection uses all available attributes that do not require reference translations. The attributes are normalized and the used data set contains no duplicate entries and is additionally shortened by up to 5% due to outlier detection.

Given the smaller amount of available attributes and the non-consideration of reference-based attributes, the number of different evaluations is shortened. The following table contains the results with the highest accuracies for each tested candidate and algorithm. Table 4.13 shows the highest achieved results for each algorithm.

Candidate	Accuracy	F1-Automated	F1-Professional
Decision Tree			
Google	56.79%	0.562	0.574
Bing	60.50%	0.593	0.612
Artificial Neural Network			
Google	55.77%	0.539	0.574
Bing	60.00%	0.564	0.630
k-Nearest Neighbor			
Google	59.27%	0.625	0.555
Bing	**62.93%**	0.654	0.601
Naive Bayes			
Google	52.92%	0.617	0.389
Bing	55.99%	0.608	0.496
Support Vector Machines			
Google	52.87%	0.563	0.489
Bing	52.45%	0.637	0.310

Table 4.13: Overview over the best achieved results for research question 2.

The presented numbers result of 50.000 optimizations for Decision Trees, 150 for Artificial Neural Networks, 50 for k-Nearest Neighbor and 50 for Support Vector Machines.

Notable Results	
SVM: Average Recall for Automated	84.56%
SVM: Average Recall for Professional	19.51%

Table 4.14: Overview over the most notable results for the second research question.

Table 4.14 shows notable results that appeared during the optimization process and will be further discussed in subsection 5.1.3.

Document-Based Results

Following, the results of applying the sentence-based approach on the original and manufactured documents are presented. Each candidate document length uses an individually optimized Decision Tree model. Each used model is the result of 729 individual optimizations.

Evaluating the original technical documents lead to the results presented in the following table 4.15.

	Professionally Translated Documents		**Automatically Translated Documents**	
Document Length	% Prediction Professional	Predicted Class	% Prediction Automated	Predicted Class
213	49.30%	0	62.44%	0
360	66.94%	1	56.39%	0
722	49.31%	0	54,02%	0
790	53.92%	1	61.39%	0
903	64.23%	1	56.37%	0
1081	58.09%	1	60.41%	0
1175	64.17%	1	56.77%	0
1387	60.27%	1	51.12%	0
1607	59.61%	1	60.61%	0
1973	56.61%	1	60.52%	0
2461	52.95%	1	59.37%	0
3065	58.30%	1	51.48%	0
3076	61.35%	1	58.06%	0
3514	69.24%	1	50.88%	0

Table 4.15: Classification of the original documents without knowledge of the German translation.

Applying the Decision Tree-based document evaluation on the manufactured documents yields in the following data.

Document Evaluations						
Document Length	Number of Documents	Class	Average % Automated	Average % Professional	Misclassifications	% Misclassification
5	5000	1	39.54%	60.46%	1561	31.22%
10	2500	1	37.95%	62.05%	765	30.60%
20	1250	1	39.89%	60.11%	316	25.28%
50	500	1	37.45%	62.55%	55	11.00%
100	250	1	38.00%	62.00%	7	2.80%
250	50	1	35.86%	64.14%	0	0.00%
500	25	1	36.99%	63.01%	0	0.00%
1000	10	1	39.63%	60.37%	0	0.00%
3000	10	1	40.36%	59.64%	0	0.00%
5	5000	0	58.66%	41.34%	1720	34.40%
10	2500	0	57.23%	42.77%	534	21.36%
20	1250	0	58.43%	41.57%	216	17.28%
50	500	0	56.54%	43.46%	72	14.40%
100	250	0	56.81%	43.19%	13	5.20%
250	50	0	55.47%	44.53%	0	0.00%
500	25	0	56.77%	43.23%	0	0.00%
1000	10	0	58.54%	41.46%	0	0.00%
3000	10	0	58.43%	41.57%	0	0.00%

Table 4.16: Document classification results considering all attributes for the second research question.

4.2.3 Quality Ranking of Technical Documentation

Additionally to the classification systems described above, we propose an evaluation pattern to rank technical documents. This is done by combining the classification results of a Decision Tree and an Artificial Neural Network with a weighted mistake score as described in subsection 3.6.2. The four quality classes have been tested with sample checks and will be shown in table 4.17. A detailed discussion concerning the chosen classes will be given in subsection 5.1.6.

Classes for Evaluation of Translation Quality			
Class	Prediction Decision Tree	Prediction Neural Network	Weighted Mistake Count
1	1	1	0
2	1	0	0
	0	1	0
	1	1	< 2.5
3	0	0	0
4	1	0	> 0
	0	1	> 0
	0	0	> 0
	1	1	≥ 2.5

Table 4.17: Overview over the different quality classes for the evaluation of technical documents.

4.2.4 Deliverables

The following deliverables will be handed in with this thesis:

- This work produced two classification systems. The first one, having access to the original document, is able to predict whether a text has been translated automatically or professionally, with an accuracy of 72.24% on a sentence level and with correct predictions on a document level given a size of more than 100 sentences.

- The second classification system with no access to the original document, achieves correct classification rate of 62.93% on a sentence level and correct predictions on a document level, given a size of 500 or more sentences per document.

- A framework is handed in that allows the ranking of translated text fragments into four classes, based on a combination of two algorithms and a weighted mistake score.

- For future research, a translation database is provided, containing 22327 sentences including context information and their professional translations.

Chapter 5

Discussion

This chapter will address the evaluation and discussion of the achieved results, the chosen methodology and the validity as well as reliability of the experiments. Section 5.1 will reflect on the results and outline what has been achieved as well as pointing out the main problems concerning the experiments. Furthermore, an answer to the two research questions will be given as well as an evaluation and interpretation of the results will be performed. Section 5.2 will reflect on the taken approach towards the research tasks, point out its benefits and flaws as well as address whether the right method was chosen to solve the given problem. Finally, section 5.3 evaluates the validity of the used data sets, the calculated attributes as well as the chosen algorithms and the overall experiment setup. Considering these validity points, a conclusion on the reliability of the experiments' results will be drawn. Therefore, this chapter will focus on answering the following questions:

- What conclusions can be drawn out of the presented results?
- Was the applied method a suitable solution for the given task?
- Which benefits and shortcomings exist concerning the presented work?
- How valid and reliable are the used data sets and the presented results?

5.1 Results Interpretation

5.1.1 Sentence-Based Classification with Knowledge of the Original Document

The best results among all algorithms for the first research questions were achieved by Artificial Neural Networks with an accuracy of 72.24% and 71.67%. However, three of the five tested algorithms do not deviate substantially from one another when it comes to accuracy. The average accuracies among the three algorithms differ by less than two percentage points over all tested data sets, with the Neural Network achieving the highest average accuracy of 70.33%. Concerning fluctuations from the mean values, Decision

Trees and Artificial Neural Networks deviate with the same amount ($\sigma = 0.014$) from their averages with k-Nearest Neighbor classifier achieving the minimal deviation among all algorithms with $\sigma = 0.009$.

This contrasts with the other two algorithms, the Naive Bayes classifier and Support Vector Machines. The latter achieved a maximum accuracy of 63.25% losing almost nine percentage points in terms of accuracy compared to the Neural Network and had substantially worse computation times, which led to its removal from the experiments early in the optimization cycles. The Naive Bayes classifier achieved a maximum accuracy of 68.76%. However, it is important to note that its average accuracy is only at 65.84%, resulting in the highest standard deviation among the algorithms. The reason for this might be the simple structure of the Naive Bayes classifier, not allowing for any optimization processes to be done and the ignoring of dependencies between attributes, which definitely exist in the present case. Example given, since multiple intermediate results of the Meteor metric are used as individual attributes, it is impossible that no dependencies exist among some attributes.

An additional subject of interest is the comparison of results using one reference and results using multiple references. The main point of concern is that an additional reference does not necessarily lead to improvement of the algorithms' accuracies. This is proven by the shown results, since the best Neural Network uses a single machine translation system as a reference. On the other hand, the second best algorithm uses two algorithms, which leads to the assumption that the achieved accuracy of an algorithm is in part independent from the number of references used. The reason for this is the aim of the machine learning approach for the given task. In contrast to many evaluation approaches concerning machine translation systems, the presented approach tries to classify the given text fragments into two classes instead of rating its quality. Therefore, the use of a single pseudo reference serves a different goal as the use of multiple pseudo references. Since many of the used attributes calculate similarity scores between the reference and the candidate translation, a machine translated reference can be used to identify automated translations due to high similarities with the given reference, while professional translations might deviate more strongly from it. In contrast to that, the use of multiple pseudo references aims to generate a high quality translation reference by combining the given machine translation systems [8, 9, 7]. For the present task, the use of multiple references seems to increase the accuracy slightly, especially compared to using only one of the two references. Example given, the addition of Freetranslation as a second reference to a data set having Google as candidate and Bing as a reference, tends to lead to an improvement in terms of accuracy.

Furthermore, the question, which metrics have the most influence to solve the given problem is an important topic to address. An analysis has been done on the results of the built

Decision Trees during the experiments. This is, due to the easy interpretability of Decision Tree models, including an easy extraction of the most important attributes. Since, Decision Trees look for the most influential attribute at every splitting point as described in subsection 2.3.1, the most significant attribute is always placed as the root node of the given Decision Tree. Figure 5.1 gives an example of an optimized Decision Tree using Google Translate as the candidate translation and Freetranslation as a reference.

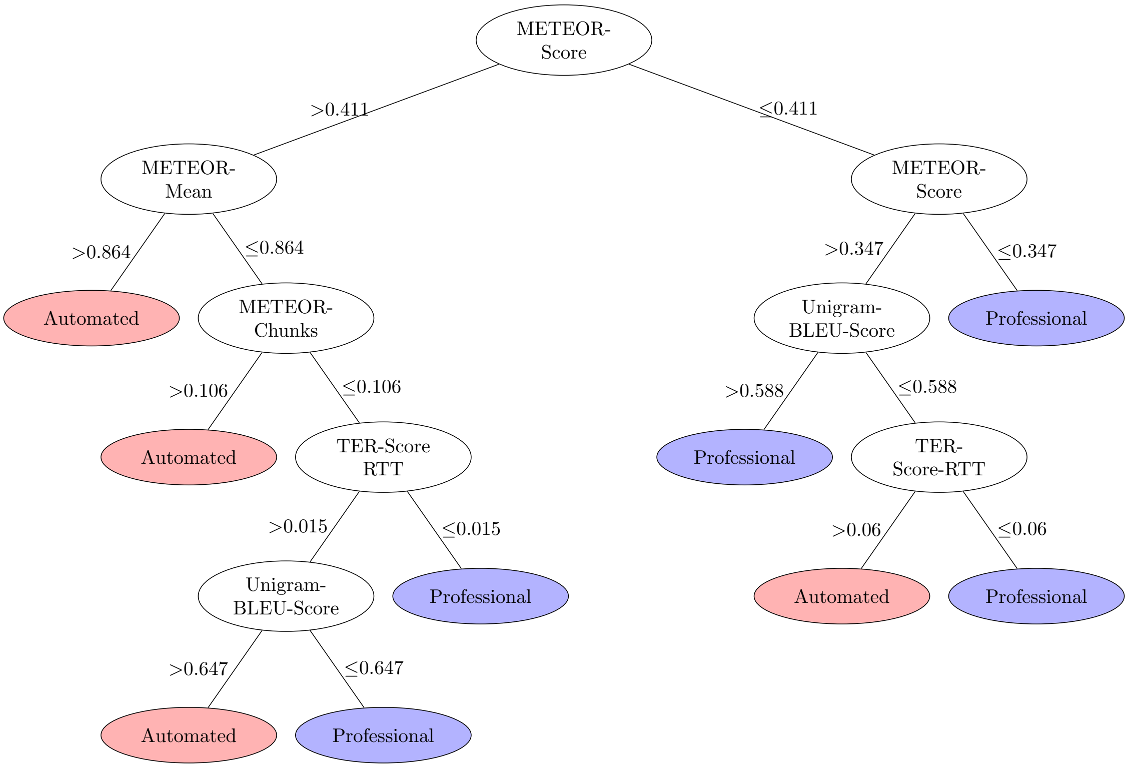

Figure 5.1: Optimized Decision Tree.

This example has been chosen, since it represents the most commonly used attributes for the upper levels of the Decision Tree. The most used attributes by a substantial margin were the Meteor score and its intermediate results, such as Meteor Mean and Meteor Chunks. The next most important attributes, showing up the most frequently in the top levels of the Decision Tree, are the BLEU score and the difference in translation lengths. It is clearly observable that attributes using reference translations are more influential, than metrics having no access to them.

Furthermore, as shown in table 4.10, using Google Translate as a candidate, results in substantially higher recall values for *automated translation* and therefore higher precision values for *professional translation*. This is due to the fact that Google Translate seems to possess a treat that makes the algorithm predict text fragments more often as *automated translation*. A first estimated reasoning could be, that Google's translating machine produces output that partially shares characteristics of professional translations.

Assuming, that all Google produced sentences share a characteristic A, while manually translated sentences show either characteristic A or B. Since the higher percentage of sentences with characteristic A are produced by Google, new sentences containing the characteristic A are assumed to belong more likely to automatic translations and are in conclusion classified as Google translations. On the other hand, this would normally result in Google Translate achieving worse results in terms of accuracy, since the classification is harder to do for the given algorithm. This does not show in the presented results, however this might be caused by an insufficient sample size or different features of the machine translation systems that equalize this flaw. In contrast to Google Translate, Bing Translator stands out for its very equal results in terms of predicting automated as well as professional translations. Bing Translator achieves an average difference of 1.60 percentage points compared to the 7.97 of the Google Translate system, resulting in better precision values for automated translations and worse values for professional ones.

The addition of a round-trip translation to be used as an additional reference, led to further improvements in terms of accuracy. The results without round-trip translations are shown in appendix B. Adding additional attributes to a database, is generally helpful for the algorithm and therefore the observed results were expected, but the main advantage using the round-trip translation shows in its use for the second research question.
The Freetranslation system was used as a round-trip reference, since using the round-trip translation system as a candidate would bias the algorithm and Freetranslation achieved the lowest accuracies in the first stages of optimizations. The average of Freetranslation accuracies records a loss of 1.96 and 2.52 percentage points respectively compared to the other candidates. However, it is important to note that the addition of the Freetranslation system as a reference to an existing "one-reference setup" almost always leads to improvements in terms of accuracy. This is most likely due to the fact that the Freetranslation system provides different and supposedly worse translations than the systems provided by Google and Bing. Therefore, the combination of two stronger differing translation systems leads to better results.

Summarizing, the best achieved results show a classification accuracy of 72.24%, resulting in a gain over a random classifier of 22.24% using Bing Translator as a candidate and Google Translate as a pseudo reference.

5.1.2 Document-Based Classification with Knowledge of the Original Document

Due to the lack of a sufficient amount of technical documents to train machine learning algorithms on a document level, the approach to classify documents into the classes *professional translation* and *automated translation* consisted of recombining the documents' respective sentences and aggregating the sentence-based classification results. We chose

Bing Translator as the candidate, since the result showed fewer deviations for the Bing Translator experiments and a Decision Tree as an algorithm, due to the more efficient optimization process without a substantial loss in terms of accuracy. The final splitting value to decide whether a document belongs to the automated or the professional class, was calculated by averaging the respective distributions of the two labels. This was done for the original 14 manuals that were used as a database by training a model on all but one document and testing the left out documentation on the built model. Building a model on the complete data set was not possible due to the algorithm being biased by having trained on the test data. The classification results of the original documents are shown in table 4.11. It clearly shows that by having knowledge of the original document, the correct classification of documents works well for real world data, since all documents are classified correctly.

To increase the significance of the document-based approach, an additional 19190 manufactured documents were taken into account to evaluate the classification possibilities for different sizes of documents. Due to the limited data set, it was impossible to train a single model for all the documents and therefore different models were trained and optimized for the different sizes of documents ranging from five to 3000 sentences.
As expected, the results show that a higher amount of sentences in a document leads to fewer misclassifications by the algorithm. While the average error for documents consisting of five sentences is relatively high with 23.14%, the average percentage of incorrect classifications drops to 0.50% for documents with 100 sentences. Having knowledge of the original document, leads to no misclassifications for documents containing 250 or more sentences.
An important fact concerning the original document, are the clearly visible differences in prediction distributions. On a percentage basis, the distributions range between 59.38% and 75.13%. This shows that real-world documents can differ substantially in terms of ease of classification for the algorithms. Especially compared to the manufactured document database where sentences are combined randomly out of all documents, the classification of real-world documents has clearly higher deviations. By combining sentence randomly from a set of partly easy to classify documents and partly hard to classify documents, the classification deviates less from the average correct classification rate. The difference between maximal and minimal prediction percentage for the respective class is 15.75 percentage points for the original documents and substantially lower with 3.59 percentage points.

Another important point of concern, is the question of the percentage share distribution of the respective labels. To give an extended view on this concern, figure 5.2 shows the range containing 95% of the classification distributions for each of the two labels based on the document size.

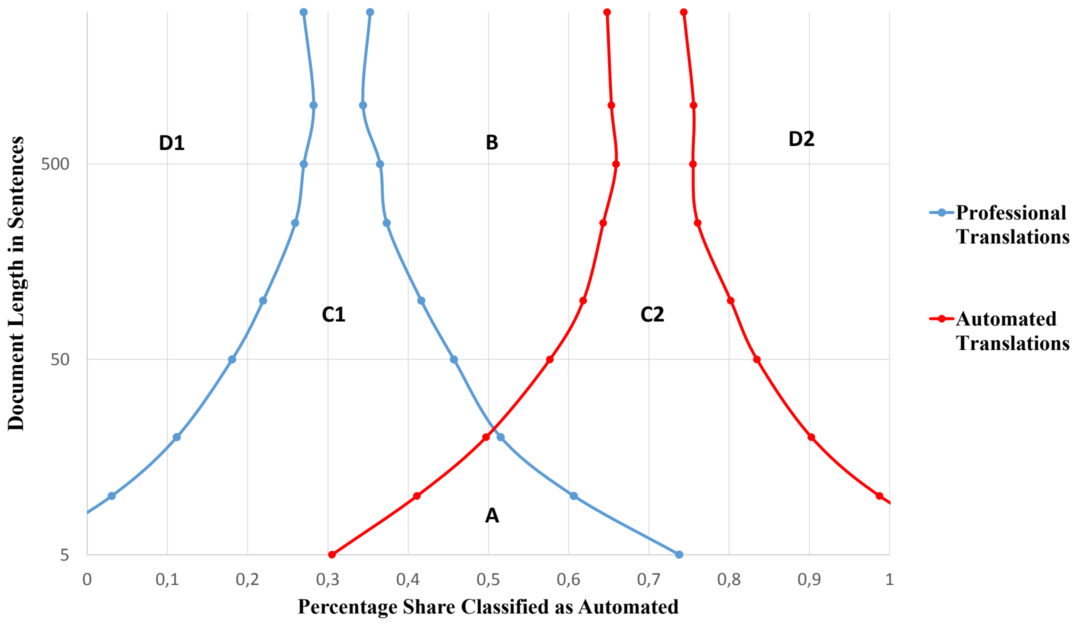

Figure 5.2: Classification distribution for research question 1.

It is clearly visible that longer documents deviate less in terms of classification distributions than shorter documents. Since the algorithm achieves an appropriate accuracy of 70%, the distributions converge towards 0.3 and 0.7 respectively. The figure shows four different areas of interest:

- **A:** The first area of interest, is the overlapping part of the two distributions. Since both distributions signify that this area belongs to 95% of their classification distributions, no conclusion can be drawn for documents belonging in that area. Therefore, the proposed approach has no sufficient certainty about documents with 20 or fewer sentences and a percentage share for class *automated translation* between 0.3 and 0.7. This is an expected result, since the smallest documents with the most even distributions are the hardest to classify correctly.

- **B:** The second area of interest, is area B, which does not belong to either of the two funnel-shaped distributions. This area emerges inevitably for every algorithm that produces any additional accuracy gain. Since this means that the used algorithm has a better performance than a random classifier, even an accuracy gain of 1% would be sufficient to create a gap between the two distributions at a certain document size. This is because the larger the documents become, the more the standard deviation decreases and the distributions approximate the accuracy of the used algorithm. Therefore, the creation of documents with an even distribution for both labels, becomes more unlikely with increasing document sizes. For the presented algorithm, the described gap starts at a document size of 20, containing a small amount of the manufactured documents that do not meet the 95% requirements.

- **C1 & C2:** These areas fit the 95% requirement and contain the majority of document entries. If a new documentation would be classified with a distribution and size fitting into one of the funnels, the likelihood of it belonging to that class is significantly higher, than for the other class.
- **D1 & D2:** The final areas are extreme distributions for mostly high-size documents. Similar to area **B**, it is highly unlikely that these documents appear, due to the given accuracy of the used algorithm on a sentence level. However, if that would be that case, the probability of the document belonging to the class with the smaller percentage share distance is substantially higher.

Concluding, although the algorithm classifies every document with one of the two given classes, the results cannot always be taken for certain, since the resulting percentage share of sentence level classification plays an important role for determining the certainty of the classification. The present approach proposes a classification with certainty, if the classified document fits into one of the following areas:

- **C1, D1** for a classification with *professional translation* or
- **C2, D2** for a classification with *automated translation*.

5.1.3 Sentence-Based Classification without Knowledge of the Original Document

As mentioned in section 3.3 there was reasonable concern that the amount of existing attributes using no reference translations was insufficient to create a useful classifier for the given problem. This concern was backed up by first results using the few attributes that do not require a reference translation, achieving accuracies between 51 and 54%. This would mean close to no gain compared to a random classifier. Therefore, the round-trip translation was introduced to create a fictive computerized reference translation by translating the candidate sentence with an unknown machine translation system to foreign language and back to the candidate language. The expected result was that the sentence would be more similar to the candidate sentence, if it belonged to class *automated translation* than for the other class. This is due to the observation of machine translation system translating words and phrases in the same manner, resulting in better round-trip translations than for professionally translated sentences. This resulted in a maximum achieved accuracy of 62.93% by a k-Nearest Neighbor classifier using Bing Translator as a candidate. On average, the k-Nearest Neighbor achieved the highest results as well with a score of 61.1% compared to the next highest by the Decision Trees with $58,65\%$ and the lowest average by Support Vector Machines with 52.66%. Similar to research question 1, the three algorithms with that highest accuracy were the k-Nearest Neighbor classifier, Artificial Neural Networks and Decision Trees. The results are a clear improvement compared to the achieved accuracies without the round-trip translation reference and allows attempts on a document-based approach. The Support Vector Machine achieves the lowest accuracy by a substantial margin and the results show that even with an optimization process, it was not possible to generate significant accuracy gains compared to a random classifier. This is well illustrated by the unequally distributed recall values. As shown in table 4.14, the average recall for *automated translation* is 84.56%, while *professional translation* lies at 19.51%. This is a typical value distribution for classifiers that predict a very high amount of data entries with one of two labels, while almost ignoring the second label. It is important to note that the use of Bing Translator as a candidate seems more promising for the approach without knowledge of the original document, since, especially for the better performing algorithms, Bing Translator achieves clearly better results than Google Translate with an average accuracy of 61.14% compared to 57.28%.

5.1.4 Document-Based Classification without Knowledge of the Original Document

The results discussed above, were used to create a document-based classification system similar to the one discussed in subsection 5.1.2. The results for the original documents are shown in table 4.15. The first apparent fact is that in this case, not every document was classified correctly, having two documents containing 213 and 722 sentences respectively falsely classified as *automated translation*, predicting 49% of the document's sentences correctly. This goes hand in hand with the observed recall values for close to all results of the done experiments. The recall values for automated translations is constantly slightly higher and the automated class is predicted slightly more often. Furthermore, having an algorithm with half the accuracy gain over a random classifier, compared to approaches having access to the original document, clearly leads to higher misclassifications. Although the algorithm correctly classifies about 60% of the given instances and for larger documents there should be no misclassifications, a real-world document can be an outlier and therefore hard to classify for the given algorithms compared to the manufactured documents. This is clearly visible for the given documents, ranging from percentage shares for the document class of 49.30% to 69.24%. This leads to significantly more deviating values for the real-world data with a range of 19.94 percentage points than for manufactured values with a range of 8.67 percentage points. The documents containing 722 and 213 sentences are one of the described outliers, since they do not fit the observed misclassification patterns for the manufactured database shown in table 4.16

The evaluation shows an average misclassification rate of 32.81% for the shortest documents, 25.98% for documents containing ten sentences and an average error of 4% for document length 250 before dropping to 0% for larger test files.
As for Research Question 1, the sentence level classification distributions of the documents were analyzed and are illustrated in figure 5.3.

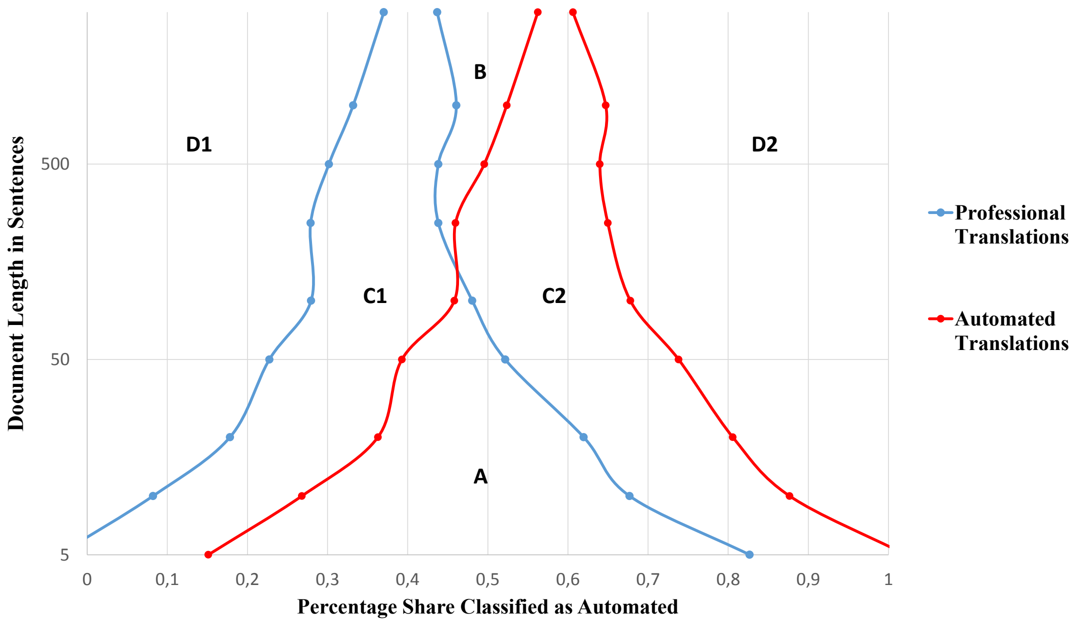

Figure 5.3: Classification distribution for research question 2.

The areas of interest are the same as described in subsection 5.1.2. The deviations in the areas of larger documents are due to the comparably small amount of created documents for lengths of 1000 sentences and more. The present approach proposes a certain classification, if the classified document fits into one of the following areas:
C1, D1 for a classification with *professional translation* or
C2, D2 for a classification with *automated translation*.

5.1.5 Comparison of the two Research Questions

Concerning the setup of the two research questions, research question 1 was a more straight forward task. Allowing the use of reference translations made the creation of a valid attribute base easier and the results were more promising in the first optimization runs. In contrast to that, the second research question created the problem of having to solve a document quality problem without access to semantics of the document, since the use of reference translations was not allowed. The addition of round-trip translations as a reference allowed the creation of a sufficient attribute base to perform the experiments. The expected results were that algorithms trained and tested on research question 1 would outperform the algorithms for research question 2 by a substantial margin, since the information of the original document is a highly valuable, additional piece of information. These expectations were confirmed by the experiments, showing a maximum accuracy of 72.24% compared to the 62.93% of research question 2 for the sentence-based approach. This is an increase in accuracy of 9.31 percentage points and an accuracy gain increase over the used benchmark of a random classifier by 172%. The respective results for the different algorithms were comparably equal, resulting in Decision Trees, Neural Networks and the k-Nearest Neighbor classifier achieving the best results, and Naive Bayes as well as Support Vector Machines falling off in terms of accuracy. On a document level, the classification system with knowledge of the original document was able to classify all 14 documents correctly for both, its professional and its automated translation, while the system without the additional knowledge failed the correct classification for two professionally translated documents.

Furthermore, the misclassification rate is clearly higher for research question 2, with a total number of 5259 misclassification for the complete data set of 19190 documents, while the system for research question 1 classifies 2962 entries falsely. This results in a total accuracy of 72.59% for the system with no knowledge of the original document and an accuracy of 84.56% for the system with this knowledge. This accuracy has to be taken with caution, since the results are highly dependent on the length of the given document, such as documents containing five sentences having misclassification rates of 19% to 34% and documents containing 1000 and more having no misclassifications at all. Figure 5.4 compares the development of misclassification rates based on the respective document sizes for both systems.

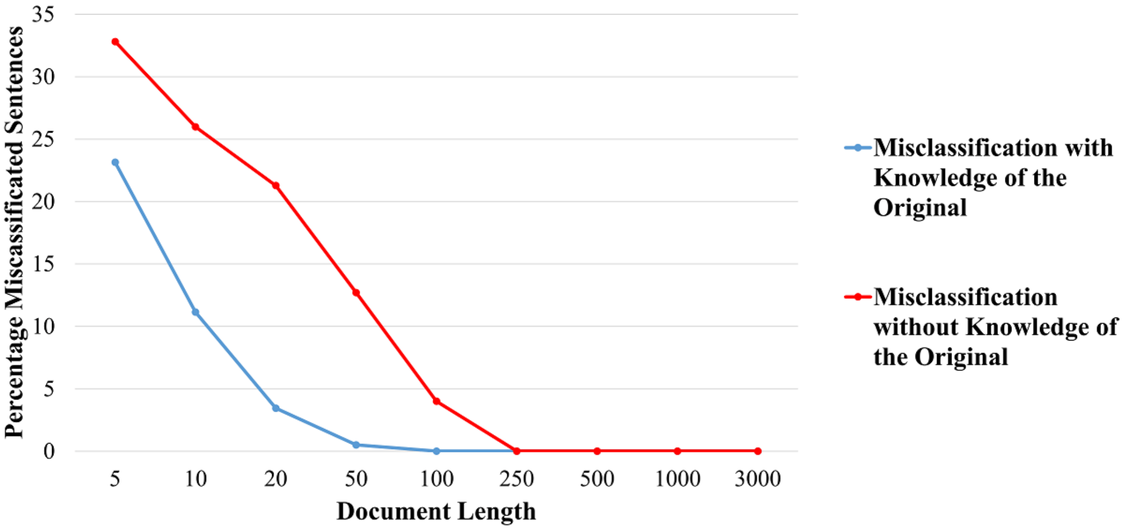

Figure 5.4: Development of the misclassification rate for the manufactured documents with relative to the document size.

The figure above shows that misclassifications are strictly lower for every document size when using the system developed for research question 1 and furthermore, the required document length to avoid any misclassifications is significantly lower.

To further examine the performance between the two systems, it is important to look at the percentage share of the different distributions relative to the given document sizes as shown in figures 5.2 and 5.3. The main difference between the two figures is clearly the larger overlapping part, resulting in more documents that cannot be classified with certainty. Example given, a document containing 50 sentences with 40% being classified as *automated translation*, could not be classified with certainty using the system for research question 2, since the respective point still belongs to the overlapping part and therefore to area **A** from which no conclusions can be drawn. However, since area **A** is smaller in the system used for research question 1, if the document would be classified similarly, it can safely be labeled as a professionally translated document. This is due to the distribution belonging to area C, which contains 95% of the professionally translated documents. If the sentence level algorithm has a lower accuracy, the average distributions for the two classes are closer to one another (compare 0.3 and 0.7 to 0.4 and 0.6 respectively). This results in less certainty for classified documents, since the distance between the two funnel-shaped distributions is smaller and errors are more likely.

Concluding, the results of the experiments confirmed the expectations with research question 1 being easier to solve using classification techniques, than research question 2, due to higher overall and averaged accuracies on a sentence level, smaller misclassification rates and more certain classifications for all sizes of documents.

5.1.6 Evaluation Framework

The proposed framework uses the results of the classification system described in subsection 5.1.1 and therefore assumes that knowledge of the original document is available. The framework works on a sentence level and the evaluation process consists of classifying the given sentence with two well performing algorithms, an Artificial Neural Network and a Decision Tree. The reason the k-Nearest Neighbor classifier has not been used for the evaluation of translation quality, is due to the structure of this classifier. Since the main computational effort in the k-Nearest Neighbor approach is needed for the classification of new entries, by calculating the k nearest neighbors to given data point. Therefore, a use of k-NN as a classification system is not reasonable for the given task.

Additionally, a weighted mistake count is added as additional context information. These three attributes (a boolean value for the classification using the Decision Tree, a boolean value for the classification using the neural network and a floating point number for the weighted mistake count) are used to classify the sentence into one of the four existing classes. Class 1 represents the best and class 4 the worst document quality. It was observed and literature agrees on the fact that professional translations are generally of higher quality than automated translations. Therefore, a sentence classified with *professional translation* has to be ranked higher than a sentence classified as *automated translation*. In this case, the classification results of Decision Tree and neural network are treated equally, which means that it is distinguished between an algorithm having 0, 1, or 2 classifications as *professional translation*. As described in section 3.3, the mistake count is a value calculated by using a rule-based dictionary for identifying different categories of grammatical mistakes. The present database showed a total of 94 mistake categories and each of these categories are weighted with a score depending on their influence concerning the correctness of the given sentence. Example given, certain mistakes categories include suggestions for style improvements and cannot be treated equally with actual grammatical mistakes. Additionally, the majority of the found categories were removed, due to them having no impact on the correctness of the sentence. The categories were separated into eight severeness categories ranging from 0.1 to 7.5. Figure 5.5 shows the distribution of the different mistake categories over the severeness categories.

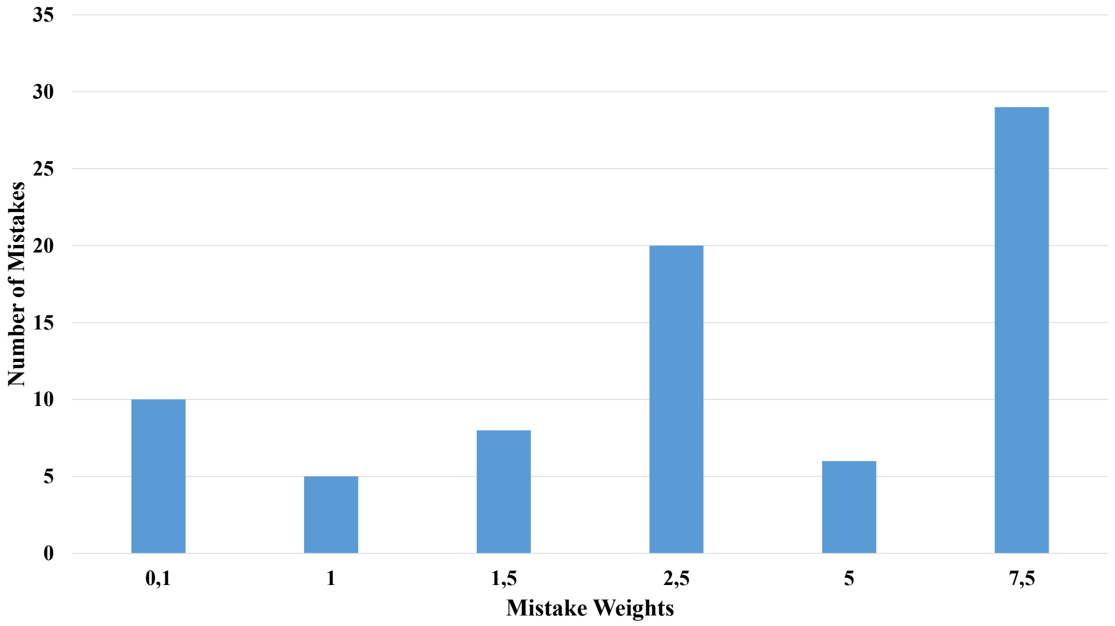

Figure 5.5: Overview over the distribution of the different mistake categories relative to their weights.

The mistake count had more influence on the sentence quality than the classification system, resulting in most sentences with a mistake score being classified with the worst class. This is due to the fact that the mistake count has a high relevance when mistakes occur. The amount of sentences containing mistakes relative to the overall sentence count is comparably high at a rate of 51.61%. This means that 48.39% of the sentences do not contain mistakes that can be identified with the given metric. However, due to the removal of many mistake categories that do not influence the text quality in a meaningful manner, the remaining categories are represented in 10% of the given used sentences. In order to evaluate the quality of a document, its sentences are binary classified and the document quality is calculated using a weighted average over all sentences, with lower quality classes receiving higher weights. A weighting of the classes relative to their quality level proved to be appropriate, resulting in sentences belonging to class 2 being weighted two times higher, than sentences from class 1. The same applies for class 3 (weighted three times higher) and class 4 (weighted four times higher). Figure 5.6 gives an overview over the document distribution among the quality classes, using the proposed framework.

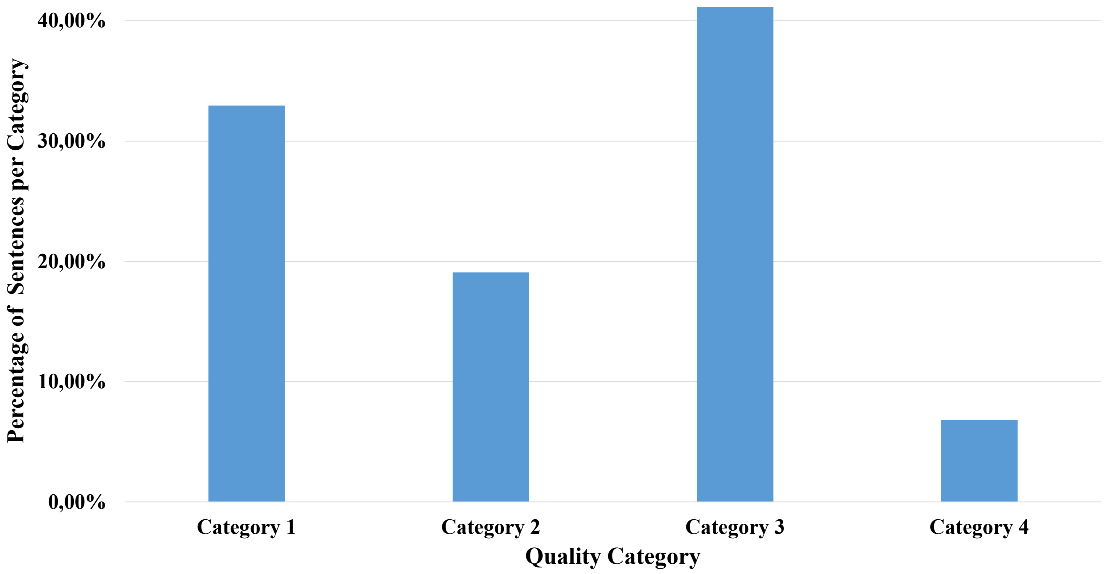

Figure 5.6: Illustrated sentence evaluation by ranking them into their respective classes.

Concerning figure 5.6, it is clearly observable that Category 1 and 3 are the most frequent classes, which is due to them containing the entries in which the machine learning algorithms agree in their classification and the mistake count is 0. It is important to note that category 4 has the least entries by a substantial margin, however sample-based tests show that sentences classified as category 4 seem to be clearly of the worst quality compared to the other classes. Furthermore, although it is not relevant for the evaluation of document qualities, mistake counts have been further weighted with weights above 2.5, to allow further examinations of the classified sentences through analysis of their overall mistake score.

Figure 5.7 shows the evaluation of the different documents using the proposed framework.

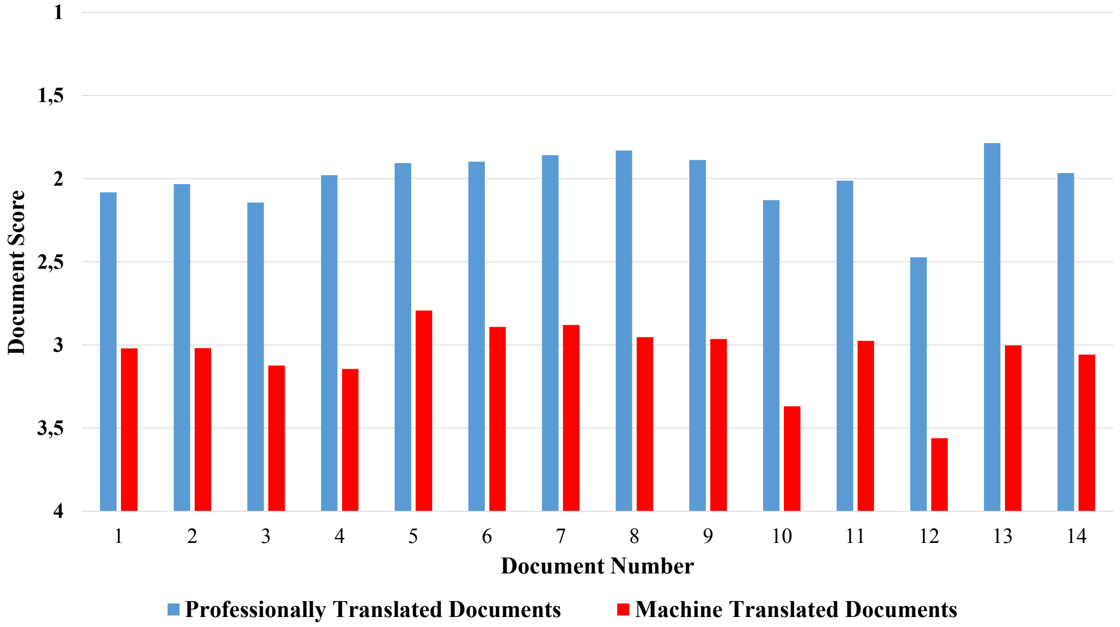

Figure 5.7: Illustrated document evaluation by ranking them using the averages of the respective sentences.

It is clearly visible that the professionally translated documents are rated higher than the respective automated documents. The reason for the professional documents being rated with an average score of 2.00 is mostly due to sentence extraction problems in the preprocessing step, resulting in additional mistakes that have not been caused by erroneous translations. The clear difference to the average score of 3.05 for automated translations shows the quality difference between professionally and automatically translated sentences. It is important to note that this is a proposed framework that has been validated by sample-based tests and seems to produce reasonable results. It still remains to be verified by professionally qualified institutions.

5.2 Method reflection

The used KDD process proved to be suitable for finding a solution to the two research questions. The standardized procedure allowed for the flexibility of studying different experiment setups without neglecting the initial goal. Beginning with the data extraction and preprocessing, the presented method proved to be able to generate a functioning, sentence-based data set in a sufficient quantity and quality for further use on different algorithms. The choice of several evaluation metrics allowed combining individual strengths of the attributes and therefore built the groundwork for the computation of valid results. To meet the requirements of some scoring metrics concerning reference translations, the use of machine translations as pseudo references proved to be reasonable. Additionally, the use of pseudo references provided the opportunity to answer both research questions properly and eliminated the need for professionally translated references. Furthermore, the used attributes were distributed well between the purpose of both research tasks resulting in 18 attributes usable for having no knowledge of the original document and an additional 14 attributes for having knowledge of the original document. The preprocessing of the used data set and the respective attributes consisted of outlier removal, the detection of highly correlating attributes and the normalization of the calculated attribute values, resulting in a well set up database for classification purposes.

To address possible translation system specific characteristics, three different machine translation systems were used. The results show that machine translated references can be sufficient for the evaluation of technical document translations. However, it is of interest to compare the differences between the use of professional and automated references and therefore advised to repeat the experiments with the use of professional references, as this was not possible due to the restricted resources for this work.

Using different machine learning algorithms to evaluate the quality of sentences and even documents, proves to be effective and the results show the adequacy for the task at hand. The most known algorithms were tested with sufficient optimizations to generate significant gain over a random classifier benchmark. While the results allowed an adequate judgement on the evaluation of technical documents and their translations, more detailed studies and optimizations remain to be done, as the experiments were limited by the available computation resources. Especially concerning the elaboration on algorithms such as the k-Nearest Neighbor classifier and the Artificial Neural Network, the relatively small number of optimizations suggest that the presented results do not represent the achievable optimum. Nevertheless, they present valid results for the given task of classifying sentences and documents.

Focusing on the initial document quality evaluation on a sentence level ensured the significance of the approach, since the used data set consisted of over 20.000 different sentences in contrast to the original 14 documents. Besides the increased significance, the focus on smaller text parts extends the usability for the quality evaluation, since the classification system can be used on single phrases as well as on large documents by recombining the calculated scores. However, the taken approach has a clear shortcoming. By not classifying on a document level, but instead recombining respective sentences to create a document classification leads to a focus on a very specific training set, since the initial experiment setup uses a small amount of 14 documents. Furthermore, by manufacturing random documents out of the existing sentences, the validation of the systems' performances on a document level was underpinned. However, the created files are not entirely comparable to original documents. The randomized combination of sentences to documents builds largely unified documents and in combination with the classification on sentence level, limits the standard deviation of the classification task, consequently resulting in fewer misclassifications. This assumption is supported by the evaluation of the 14 original documents, which show stronger fluctuating classification percentages, than the generalized document evaluation. Nevertheless, the general validity of the document classification is supported by the classification results for the original data and still generally reliable.

Combining the binary classification with a reference independent spell and grammar check allowed final quality statements on a sentence and document level. Although first evaluations of the proposed framework for quality estimation seems promising, a more detailed ranking of documents is a challenge that has not been addressed in the presented work, because the building and verification of a more in-depth document evaluation would have gone beyond the scope of this thesis. Concerning the advantages of focusing on technical documents, it is assumed that the focus on a field that uses a specific and smaller vocabulary results in additional gain compared to a more general approach on a not specified domain. In order to verify this assumption, future research on this topic should apply the presented approach on different data sets and evaluate the differences in terms of classification results. Finally, it is noteworthy that the presented approach is based on the implicit quality advantages of professional translations and therefore, the better computerized translation systems become, the harder the distinction between professional and automated translations will be. This is due to the fact that more similar translations are expected to reduce the effect of machine learning algorithms on this type of classification problem and therefore complicate future research on this topic.

In conclusion, the proposed method led to a successful attempt on classifying technical document translations and evaluating translation quality with and without knowledge of the original document, without requiring additional high quality references as used in common translation evaluations.

5.3 Reliability

This final section aims to examine the validity and reliability of the shown results in section 4.2.

The achieved results show substantial improvement over a given benchmark on a sentence level for both research questions. Recombining the labeled sentences to classify on a document level is a valid strategy, since the same information is used that would have been taken into account by an algorithm classifying directly on a document corpus. None of the presented machine learning algorithms has been trained on data entries, which have been used for testing purposes in the same process iteration, which underpins the validity of the presented classification accuracies. As mentioned in section 5.2, the break down of the document database onto sentence level created a large enough database to generate reliable results, with sufficiently large testing and training data sets. Furthermore, the precise preprocessing removed redundant data, detected erroneous data and further improved the applied data set. Using the general structure of the knowledge discovery process allowed an iterative setup using multiple iterations concerning the database creation, the attribute and metric selection as well as the algorithm optimization, leading continuously improving the classification results.

In contrast to that, a few shortcomings concerning the presented results have to be mentioned. First, the results were supposed to focus on a technical documentation domain and therefore no statement can be made concerning the classification of documents not belonging to the domain. Second, as noted in section 1.5, only syntactical translation aspects are taken into account, which means that the classification of a text as a high quality document says nothing about the actual meaning of the translation compared to the original document. Third, to further evaluate the classification accuracies, an in-depth comparison to related experiments using machine learning algorithms on different domains should be done to set the achieved results into the scientific context of machine translation evaluation on a more general methodology.

Concerning the proposed evaluation approach, the presented framework has been validated using sample-based verifications. However, the system still has to be verified by professionally qualified institutions to further validate and possibly adapt the proposed evaluation classes. In conclusion, the presented approaches answer both research questions with success, being able to classify and evaluate technical documents and their translations.

Chapter 6

Conclusion

6.1 Conclusions

This work answered the following research questions:

- How can the translation quality of technical documents be evaluated, given the original document is available?
- How can the translation quality of technical documents be evaluated, given the original document is not available?

This was done by using a knowledge discovery process consisting of the phases, gathering data, preprocessing data, choosing an appropriate data mining approach to find patterns among the data and interpreting them. Finally, the results were used for further research. The document database was broken down into a sentence level, producing nine data sets each containing $22,327$ data entries for each of two translation types (*automated translation* and *professional translation*). 32 metrics and attributes were chosen and implemented, of which 18 needed a reference translation for the calculation process and 14 did not. To create a reference translation, one or two computerized translation systems were used respectively to translate the original document and generate a reference for the given candidate texts. The described data set was preprocessed, removing 5% outliers, and attributes correlating to one another with more than 90% as well as normalizing the data to generate comparable attribute values. The preprocessed data was used in multiple iterations for five machine learning algorithms, Decision Trees, Artificial Neural Networks, k-Nearest Neighbor, Naive Bayes and Support Vector Machines. The algorithms were optimized on their parameters and tested on a holdout set that was split up from the database before training the models. The maximum results were achieved by the k-Nearest Neighbor classifier, scoring 72.24% while having access to the original document and 62.93% without access to it. To make a statement on document level classifications, the optimized algorithms were used on the sentences of each original document and the results were recombined to classify the respective documentation. Having access to the original text, resulted in no classification errors for the 14 documents, while having no access to it showed a misclassification rate of 14.29%. To further validate the document-based results, a set of $19,190$ manufactured documents was created by randomly combining sentences to fictive documents of sizes varying from five to 3000 sentences. The observed misclassifications ranged from 34.40% for the smallest documents to no misclassifications for documents containing 250 or more sentences for algorithms trained without knowledge of the original document and from 26.90% for the smallest documents to no misclassifications for documents containing 100 or more sentences for algorithms trained with knowledge of the original document.

Furthermore, an evaluation framework was constructed to rank sentences and documents based on their quality regardless of their translation type. The proposed model consists of four classes using two optimized machine learning models for classifying the sentences and an additional reference independent grammar and spell check tool to generate a weighted mistake count for each sentence. In order to evaluate document quality, the quality classes of the respective sentences are averaged with additional weight for higher mistake counts.
Additionally, a translation database has been developed, containing 22000 professionally translated sentences. Their translation to German and further context information can be used for future research in the area of machine translation evaluation.

6.2 Further research

This work presents a classification system for technical documents using machine learning methods and an evaluation approach for document quality rankings. Extending the work on this topic, an attempt to further improve classification quality on a document level could be made by aggregating sentences to document level based on calculated confidences of the resulting classification models. This approach would result in a more fine-grained document classification, since the algorithm's certainty for the sentence-based classifications is taken into account. Example given, a sentence with the predicted label *professional translation* could have its prediction value converted to a confidence score of 0.873, while the value of a sentence with the predicted label *automated translation* could be converted to 0.421. These confidences would represent the calculated probability of the algorithm that the given candidate translation was a professionally translated sentence.

As mentioned in chapter 5 some used machine learning approaches could not be optimized as extensively as it would be desirable due to computation resource limitations. Therefore, especially concerning k-Nearest Neighbor classifiers and Artificial Neural Networks, a more in-depth optimization process would be desirable to come closer to a global optimum in terms of classification results. An additional point of interest is the preprocessing step for generating a sentence-based data set. As described in section 3.2, the data set is extracted out of PDF documents, which are focused on visualization. This complicates the extraction process and leads to erroneous data entries in the resulting data set. An improvement in the extraction process could further improve classification and evaluation results. Another option would be the use of documents in XML format, allowing for a standardized extraction process without mistakes. To further improve the evaluation of document quality, the mistake count metric can be upgraded. In areas of highly technical vocabulary, attempts such as the use of neutral nouns as a replacement for technical terms or proper nouns has been introduced. Additionally, a more fine-grained examination of different mistake categories can be done to further elaborate on their different influences on text quality and correctness.

Concerning the suggested evaluation framework, an in-depth analysis of the proposed classes is necessary to verify and optimize them, since the recommended method has only been validated using sample-based testing. In order to put the proposed approach into context with different approaches on machine learning methods for evaluation of machine translation, a second database could be created using domain independent data to compare the results on a non-specific data set with the results of a domain focused one. This would allow drawing conclusions concerning the implicit additional knowledge that is gained by focusing on the domain of technical documentation. Extending further, comparisons could be made to different approaches for machine translation evaluations, such as the development of new metrics for similarity calculations between candidate and reference translation. Such comparisons would put the use of machine learning approaches for machine translation evaluation tasks into the context of different methodologies.

In conclusion, the presented work extends the related research on this topic by combining multiple machine translation evaluation metrics with the use of machine learning methods focusing on domain-specific documents.

Bibliography

[1] H. Somers, “Round-trip translation: what is it good for,” in *In proceedings of the Australasian Language Technology Workshop*, pp. 127–133, 2005.

[2] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: a method for automatic evaluation of machine translation,” in *Computational Linguistics (ACL), Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia*, pp. 311–318, 2002.

[3] G. Doddington, “Automatic evaluation of machine translation quality using n-gram co-occurrence statistics,” in *Proceedings of the Second International Conference on Human Language Technology Research*, HLT ’02, (San Francisco, CA, USA), pp. 138–145, Morgan Kaufmann Publishers Inc., 2002.

[4] A. Lavie and A. Agarwal, “Meteor: An automatic metric for mt evaluation with improved correlation with human judgments,” pp. 65–72, 2005.

[5] A. Kulesza and S. M. Shieber, “A learning approach to improving sentence-level mt evaluation,” in *In Proceeding of the 10th International Conference on Theoretical and Methodological Issues in Machine Translation*, 2004.

[6] M. Popović, D. Vilar, E. Avramidis, and A. Burchardt, “Evaluation without references: Ibm1 scores as evaluation metrics,” 2011.

[7] M. Gamon, A. Aue, and M. Smets, “Sentence-level mt evaluation without reference translations: Beyond language modeling,” in *In European Association for Machine Translation (EAMT)*, 2005.

[8] J. S. Albrecht and R. Hwa, “2007b. regression for sentence-level mt evaluation with pseudo references,” in *In Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 296–303, 2007.

[9] J. S. Albrecht and R. Hwa, “The role of pseudo references in mt evaluation,” in *In Proceedings of ACL*, 2008.

[10] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, “From data mining to knowledge discovery in databases,” *Ai Magazine*, vol. 17, pp. 37–54, 1996.

[11] J. H. Friedman, “Data mining and statistics: What’s the connection,” in *Proceedings of the 29th Symposium on the Interface Between Computer Science and Statistics*, 1997.

[12] O. Maimon and L. Rokach, eds., *Data Mining and Knowledge Discovery Handbook, 2nd ed.* Springer, 2010.

[13] L. Rokach and O. Z. Maimon, *Data mining with decision trees: theory and applications*, vol. 69 of *Series in machine perception and artificial intelligence*. World Scientific Publishing Co., 2008.

[14] B. P. Battula and R. S. Prasad, “An Overview of Recent Machine Learning Strategies in Data Mining,” *International Journal of Advanced Computer Science and Applications*, vol. 4, 2013.

[15] C. Sammut and G. I. Webb, eds., *Encyclopedia of Machine Learning*. Springer, 2010.

[16] F. Camastra and A. Vinciarelli, *Machine Learning for Audio, Image and Video Analysis - Theory and Applications, Second Edition*. Advanced Information and Knowledge Processing, Springer, 2015.

[17] J. G. Carbonell, R. S. Michalski, and T. M. Mitchell, “Machine learning: A historical and methodological analysis,” *AI Magazine*, vol. 4, no. 3, pp. 69–79, 1983.

[18] J. R. Quinlan, “Induction of decision trees,” *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.

[19] S. Salzberg, “Book review: C4.5: programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993,” *Machine Learning*, vol. 16, no. 3, pp. 235–240, 1993.

[20] L. Liu and M. T. Özsu, eds., *Encyclopedia of Database Systems*. Springer US, 2009.

[21] J. Su and H. Zhang, “A fast decision tree learning algorithm,” in *AAAI*, pp. 500–505, AAAI Press, 2006.

[22] D. Y. Singh and A. S. Chauhan, “Neural networks in data mining,” *Journal of Theoretical and Applied Information Technology*, vol. 5, no. 1, pp. 37–42, 2009.

[23] R. J. Erb, “Introduction to backpropagation neural network computation,” *Pharmaceutical Research*, vol. 10, no. 2, pp. 165–170, 1993.

[24] Y. Dodge, ed., *The Concise Encyclopedia of Statistics*. Springer New York, 2008.

[25] T. J. Cleophas and A. H. Zwinderman, eds., *Machine Learning in Medicine*. Springer Netherlands, 2013.

[26] W. Commons, “Zwei mögliche trenngeraden mit verschiedenen randgrößen,” 2010.

[27] I. Steinwart and A. Christmann, *Support Vector Machines*. Springer Publishing Company, Incorporated, 1st ed., 2008.

[28] D. M. W. Powers, “Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation,” Tech. Rep. SIE-07-001, School of Informatics and Engineering, Flinders University, Adelaide, Australia, 2007.

[29] W. J. Hutchins and H. L. Somers, *An introduction to machine translation*. Academic Press, 1992.

[30] C. M. Amine, “Theoretical overview of machine translation,” in *Proceedings of the 4th International conference on Web and Information Technologies, ICWIT 2012, Sidi Bel Abbes, Algeria, April 29-30, 2012*, pp. 160–169, 2012.

[31] M. Nagao, “A framework of a mechanical translation between japanese and english by analogy principle,” in *Proc. Of the International NATO Symposium on Artificial and Human Intelligence*, (New York, NY, USA), pp. 173–180, Elsevier North-Holland, Inc., 1984.

[32] H. L. Somers, “Review article: Example-based machine translation,” *Machine Translation*, vol. 14, no. 2, pp. 113–157, 1999.

[33] S. Nirenburg, C. Domashnev, and D. J. Grannes, “Two approaches to matching in example-based machine translation,” *TMI-93: The Fifth International Conference on Theoretical and Methodological Issues in Machine Translation*, pp. 47–57, jul 1993.

[34] B. J. Dorr, J. Olive, J. McCary, and C. Christianson, *Machine Translation Evaluation and Optimization*, pp. 745 – 843. Springer New York, 2011/// 2011.

[35] J. P. Turian, L. Shen, and I. D. Melamed, “Evaluation of machine translation and its evaluation,” Tech. Rep. NYU-CSE-03-005, Department of Computer Science, New York University, February 2003.

[36] M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul, “A study of translation edit rate with targeted human annotation,” in *In Proceedings of Association for Machine Translation in the Americas*, pp. 223–231, 2006.

[37] M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul, “A study of translation edit rate with targeted human annotation,” in *In Proceedings of Association for Machine Translation in the Americas*, pp. 223–231, 2006.

[38] D. Shapira and J. Storer, *Edit Distance with Move Operations*, pp. 85–98. Springer Berlin Heidelberg, 2002.

[39] M. Denkowski and A. Lavie, “Meteor 1.3: Automatic metric for reliable optimization and evaluation of machine translation systems,” in *Proceedings of the Sixth Workshop on Statistical Machine Translation*, WMT ’11, (Stroudsburg, PA, USA), pp. 85–91, Association for Computational Linguistics, 2011.

[40] L. Kothes, “Grundlagen der technischen dokumentation : Anleitungen verständlich und normgerecht erstellen,” 2011.

[41] RapidMinerGmbH, “Rapidminer documentation detect outlier cof,” 2016.

[42] J. V. Tu, “Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes,” *Journal of Clinical Epidemiology*, vol. 49, no. 11, pp. 1225 – 1231, 1996.

Appendix A

Optimization Ranges

	Minimum	Maximum
Decision Tree Parameter		
Maximal Depth	1	50
Minimal Gain	0.01	0.3
Minimal Leaf Size	50	500
Number of Prepruning Alternatives	0	100
Minimal Size for Split	50	5000
Confidence	0.001	0.5
Neural Net Parameter		
Learning Rate	0.1	0.9
Momentum	0.001	0.7
Epsilon	0.0001	0.1
Training Cycles	50	
Local Random Seed	false	
k-Nearest Neighbor		
k	1	100
Support Vector Machine		
Maximal Iterations	10	1000
Convergenz Epsilon	0	1
C	-1	1
Kernel Cache	0	100
Kernel A	-0.9	0.9
Kernel B	-0.9	0.9
Kernel Degree	0.001	0.9
Kernel Shift	0.01	0.9
Sigma (1-3)	0.01	0.9
Kernel Gamma	0.01	0.9

Table 1.1: Overview over the evaluated parameter optimization ranges

Appendix B

Additional Results

Decision Tree					
Candidate	Reference 1	Reference 2	Accuracy	F1-Automated	F1-Professional
Google	Bing	—	67.82%	0.691	0.664
Google	Freetranslation	—	66.37%	0.678	0.648
Google	Bing	Freetranslation	69.44%	0.725	0.655
Bing	Google	—	68.44%	0.684	0.685
Bing	Freetranslation	—	66.88%	0.677	0.660
Bing	Google	Freetranslation	69.69%	0.690	0.704
Freetranslation	Bing	—	67.33%	0.684	0.662
Freetranslation	Google	—	67.15%	0.689	0.652
Freetranslation	Bing	Google	67.99%	0.704	0.652

Table 2.1: Overview over the best Decision Tree results for the respective candidate-reference combinations, without round-trip attributes.

Appendix C

Detailed Working Steps

The following part describes the detailed procedure on a technical level that has been taken to solve the given task.

Data gathering:

The first step after setting the goals for the task at hand, was to find technical documents existing both in English and German versions. Both versions needed to be professionally created and exactly translated in order to be suitable for the given task. The first approach to extract usable sentences from existing PDF files used a very raw punctuation based approach. Sentences were split at common punctuations, such as . : ! ?. Due to the technical nature of the used documents, this proved to be very ineffective. Therefore, the second approach aimed to improve this initial, simple Regex-based approach by taking different scenarios into account, for example the need for an ending punctuation to be followed by a blank. The second approach resulted in an improved dataset, which was used for the first evaluations. However, because of existing text extraction problems due to the given PDF format there were still lots of erroneous sentences in the dataset. After conducting the first evaluations and determining how the input data quality influences the results, a third iteration used a part of speech tagging approach to identify and split sentences more precisely. It showed that part of speech tagging allowed a more precise sentence detection. However, difficulties resulting from the initial text extraction from the visualization based PDF format to a plain text format remained. Therefore, we conducted several more improvement steps, such as identifying paragraphs first and extracting sentences from paragraphs. This allowed a removal of the most footnote entries and picture titles as well as several headlines. To produce a high quality data set, the resulting sentences were reevaluated using a second set of Regex rules similar to the ones used for the first two approaches. Data entries, which did not match the Regex-criteria were excluded from the experiments as they were mostly erroneous due to text extraction difficulties. After extracting correct sentences, it was necessary to match the gained English sentence to its

German translation. As the sentence extractions above did not result in the same amount of sentences for both languages, an exact matching was not possible. Therefore, we developed an algorithm in which the German sentences were automatically translated into English and afterwards the sentences were aligned by using the BLEU algorithm with a low similarity score limit. The following figure illustrates the sentence matching process: First, each file has an offset counter indicating the sentence number that is currently checked. Given sentence 1 in the candidate file does not match sentence 1 in the reference file, the algorithm raises the offset counter for the reference file by one and compares the first candidate sentence with the second reference sentence. The reference counter is raised up to a threshold for each candidate row. If no match was found, the candidate sentence is assumed to be unmatchable and removed from the data set. The reference counter is reset and the process starts with candidate row 2 again. If a match is found, the candidate and reference counters are raised by one, and the process starts with the next sentence pair. The limit of compared reference sentences is necessary to ensure that only the real matches are found and not similar sentences from a different position of the document are matched. This approach lead to the first initial dataset, consisting of around 22,000 sentences in a professional German version with a matching, professional English translation. Afterwards more automated translations were generated and added, which resulted in the dataset presented in section 3.2.

Implementation:

The approaches described above are implemented using oracles java 1.8. The initial two sentence splitting methods are self-programmed. Given the complexity of state of the art parts of speech tagging, we decided to avoid programming a custom parts of speech tagger and utilize the java based Stanford core NLP packages to implement the final sentence splitting method.[1] In order to ensure the sentence splitting produced valid sentences, we implemented several requirements as rules, the sentences had to meet. After splitting the documents into sentences, we implemented the sentence matching algorithm described above. The unigram BLEU similarity is calculated using the Stanford core NLP suite.

Attribute calculation:

The calculation of the attributes presented in this thesis consisted of multiple iterations. The initial calculations included mainly the uni-, 2- and 3-gram BLEU scores and the reference length. Afterwards more complex metrics, such as Meteor or TER were added. As the calculation of scores like Meteor produces several steps, we added the intermediate results to the dataset to evaluate the impact of interim steps compared to the complete score. Covering the most common metrics for machine translation evaluation, resulted in

[1] http://nlp.stanford.edu/software/ , 22.02.2016

the first significant evaluation results. To address both research questions, non-reference metrics like the Flesch Reading Ease were implemented. This first attempt to answer research question 2, the quality evaluation of a technical documentation without knowledge of the original, proved to be ineffective. Therefore, we implemented a round-trip based approach and added the attributes mentioned in section 3.3, leading to the final dataset used to run evaluations.

Implementation:

The calculation of the used attributes are implemented into the Java project handed in with this thesis. The metrics are all implemented in Java. To reduce the complexity in validating the correct implementation of all metrics, the Translation Edit Rate and Meteor were implemented using the Java implementations provided by the University of Maryland (http://www.cs.umd.edu/ snover/tercom/, 22.02.2016) and the Carnegie Mellon University (http://www.cs.cmu.edu/ alavie/METEOR/, 22.02.2016).

Evaluation

As already mentioned, the experiments presented in this thesis are of highly iterative nature. Therefore, we conducted several evaluations with different subsets of the finally implemented attributes. In the early stages of our work, we ran the first evaluations using only reference based metrics such as the calculated BLEU scores. The following iterations considered also the Translation Edit Rate and Meteor scores, as well as Reference Length. Further different setups with the first reference-free metrics were tested. The final evaluation step started after the final round-trip attributes were added. With this final status, we started to conduct extensive evaluations, considering all different data mining algorithms and multiple optimization scenarios, leading to the results presented in section 4.2.

Implementation:

Due to the iterative nature and the complexity that would come along with the implementation and optimization of several data mining algorithms, the evaluations in this thesis are conducted by using Rapidminer Studio 6[2] as it allows effectively optimizing data mining parameter setups and efficient evaluations. The processes used to calculate the results presented in this work are handed in with this thesis.

[2] http://www.rapidminer.com, 22.02.2016