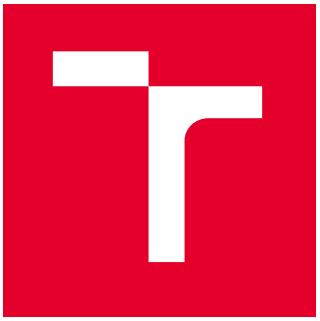

BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF CONTROL AND INSTRUMENTATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

TEXT ANALYSIS IN SPECIALIZED TRANSLATION: ACCURACY AND ERROR RATE

ANALÝZA TEXTŮ A JEJÍ UŽITÍ V ODBORNÉM PŘEKLADU: PŘESNOST A CHYBOVOST

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR — **Alžbeta Parobková**
AUTOR PRÁCE

SUPERVISOR — **Mgr. Přemysl Dohnal**
VEDOUCÍ PRÁCE

BRNO 2023

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

Bachelor's Thesis

Bachelor's study program **Automation and Measurement**

Department of Control and Instrumentation

Student: Alžbeta Parobková ***ID:*** 217515

Year of study: 3 ***Academic year:*** 2022/23

TITLE OF THESIS:

Text Analysis in Specialized Translation: Accuracy and Error Rate

INSTRUCTION:

1. Explore data mining methods and discuss their applicability in text processing. To perform the task, use Matlab Text Analytics Toolbox, the Python language, and other relevant tools, evaluating and comparing the properties and capabilities of these softwares.
2. Discuss the most frequent mistakes in specialized translation (the Czech-to-English stream in particular).
3. Propose and design a text analysis approach, the focus being on Czech-to-English translation within electrical engineering and communication.
4. Analyze the texts that you have selected to outline the problem.
5. Present the results in relevant context.

RECOMMENDED LITERATURE:

1] JO, Taeho. Text Mining. B.m: Springer, Cham, 2019. ISBN 2197-6503.

[2] EL MOUTASIM, Abdelkrim a Oudaani JAOUAD. Topic Classification of Arabic Text in Quran by Using Matlab [online]. 2018. Dostupné z: doi: 10.1007/978-3-030-12048-1

Date of project specification: 6.2.2023 ***Deadline for submission:*** 22.5.2023

Supervisor: Mgr. Přemysl Dohnal

doc. Ing. Václav Jirsík, CSc.
Chair of study program board

ABSTRACT

The focus of the thesis is on researching and applying text analysis and machine translation methods to quality evaluation of machine translated technical texts. The experimental part uses these methods to implement error identification and classification algorithm. The error and grammar correction neural model was also applied. The comparison of error rate and accuracy of different language tools is presented via error typology and standardized translation evaluation metrics.

KEYWORDS

text analysis, text mining, text preprocessing, NLP, machine translation, machine translation quality evaluation, error classification, translation metrics, error and grammar correction, pre-trained Transformer model, Python

ABSTRAKT

Práca sa zameriava na prieskum a aplikáciu metód textovej analýzy, strojového prekladu na vyhodnotenie kvality technických textov, preložených práve pomocou strojového automatického prekladu. Praktická časť využíva tieto metódy na implementáciu algoritmu pre identifikáciu a klasifikáciu chýb. Ďalšou časťou praktickej časti je aj aplikácia a natrénovanie neurónového modelu pre korekciu týchto chýb. Porovnanie chybovosti a presnosti prekladu rôznymi prekladačmi je potom preukázané nie len kvalitatívne, ale aj kvantitatívne pomocou štandartných metrík.

KLÍČOVÁ SLOVA

textová analýza, dolovanie textu, textové predspracovanie, NLP, strojový preklad, vyhodnotenie kvality strojového prekladu, klasifikácia chýb, prekladové metriky, chybová a gramatická korekcia, predtrénovaný Transformer model, Python

Rozšírený abstrakt

Cieľom bakalárskej práce je využitie textovej analýzy pri overovaní presnosti a chybovosti strojového prekladu. Overovanie je zamerané na strojový preklad technických textov z českého do anglického jazyka. Dôležitosť overenia kvality spočíva v popularite rôznych automatických (strojových) prekladačov a ich používanie v bežnej praxi. Zistenie, ktorý prekladač je najpresnejší pri preklade technických dokumentov, môže pomôcť pri jeho výbere a využitia pri preklade napríklad vedeckých článkov a publikácii.V dnešnej dobe sa vyžadujú aj anglické texty a rôzne automatické prekladače dokážu zefektívniť a zjednodušiť prácu pri ich písaní.

Prvá časť práce je spracovanie teoretickej rešerše možností textovej analýzy, strojového prekladu a ich realizácie. Na začiatku je predstavená textová analýza a jej najpoužívanejšie metódy. Textová analýza sa zameriava na dolovanie informácii, vyhľadávanie rôznych vzorov a odhaľovanie súvislosti v textových dátach. Text je neštrukturovaný typ dát, preto jeho predspracovanie je nevyhnutnosťou pre ďalšie spracovanie. Základnými metódami textového predspracovania sú čistenie dát, tokenizácia, identifikácia slovných druhov, zaznamenanie vetných skladieb pomocou parsovania, a získavanie základných foriem slov. Všetky tieto metódy predspracovania dokážu sprostredkovať hlbšie informácie o jazykovej štruktúre, preto boli aj zvolené ako hlavné metódy v praktickej časti.

Nerozdeliteľnou súčasťou textovej analýzy sú metódy, ktoré dokážu získať informácie o kontexte viet a na základe ich významu spracovať ďalšie súvisloti medzi nimi. Týmito metódami sú textová klasifikácia, zhlukovanie textu a vytvorenie súhrnu dokumentov. Najefektívnejšie využitie majú pri dolovaní dát vo viacerých dokumentov. Výnimkou je textová klasifikácia, ktorá dokáže efektívne pracovať aj s individuálnimi slovami, ak je kombinovaná s metódami určovania charakteristických rysov daných slov.

Teoretická časť zameraná na textovú analýzu je zakončená zhodnotením ako by sa dokázalo textové dolovanie dát využiť pri strojovom preklade. Najčastejším využitím je predspracovanie dát pre procesy strojového prekladu a zhodnotenie kvality samotného prekladu. Strojový preklad dokáže byť tiež výhodný pri textovej analýze, ktorá pracuje s viacjazyčnými dokumentmi.

Druhá teoretická časť predstavuje tématiku strojového prekladu. Výsledkom prieskumu sú rôzne možnosti metodiky aplikovateľnosti tohto prekladu, štandardizované metriky pre zhodnotenie kvality prekladu a aké prekladače sú najpopulárnejšie. Metodiky strojové prekladu sú tri: založený na pravidlách, využitie princípov štatistiky a nasadenie neurónových sietí. Určovanie pravidiel je najstarší spôsob implementácie strojového prekladu, ale vyžaduje si veľké jazykové znalosti a sú časovo náročné na vytvorenie. Štatistický princíp používa najme strojové učenie a štatis-

tiku, čo spôsobilo zvýšenie efektívnosti, aj keď zďaleka to nie je najpresnejší spôsob prekladu. V poslednej dobe sú veľmi rozšírené prekladové neurónové siete, ktoré využívajú aj všetky najpopulárnejšie prekladače, ako sú Bing, Google Translate, DeepL a Systran.

Posladná časť prieskumu sa zaoberá nástrojmi, vďaka ktorým vieme textovú analýzu a strojový preklad využiť. Najpopulárneišími programovacími jazykmi sú Python, MatLab a R. Jazyky Python a MatLab dokážu pomôcť riešiť analytické, aj prekladové problémy. Python je open-source, preto ponúka oveľa väčšie množstvo možností metód pri riešení týchto problémov. MatLab je zase výpočtovo výkonejší. R sa najľahšie využíva pri analýze. Každopádne pri ich hlbšom porovnaní, zavážil faktor možností trénovanie predtrénovaných neurónových modelov. MatLab ponúka predtrénované modely, ale hlavne pre počítačové videnie. Z tohto dôvodu Python zvíťazil ako programovací jazyk pre túto bakalársku prácu.

Súčasťou rešerše zahŕňalo zistenie aké architektúry sa používajú pri spracovaní prirodzeného jazyka. Z prieskumu vyplýva, že všetky architektúry vychádzajú z Transformera a sú iba jeho deriváciami. Transformer obsahuje dekódovacie a enkódovacie vrstvy pre hlboké učenie. Rovnako obsahuje takzvaný 'Attention' mechanizmu, ktorý pomáha učiť sa rozoznať aké časti dát sú dôležité. Transformer architektúru využívajú aj predtrénovahé modely ako sú GPT, BERT a T5. Predtrénované modely využívajú princíp transferného učenia. Najskôr sa trénuje veľký model na obrovskom objeme dát bez zamerania na spracovanie konkrétnej úlohy. Jeho naučené vrstvy sa uložia a posledné vrstvy v architektúre sa nahradia novými vrstvami. Nové vrstvy sú nanovo trénované, tentokrát na menšom data sete zameraného na vyriešenie konkrétnej úlohy.

Praktická časť práce najskôr vysvetľuje návrh riešenia pre zhodnotenie kvality strojového prekladu a potom opisuje jej realizáciu. Práca je písaná v jazyku Python a implementácia obsahuje dve časti: identifikáciu najčastejších chýb a model pre korekciu gramatických chýb. Najčastejšie vyskytujúce chyby pri preklade sú nesprávne použitie členov, zlé časovanie slovies, preklepy, nesprávna voľba slovnej zásoby a nechcená úprava textu. Tieto úpravy môžu byť pridanie alebo odstránenie slov. Praktická časť využíva najmä porovnania textov k referenčným, ktoré boli preložené človekom a sú považované za bezchybné pre účely tejto bakalárskej práci.

Princíp realizácie je taký, že sa využíva klasifikácia slov na základe predom daných pravidiel a určí sa ich štatistický výskyt v texte. Dáta sú predspracované, konkrétne metódami čistenia dát, tokenizácie, určovania slovných druhov a hľadania základného tvaru slova. Výsledok nájdenia a klasifikácie chýb je zobrazený v chybovostnej tabuľke. Táto analýza ukazuje, ktoré prekladače sú najpresnejšie, ale aj na aké chyby sa zamerať pri neskorjše korektúre.

Druhá časť je korekčný model, ktorý je založený na modely hlbokého učenia.

Využíva sa predtrénovaný model BART, ktorý dokáže z textového vstupu generovať textový výstup. V tomto prípade sa na korekciu hľadí ako na preklad z chybného jazyka do toho správneho. Model dokáže určiť správny tvar slovies, dosadiť členy a opraviť preklepy alebo nahradiť neznáme slovo lepšou terminológiou.

Na záver boli porovnané prekladače medzi sebou. Najpresnejším nástrojom pre preloženie technických textov je DeepL. DeepL ako jediný nemal tendenciu pridávať alebo odstraňovať zbytočné slová do textu. Rovnako aj jeho kvantitatívne metriky presnosti, ako je podobnosť, BLEU a METEOR, vypočítali najpresnejšie výsledky. Vo všeobecnosti sa dá povedať, že prekladače sú pomerne presné.

Na záver, bakalárska práca obsahuje prieskum a aj následne porovnanie rôznych metód textovej analýzy a strojového prekladu. Využívali sa všetky tri typy metodológie, a to štatistické, založené na pravidlách a aj neurónové siete. Implementácia riešenia dokáže sprostredkovať prehľad o presnosti a chybovosti preloženého textu, ako aj následného opravenia chýb pomocou natrénovaného modelu. Ďalšie rozšírenie práce by mohlo byť zamerané na využitie čisto neurónových sietí, ako aj pre overenie chybovosti textu, tak aj oprave chýb. Samotný model by sa určite tiež dal ešte lepšie natrénovať, aby dosahoval presnejšie výsledky než teraz.

Author's Declaration

Author:	Alžbeta Parobková
Author's ID:	217515
Paper type:	Bachelor's Thesis
Academic year:	2022/23
Topic:	Text Analysis in Specialized Translation: Accuracy and Error Rate

I declare that I have written this paper independently, under the guidance of the advisor and using exclusively the technical references and other sources of information cited in the paper and listed in the comprehensive bibliography at the end of the paper.

As the author, I furthermore declare that, with respect to the creation of this paper, I have not infringed any copyright or violated anyone's personal and/or ownership rights. In this context, I am fully aware of the consequences of breaking Regulation § 11 of the Copyright Act No. 121/2000 Coll. of the Czech Republic, as amended, and of any breach of rights related to intellectual property or introduced within amendments to relevant Acts such as the Intellectual Property Act or the Criminal Code, Act No. 40/2009 Coll. of the Czech Republic, Section 2, Head VI, Part 4.

Brno ....

author's signature*

*The author signs only in the printed version.

ACKNOWLEDGEMENT

I would like to thank the advisor of my thesis, Mgr. Přemysl Dohnal, for his valuable comments and insightful advises.

Contents

Introduction **13**

1 Text analysis **14**
1.1 Data mining and text analysis 14
1.2 Preprocessing 14
1.2.1 Tokenization 15
1.2.2 Stop words 16
1.2.3 Stemming and lemmatization 16
1.2.4 Parsing 17
1.2.5 Part-of-speech tagging 18
1.3 Categorization 18
1.4 Clustering 20
1.5 Information extraction 20
1.6 Text summarization 22
1.7 Text analysis in machine translation 22

2 Translation **23**
2.1 Machine translation 23
2.1.1 Rule-based machine translation 24
2.1.2 Statistical machine translation 24
2.1.3 Neural machine translation 25
2.2 Quality evaluation of translation 27
2.2.1 Metrics 28
2.2.2 Error analysis profile 29

3 Tools **31**
3.1 Programming languages 31
3.1.1 Python 31
3.1.2 MatLab 32
3.1.3 R 33
3.1.4 Comparison of programming languages 33
3.2 Pre-trained language models 34
3.2.1 Transformer architecture 34
3.2.2 Types of pre-trained language models 35
3.2.3 Examples of pre-trained models 36
3.3 Existing translators 37

4 Solution Proposal **38**
4.1 Proposed methodology 38
4.2 Proposed tools 39

5 Implementation **40**
5.1 General workflow 40
5.2 Tools 40
5.3 Data sets 41
5.4 Data preprocessing 42
5.5 Error identification and classification 42
5.5.1 Identification and classification of the article error type 43
5.5.2 Identification and classification of unnecessary translation actions 44
5.5.3 Identification and classification of verb inflection error type . . 44
5.5.4 Identification and classification of mispelling and mistranslation error types 45
5.6 Pre-trained model for error correction 46
5.7 Error analysis metrics 47
5.8 Example of the error identification and classification algorithm on a sample data set 48
5.9 Comparison of existing translating tools 51

Conclusion **53**

Bibliography **54**

A Electronic attachment **60**

List of Figures

1.1 Parsing 17
1.2 Classification 19
1.3 Clustering 20
2.1 The Vauquois triangle 24
2.2 Statistical Machine Translation 25
2.3 RNN and Feed-Forward Networ 26
2.4 Neural machine translation 27
3.1 Transformer model architecture 35
3.2 Comparison of all transformer architectures 37
5.1 Implementation workflow 41
5.2 Loss function curve 47

List of Tables

1.1 Tokenization examples 15
1.2 Lemmatization and stemming 17
1.3 Part-of-speech tagging 18
5.1 Example of data processing 42
5.2 Final error table 46
5.3 Examples of different fine-tuning parameters. 48
5.4 Evaluation translation metrics 48
5.5 Translation example 49

Introduction

Text analysis offers many different methods that can help to understand huge sets of various texts. The most popular methods are text preprocessing, text classification, text clustering, information retrieval, and text summarization. This thesis focuses on researching these methods and applying the most suitable ones to get information about the error rate and accuracy of translated texts. Another research part of the thesis explores machine translation, its approaches and main applications. There are benefits in combining text analysis and machine translation approaches that can be useful, when evaluating translation quality. The last theoretical part the thesis deals with a research with the emphasis on different options of practical tools, frameworks, and toolkits that might prove useful when analysing the translation accuracy.

The experimental part of the thesis proposes the implementation solution. The goal is to implement an evaluating system that combines both text mining and automatic translation. During the quality assessment, error profile analysis is essential as it identifies and classifies different types of errors. After that, error correction can be proposed. The text analysis and error correction model are then applied to a data set that contains machine translated technical texts. The translation is done from Czech language to English language. The result of the quality evaluation system is a comparison of different translating tools, and assessing their error rate together with the accuracy. It is important to know these results as research papers in English are a necessary part of the academia, and translating tools might be a great help when doing technical and scientific translations.

1 Text analysis

The first chapter explains what text analysis is and what methods are used for different applications. The most common techniques are presented and their fundamentals are described. At the end of the chapter, the connection between text analysis and machine translation is covered. It also shows the application of text analysis to machine translation, too.

1.1 Data mining and text analysis

Data mining is a discipline that focuses on identifying patterns, finding connections and similarities, and extracting useful information from huge data sets. The gathered knowledge is further analysed in order to get a novel data-driven conclusion of a domain-specific research. Data mining is considered an interdisciplinary field combining computer science, statistics, machine learning, and database technologies. It mainly works with unsupervised structured data. [1]

Data mining covers a lot of subfields, one of them is text analysis (also known as text mining). Although text analysis uses similar methods and approaches, the application is only on textual data. Texts are considered unstructured data, therefore, preprocessing is required in order to work with them. The text analysis combines approaches of computational linguistics, statistics, natural language processing, content analysis, data mining, and machine learning. Its process consists of information retrieval via collecting the document samples, information extraction, and mining new associations between texts. [2]

1.2 Preprocessing

Preprocessing is the first necessary step when applying any natural language processing tasks. It prepares and cleans data for further analysis. It is a crucial part of text mining tasks, as it reduces the data size, and removes any unnecessary, useless information from the text that does not enhance further text operation.

The process usually starts with standardization, where characters in sentences are lower-cased, any special characters, numbers and punctuation are removed. This useful prerequisite is needed to prevent word duplicates or data noise.

Only after the proper clearance and identification of crucial structures in textual data, efficient analysis can be done. The whole preprocessing could be summarized into two main applications: the data cleaning and the characteristic feature extraction. After the initial standardization, next feature extraction procedures, such as

tokenization, stop-word removal, stemming and lemmatization, parsing and part-of-speech tagging, take place. These preprocessing methods are explained in the next chapter's subsection in more details.

1.2.1 Tokenization

Tokenization is a process where a text is divided into a list of individual words or sentences, called tokens. It is an important step in text analysis, as it identifies words, terms, phrases or any other sentences. Tokenization can be done either on a word-level or a sentence-level, even though tokens representing words are more common. Its output list is a crucial part of lexical analysis and serves as an input for further text mining. The tokens can be identified by different characteristic features, too. The most used structures for token identification are white spaces, prefixes and suffixes, punctuation characters.[3]

Tab. 1.1: Examples of different tokenization done on the sentence-level, on the word-level, and on the cleaned data.

Text	An electronic circuit is composed of individual electronic components, such as resistors, transistors, and capacitors.', 'Signals can be amplified, computations can be performed, and data can be moved from one place to another.'
Tokenized on the sentence-level	['An electronic circuit is composed of individual electronic components, such as resistors, transistors, and capacitors.', 'Signals can be amplified, computations can be performed, and data can be moved from one place to another.']
Tokenized on the word-level	['An', 'electronic', 'circuit', 'is', 'composed', 'of', 'individual', 'electronic', 'components', ',', 'such', 'as', 'resistors', ',', 'transistors', ',', 'and', 'capacitors', '.']
Tokenized cleaned data on the word-level	['an', 'electronic', 'circuit', 'is', 'composed', 'of', 'individual', 'electronic', 'components', 'such', 'as', 'resistors', 'transistors', 'and', 'capacitors']

1.2.2 Stop words

Stop-word removal is an essential part of preprocessing in natural language processing and text mining. It improves the performance and quality of classification analysis' results. In general, the most frequent words in the text are considered stop words. They are usually words that carry less important meaning in regards of document content. Stop-word lists are language and domain specific, therefore, the right list has to be selected for a particular problem. However, in machine translation, stop-word identification is more useful than the complete removal of them. The reason is that stop words might create noise, but they also may carry grammatically important information, depending on the language. The example of stop words in the English language are articles, such as *'a'*, *'an'*, and *'the'*.[4]

1.2.3 Stemming and lemmatization

Stemming and lemmatization are preprocessing methods that help to identify the word structures. Stemming reduces words in various forms to their basic root form. It is a determining step for search engines, text clustering, summarization and categorization. Prefixes and suffixes are usually removed from the words as a part of the process. The conversion also assumes that morphological form of a word is semantically related to their stem form. For example, there are words such as *'creative'*, *'creates'*, *'creating'*, and their identified stem word would be *'create'*. Stemming does not involve context of words in sentences. There exist different types of stemming algorithms for English language. The main approaches are a truncating method, a statistical method and a mixed method. Stemming algorithms for different languages usually have different division of their categories. Truncating methods require affix removal. One of the most popular truncating stemmer is Porter Stemmer. This algorithm is rule-based, where it works with the main idea that English language suffixes consist of other smaller and more simple suffixes. It checks the rules if the word structure can be consider suffix or not. If it is identified as such, it is removed to leave only the root form of the word. The advantages of Porters Stemmer is its efficiency, lower error rate compared to other algorithms, and can be applied to language-independent solutions, too. The only limitations are false positive results, where identified stem is not a real word, and its long time duration during the stemming process. Statistical methods consist of n-gram stemmers, Hidden Markov Model stemmer, and YASS Stemmer. Statistical approach to stemming calculates the distance between two strings to determine their similarities. All of these methods are language independent and there is no need to known the language morphology. The disadvantages are that n-grams stemmer are time consuming, Hidden Markov

Models and YASS Stemmer are complex and requires a great computational power. [5]

Tab. 1.2: An example of lemma and stem forms of word tokens.

Tokens	**Lemma forms**	**Stem forms**
an	an	an
electronic	electronic	electron
circuit	circuit	circuit
is	be	is
composed	compose	compos
of	of	of
individual	individual	individu
electronic	electronic	electron
components	component	compon

Lemmatization is subtly different from stemming, even though the concepts are similar. Lemmatization also reduces a word to their root form, lemma, but it tries to understand the part of speech and a context of the word first. It involves understanding of text context to correctly identify the root word. To give an example, the word *'better'* has identified lemma as *'good'*. The context of the word was found and correctly identified to its root meaning. The lemmatization is the most essential for information retrieval tasks. [6]

1.2.4 Parsing

Parsing is an analysis determining syntactic structure of sentences ad defying their grammatical dependencies. Its application can be found in computational linguistics, information extraction, machine translation, and automatic query answering. [14]

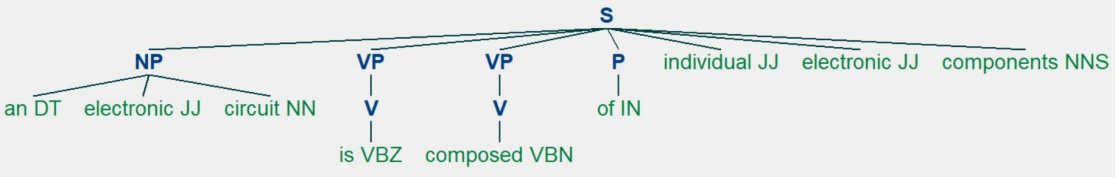

Fig. 1.1: An example of visualizing grammar structure using a parser tree. The parser tree was done via the NLTK library.

The most widely used type of parsing is a dependency parsing. It maps a sentence to its dependency structure. The main idea behind the method is that every

syntactic structure consists of words linked by binary relations, called dependency relations or dependencies. Dependencies are held between the main word called the head and their dependant which usually is a syntactically subordinate word. The dependency type is also defined. [14]

1.2.5 Part-of-speech tagging

Part-of-speech tagging, also known as POS tagging, is a preliminary process in many natural language processing applications, such as information extraction, parsing, machine translation. The main objective of this preprocessing method is to assign a correct part of speech label to every word in a sentence. Part of speech labels can be noun, verb, adjective, etc. It is also a way to categorize words according to their morphological and syntactical properties. Rule-based approach is the most common, however, unsupervised and supervised learning algorithms increase in their applicability. Part-of-speech tagging is also a crucial process for text quality evaluation because it shows the grammatical connections between the words. [15]

Tab. 1.3: An example of tokens paired with their part-of-speech tags.

Text	**Result of Part-of-speech tagging**
An electronic circuit is composed of individual electronic components.	[('an', 'DT'), ('electronic', 'JJ'), ('circuit', 'NN'), ('is', 'VBZ'), ('composed', 'VBN'), ('of', 'IN'), ('individual', 'JJ'), ('electronic', 'JJ'), ('components', 'NNS')]

1.3 Categorization

Text categorization, also known as text classification, is a method that classifies words, sentences or documents into pre-defined categories based on their specific extracted features. The algorithm tries to assign labels or tags to textual structures. Text classification is often approached by using rule-based techniques or using machine learning models. Rule-based technique requires a set of pre-defined rules in order to correctly categorize textual units. Domain knowledge is required, too. Machine learning models are data-driven. Therefore, they can classify text objects by observing data based on pre-labeled training data set.

However, both approaches follow similar step-by-step procedure. First, feature extraction is needed in order to identify deciding structures to form rules for each category. Useful feature extraction algorithms are *bag of words, n-grams*, *Naive Bayes*, *support vector machine* or *hidden Markov model*. The output is then a

classifier used further in machine learning predictions or setting classification rules. The classifiers are then used in variety of classifying models. The most popular types are feed-forward neural networks, RNN-based and CNN-based models, capsule neural networks, models with attention mechanism, graph neural networks, or pre-trained models based on these architectures.

There are many applications of a text categorization. Its application areas are sentinent analysis, topic analysis, news categorization, query answering or even natural language inference. Categorization in natural language inference is useful mainly when analysing paraphrasing or when the semantic similarity of different texts is measured. [7]

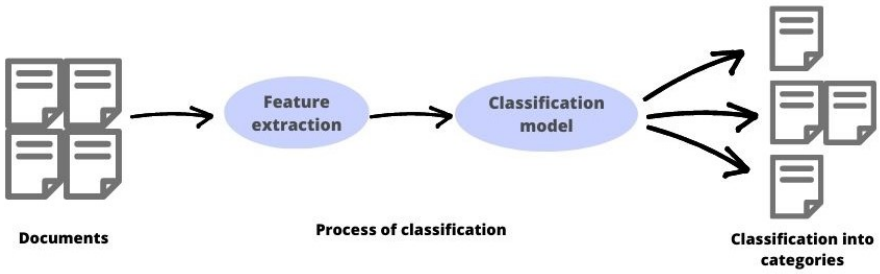

Fig. 1.2: Classification, or categorization general workflow, when dividing documents into three pre-defined categories.

The most popular statistical classification algorithms are *k-nearest neighbour classifier* (KNN), *Naive Bayes classifier*, *support vector machines* (SVM), and decision trees. *K-nearest neighbour classifier* is based on the calculation of the shortest distance from the labeled training data to text objects in the text. *Naive Bayes classifier* calculates the probability of the attribute value that represents one of the categories. The advantage is that it should categories unknown attributes of text structures, too. *Support vector machines* maps textual data in order to transform them into a higher dimension to find the optimal separating hyper plane. It tries to optimize the weights of the training examples and input vectors to correctly categorize text structures. The last popular statistical method is a decision tree. Decision tree is based on recursive partition of the instance space. It is built on greedy search algorithms to find the most probable category. The biggest disadvantage of these statistical methods is its time inefficiency and long learning process. [8]

1.4 Clustering

Clustering methods find and group similar text objects together. Classification and clustering share similar approach, although clustering tries to find the groups to put clusters into and categorization sorts into pre-defined groups. Therefore, clustering is considered an unsupervised learning method. The textual structures can be of a different granularity - documents, paragraphs, sentences or even words.

To ensure effective clustering, word frequency normalization is essential. A lot of words appear in many documents without carrying different meanings and their appearance might affect text analysis negatively. One of the most common representation for text processing is a vector-space based TF-IDF representation. In this representation, the word frequency is determined by an inverse document frequency, also known as the IDF. The IDF normalization ensures reduction of the most common words by lowering weights of the frequent words. The method highlights the less represented words to be a determinative factor for creating clusters. Another method is k-means clustering where the algorithm iterates through the document and is calculating the distance between the cluster word and the cluster center. It is unsupervised machine learning method. These two methods are usually used together at the same time. TF-IDF is used for feature extraction and k-means as a sorting algorithm to create clusters. [9]

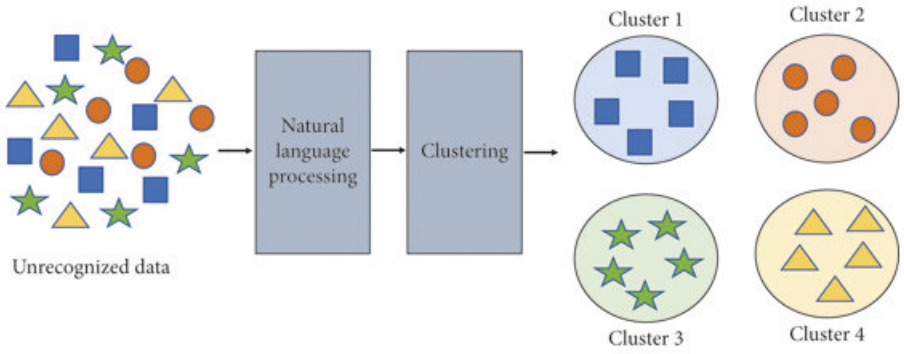

Fig. 1.3: An example on how clustering works. [10]

1.5 Information extraction

The aim of information extraction is to uncover structured data from unstructured or semi-structured texts. Examples of the fields of its application are biomedical research, finance, data analytics and web search engines used in marketing.

Information extraction systems benefits from supervised learning algorithms. Examples of these systems might be support vector machines and maximum entropy models. As information extraction can be looked at from the perspective of a classification problem solving task, sequence labeling methods, such as a hidden Markov model and conditional random fields. This method focuses on two types of classification that are the named entity recognition and relation extraction. Both of them rely on statistical machine learning methods, rather than rule-based approach.

The fundamental structure of named entity recognition is a named entity which is represented by a sequence of words that stand for an actual real-world entity, for example "VUT". The goal is to identify those named entities from free-form text and then classify them into predefined categories such as a person, an organization or a location. A lot of named entities are context-dependent so the right corpus has to be chosen. One of the main models of statistical learning approach is *the Markov model*. Both *Hidden* and *Maximum Entropy Markov* models are frequently applied. The difference between those two probabilistic frameworks is that the *Hidden Markov model* describes generative model of the data and *Maximum Entropy Markov* model relies on a discriminative representation of data which ensures a lower prediction error rate.

Relation extraction focuses on a detection and characterisation of semantic relations between entities in the text on a sentence-level. Here, a feature-based classification or kernel models are the options to choose for the application in the information extraction systems. Kernel models try to define kernels and apply kernel machines, such as a support vector machine. Feature-based classification relies on word sequences, dependency trees, dependency paths, and parse trees. Disadvantage of the methods is a need for a large amount of training data. [11]

Other useful and practical methods for information retrieval are *bag of words* and *bag of n-grams*. *Bag of words* model transfers text into vectors with their own coordinates. First vector coordinates represent the index of a token where a particular word occurs. The second coordinate is an iteration of all new words. If the words are the same, the second coordinate is the same for all the same words. The result of the model is a frequency of certain words or summation of all tokens.

Similar model is *bag of n-grams*. Its advantage is that it can identifies compound nouns or phrases. This is possible by defining different sizes of n-grams. If n-gram is equal to one, it identifies only single words. However, if n-grams are defined to be clusters of two, compound nouns are being identified instead. [12]

1.6 Text summarization

Automatic text summarization aims to create a short and coherent summary containing the key points and ideas of the original text, as well as preserving the same overall meaning. Two approaches to text summarization exist - extractive and abstractive summarization.

Extractive summarization identifies the most important sentences from the text and then puts it together into a short summary. On the other hand, abstractive summarization creates a completely novel shorter text that convey the most critical information to have the same meaning as the original text remained. Advanced natural language processing methods need to be implemented to achieve understandable summary generated via the abstraction. Existing abstractive summarizers tend to rely on extractive summary as a part of their data preprocessing. Both of the methods benefit from applying machine learning models and representation weighting using TF-IDF normalization. [13]

1.7 Text analysis in machine translation

Text analysis and machine translation are both part of natural language processing. Both of these subfields use similar approaches and methods to solving their problems applied to textual data. In this bachelor thesis, there is a focus on using text analysis on machine translated specialized texts. Even though, both subfields are different, they are mutually beneficial. Text analysis preprocessing of text data is useful as a part of preprocessing for machine translation as it prepares raw text data for further utilization.

Another great application of text analysis to machine translation is its quality evaluation. A lot of methods can determine the quality of machine translated texts, error rate and accuracy of machine translation approach. Then the result of quality assessment can be visualized and presented via different text mining visualization methods. However, machine translation can be beneficial for text analysis as well. One of the applications is automatically translating huge data sets provided for multi-lingual analysis.

2 Translation

In the second chapter, the main focus is on machine translation. It starts with a broad explanation what translation is, what aspects it consists of and types of translation. Then the chapter moves on machine translation, its main principles and workflow, and its main types. The chapter ends on quality assessment of machine translation where methods and important metrics are explained, as well as common mistakes and errors, that might occur in translated texts, are discussed.

2.1 Machine translation

Translation studies combine different perspectives and methodologies from linguistics, anthropology, psychology, literary theory, cultular studies and other important fields regarding the study of a human nature. Its main goal is to deliver a semantic content of the text from one language to another.

There are two types of translation: literary and technical. Literary translation focuses on fiction, poetry and dramatical texts. Technical translation concerns with a translation of texts from different disciplines in science, technology, commerce. There is a need for preserving the content and a background knowledge of the right terms regarding different fields. Machine translation (also known as an automated translation) and computer-aided translation is beneficial for the technical type of translation. This approach ensures a faster process of translating and keeping high quality of texts where it can help with terminology consistency.

Machine translation serves the same purpose as a human translation. It is a sub-field of computational linguistics where a software translates a text or speech to a different language. Its core approaches can be described by the Vauquois triangle that is also shown in Figure 2.1. These strategies, together with their various combinations, are still applied in machine translation systems. The direct approach involves immense string matching with a possible target string reordering to accordance to the target language syntactic rules. Transfer translation systems analyze and then generalize different grammar structures in order to achieve the transfer function from source language to generate the correct forms in the target language. The transfer function is usually a very abstract representation of the grammar. The last strategy is interlingua, where the source language texts are analysed to create its language-neutral representation. Then the generation of texts in the target language takes place. [16]

Then were developed different types of machine translation systems that derive from these core strategies. These are rule-based, statistical and neural machine translation.

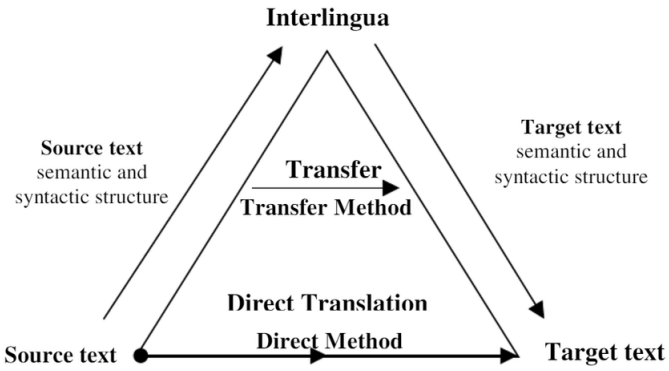

Fig. 2.1: The Vauquois triangle. [17]

2.1.1 Rule-based machine translation

Rule-based machine translation is a translation system based on applying rules representing grammar and dictionaries that provide linguistic information about morphology, syntax and semantics of both target and source languages. The process of translation consists of syntax and semantic analysis, and syntax and semantic generation. It also links structures between the input source language and output target language with preserving their meaning. Even though the rule-based machine translation was one of the first developed approaches to machine translation with a lot of done research, the need for huge and correct dictionaries, created by linguists, is one of its biggest downside. The rule-based machine translation uses all of the strategies mentioned in the Vauquois triangle - direct, transfer and interlingua approaches. [18]

2.1.2 Statistical machine translation

Statistical machine translation is characterized by the utilization of machine learning methods as it handles translation of a natural language as a solely machine learning problem. The developed learning algorithm is applied to a previously translated text, known as a parallel corpus, parallel text, a bitext or multitext. In the next stage, the algorithm can translate completely new sentences based on the corpus. The benefit of statistical machine translation is its time efficiency where translation can be done on a less known language pairs in a quite short period of time.

In general, the goal of statistical machine translation is to take a sequence of tokens in the source language and convert it to a sequence of tokens in the target language. In the usual case, tokens represent words and sequences represent sentences. The requirement for the translation is a precise and consistent preprocessing

of all data as statistical translation systems are sensitive to discrepancies. The whole translating problem can be divided into four steps: defining a translation equivalent, apply parametrization, identify score parameters via parameters estimation, and decode it.

Translation equivalent, also called simply a model, is a set of rules to transform source language into the target language. The most popular models are finite-state transducer and context-free grammar. Both of them try to find two outputs and find alignment between them. Finite-state model can be also very beneficial during the text analysis preprocessing stage of part of speech tagging. This method is further divided into word-based and phrase-based models. [19]

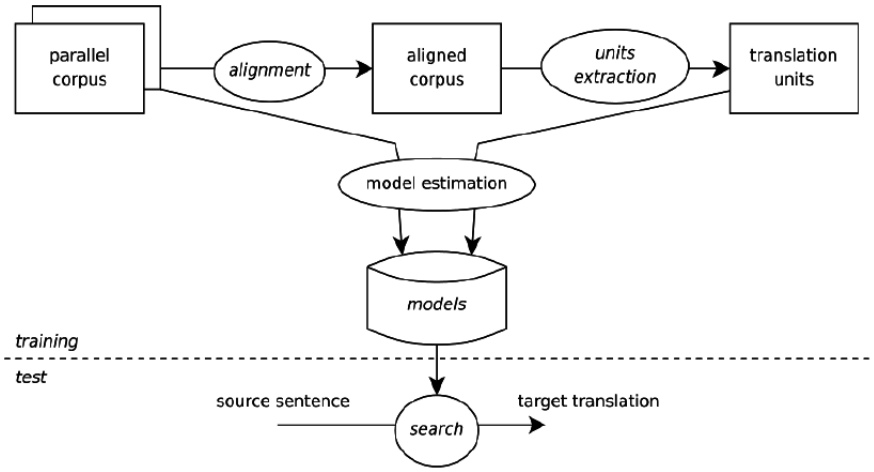

Fig. 2.2: An example of the general statistical machine translation pipeline. [20]

2.1.3 Neural machine translation

Neural machine translation is the most researched machine translation in academics and commercial use. In comparison to the statistical machine translation systems, neural translation systems do not require any words alignment, translation rules definitions, or any other feature extraction. One of the main used approaches is the Embed-Encode-Attend-Decode modelling. This is the core of the most used neural architectures in machine translation, such as a recurrent neural network (RNN), convolutional neural network (CNN), and self-attention/feed-forward neural network (SA/FFN).

Neural machine translation systems can process multiple language pairs with a great efficiency due to better generalization based on application of transfer learning. Transfer learning, also known as knowledge transfer, benefits from the exposure of different languages and improves quality of the bilingual machine translation

system. Low-resource languages with small parallel corpora can make the most of this approach.

To translate texts, the encoder converts words in the source language into a word embedding. These are processed by neural layers and converted into the encoder representation that carries the contextual information about these words. Neural layers are usually stacked into multiple levels. The decoder then works with an attention mechanism, encoder representations, and previously generated words that create a decoder representation, to translate words into the target language. Both the encoder and decoder can be RNN, CNN or self-attention networks. However, the self-attention or feed-forward are the most widely used networks in machine translation.

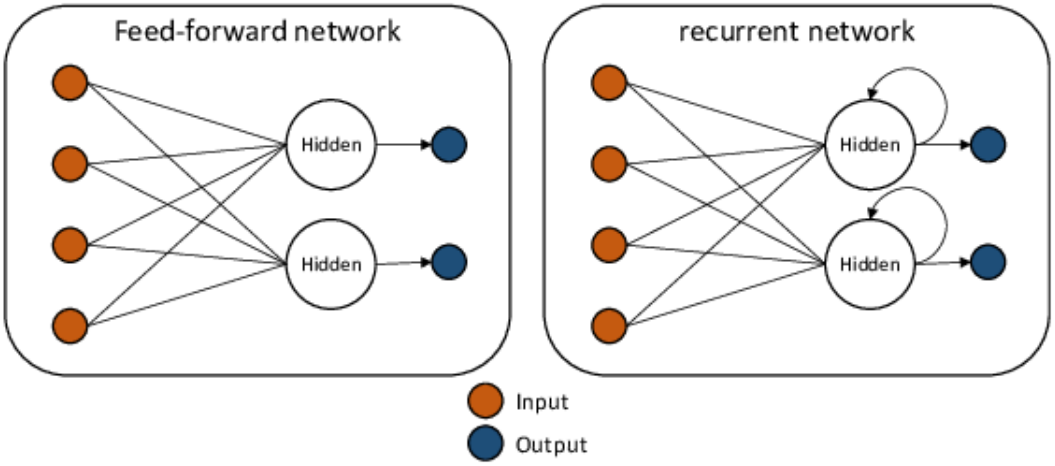

Fig. 2.3: Difference in architectures between feed-forward network and recurrent neural network. [21]

Before training the learning model, data has to be preprocessed via text analysis methods. Then, a new vocabulary consisting of the most frequent words is established and remaining words are mapped to a single token representing an unknown vocabulary. To train models efficiently, minimizing of cross-entropy between the predicted and actual target words is an essential step of the machine and deep learning process. The training is usually finished when it was done on a huge amount of iterations or the model converges adequately. The model convergence is reached when additional training iteration does not improve the result in any significant way. Other factors that influence the model training are a learning rate, hidden dimension size, number of layers, and so on. [22]

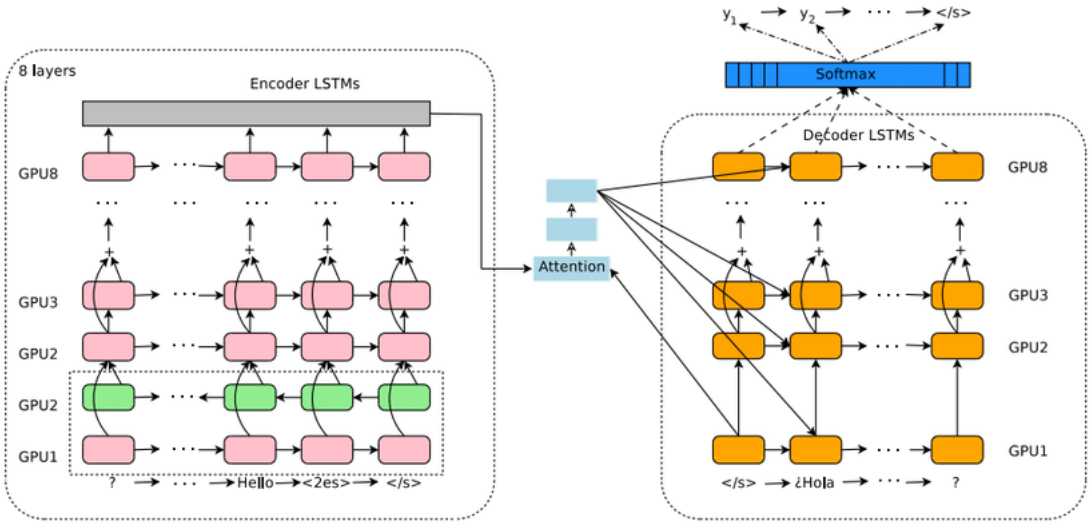

Fig. 2.4: Google Neural Machine Translation architecture for translating multilingual language pairs. [23]

2.2 Quality evaluation of translation

Translation is a complex process influenced by cognitive, linguistic, social, cultural and technological factors. Therefore, the translation quality assessment can be a difficult task to proceed as it reflects the complexity. The translation quality evaluation, together with machine translation, is a huge part of the recent researched topics.

In order to reflect the high quality translation, the text needs to have a certain degree of accuracy, fluency, readability, and comprehensibility. Accuracy, interchangeably adequacy, is an extent to which the meaning of the text in the source language is transferred into the target language. Fluency goes alongside the accuracy, and measures the grammar correctness, mistranslation, and undertranslation. Readability is affected by linguistic features, such as a word frequency, a sentence length, formatting, and spacing. It represents how smoothly the text can be read. The last factor is comprehensibility, which illustrates how understandable the text is for the reader.

Machine translation quality is assessed either manually or automatically. The automatic approach is based on comparison of a machine translated output to one or more reference translations which represents the most accurate outputs done by translators. There exist several metrics that calculate their score of the highest accuracy and similarity between machine translated texts and reference texts. These are part of a quantitative evaluation. [24]

2.2.1 Metrics

The quantitative quality evaluation metrics compare the machine translated outputs to the human-translated reference text. Automated metrics usually measure the overlap in an occurrence of words, word sequences, their order and edit distance. These metrics are usually based on n-gram statistical word matching modelling.

The first metric is the word error rate (WER) that derives from the Levenshtein distance. It calculates the minimum number of editing steps needed to transfer the machine translated output into the reference translation. WER metric takes word order into account, as well as the operations such as deletion of words, insertion of different words, and the replacement of words. The general formula for computing the WER metric is represented by Equation 2.1

$$WER = \frac{S + D + I}{N} \tag{2.1}$$

, where S is a number of substitution, D is a number of deletion, I is a number of insertion, and N is the number of words in the reference text.

The position-independent word error rate (PER) is based on the same principle than the WER metric, however, it does not take word order into account and different word order does not affect the final result. It counts the number of identical words present in the machine translated and reference text. [25]

Another metric is METEOR. The METEOR metric is based on flexible unigram matching using precision and recall, even between morphologically similar words with identical word stems, or synonyms. It measures fragmentation that captures how well-ordered the matched words are in the machine translated texts, compared to the reference one. Precision represents the ratio between the number of unigrams in the translated texts mapped to unigrams in the reference, and the total number of unigrams in the translated text. Recall is similar to the precision. It calculates the ration between total unigrams in translated text mapped to the reference text, and the total number of unigrams in the reference text. First, *Fmean* needs to be calculated. It combines the precision P and recall R via harmonic mean that puts bigger weight on recall. The formula is Equation 2.2.

$$Fmean = \frac{10 * P * R}{R + 9 * P} \tag{2.2}$$

. It also uses a penalty score when calculating the final score. The penalty score is based on the percentage of number of chunks divided by number of unigrams. Chunks are created based on all unigrams in machine translated text mapped to all unigrams in the reference. If there is only one chunk created, the translated text reflects precisely the reference. It is calculated by using Equation 2.3. [26]

$$Penalty = 0.5 * (\frac{chunks}{matchedunigrams}). \tag{2.3}$$

Therefore, the final METEOR score is computed using both Fmean and Penalty score. Its equation is

$$METEOR = Fmean * (1 - Penalty). \tag{2.4}$$

The most widely used metric is a BLEU score. The BLEU score compares n-grams of the candidate with the n-grams of the reference translation. Then the number of their matches is counted. The ratio of matching n-grams to the total number of n-grams makes the precision for each n-gram order. BLEU score is also sensitive to raw data, so the compared translated texts should be fully preprocessed, including tokenization, lower-casing, deleting punctuation, and other preprocessing tasks are recommended, as well, to get better results.

To get the BLEU score, first, a brevity penalty BP needs to be calculated. In BLEU score formula, c is the length of the candidate translation and r is the length of the reference corpus.

$$BP = \begin{cases} 1 & c > r, \\ e^{1-\frac{r}{c}} & c \leq r. \end{cases} \tag{2.5}$$

Then the BLEU ranking metric is calculated using Equation 2.6, where w_n are the uniform weights and N is equal 4. It computes the geometric average of modified n-gram precision p_n, using n-grams of the maximum length N and positive weights w_n. These are summed together to one.

$$BLEU = BP * exp(\sum_{n=1}^{N} w_n log(p_n)). \tag{2.6}$$

Because BLEU's range of values are from zero to one, it is more convenient to calculate the metric using the log domain, as it is represented by Equation 2.7. [27]

$$log(BLEU) = min\left(1 - \frac{r}{c}, 0\right) + \sum_{n=1}^{N} w_n log(p_n). \tag{2.7}$$

2.2.2 Error analysis profile

The aim of error analysis is to pinpoint and classify individual errors made by machine translation systems. The analysis provides a detailed error profile for individual machine translation engines. The error profile helps to highlight both strong and problematic areas of the system which might be harder to be identified using only automated metrics. The analysis represents a qualitative approach to the machine translation quality assessment. [28]

The various error types can be found in the machine translated texts. The most common and frequent ones differ based on both the target language and source

language. It is also affected depending whether the language pairs are from the same language group, or if the languages share similar structures and rules in their grammar. However, the errors can be divided into two categories: the fluency and the adequacy errors. The fluency errors are incorrect verb and pronoun forms, wrong word order, incorrectly split compound words, incorrectly identifying foreign and unique words, spelling errors, incorrect prepositions and other errors affecting the grammar precision of the translation. The adequacy errors also include the wrong word order and verb form. The difference in these two cases is the adequacy types might be grammatically correct, however, the correct meaning of the source text is not reflected in the text. Other errors in the second category are mistranslation, adding extra words, ommision. These categories can be further divided into other subgroups of different errors, too. [29]

Additionally, technical documents can contain terminology errors. These are usually domain-specific and the quality of machine translated technical terms is heavily based on the quality of training corpora or data sets. The machine translation process also relies on a huge bilingual data set, which is quite problematic to obtain in case of less known languages. In case of terminology, the most common mistakes are: term being redundantly translated, mistranslated as a wrong term, or mistranslated due to wrong acronym. To solve this issue, many neural machine translation systems are using the injection of terminology approach, where the decoder of the neural network is enhanced during training process. Another popular solution is to use a placeholder instead of directly translating the term. The placeholder is usually marked with some identifier, such as its respective part-of-speech tag, its inflection form or morphological features. [30]

3 Tools

This chapter discusses options for programming languages applied to text analysis, text quality evaluation, and machine translation. Their advantages, disadvantages, and their comparison to each other are explained. The other half of this chapter talks about deep learning networks that can be applied to error and grammar correction. Types and application of different pre-trained language models are explained, too. At the end of the chapter, a short summarization of existing translating tools is presented.

3.1 Programming languages

The most popular programming languages are Python, MatLab and R. In this chapter, research about possibilities for text analysis tasks are explored, as well as comparison of these three programming languages.

3.1.1 Python

Python is a high-level programming language and is an open-source with a huge variability of importing libraries and modules to suit many different purposes. It supports both paradigms of scripting and object-oriented language. Python can handle working with a good amount of data, therefore, domains, such as database programming, data and text analytics, machine learning, opt for Python as the main programming language. The only disadvantages are slower execution speed compared to low-level programming languages and its incompatibility between different versions of Python.

There are many different libraries, tools and toolkits for natural language processing tasks in Python. The most popular ones are NLTK, spaCy, scikit-learn, Gensim, TextBlob. NLTK is one of the most popular one because of its flexibility. It is mainly used in academic research as the toolkit provides many different approaches that can be implemented through it. spaCy is the competitor of the NLTKs, however, only provides one optimal implementation solution for the tasks. Although these two libraries differ in the number of implementation possibilities, they can be applied to the same natural language processing tasks. Another slight difference is that spaCy includes machine learning models and store textual data as vectors which is beneficial for more complex tasks. Regarding the application, both libraries can be used interchangeably. Other packages are more specific than these two. Scikit-learn library consists of machine learning modelling for classification,

regression, clustering, and even preprocessing tasks. [31] Gensim also uses unsupervised machine learning. Its main applications are topic modelling, document indexing and similarity retrieval. It also works with textual data as semantic vectors. To be able to use this librariy, Numpy and Scipy libraries need to be installed, too. [32] And the last popular Python library for natural language processing is TextBlob. TextBlob library can be applied for the main text analysis and processing tasks, such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, spelling correction, text preprocessing, and more. However, it is dependant of NLTK library that dowloands automatically with TextBlob. [33]

In order to create, train or use artificial neural network models, deep learning frameworks are necessary. The application of deep learning is also broadly spread in natural language processing and machine translation as there is a need for recurrent neural network architectures. One of the most prevalent and used deep learning frameworks is TensorFlow. TensorFlow is an open source computational and deep learning framework developed by Google Brain team. It is based on computational graphs where their nodes represent mathematical operation and their edges show the data flow between these nodes. TensorFlow provides visualization tool depicting the data flow during modelling. It has CUDA support, too. The NVIDIA CUDA deep learning library is a necessity for GPU acceleration of computational power. The advantages of TensorFlow are its building variety of different neural network architectures, visualization tool, model parallelism where different parts of model can be trained on different devices. TensorFlow also benefits from using Keras, research-based neural network library. It allows better and easier prototyping of models and supports CNN networks too. The disadvantage of TensorFlow is its complexity and is more difficult to learn. [34]

3.1.2 MatLab

Matlab is a unique programming language and a scientific computing software environment developed by MathWorks. Matlab allows matrix manipulation, visualizing data, implementing algorithms and user interfaces, and integrating with other programming languages. It has a great computational power in order to work with huge data sets.

For text analytics, there is the Text Analytics Toolbox. The library contains functions applicable in preprocessing, visualization, and modeling text data. It is a great option for text categorization and clustering, sentinent analysis, text summarization, search engines, computing similarity between documents, information extraction and presenting the text data via scatter plots and word clouds. It provides support for both statistical and machine learning approaches by implementing

Transformer models for any text analysis process mentioned in this chapter section. [35]

3.1.3 R

R is a statistical programming language used heavily in data mining and analysis, bioinformatics and statistical software development. Its software environment is free and open-source with a possibility of adding various packages and libraries to enhance its purpose. It also supports object-oriented programming approach.

As it is an industrial standard for data mining, R can be applied in various use cases for text mining as well. Textual data preprocessing, word and document frequency analysis, sentiment analysis, correlations analysis between words, topic modelling, and other techniques can be achieved by R and its various packages. One of the examples of available packages providing the main text mining techniques is 'tidytext'. [36]

3.1.4 Comparison of programming languages

To conclude the researched options for programming language, their comparison is presented in this part of the chapter. Python, MatLab and R are the most popular languages for text analysis purposes. However, the aim of the thesis is to also use machine translation methods for evaluation of quality. Programming language R uses statistical approaches and therefore, it is most useful when applying to text mining. It is suitable for evaluation of translated texts, however, to train neural networks for error correction using machine translation approach is very limited. In contrary, both MatLab and Python do have similar application and methodology options for both text analysis and machine translation. The goal of the thesis experimental part is to leverage pre-trained language models, as the error and grammar correction is a very specific task with a much smaller training data set available than it is needed for training language models from scratch. MatLab utilizes a great computational power for training deep learning networks but has limited options for pre-trained language models. Although MatLab has a great variety of pre-trained models for image processing, available pre-trained language models are only BERT for text classification and GPT-2 for text generation. As Python is open-source with a huge community, options for pre-trained language models are much greater.

Therefore, the deciding factor for the most suitable programming language for this thesis are pre-trained models. Python provides variety of text analysis methods, as well as many deep learning networks using transfer learning applied to error and grammar correction task.

3.2 Pre-trained language models

Natural language processing problems usually combines a classification method and a latent feature representation. It started with applying traditional statistical approaches to design the classification feature and then applying machine learning model to learn the classification function. Then deep learning approach identifies the unique latent feature representation for each task via neural networks alongside the classification function. However, majority of natural language processing tasks share similarities, a general feature representation can be established and reused for more specific tasks. In general, language models predict and generate words based on the previous ones, therefore, they can be pre-trained on huge general data. Then the existing pre-trained models can be exploited via restructuring task into text generation, fine-tuning, or prompting, in order to be applied to much more specific problems.

The most used architectures are autoregressive, masked, and encoder-decoder language models. All of them are based on the Transformer model and benefit from different structures within this architecture. [37]

3.2.1 Transformer architecture

Google's neural model was the first Transformer model for machine translation. It consists of an encoder network, a decoder network, and an attention network. The encoder and decoder networks are connected through the attention network which enables the decoder to go through various parts of the source sentence during its process. The encoder serves the purpose of transforming source sentences into a list of vectors and each vector is then labelled as a symbol value. The decoder works on a principle of a recurrent neural network (RNN) and a softmax layer. The RNN predicts the next symbol in the sentence and the softmax layer generates its probability distribution over the candidate outputs. The model itself consists of 8 layers of Long Short Term Memory network for both encoder and decoder. The layers are then stacked together with residual connections. Residual connections mean that the input provided for a bottom LSTM layer is also added element-wise to the output from the bottom layer. Their sum is then sent to the top layer as the new input. Another component of the transformer architecture is a word-piece model which benefits from using a sub-word units approach. It is data-driven and it generates a deterministic segmentation for any possible sequence. This approach ensures a better translation of rare words in different languages without the need to include them in a training data set. [38]

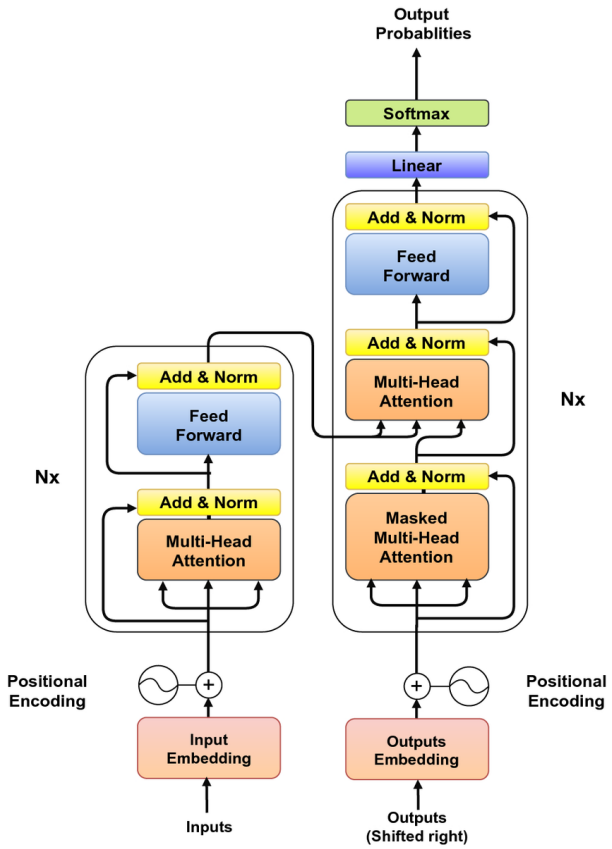

Fig. 3.1: Transformer model architecture. [39]

3.2.2 Types of pre-trained language models

There are many language models that leverages the Transformer architecture. They are trained on massive general data sets, in order to build pre-trained models for more specific tasks in natural language processing. The examples of derivative architectures are Language Understanding Models (NLU), Text Generation Models (NLG), models with size reduction, models pre-trained for information retrieval in particular, long sequence models, and computationally efficient architecture. The first two categories cover the most known architectures for natural language processing, therefore, these two are explained in more details in this part of the chapter. Another reason is the fact that the other architectures are improved the NLU and NLG architectures for these particular use cases.

The NLU models are more generalized and were trained further on larger data. The examples of these models using the NLU architecture are Transformer, GPT-I, BERT, XLNet, Megatron, etc. The main goal is to understand the natural language, so the dialect is mapped to a formal interpretation to create a conventional phrase representation. However, the fine-tuning approach to pre-training them is

challenging for task-specific purposes as other layers have to be implemented and dedicated to fine-tuning. It increases the size of the model itself, as well as need for a large data set. Therefore, the NLG models were created. The training process is the opposite to the training of NLP models. The generated language representation is learned from the corresponding masked or corrupted semantics. They are also called sequence-to-sequence generation models and are the optimal choice for machine translation. Models such as T5, BART, mBART, GPT-II belong to this category. The downside and need for further development is the size of these models, the need for several GPUs when the training data set is large, or the inefficiency when dealing with longer text sequences in data sets. [40]

3.2.3 Examples of pre-trained models

The most well-known Transformer-based pre-trained models are BERT, GPT and T5 models. All of them are can be used for natural language generation tasks, which also include text summarization, query answering, and language translation.

GPT models are decoder-based and goes through unsupervised learning during the initial training. The main advantage of its training is no need for data labelling which is a time-consuming process. The pre-trained model can be then fine-tuned on small supervised data sets. The model performs complex language understanding via common-sense reasoning, semantic similarity, and reading comprehension.

BERT is a bidirectional encoder model. It has a stack of pre-trained Transformer encoders that provide a token with several contexts through these multiple layers. The bidirectional network increases the diversity of learning. The downside might be a chance of foreseeing the future better tokens during pre-training and cause the trivial predictions. Hence, the application of masked language modelling that masks a certain proportion of all input tokens randomly in each sequence. To maintain a bias towards the correct prediction, the cross-entropy loss is employed. The randomized masked tokens ensure the contextual representation of each token, too.

The last well-known model is T5 which is an encoder-decoder Transformer-based neural network. T5 utilizes the most effective transfer learning practices when pre-training. It frames the input and output strings into an unified text-to-text structure. It also includes a fully visible masking for the input sequence and occasional masking for the target prediction of the output sequence. The advantage of this model is its outperformance of decoder-based language models. The T5 model can be beneficial for domain specific tasks, although there is a possibility of overfitting when training on small data sets. [40]

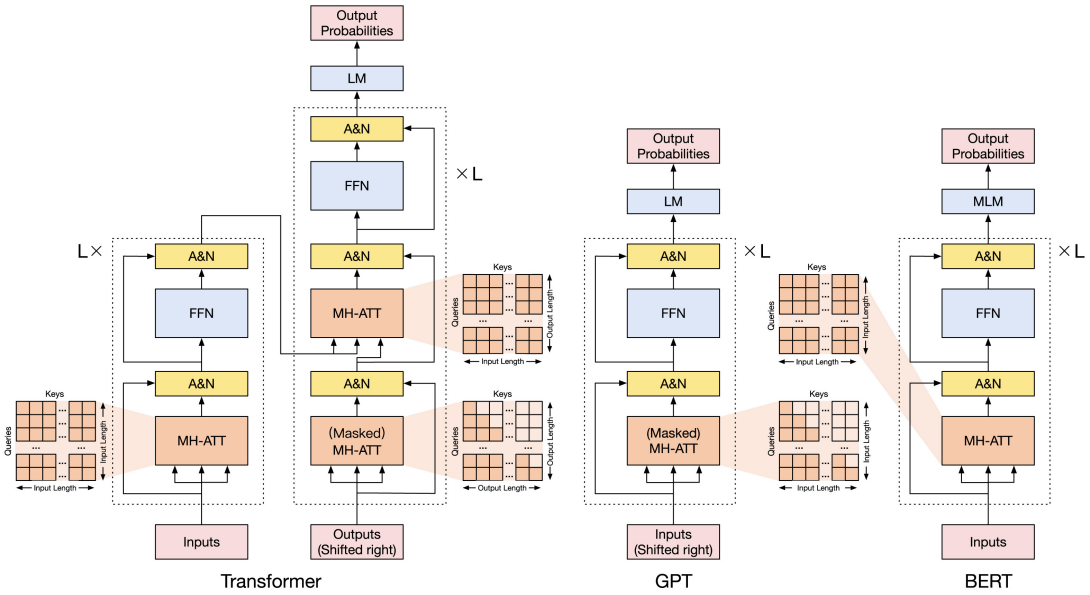

Fig. 3.2: Comparison of Transformer, GPT and BERT model architectures. [41]

3.3 Existing translators

There have been several advancements in the development of already-existing translating systems, too. Nowadays, majority of them are also Transformer-based neural machine translation engines. The most popular translating tools are Google Translate, DeepL Translation, Systran, and Microsoft Bing Translator.

Google Translate and DeepL Translator are both multilingual neural machine translation tools. The difference is in their model architectures. Google Translate works on the Google's Neural Machine Translation network that utilizes the encoder-decoder Transformer model based on the recurrent neural network. On other hand, DeepL Translator is based on the convolutional neural network architecture. Systran Translate has developed its own Pure Neural Machine Translation architecture where the initial translation is done by machine translation system, and then provides users with post-editing interface. [42]

4 Solution Proposal

In this chapter, the solution, to evaluating accuracy and error rate in machine translated technical texts using text analysis approach, is proposed. The proposal is made according to the research done in previous chapters by selecting the best methods, tools, and approaches.

4.1 Proposed methodology

The main goal of the thesis is to explore possibilities of different text analysis methods when evaluating the quality of machine translated technical texts. In this experiment, both machine translation and text analysis approached are combined together to create a quality assessment of various popular translating tools, such as Google Translate, DeepL, Systran, and Microsoft Bing.

The process starts with text analysis preprocessing tasks, which are crucial for language grammar correction. In this part, tokenization and cleaning textual data is the first step, followed by part of speech tagging. Usage of stop-word removal, lemmatization and stemming might be considered for the last part of the experiment. However, in this particular case, important data can be lost via these methods, as language grammar information are removed or altered.

After the textual data is ready, machine translated texts are compared to the reference text. Reference text is considered grammatically correct for the purpose of this experiment. After the comparison is done, metrics, such as BLEU score, similarity score, and METEOR are calculated and the results are stored in the dataframe for a better visualization. Sentences are evaluated on token-level, where different words are stored in a separate table with error classification categories. Error classification task consists of rule-based and statistical methods via determined rules and statistical functions to identify the correct error category. The final evaluation result is to present error rate and accuracy of different translating tools, when applied to more technical texts.

The last part of the experiment is to train and apply pre-trained language model by using transfer learning approach. The deep learning model is a grammar correction model that learns to identify and correct corrupted sentences. This is the proposed solution on how to improve machine translation post-editing process.

In conclusion, the main proposed methods are preprocessing, information retrieval, feature extraction, token classification and text generation. Other methods, such as text summarization and text clustering, do not provide any useful information about error rate, accuracy, and quality of machine translated texts. Although

these methods can be used on individual texts to provide better semantic understanding of the text, mistakes in translated texts could affect their result. Therefore, other methods providing improvements need to be applied first.

4.2 Proposed tools

Python seems to be the right programming language for this thesis experiment. The reason is its accessibility of different tools and right frameworks. As there are several Python libraries and packages for natural language processing tasks, comparison was done on the most popular ones: NLTK, spaCy, Gemsin, and scikit-learn. For the text analysis evaluation tasks, the main package is NLTK as it provides all text analysis methods without the need to incorporate other packages to include the needed methods to reach the aimed result. NLTK is also the most used library for research, because it has selection of various functions for the same task. Gemsin and scikit-learn provide machine learning approaches. For the purpose of this thesis, different machine learning methods are not applied. Although spaCy is the biggest competitor to the NLTK library, it provides only the most optimal solution, which might not be suitable for the quality evalution of machine translated texts. NLTK is the proposed library for natural language processing tasks.

Other reasons why Python is proposed is the possibility to implement the proposed solution in more efficient way. Python is open-source, therefore has a huge community that provides larger variety of different approaches to solving the problem. The other advantage are the resources for pre-trained neural models that can be applied to natural text processing.

5 Implementation

In this chapter, the whole process of the final implementation is described. The goal of the result is to create a language error identification and classification algorithm using known text analysis and machine translation methods.

5.1 General workflow

Thesis experimental part consists of two main functions: error analysis and error correction. Error analysis serves to identify and categorize the most common translation errors using text analysis methods, such as feature extraction, classification, and preprocessing. Error correction uses the machine translation neural model to correct different translation mistakes.

The general workflow of the error analysis starts with cleaning the text data. Many methods are dependant on working with clean data in order to avoid analysis inaccuracy. The second step is to apply error identification and classification analysis algorithm. The result is an error analysis profile with a conclusion what error categories were detected in the machine translated texts. The last part of text analysis evaluation is to conclude the accuracy of each translation tool. The most common translation evaluation metrics are calculated and presented. The results are always stored in DataFrame in order to export the analysis results into the CSV file format.

After getting the quality evaluation analysis, the error correction neural model can be applied. It can correct every error category detected by the text analysis algorithm. The model serves as a solution to help post-edit machine translated texts and increase efficiency of the whole translation process.

5.2 Tools

According to the solution proposal, the main programming language is Python. One of the main reason is its accessibility of having text analysis, machine translation methods and deep learning framework in one place. The main library for natural language processing is NLTK, which provides all text analysis methods and translation evaluation methods, too. Other important libraries are numpy [44] and pandas [45], which are beneficial when dealing with data mining. The last used library for text analysis is the FuzzyWuzzy library [46]. It provides fuzzy string matching and computational capabilities for calculating Levenshtein Distance between strings.

The libraries needed for the error correction model are HappyTransformer [47], and datasets [48] library. The other prerequisites for the thesis experiment is to

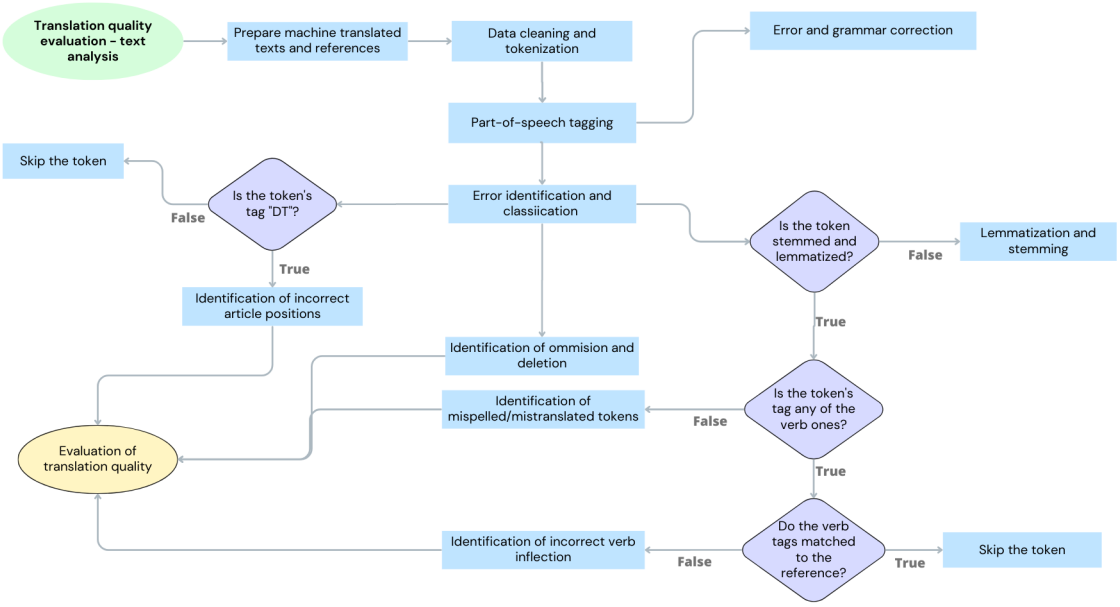

Fig. 5.1: General workflow of the implemented solution.

have TensorFlow, NVIDIA GPU driver and support for the CUDA architecture. The reason is to be able to train the deep neural network efficiently. Training on CPU might not work.

5.3 Data sets

The main data sets are machine translated texts. Texts were translated from Czech language to English language by using the four most popular translation tools, such as Bing, DeepL, Gooogle Translate, Systran. The main represented topics were related to electrical engineering to have domain-specific terminology used in the texts.

In order to provide a certain benchmark for quality evaluation analysis, one reference text for each set of machine translated texts were provided. The reference data set contains human-translated texts that are considered correct for the purpose of this experiment. Every set of texts is stored in the .zip file "textAnalysisDataset.zip". The file is provided via the electronic attachment.

The error correction model was pre-trained on one of the standard benchmark data sets, "JFLEG: A Fluency Corpus and Benchmark for Grammatical Error Correction" [43]. The data sets was extended for sentences used in each training cycle to contain more technical terminology. The reason for the extension was to increase

error correction model accuracy for this particular use case of correcting technical texts.

5.4 Data preprocessing

The first step of text analysis is cleaning textual data. A new function called 'dataCleaning.py' was implemented for this process. Its input is the loaded data that goes through tokenization via a simple string splitting into a list of strings. Lower casing and removing punctuation is done separately by iterating through each token in the list. Then, the function returns a list of cleaned word tokens. To disclaim, sentence tokenization is also provided in order to ensure text analysis in more depth on the sentence-level, rather than document-level. However, the experiment works on the word-level, when applying error identification and classification algorithms.

After the first preprocessing step, part-of-speech tagging was applied. For the part-of-speech-tagging, a function from the NLTK library was used. The output of the process is a tuple containing the word token and its part-of-speech tag. These tuples are stored into a DataFrame table for better accessibility.

In case of grammar correction and error analysis, lemmatization, stemming and stop words removal are not beneficial for the whole text analysis as these methods could affect the grammatical structures of translated texts and the result would not be correctly evaluated. However, some of the error identification implementations works better with these preprocessing methods. If that is the case, the applied method is described in the subsection of the chapter about the error identification and classification program.

Tab. 5.1: An example of data cleaning and part-of-speech tagging.

Original sentence	**Data cleaned sentence**
Gallium arsenide (GaAs) is a III-V direct band gap semiconductor with a zinc blende crystal structure.	('gallium', 'NN'), ('arsenide', 'NN'), ('gaas', 'NN'), ('is', 'VBZ'), ('a', 'DT'), ('iiiv', 'JJ'), ('direct', 'JJ'), ('band', 'NN'), ('gap', 'NN'), ('semiconductor', 'NN'), ('with', 'IN'), ('a', 'DT'), ('zinc', 'NN'), ('blende', 'NN'), ('crystal', 'NN'), ('structure', 'NN')

5.5 Error identification and classification

The most frequent mistakes during translation are spelling, replacing the original vocabulary with different words and terminology, deleting or adding unnecessary

words, wrong grammatical inflection of verbs, and wrong placement of articles. In this thesis, these error types contribute to the final text analysis result of the different translating tools, and their comparison. This part of the thesis leverages rule-based and statistical approach to the token classification. The final result is the error table, where each error category is represented and the error occurrence is noted. The main error categories are labelled as 'Article position error', 'Ommision/Addition error', 'Verb inflection error', and 'Spelling/Translation error'.

5.5.1 Identification and classification of the article error type

Articles are difficult to translate via machine translation tools as many languages do not have them. Therefore, it is one of the most difficult aspect to consider when identifying mistakes, as well as correcting them. The idea behind the implementation of the incorrect articles identification is that the respective tokens would have different index number than the reference. Firstly, tokens classified as articles are identified by their part-of-speech tag. Then, the index values are stored in a list of arrays. The index values are compared to the reference through the for loop algorithm. If the values are different, they are classified as wrong article error type. This is represented by 'Incorrect' marker in the error table. If the index values are the same, they are represented by 'Correct' label in the error table. After the error analysis of article position, all article tokens are removed as they are considered as stop-words. The error correction model also works better without articles contained in the text, because they create noise.

The reason for a wrong article position might be adding or removing different words and therefore, the results might be affected by this another error type. As a solution to still give an insightful text analysis result, individual appearance of each article is analyzed. The articles could be either *'a', 'an'* or *'the'*. The whole list of present articles are displayed and their number of occurence, together with its index position value, as well as the bi-grams with the article and the next noun, are shown. The result is then stored in a separate table to provide more information of this error type for further post-editing done by a human translator.

The main text analysis function implemented for identification and classification of this error type can be found in 'analysisFunctions.py' file under the name 'article-Analysis'. The function's inputs are the articles' index position of a reference text and a translated text. It returns the most frequent error type category.

5.5.2 Identification and classification of unnecessary translation actions

Another error type category is 'Ommision/Deletion'. Here, the algorithm compares the length of translated token lists to the reference. If the translated sentences are longer, addition of unnecessary words takes place. If the translated sentences are shorter, unnecessary deletion of words happens. The possible labels, representing this error category, are 'Ommision', 'Addition' and 'Unchanged'. This part of text analysis is very useful on sentence-by-sentence lever, rather than on the document-level. Even though the algorithm counts the number of appearance of each category in the whole text, and then the most occurred mistake category is displayed as a result, comparing each sentence separately shows exactly what action happened where and needs to be post-edited by a human translator.

The implementation of the identification and classification of unnecessary translation actions error type can be found in 'analysisFunctions.py' file under the function definition of 'additionOmmisionAnalysis'. Its two inputs are a lenght of the reference and the translation. It returns the most frequent error category when analysing the whole text. If the analysis is done on sentence-by-sentence level, the individual sentences go through the analysis function separately. Therefore, the input is corresponding reference sentence and the translated sentence. In this case, it returns the identified category.

5.5.3 Identification and classification of verb inflection error type

One of the most occurring mistakes during translation process is the incorrect inflection of the verb forms. The NLTK's part-of-speech tagging function provides tags with detailed information about the tokens' grammar structures. Therefore, instead of identifying verb structures, the identification of correct verb-form tags are detected. After the verb detection is done, the statistical occurrence frequency is calculated. If the number of certain verb-form tags does not match with the reference, the sentence is considered to contain wrong verb inflection forms. The verb forms that do not match, are stored separately. Then they are grouped by their lemma forms to get a detailed information of the verb structures. The lemmatizer used is *'WordNetLemmatizer()'* provided by NLTK library. To have a complete text analysis of word forms, stemming is applied, too. Porter's Stemmer is used.

Its implementation can be found in 'analysisFunctions.py' file and the corresponding implemented functions are 'verbErrorCountAnalysis' and 'verbErrorAnalysis'. The first function identifies the respective verb part-of-speech tags and displays their frequency number as a result. The second function is a categorization function

that takes the result of the first function and according to that, catagorizes whether it is a correct or incorrect verb form. The reason for having two function is the ability to better provide text analysis' results of what verb forms are incorrect. This can be displayed in a different verb table that also returns the whole verb list in their lemma and inflected forms, too.

5.5.4 Identification and classification of mispelling and mistranslation error types

The last category is mispelling and mistranslation. The identification of these two error categories is based on the idea that the incorrect word is the least frequent one, in comparison to the reference text, as it is not clustered with any identical tokens. Lemmatization is needed for the word frequency identification. The reason is that different grammatical forms of verbs in different translation would be labelled as incorrect vocabulary, even though the verb carries the same meaning. Therefore, the classification into the incorrect vocabulary category is based on the lemmas of tokens. For incorrect vocabulary, the words with the count of one are labeled either as incorrect vocabulary or wrong spelling. To identify misspelled tokens, the fuzzy matching algorithm is used. If one of the least frequent tokens match of at least 70 percent with the reference one, they are categorized as a 'wrong spelling' error type. If tokens do not have matches of at least 70 percent, they are classified as the 'wrong vocabulary' error type. The prerequisites for misspelling identification is that tokens need to be stemmed. The reason is that verbs with only different suffixes can appear as misspelling, even though they can be spelled correctly. Although mispelling might happen, the expected result is that mistranslation happens more frequently. The reason is that data sets used for neural machine translation are less likely to contain spelling mistakes as they undergo data cleaning processes, and therefore, it is more likely that machine translation system misunderstands certain words and their context more often.

The implemented function 'vocabularyAnalysis' can be found in 'analysisFunctions.py' file. The function's inputs are the reference text and the translated text. The implementation returns the resulting error category of the applied text analysis. However, in the main program the function is used twice on original texts and stemmed texts in order to compare their result without the limitation mentioned before. The reason is to return more accurate result of the text analysis. This comparison can be displayed in a separate table to give more insights for the further post-editing process of translation.

Tab. 5.2: Final error table displaying average results for error identification and classification applied to all texts in data set.

	Article error	**Ommision, Addition error**	**Verb inflection error**	**Spelling, translation error**
Reference	Correct	Unchanged	Correct verb form	-
Bing	Incorrect	Ommision	Incorrect verb form	Mistranslation
DeepL	Incorrect	Unchanged	Incorrect verb form	Mistranslation
Google Translate	Incorrect	Addition	Incorrect verb form	Mistranslation
Systran	Incorrect	Ommision	Correct verb form	Mistranslation

5.6 Pre-trained model for error correction

The proposed solution for grammar and error correction is the application of a pre-trained neural model. According to the research, sequence-to-sequence models would be the best option for this task as they transform the textual input into another textual output. Grammar correction can be looked at as machine translation process, where the corrupted input is being translated into an output in the grammatically correct form. Pre-trained models leverage the transfer learning approach. In the pre-trained models only last layers are trained again on smaller and more task-specific data sets.

In this experimental part of the thesis, pre-trained models by Happy Transformer were selected. Their *'HappyTextToText()'* library object can initialize two types of text-to-text Transformer-based pre-trained models, T5 and BART. The models were pre-trained on huge data sets. Therefore, they could be used during the transfer learning process, where they are trained again for more specific task via fine-tuning. During the fine-tuning process, settings parameters and model parameters were adjusted. The number of beams were changed to 10 from the default value of 1. This parameter determines a number of steps for each search during the beam searching. The algorithm, that goes through the best possible options when learning, was the beam search. It showed better results than the greedy search algorithm during the experiment. Another adjusted parameter was a number of epoch that was changed to 6 instead of default value of 3. Epochs represent a number of complete iterations through the whole training data set. At the beginning, I started fine-tuning with the default value. However, the increasing of epochs and decreasing of batch size showed the best possible result when learning. The batch size was set to 1 as the learning was the most precise with the lowest possible batch size. For the purpose

of this experiment, both T5 and BART models were trained. However, BART model showed better results when encountered with edge cases. It could detect the basic form of words better, so it reacted better to odd spelling. BART uses masking, which also helps with inserting correct articles to the sentences. As the final solution, BART was chosen for the error and grammar correction model. It has 6 encoder and 6 decoder layers. The number of attention layer heads are 12 for each learning structure. During training, its loss function is calculated and its characteristic is shown in Figure 5.6. It indicates the effectiveness of the learning model.

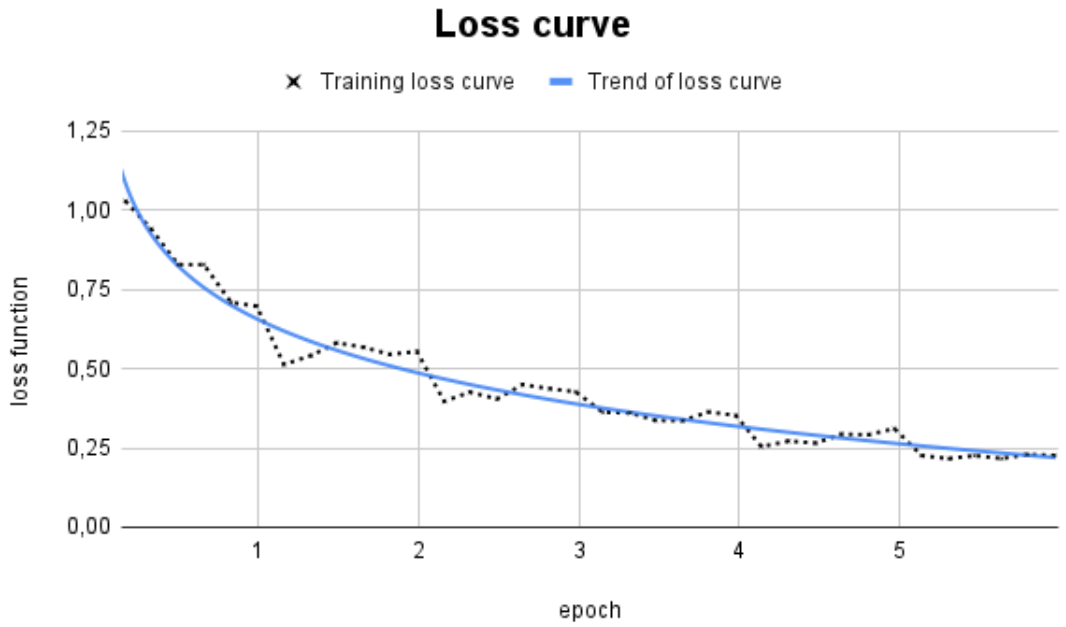

Fig. 5.2: The loss function curve representing learning process of the error and grammar correction model.

It is important to find the right model parameters during the fine-tuning process, because when optimal settings is not found, overfitting or underfitting behaviour can occur as a result after model training. The example how different results might be is shown in Table 5.3

5.7 Error analysis metrics

Error table represents more qualitative approach to quality evaluation of machine translated texts. To get more quantitative results, standard translation metrics

Tab. 5.3: Example on how different settings parameters can affect the final result after training the model.

Original incorrect sentence	Sonar uses sound propagationio to detectign objectos
Optimal fine-tuning mask fitting	Sonar uses sound propagation as a means to determine the range of objects
Underfitting mask	Sonar uses sound propagation to navigated , communicat , or detectign object
Overfitting mask	A camera uses sound propagationio to detecting objects

scores are calculated, too. The reason is to get more standardized evaluation result. The NLTK package provides functions for calculating translation metrics scores. Similarity, BLEU score, and METEOR are calculated for all machine translated texts. The average result of each score is represented in Table 5.4. According to the standardized metric scores, DeepL's translation is the most accurate one when applied to technical texts and when translating from a highly inflecting language. All error analysis metrics are defined in the 'metrics.py' file.

Tab. 5.4: The calculated average result metric score of each translating tools.

	Similarity score [%]	BLEU score [-]	METEOR score [-]
Bing	95	0.07795698924731183	0.83
DeepL	97	0.07474226804123711	0.94
Google Translate	95	0.07323232323232323	0.88
Systran	94	0.07774798927613942	0.88

5.8 Example of the error identification and classification algorithm on a sample data set

According to the metric score values, DeepL is the most precise when it comes to the technical sample texts. On the other hand, Systran is the least precise one when it comes to the technical sample texts. In this section of this chapter, the more detailed result of one set of analyzed sample texts is presented. In the Table , there are the sample texts used for this experiment. The table contains its original in Czech language, reference translated text and two machine translated texts.

Tab. 5.5: Example of all translations, including reference, in order to show the functionality of error identification and classification algorithm. [49]

Original in Czech	Arsenid gallitý (GaAs) je III-V polovodič s přímým zakázaným pásmem s krystalovou strukturou směsi zinku. Arsenid gallitý se používá při výrobě zařízení, jako jsou mikrovlnné frekvenční integrované obvody, monolitické mikrovlnné integrované obvody, infračervené diody vyzařující světlo, laserové diody, solární články a optická okna. GaAs se často používá jako substrátový materiál pro epitaxní růst jiných III-V polovodičů, včetně arsenidu india gallitého, arsenidu hlinitého gallitého a dalších.
Reference translation	Gallium arsenide (GaAs) is a III-V direct band gap semiconductor with a zinc blended crystal structure. Gallium arsenide is used in the manufacture of devices such as microwave frequency integrated circuits, monolithic microwave integrated circuits, infrared light-emitting diodes, laser diodes, solar cells and optical windows. GaAs is often used as a substrate material for the epitaxial growth of other III-V semiconductors, including indium gallium arsenide, aluminum gallium arsenide and others.
DeepL Machine Translation	Gallium arsenide (GaAs) is a III-V direct bandgap semiconductor with a zinc compound crystal structure. Gallium arsenide is used in the manufacture of devices such as microwave frequency integrated circuits, monolithic microwave circuits, infrared light emitting diodes, laser diodes, solar cells and optical windows. GaAs is often used as a substrate material for epitaxial growth of other III-V semiconductors, including indium gallium arsenide, aluminum gallium arsenide, and others.
Systran Machine Translation	Gallium arsenide (GaAs) is an III-V semiconductor with a direct bandwidth forbidden by the crystal structure of a zinc mixture. Gallium arsenide is used in the manufacture of devices such as microwave integrated frequency circuits, monolithic microwave circuits, light-emitting infrared diodes, laser diodes, solar cells and optical windows. GaAs is often used as a substrate material for epitaxic growth of other III-V semiconductors, including indium gallium arsenide, aluminum gallium arsenide, and others.

The first part of the error identification and classification algorithm is the incorrect article position identification. The incorrect article position identification might

indicate addition or ommision of words as these unnecessary translation actions affect the final position of the articles. After applying this part of text analysis, the reference's article positions are '[4 11 21 50 54]', DeepL's article positions are '[4 10 20 49]', and Systran's article positions are '[4 8 13 17 25 53]'. According to text analysis, the number of articles in each text is different, too. In case of DeepL, the ommision of an article happened. In case of Systran, the addition of several articles took place. Therefore, the results of the first part of article error type identification is that machine translations are not precise. The reason for this behaviour might be a fact that Czech language does not have articles in the language and English does. Therefore, both human and machine translators need to have a deep grammatical knowledge where articles need to be added in different sentences. It is difficult to mirror this grammar and implement it in neural machine translations. In this case, human translator is needed for editing these mistakes.

After the previous evaluation of article precision, articles are considered as stop-words in further text analysis. Therefore, the stop-words removal takes place and the algorithm works with texts without containing them.

The second error type identification and classification is vocabulary addition and ommision. In case of DeepL, there was omission of one word, which was *'bandgap'*. There comes the limitations of using a reference translation, because the word *'bandgap'* can be written as two words and is still correct. In case of Systran, there was many additions and overall mistranslations as well. On a whole-document-level, Systran added more words than removed, however, when the text analysis algorithm was used on the sentence-by-sentence basis, different sentences had ommisions and additions of words. When looked closely on the results, almost no sentences were translated correctly.

The third error type classification is the correct verb inflection form. In this case, DeepL translates verbs correctly. All verb tokens were identified in the same form as the reference. On the other hand Systran does not inflect verbs correctly. In the case of this example sample text, generally the wrong form was inflecting verb to 3rd person present tense. Other verb forms seemed correct. However, final result, when comparing all results of text data set, was different.

The last error type is mistranslation and mispelling. In this case, mispelling does not happen. As the first and second error types indicated, mistranslation is the most common mistake. In case of DeepL, its detected mistake can be ignored as this error was indetified based on the limitation of using reference texts. Systran translator changed quite a lot of sentences with a different vocabulary. The reason behind mistranslation is different learning data set for different neural machine translation systems. These data sets are usually downloaded from different web sites. The machine translation systems are usually trained on general data sets that do not

have to include all domains. This fact affected the enhancement of training data set of the error correction model (that was implemented as a part of this thesis) in order to include more technical terminology and increase accuracy.

To conclude the precision and error rate when applied to one sample set of translated texts, almost all error types are present and human translators are always needed, at least for post-editing process. However, this error identification and classification algorithm provides more detailed information about these mistakes and can increase efficiency when post-editing by human translators.

5.9 Comparison of existing translating tools

To identify translation error rate and accuracy, several translating tools are compared. The machine translation is a significant help to providing research papers in English by non-native English speakers. Therefore, it is useful to know how these popular tools behave when translating technical texts from highly inflecting languages. Ten texts were machine translated and their average result was calculated. Their metric score comparison is shown in Table 5.4.

According to the error type evaluation, article position is the most problematic aspect of machine translation. One of the reasons might be that in the source language, in this case in Czech, articles do not exist and translation algorithm has to generate them. This reason causes the differences between human-translated and machine translated texts. The article positions are completely different, when comparing machine translation tools between each other. Therefore, the manual addition of articles is still the best approach.

Another consistent error category is addition or deletion of different words. According to the evaluation of error typology, DeepL seems to be the most precise when evaluating the unnecessary translation actions. In this experiment, ommision was more prevalent than addition. The reason might be encountering unrecognizable foreign words, different punctuation or special characters.

When looking at the result of verb inflection error, the most difficult verb forms to translate are verbs in gerund form and verbs in the third person singular. Gerund form is when verb has the "-ing" suffix. The verb inflection is caused by different inflection rules between the target and the source language. To resolve this problem, machine translator has to have implemented both natural language understanding and natural language generation, too. Based on the research, combination of two different neural models of each type could be the solution.

And the last frequent error category is vocabulary error. Misspelling usually occurs when reference text also contains misspelled words. In the comparison between misspelling and mistranslation, mistranslation is more frequent. The reason might

be detection of misspelled words as foreign words, and then, replacing them with a synonym. Mistranslation depends on different training data sets when modelling these machine translation networks.

Conclusion

The aim of the bachelor thesis was to research text analysis methods and tools to apply into evaluating accuracy and error rate of machine translation. The first theoretical part explains the main text analysis methods and their application. During the research, the focus was on the most popular methods and ones that can evaluate natural language structures. It appears that text mining is the most applicable to the translation quality measurement and to the identification of error typology in translated texts.

The second part of the research describes machine translation. The different types of machine translation systems are described. Nowadays, the most used one is neural machine translation which shows the best result.

The last part of the theoretical research contains information about various tools applicable for both text analysis and machine translation. Advantages and disadvantages of different programming languages are researched. Python appeared to be the most suitable programming language as it offers practical applications of text analysis and machine translation in the most efficient way. Different pre-trained models were researched. Their benefits were considered, too.

The goal of the proposed solution was implemented with the emphasis on error detection and classification, and overall translation quality evaluation. The experiment consists of two parts: text analysis and error correction model. text analysis methods were applied to identify five most common mistakes made by machine translation tools. The grammar correction model is the solution to correct the mistakes made by automatic translation to make post-editing translation process more efficient.

At the end, four most popular translation tools were compared. In conclusion, machine translation is reliable in comparison to statistical approach. However, it is still not perfect and the most common mistakes still occur. According to both error analysis and evaluation metrics, DeepL is the most suitable translation tool for technical texts.

The thesis solution shows how text analysis and machine translation approaches can be beneficial for determining accuracy and error rate of machine translation tools. To further the research, neural models could be applied for error identification and classification, too. The neural approach is the most efficient for machine translation, therefore, its application to text analysis methods might be worth exploring further. The thesis evaluation is based only on few error categories and evaluating different error categories could bring even more deep insight into the accuracy and error rate of machine translation. Other more sophisticated approaches could be explored in further research, too.

Bibliography

[1] CIOS, J. Krzysztof, Witold PEDRYCZ, Roman W. SWINIARSKI and Lukasz A. KURGAN. *Data Mining: A Knowledge Discovery Approach*. USA: Springer Science+Business Media, 2007. ISBN 978-0-387-33333-5.

[2] MURUGAN, A., C. HILL, T. NOLAN. *Practical Text Analytics: Maximizing the Value of Text Data*. USA: Springer International Publishing, 2019. ISBN 978-3-319-95663-3.

[3] VIJAYARANI S., R. JANANI. Text Mining: Open Source Tokenization Tools - An Analysis. In *Advanced Computational Intelligence: An International Journal*. 2016.

[4] RAKHOLIA M. Rajnish, Jatinderkumar R. SAINI. *Lexical Classes Based Stop Words Categorization for Gujarati Language*. IEEE, 2016. ISBN 978-5090-3480-2.

[5] JIVANI Anjali Ganesh. A Comparative Study of Stemming Algorithms. In *Int. J. Comp. Tech. Appl.*. 2011, 2, p. 1930-1938. ISSN: 2229-6093.

[6] BALAKRISHNAN, Vimala, Ethel LLOYD-YEMOH. *Stemming and lemmatization: A comparison of retrieval performances* [online]. University of Malaya, Malaysia: 2014 [cit. 2023-05-10].

[7] MINAEE Shervin, Nal KALCHBRENNER, Erik CAMBRIA, Narjes NIKZAD, Meysam CHENAGHLU, Jianfeng GAO. Deep Learning-based Text Classification: A Comprehensive Review. In *ACM Comput. Surv.*. 2021, 54, 1-40. Available from: doi: https://doi.org/10.1145/3439726.

[8] PATRA, A. D. SINGH. A Survey Report on Text Classification with Different Term Weighing Methods and Comparison between Classification Algorithms. In *International Journal of Computer Applications (0975 – 8887)* [online]. 2013, 75, 14-18 [cit. 2023-05-09].

[9] AGGARWAL Charu, ChengXiang ZHAI. A Survey of Text Clustering Algorithms. In *Mining Text Data*. NY, USA: Springer New York, 2012. ISBN 978-1-4614-3222-7.

[10] LI, Guang; Fangfang, LIU; Ashutosh, SHARMA; Osamah, KHALAF; Youseef, ALOTAIBI; Abdulmajeed, ALSUFYANI; Saleh, ALGHAMDI. Research on the Natural Language Recognition Method Based on Cluster Analysis Using Neural Network. In: *Mathematical Problems in Engineering* [online]. 2021 [cit. 2023-05-18]. Available from: do:10.1155/2021/9982305.

[11] JING, Jiang. Information Extraction from Text. In: *Mining Text Data* [online]. Boston, MA: Springer, 2012, s. 11-41 [cit. 2022-11-30]. ISBN 978-1-4614-3223-4. Available from:doi: https://doi.org/10.1007/978-1-4614-3223-4.

[12] KOZÁK, Ondřej. *Metody dolování dat pro analýzu textů* [online]. Brno, 2022 [cit. 2023-05-20]. Available from URL: https://www.vutbr.cz/studenti/zav-prace/detail/141631. Bachelor thesis. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Thesis supervisor Přemysl Dohnal.

[13] ALLAHYARI M., S. POURIYEH, M. ASSEFI, S. SAFAEI, E.D. TRIPPE, J.B. GUTIERREZ, K. KOCHUT. *Text Summarization Techniques: A Brief Survey.* USA, 2017.

[14] NIVRE Joakim. Dependency Parsing. In *Language and Linguistics Compass 4/3.* Blackwell Publishing Ltd, 2010, p. 138-152. Available from: doi: https://doi.org/10.1111/j.1749-818X.2010.00187.x.

[15] ANTHONY P.J., K.P. SOMAN. Parts of Speech Tagging for Indian Languages: A Literature Survey. In *International Journal of Computer Applications.* 2011, 34, p. 22-29.

[16] TRUJILLO, Arturo. *Translation Engines: Techniques for Machine Translation.* London: Springer, 1999. ISBN 978-1-85233-057-6

[17] TINOTENDA, Chemvura. *LARMAS - Language Resource Management System* [online]. 2017 [cit. 2023-05-15]. Available from: doi:10.13140/RG.2.2.34784.38405.

[18] OKPOR, M. D. Machine Translation Approaches: Issues and Challenges. In *IJCSI International Journal of Computer Science Issues* [online]. 2014, 2014(11, 5), 159-165 [cit. 2023-03-23]. ISSN 1694-0784.

[19] LOPEZ, Adam. Statistical Machine Translation. In *ACM Computing Surveys* [online]. ACM, 2008, 2008(40, 3) [cit. 2023-03-23]. Available from: doi:http://doi.acm.org/10.1145/1380584.1380586.

[20] SINGLA, Karan. *Methods for Leveraging Lexical Information in SMT* [online]. 2015 [cit. 2023-05-19]. Available from: doi:10.13140/RG.2.1.2138.7367.

[21] ALFARRAJ, Motaz; Alregib, GHASSAN. *Petrophysical-property estimation from seismic data using recurrent neural networks* [online]. 2018 [cit. 2023-05-19]. Available from: doi:10.1190/segam2018-2995752.1.

[22] DABRE, Raj, Chenhui CHU, Anoop KUNCHUKUTTAN. A Survey of Multilingual Neural Machine Translation. In *ACM Computing Surveys* [online]. 2020, 2022(53, 5) [cit. 2023-03-27]. Available from: doi: https://doi.org/10.1145/3406095.

[23] WU, Yonghui; Mike, SCHUSTER; Zhifeng CHEN; et al. *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation* [online]. 2016 [cit. 2023-05-18]. Available from: doi: https://doi.org/10.48550/arXiv.1609.08144.

[24] MOORKENS, Joss, Sheila CASTILHO, Federico GASPARI, Stephen DOHERTY. *Translation Quality Assessment: From Principles to Practice* [online]. Cham, Switzerland: Springer, 2018 [cit. 2023-03-29]. ISBN 978-3-319-91241-7. Available from: doi:https://doi.org/10.1007/978-3-319-91241-7.

[25] HAN, Lifeng, Gareth J.F. JONES, Alan F. SMEATON. Translation Quality Assessment: A Brief Survey on Manual and Automatic Methods. In *Proceedings for the First Workshop on Modelling Translation: Translatology in the Digital Age* [online]. Association for Computational Linguistics, 2021, 15-33 [cit. 2023-03-29].

[26] BANERJEE, Satanjeev, Alon LAVIE. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization* [online]. Association for Computational Linguistics, 2005, 65-72 [cit. 2023-03-30].

[27] KOEHN, P. and Ch. MONZ. Manual and Automatic Evaluation of Machine Translation between European Languages. In *Proceedings of the Workshop on Statistical Machine Translation* [online]. New York City, USA: Association for Computational Linguistics, 2006, 102-121 [cit. 2022-12-30].

[28] STYMNE, Sara, Lars AHRENBERG. On the practice of error analysis for machine translation evaluation. In *LREC* [online]. 2012, 1758-1790 [cit. 2023-03-30].

[29] STYMNE, Sara; Stymne, Sara. Using a grammar checker and its error typology for annotation of statistical machine translation errors. In *Proceedings of the 24th Scandinavian conference of linguistics* [online]. 2013, 332-334 [cit. 2023-04-01].

[30] ZAHRADNÍK, Petr. *Translating Science and Technology: Expert and Popular Science Texts in English-to-Czech Translation using NMT* [online]. Brno, 2022

[cit. 2023-05-18]. Available from URL: https://is.muni.cz/th/o2375/. Master thesis. Masarykova univerzita, Filozofická fakulta. Thesis supervisor Přemysl Dohnal.

[31] PEDREGOSA, F. *Scikit-learn: Machine Learning in Python* [online]. JMLR 12, 2011, 2825-2830 [cit. 2023-05-09].

[32] REHUREK, Radim; Petr, SOJKA. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks* [online]. Valletta, Malta: ELRA, 2010, 45-50 [cit. 2023-05-09].

[33] LORIA, Steven. *TextBlob: Simplified Text Processing* [online]. 2020 [cit. 2023-05-09]. Available from URL: https://textblob.readthedocs.io/en/dev/.

[34] PARVAT, A.; J., CHAVAN; S., KADAM; S., DEV; V., PATHAK. A Survey of Deep-learning Frameworks. In *International Conference on Inventive Systems and Control (ICISC-2017)* [online]. IEEE, 2017 [cit. 2023-05-03]. ISBN 978-1-5090-4715-4.

[35] BANCHS, Rafael E. *Text Mining with MATLAB* [online]. Mountain View, CA, USA: Springer, 2021 [cit. 2023-03-03]. ISBN 978-3-030-87695-1.

[36] SILGE, Julia; David, ROBINSON. *Text mining with R: a tidy approach* [online]. Sebastopol: O'Reilly, 2017 [cit. 2023-01-07]. ISBN 978-1491981658.

[37] MIN, Bonan; H., ROSS; E., SULEM; A., VEYSEH; T.H., NGUYEN; O., SAINZ; E., AGIRRE; I., HEINZ; D., ROTH. *Recent Advances in Natural Language Processing via Large Pre-Trained Language Models: A Survey* [online]. arXiv, 2021 [cit. 2023-04-02]. Available from: doi: https://doi.org/10.48550/arXiv.2111.01243.

[38] YONGHUI, Wu; Mike, SCHUSTER; Zhifeng, CHEN; Quoc, V. LE; Mohamman, NOROUZI. *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation.* arXiv, 2016. Available from: doi:https://doi.org/10.48550/arXiv.1609.08144.

[39] AURPA, T.T.; R. SADIK; A., MD SHOAIB. *Abusive Bangla comments detection on Facebook using transformer-based deep learning models. Social Network Analysis and Mining* [online]. 2021 [cit. 2023-05-15]. Available from: doi:12.10.1007/s13278-021-00852-x.

[40] SINGH, S.; Ausif, MAHMOOD. The NLP Cookbook: Modern Recipes for Transformer Based Deep Learning Architectures In *IEEE Access* [online]. IEEE, 2021 [cit. 2023-04-27].

[41] HAN, Xu; Zhengyan, ZHANG; Ning, DING. Pre-trained models: Past, present and future. In *AI Open* [online]. 2021, 2, 225-250 [cit. 2023-05-15]. Available from URL: https://www.sciencedirect.com/science/article/pii/S2666651021000231.

[42] KADLEC, Lukáš. Technologie strojového překladu. Brno, 2022. Available from URL: https://www.vutbr.cz/studenti/zav-prace/detail/142542. Bachelor's thesis. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav jazyků. Thesis supervisor Kenneth Froehling.

[43] NAPOLES, Courtney; Keisuke, SAKAGUCHI; Tetreault, JOEL. JFLEG: A Fluency Corpus and Benchmark for Grammatical Error Correction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers* [online]. Valencia, Spain: Association for Computational Linguistics, 2017, 229-234 [cit. 2023-04-24]. Available from URL: http://www.aclweb.org/anthology/E17-2037.

[44] HARRIS, C.R.; K.J., MILLMAN; S.J., VAN DER WALT; et al. Array programming with NumPy. In *Nature 585*. 2020, 357-362. Available from: doi: 10.1038/s41586-020-2649-2.

[45] MCKINNEY, Wes. Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference* [online]. 2010, 56-61 [cit. 2023-05-05]. Available from: doi:10.25080/Majora-92bf1922-00a.

[46] COHEN, Adam. *FuzzyWuzzy* [online]. Available from URL: https://pypi.org/project/fuzzywuzzy/.

[47] FILLION, Eric; Ted, BROWNLOW. *Happy Transformer* [online]. Available from URL: https://happytransformer.com/.

[48] LHOEST, Quentin; Albert, VILLANIVA DEL MORAL; Yacine, JERNITE; et al. Datasets: A Community Library for Natural Language Processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* [online]. Association for Computational Linguistics, 2021, 175-184 [cit. 2023-04-24]. Available from URL: https://aclanthology.org/2021.emnlp-demo.21.

[49] MOSS, S. J.; LEDWITH, A. *The Chemistry of the Semiconductor Industry* [online]. Springer, 1987 [cit. 2023-03-05]. ISBN 978-0-216-92005-7.

A Electronic attachment

The electronic attachment contains the experimental part of the thesis. It contains the algorithm that solves different error types identification and classification with one sample set of translated texts by different translating tools. It also contains the error and grammar correction model with its enhanced training set.

- `/` ..... root of the attached archive
  - `textAnalysis` ..... text analysis methods, error identification and classification
    - `main.py`
    - `dataCleaning.py`
    - `analysisFunctions.py`
    - `metrics.py`
    - `text1.txt`
    - `text1Bing.txt`
    - `text1Deepl.txt`
    - `text1Google.txt`
    - `text1Systran.txt`
  - `languageModel` ..... error and grammar correction model
    - `languageModel.py`
    - `eval.csv`
    - `train.csv`