UNIVERSITY OF PIREAUS

Department of Digital Systems

Postgraduate Programme “Information Systems & Services”

The Impact of Distributed Neural Machine Translation in Sentiment Analysis

George Manias (AM: 1816)
Piraeus, February 2020

Abstract

In recent years there has been a steady increase in the interest of the research and business community in the field of Artificial Intelligence (AI). The ability of modern user devices to implement and apply highly sophisticated Machine Learning techniques has increased the importance of using and developing applications that will use Machine Learning techniques and algorithms to process and analyze user data.

Undoubtedly, progress in making computer systems more human-friendly requires the inclusion of Natural Language as an integral means of a wider communication interface. Nowadays, the overarching goal of natural language research is to enable communication between humans and computers without resorting to memorable and complex processes.

Modern chatbots, automatic translation engines, search engines and more are included in these applications. These applications implement algorithms and techniques in the field of Natural Language Processing (NLP), which is also a subtask of Machine Learning. It is one of the most evolving research and business areas of Machine Learning. While Natural Language Processing is not a new field of research, however, the explosion of technology and the need for more effective human-machine communication and understanding as well as the increased availability of big data, powerful computers and enhanced algorithms has led the last five years to the increasing of interest in the production of new algorithms and applications that will bring humans closer to the machines.

Natural Language Processing is widely used in the development of sophisticated user interfaces, as we can see in the multitude of modern mobile and web applications, bypassing the classic keyboard-mouse-screen model. In such an environment, the efficiency and speed of applications are significantly increased as it becomes more user-friendly, while new possibilities such as voice telephony interconnection are constantly being opened.

Natural Language Processing helps computers communicate with people, thereby increasing efficiency in tasks involving the use of written or oral speech. For example, NLP enables computers to read text, recognize speech, interpret, measure sentiment, and determine which parts of written or oral speech are most important to proceed with Q&A creating specific proposals.

Nowadays machines can analyze more human-based language data at no particular cost and in a coherent, non-discriminatory way. Given the staggering and ever-increasing amount of unwanted data generated daily from medical records to social media, automating their processing will be critical to effective data and speech analysis.

The continued interest of the scientific and business community in analyzing application users' sentiment has led to an increase in the search for satisfactory techniques and algorithms that will help users deliver services more effectively and efficiently. The research community has effectively proposed and developed methods for identifying and categorizing users' suggestions and feelings as expressed in a variety of different types of texts, such as blogs, reviews, tweets, and more.

But the needs and trends of modern intercultural and multilingual societies are increasingly demonstrating the need to create multilingual solutions and applications that will operate in a wider context and make full and effective use of the techniques of Machine Translation. Despite advances in the recognition of emotion in individual high-frequency and use languages, such as English, few applications and studies have been performed in less used and more difficult languages, Greek, and even less has been done on how to mechanically translate a text, from a smaller use language (Greek) to a more widely used language (English). Mistaken or with errors translation can result in falsification of the categorization of this text and the discovery of emotion through it. Many companies, giants e.g. TripAdvisor, Facebook, Google, NetFlix etc. focus their products and services in this direction. Recent years research teams of these companies try to implement and provide state-of-the-art machine translation systems. *But is the translation result always correct, for example a critic in a movie and the sentiment that is derived from that particular translation over the original text? Can - and to what extent - the discovery of emotionality be falsified if we proceed to the Machine Translation of a text?*

The purpose of this thesis is to investigate the impact of Neural Machine Translation on Sentiment Analysis by applying deep learning techniques such as Deep Neural Networks and multilingual sensing analysis tools. Recent advancements in the field of Neural Machine Translation enhance an interest of its use in Sentiment Analysis. A comparative analysis of different approaches and Neural Machine Translation open source services for multilingual sentiment analysis will be examined. These approaches are divided into two parts: one using classification of text without language translation and second using the translation of our analyzed data to a target language, such as Greek and German, before performing Text Classification and Sentiment Analysis.

Keywords: Sentiment Analysis, Machine Translation, Natural Language Processing, Multilingual Sentiment Analysis, Neural Machine Translation, Deep Learning

Acknowledgments

Firstly, I would like to thank Professor Kyriazis Dimosthenis for his valuable help and guidance provided throughout my postgraduate studies. I would also like to thank him for the opportunity and inspiration he has given me to deal with this subject and expand my knowledge area.

Of course, I could not omit my family and my beloved ones who are constantly beside me, in every step of my life, supporting my decisions. Their patience and their trust in me gave me courage, confidence and strength in difficult times.

Table of Contents

List of Figures

List of Tables

1 Introduction

Natural Languages are one of the most powerful tools and characteristics of the human gender and one of the most fascinating discoveries in the history of mankind. If it was not for natural languages, it would be no civilization, evolution and communication. Human communication, and therefore natural language, is subjective and ambiguous.

Everything we express, either verbally or written, carries huge amounts of information. In most human languages the sounds that represent the different characters are intertwined with each other, so converting the analogue signal to justified can be a difficult process. Also, in a physical speech it has been observed that a person does not use the same pause or ordering between words and the system should be able to distinct these words and moreover should get into consideration the grammatical and semantic restrictions in which the word is performed within the context. In theory, we can understand and even predict human behavior using that information, but if you want to scale and analyze several hundreds, thousands or millions of people or declarations in a given geography, then the situation is unmanageable.

The term “natural language” is used to distinguish human languages (Greek, English, Italian) from computer programming languages (C++, Java etc.). NLP is a scientific field that combines the science of linguistics and computer science and deals with the computational aspects of human language. It belongs to the cognitive sciences and mainly to the field of Artificial Intelligence (AI).

Understanding and generating natural language is a very difficult problem and is referred very often as a NP-Complete complexity problem. Understanding the natural language requires extensive knowledge of the outer world and the human capability to handle it, while generating the natural language requires good and deep knowledge of techniques and tools from the fields of Deep Learning, Machine Learning and Artificial Intelligence. Everything that is expressed in written or oral natural language should be transformed in a machine-readable way and backwards.

1.1 Motivation

The data that is generated daily in the modern social and technological environment comes from conversations, texts, articles, posts, tweets and constantly growing. A recent article [1] based on a IDC’s research illustrates the ever-increasing volume of data produced in their new Zettabytes unit of measurement, and the fact that by 2025 30% of automatic data is projected to be derived from Real-Time for data processing and analysis.

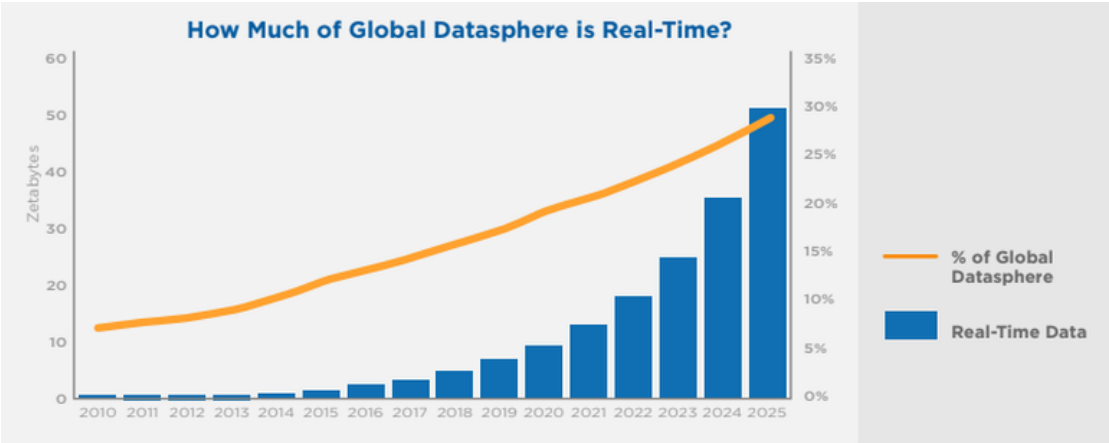

Figure 1: How much of global datasphere is real-time? 2010-2025

This chart shows us the need to find and implement new applications using advanced machine learning techniques and algorithms, so that they can process and analyze a huge amount of real-time data. Real-time data development will depend in part on consumer demand, according to the report: "As their digital world overlaps with their physical reality, consumers are expecting access to products and services wherever they are, and any device, they want instant data, on the go, and personalized. " Businesses collect an incredible amount of data every day. According to IBM, over 2.5 quadruples of bytes of data are generated per year, and it is noteworthy that in the last 2 years only 90% of existing data has been generated.

Also, examples of the above data are examples of unstructured data. Unstructured data does not fit the traditional relational database structure of rows and columns and represents the vast majority of real-world data available. They are characterized by a wide variety (Variety), a very growing volume (Volume) and a high velocity of production (Velocity) and are called Big Data.

However, thanks to advances in disciplines such as Machine Learning, there has been a great revolution in this regard. Today we are no longer going to interpret a text, or a speech based on the keywords (the old mechanical way), but to understand the meaning behind those words (the cognitive way). In this way it is possible to detect forms of speech such as irony or even to perform Sentiment Analysis or Opinion Mining.

The human language is amazingly complex and varied. We express ourselves in infinite ways, both verbally and in writing. Not only are there hundreds of languages and dialects, but within each language there is a unique set of rules for grammar, syntax, terms and aliases. When writing, we often misspell words or shorten words or omit punctuation, which can change the meaning of a sentence or even separate the sentences. Also, in oral speech, we often express the mood of our speech with the tone and style of our speech, or we can even fake or cut words or borrow words from other languages.

Data generated from conversations, declarations or even tweets are examples of unstructured data. **Unstructured data** does not fit neatly into the traditional row and column structure of relational databases and represent the vast majority of data available in the actual world. It is messy and hard to manipulate. Nevertheless, thanks to the advances in disciplines like Machine Learning a big revolution is going on regarding this topic. Nowadays it is no longer about trying to interpret a text or speech based on its keywords, but about understanding the meaning and collecting the information behind the sentences and documents. This way it is possible to detect figures of speech like irony, or even perform Sentiment Analysis and Opinion Mining.

In recent years, NLP has evolved thanks to huge improvements in data access and increased computing power, which enable professionals to achieve meaningful results in areas such as healthcare, the media, finance and human resources, including.

While supervised and non-supervised machine learning and especially deep learning techniques are widely used to model human language, there is also a need for editorial and semantic understanding and expertise in the field that is not necessarily present in these engineering approaches. learning. The modern domain of NLP is important because it helps to resolve ambiguity in the language and adds useful structure to the data for many applications, such as speech recognition or text analysis. Today, the NLP is evolving thanks to huge improvements in data access and increased computing power, which enable professionals to achieve significant results in areas such as healthcare, media, finance and human resources, among others.

A 2017 report by Tractica[1] on the interest of the modern market for natural language processing (NLP) estimates that total demand for applications, software and services related to the NLP sector will be approximately $ 22.3 billion by 2025. The report also predicts that NLP software solutions that leverage AI will see market growth from $ 136 million in 2016 to $ 5.4 billion by 2025.

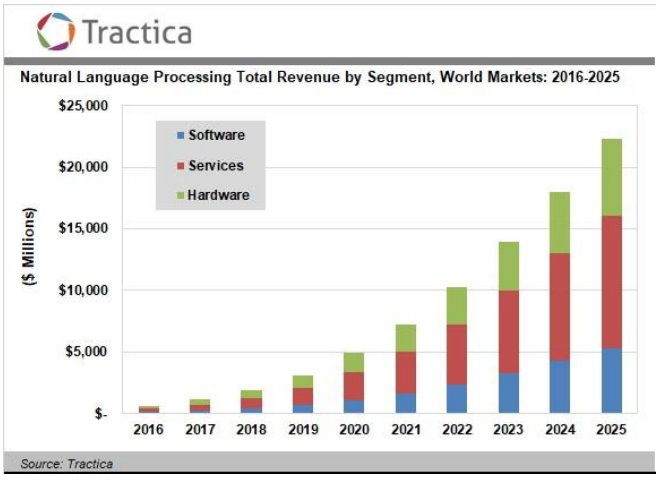

Figure 2: NLP total revenue by Segment, World Markets 2016-2025

[1] https://www.tractica.com/newsroom/press-releases/natural-language-processing-market-to-reach-22-3-billion-by-2025/

Although this great interest from the business and research community on subtasks of NLP like Sentiment Analysis most of the work is focused on data in the English language. There have been a lot of high-performance tools and applications developed for English according to any other language and that is the problem that we want to investigate on this thesis. We seek to explore if it is possible to use NMT and Sentiment Analysis techniques and tools in order to discern the sentiment of texts that were written in an internationally less used languages than English.

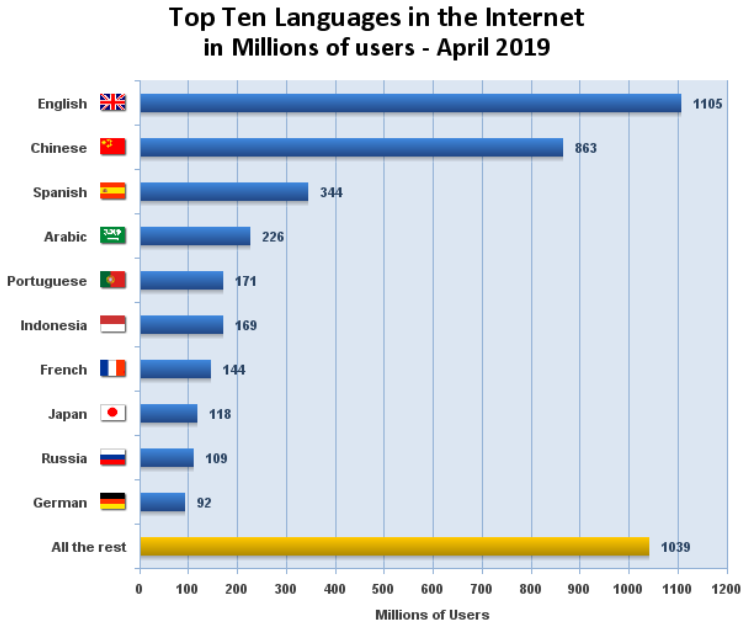

Figure 3: Top 10 Languages in the Internet in Millions of users – April 2019

As presented in the above image [2], English and Chinese together have almost the 50% of all languages that are being used in the Internet. But there is still a great amount (23.7%) of rest of the languages (e.g. Turkey, Greek, Latvia) that are still uncovered from MT and SA applications due to the low interest on these languages.

Moreover, more than 5000 languages exist in the real world which reflects the linguistic diversity. It is difficult for an individual to know and understand all the languages of the world. Hence, the methodology of translation was adopted to communicate the messages from one language to another. Developments in Information Communication and Technology (ICT) have brought revolution in the process of Machine Translation. Research efforts have been on to explore the possibility of automatic translation of one language (source text) to another language (target text).

1.2 Big Data – An Introduction

The advancement of technology in hardware and software allows us to collect, store and analyze large amounts of data from a variety of sources for all kind of businesses. Data science applies analytical techniques to data for finding underlying relationships to improve the organization's performance and value creation.

The exploitation of this data, as well as the knowledge it can derive from it, together with its human resources, is the most important resource for the company's growth, profitability and persistence in a highly competitive environment.

Successful data management and exploitation in conjunction with the so-called information systems, systems that allow data organization and analysis, are the main factor for increasing a company's productivity and are crucial for the efficiency of modern businesses, organizations and governments within a rapidly evolving global economy.

One of the main features of data in our time is high volume. It is now possible to gather information from a variety of sources: smartphones, and in general smart devices, social networks, and the health system are a small part of potential data sources. This led to the birth of one of the most widely used terms in our time: "Big Data". The first appearance of the term was made in 1997 by NASA scientists [3]. They reported that they were unable to visualize their data sets, as they were so large that they could not be stored on the main memory, on the local disk and on an external hard drive, so they said they are having a Big Data problem.

Since then, technology is constantly evolving and penetrating more and more into all segments of the population. This produces a large amount of information at a very fast pace and with a great variety of types. In fact, the growth rates are exponential. However, in the last two decades, the volume and speed with which data is generated has made this field become a topic of special interest. To be more specific according to the market intelligence company IDC, they estimated that the “Global Datasphere” reached 33 zettabytes of data in 2018 and they predict the world’s data will grow to 175 Zettabytes in 2025 [4].

Figure 1 - Annual Size of the Global Datasphere

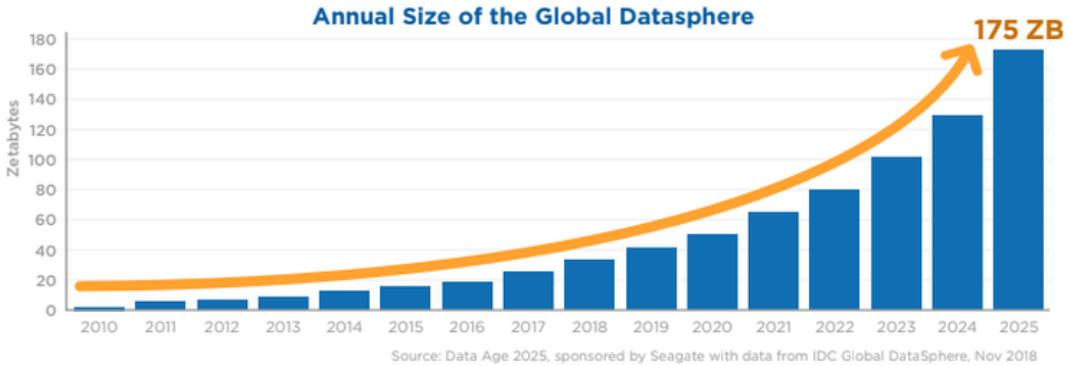

Figure 4: Annual Size of the Global Datasphere 2010-2025

This amount of data has a direct effect on every domain of our modern societies and that is the main The Big Data revolution has changed the way scientists approach problems in almost every (if not all) area of research. The Big Data field culls concepts from various fields of Computer Science.

Big data is a term for massive data sets having a wide variety (Variety), a very increasing volume (Volume) and a high speed of production (Velocity). These three concepts are also known as 3V's. However, their study above has now established a 6-dimensional model as described below:

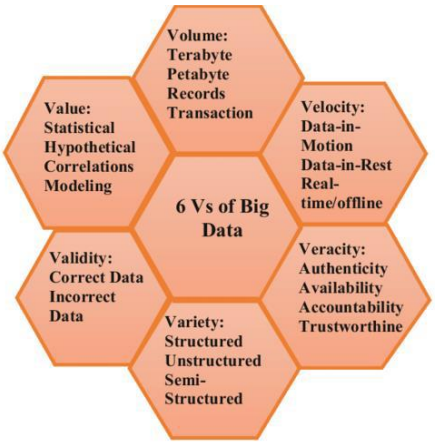

Figure 5: 6Vs of Big Data

Volume

The size of the data determines whether they will be considered "large" and it is understood that here we are not talking about a sample but about the whole data set. Imagine here companies like Facebook what volume of data they are required to store and process. Only in Greece, Facebook now has 5 million users ranging from photos and videos to updates and demographics.

Velocity

The speed at which they are produced, stored and processed to meet the challenges of an ever-evolving environment. Here, in fact, the ability to analyze them in real time opens up, for example, a new landscape in advertising: a customer who saw a product on a company's website but did not buy it and left the site would be able to target it directly from the company itself, seeing in advertising the product it was thinking of just before buying.

Variety

We exchange daily texts to audio messages and live video. We want to post a thought and a photo on social media to share a moment with our friends. Facebook now lets you see on a world map who's in live video broadcasting right now, with the ability to watch what you're saying.

Veracity

Their data and meaning are constantly changing. This means that a word that is fundamentally positive can be used in a way that denotes a negative expression. For example, the phrase "You're perfect today!" Indicates a positive attitude, while the phrase "Perfect ... I missed a two-day try for something you never used" indicates

something negative. Therefore, "perfect" is not enough to understand whether the view expressed is positive, but to read the whole phrase in order to understand the meaning. With the "explosion" of social media, "emotion analysis" is a new challenge for companies and for politicians alike, trying to figure out in millions of posts whether the opinion expressed is interpreted as negative, positive or neutral. from automatic detection. Despite the difficulty, some companies are already trying to implement user feedback mechanisms that are posted daily on social media. (Schroeck, et al., 2012)

Validity

Big data makes no sense unless the data is inaccurate. The quality of the data will therefore affect the analysis. Imagine having user behavior and wanting to automatically make a decision to buy it. If so, the data you had was inaccurate, and then the analysis that would be done would not be qualitative and accurate.

Value

Data is of no value unless we can convert it into value. Some of us have already seen their value, as we have used real-time traffic enforcement. In Formula 1, there can be thousands of data sensors in every vehicle: from tire pressure to fuel efficiency. This data is passed on to the appropriate people who analyze it and judge what adjustments will be needed to the vehicle in order for the driver to be more likely to win the race. In sports through some sensors, big data can be used to improve players' training and even to understand their opponents. The winner can even be predicted in a game or even predict the future performance.

Innovative and efficient forms of big data processing are therefore needed since the volume is enormous and it is important for us to be able to identify useful information. Therefore, adopting an integrated Big Data strategy will give a significant advantage to a business or organization since it can manage data in a very efficient way and make important decisions on data driven strategies based on conclusions drawn on the volume of data that will lead to even greater optimization of the various practices and decisions.

1.3 Natural Language Processing

1.3.1 What is NLP?

Natural Language Processing or NLP is a field of Artificial Intelligence that gives the machines the ability to read, understand and derive meaning from human natural languages. That means that NLP allows a machine to process a natural human language and translates it to a machine-readable way. Through NLP tasks machines are capable to interpret human language in order to process, analyze, and extract meaning from large volumes of natural language, spoken or written, text data.

NLP helps machines learn to communicate with each other and with humans in a meaningful way by enabling humans to use common everyday language to complete a broad range of tasks from the simple such as auto completing an online form to the most complex, such as recognizing voice or optimizing a data network. NLP is a critical technology for extracting insights and analysis from a vast amount of previously unindexed and unstructured data, such as mining video and audio files, emails, scanned documents, and more. The main drawbacks we face these days with NLP relate to the fact that language is very tricky.

The field of NLP involves making computers to perform useful tasks and extract useful information by parsing unstructured textual content from the natural language. The term of NLP involves two different procedures: **Natural Language Understanding (NLU)** and **Natural Language Generation (NLG).** Mainly with the term Natural Language Processing we mean understanding the natural language, because it is the most difficult part of the overall process.

Natural language Understanding (NLU) refers to the possibility of extracting some meaning from an oral or written sentence-phrase, so that the computer can at least convert the sentence into internal knowledge representation structures for future use. Tasks like mapping the given inputs in useful representations and analyzing the different aspects of the language are involved in this component. Natural Language Understanding (NLU) encompasses one of the narrower but especially complex challenges of AI: how to best handle unstructured inputs that are governed by poorly defined and flexible rules and convert them into a structured form that a machine can understand and act upon.

Figure 6: Understanding human language through NLU

Natural Language Generation (NLG) refers to the conversion of knowledge-based structures of knowledge into sentences of natural language. In other words, is the process of producing meaningful phrases and sentences in form of natural language

from some internal representing according to given input and it involves the above main tasks:

- Text planning: retrieving the relevant content from database. Here database can be including vocabulary, sentences, knowledge, sample data and many more.
- Sentence planning: we get our content using text planning now next step to do is choosing required words and forming meaningful sentence setting the words in right grammatical way.
- Text realization: we have all the thing need to create actual text in human's language.

In general terms, NLG (Natural Language Generation) and NLU (Natural Language Understanding) are subsections of the more general NLP domain that encompasses all software which interprets or produces human language, in either spoken or written form. The way that these three sections integrate with each other is as above:

- **NLU** takes up the understanding of the data based on grammar, the context in which it was said and decide on meaning, intent and entities.
- **NLP** converts a text into structured data. Transforms unstructured to structured data.
- **NLG** generates text based on structured data.

1.3.2 Natural Language Generation (NLG)

Natural Language Generation refers to the ability of a system to respond to a user by natural or written language. The natural language generation process can be broken down into two stages, the choice of *what to say* and the choice of *how to say.*

The stage of choosing *what to say* has to do with what information the system chooses to report to the user. The issue is with natural language communication because the user must listen or read everything the system tells him, possibly losing valuable time.

Therefore, the system has to choose what it deems necessary by leaving the rest of the information on the case that the user will request. The problem of choosing the information the system will present to the user is referred to as text planning and in the most advanced cases it is borrowed from action planning techniques.

The next step has to do with how the information will be told to the user. Usually, information is grouped into small logical sections from which sentences are then created using the rules of the grammar of the language.

One thing about voice communication is when the proposals must be spoken. There are two approaches to this end. The first is to have all words stored in audio format, with all possible variants in terms of number, personnel, etc. whenever the sentences are drafted. The second and most interesting approach is to synthesize phonemes from the word letters. In both cases there are fewer or more problems such as that

the same word can be pronounced differently depending on the words preceding or following or depending on its position in the sentence. The past few years have created many programs that express written text, such as the TTS system of Bell's labs.

1.3.3 Steps in NLP

When we first talk about natural language processing we are referring primarily to the ability to understand it. Understanding relates to the conversion of logic in unambiguous internal representational structures of knowledge.

Problems that have to do with the nature of the human language are addressed through a variety of applications. One major problem observed is the multifaceted meaning of the words. Many words have more than one meaning. The system must choose the concept that most logically fits within the context of the text.

Structural ambiguity is also a feature of natural languages. The grammar for natural languages is ambiguous; there are often more than one powerful parse trees for a specific sentence. Choosing the most appropriate editorial tree usually requires semantic and other contextual general information.

These problems of the nature language can be overcome by applying the below stages, when trying to develop NLP application or model. In its most complete form this procedure consists of five stages.

- Morphological and Lexical Analysis
- Syntactic Analysis (Parsing)
- Semantic Analysis
- Discourse Integration
- Pragmatic Analysis

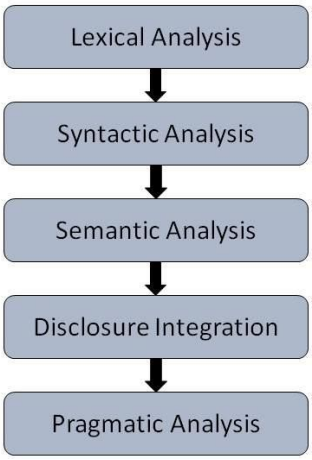

Figure 7: Stages of an NLP application procedure

These stages do not have a strict sequence of execution, but it is possible for a stage to receive feedback from subsequent stages, thereby recalculating its outputs. These steps are described in more detail below.

Morphological and Lexical Analysis

One of the critical parameters of the overall comprehension process is the lexicon, which is the collection of words and phrases in a language. Due to the large number of words it is not possible to hold all its forms for each of them. It involves identifying and analyzing the structure of words. Lexical analysis is dividing the whole chunk of txt into paragraphs, sentences, and words. What is done is to store in the lexicon only the basic forms of the word and then to derive the rest from it, applying rules of morphological analysis. Such rules are:

- to form different faces, numbers, word numbers (inflectional morphology)
- to form new words by adding known prefixes (un-) and suffixes to existing words (derivational morphology)
- to combine words to form compound words (compounding)

One last problem is the lack of recognition of words because they do not exist in the lexicon or because the words are entered incorrectly. The case of not having a word in the lexicon cannot generally be addressed unless the system manages to create it with morphological rules. The second case relates to specific cases of words, such as dates, hours, numbers, which are handled by special techniques. The third case is dealt by using spelling correction algorithms.

Syntactic Analysis (Parsing)

Aims to group words produced from the previous stage into sentences which are correct based on the grammatical and syntactic rules of the language. At this stage, the words that were produced in the previous stage are considered correct. It involves analysis of words in the sentence for grammar and arranging words in a manner that shows the relationship among the words. The sentence such as “The school goes to boy” is rejected by English syntactic analyzer. The syntactic analysis of a group of words requires the existence of a grammatical, that is, a set of rules that make up sentences of individual words. This step is also calling Parsing because it tries to determine the syntactic structure of a text by analyzing its constituent words based on an underlying grammar. For example, considering the below Grammar Rules, where each line indicates a rule of the grammar to be applied to an example sentence “Tom ate an apple”, we can have the parse tree of Image .

```
sentence -> noun_phrase, verb_phrase
noun_phrase -> proper_noun
noun_phrase -> determiner, noun
verb_phrase -> verb, noun_phrase
proper_noun -> [Tom]
noun -> [apple]
verb -> [ate]
determiner -> [an]
```

Figure 8: Example of Grammar Rules

Then, the outcome of the parsing process would be a parse tree like the following, where sentence is the root, intermediate nodes such as noun_phrase, verb_phrase etc. have children - hence they are called non-terminals and finally, the leaves of the tree 'Tom', 'ate', 'an', 'apple' are called terminals.

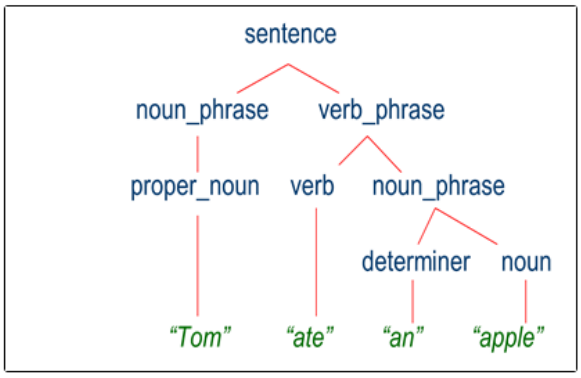

Figure 9: Example of a Parse Tree

Of course in a real system there will be many more rules, so searching for a sentence structure becomes a more complicated problem. Both semantic and pragmatic analysis can help with this problem.

Semantic Analysis

At the stage of Semantic Analysis, a first attempt is made to convert sentences into internal structures of knowledge representation, using the semantic meaning of words. For example, the proposal "John is a student" can be represented as *is(John, student)*. In general, the process of semantic representation requires more sophisticated Grammatical Definitions that are capable of distinguishing generations, numbers, persons etc. This task involves mapping syntactic structures and objects in the task domain.

The most important problem of semantic analysis is the multifaceted problem of ambiguity of words and sentences. The problem stems from the fact that a word can have many different meanings, that many words are implied or referred to with pronouns, part of the knowledge of a text being considered known and not mentioned. Doubt is usually introduced in the editorial analysis and eliminated in subsequent stages. Finding the right meaning for a sentence involves uncertainty, using data from the results of syntactic, semantic, and pragmatic analysis.

Discourse Integration

Discourse integration is closely related to pragmatics and is considered as a different step in the whole process of NLP only in the past few years. The meaning of any sequence of text depends on prior discourse. This analysis deals with how the immediately preceding sentence can affect the meaning and interpretation of the next sentence.

Pragmatic Analysis

The pragmatic analysis attempts to incorporate the sentence into the general context of the descriptions of the pads, considering the circumstances in which they were addressed. The sentence may contain unnecessary indications referring to the names of other sentences. Also important is the system's knowledge of how the world works, the chances of events, habits, scenarios, etc. It involves deriving those aspects of language which require real world knowledge. Thus, based on what is said the system can make many reasonable conclusions (default reasoning) by expanding its knowledge of the current status of the context.

1.3.4 NLP pipeline

NLP is a complex field and is the intersection of Artificial Intelligence, computational linguistics, and computer science.

Through, a typical NLP pipeline we try to overcome all the problems that could be derived from the natural languages. Primary goal is to create structured data representing parsed text from raw text. All the steps that are shown in the below Image and being analyzed below could apply in every classical NLP technique and task.

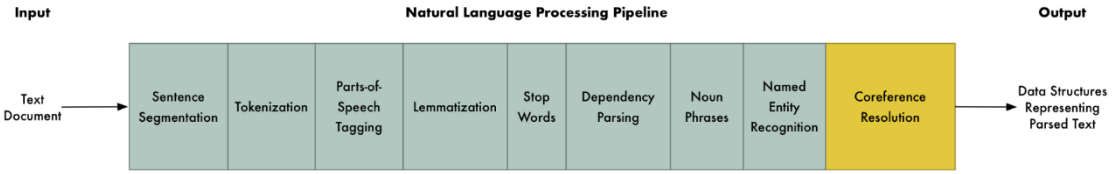

Figure 10: NLP pipeline

Step 1: Sentence Segmentation

It identifies sentence boundaries in the given text, i.e., where one sentence ends and where another sentence begins. Sentences are often marked ended with punctuation mark '.'

Step 2: Word Tokenization

It identifies different words, numbers, and other punctuation symbols. The output can be provided as input for further text cleaning steps such as punctuation removal, numeric character removal or stemming.

Step 3: Part-of-Speech Tagging

Part-of-speech tagging (POS tagging) is the task of tagging a word in a text with its part of speech. A part of speech is a category of words with similar grammatical properties. Common English parts of speech are noun, verb, adjective, adverb, pronoun, preposition, conjunction, etc.

Step 4: Text Lemmatization & Stop Words

Lemmatization of a word is the process by which the relative meaning found in dictionaries (the term of a dictionary or encyclopedia is called a lemma, hence and the name of the procedure). Consequently, every word of the document is reduced to a regular form, that is a single word, meaningfully identical and in the simplest possible form (such as a nominal decrease in the singular number (if noun) in the male gender (if adjective) or a singular person tense (for verbs), ...). This normal form is its meaning root, that is, in the entry with which it is recorded in a dictionary or encyclopedia. Performing Lemmatization helps to keep our lexicon from growing too much. Removal of stopwords also helps to achieve this goal. Stopwords are words that occur quite frequently in the written and spoken natural language and help to connect sentences without passing on any additional information, such as “and”, “or”, “to”, “the” etc. Therefore, these words do not provide any information gain and mislead the machine on deciding whether a sentiment is positive or negative. The list of words that we wish to exclude from a document may be created, either with an explicit statement or using a ready-made list from libraries like NLTK [5].

Step 5: Parsing & Noun Phrases.

The Natural Language Processing (NLP) community has made big progress in syntactic parsing over the last few years. By using syntactic parsers, a sentence’s grammatical structure can be described in such a way to help another application reason about it. Recent results have shown that spaCy offers the fastest syntactic parser in the world and that its accuracy was within 1% of the best available parser [6].

SYSTEM	YEAR	LANGUAGE	ACCURACY	SPEED (WPS)
spaCy v2.x	2017	Python / Cython	**92.6**	*n/a* ⓘ
spaCy v1.x	2015	Python / Cython	91.8	13,963
ClearNLP	2015	Java	91.7	10,271
CoreNLP	2015	Java	89.6	8,602
MATE	2015	Java	92.5	550
Turbo	2015	C++	92.4	349

Figure 11: Compare of parser’s accuracy

Step 6: Named Entity Recognition (NER)

Named entity recognition (NER) is the task of identify entities within the documents with their corresponding type, such as persons, location and time. Approaches typically use BIO notation, which differentiates the beginning (B) and the inside (I) of entities. O is used for non-entity tokens.

John	Doe	visited	Italy
B-PER	I-PER	O	B-LOC

(Table 1: Example of a NER task)

Step 7: Coreference Resolution

Coreference resolution is the task of clustering mentions in text that refer to the same underlying real-world entities. Is about defining the relationship of given the word in a sentence with a previous and the next sentence.

The following table gives a detailed overview of what can be the expected output from each of the previous steps.

Technique	Example	Output
Sentence Segmentation	Mark met the president. He said:"Hi! What's up -Alex?"	Sentence 1 – Mark met the president. Sentence 2 – He said:"Hi! What's up – Alex?"
Tokenization	My phone tries to 'charging' from 'discharging' state.	[My] [phone] [tries] [to] ['] [charging] ['][from] ['][discharging] ['] [state][.]
Stemming/Lemmatization	Drinking, Drank, Drunk	Drink
Part-of-Speech tagging	If you build it he will come.	IN – prepositions and subordinating conjunctions. PRP – Personal Pronoun VBP – Verb Noun 3rd person singular present form. PRP- Personal pronoun MD – Modal Verbs VB – Verb base form
Parsing	Mark and Joe went into a bar.	(S(NP(NP Mark) and (NP(Joe)) (VP(went (PP into (NP a bar))))
Named Entity Recognition (NER)	Let's meet Alice at 6 am in India.	Let's meet Alice at 6 am in India Person Time Location

Coreference resolution	Mark went into the mall. He thought it was a shopping mall.	Mark went into the mall. He thought it was a shopping mall.

(Table 2: Output example of NLP pipeline steps)

1.3.5 Key Tasks of NLP

With recent technological advances, computers now can read, understand, and use human language. They can even measure the sentiment behind certain text and speech. These capabilities (Image 6) allow businesses, organizations and even government agencies to recognize patterns, categorize topics, and analyze public opinion.

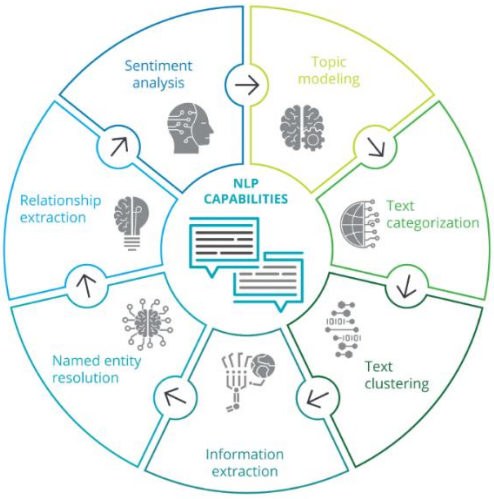

Figure 12: Key Tasks of NLP

In our everyday lives we make use of several applications and systems and that use NLP techniques and tools, while constantly new ones emerge. Below seven advanced and wide-used tasks are being introduced that are at the cutting edge of research and business interest.

Topic modeling

Topic modeling is a method based on statistical algorithms to help uncover hidden topics from large collections of documents. Topic models are unsupervised methods of NLP; they do not depend on predefined labels or ontologies. A popular method within topic modeling is Latent Dirichlet Allocation (LDA), which is used to discover latent patterns in a sea of unstructured data. The US Securities and Exchange Commission (SEC), for example, made its initial foray into natural language processing in the aftermath of the 2008 financial crisis. The SEC used LDA to identify potential problems in the disclosure reports of companies charged with financial misconduct.14

The UK government uses the same technique to better understand public comments on GOV.UK. With LDA, the government can see how customer complaints and comments relate to one another; for example, that mortgage complaints often contain allegations of racial discrimination. Uncovering such topics allows the government to address them.15

Text categorization

This method relates to the classification of texts by their content. The categories can be “political”, “athletic” and many others. Apparently, this application is useful in online newspapers and news agencies, where the user can create their own profile and depending on this system to display the news of interest first. In recent years automated data categorization systems have made remarkable progress, achieving success rates of over 90%. [7]

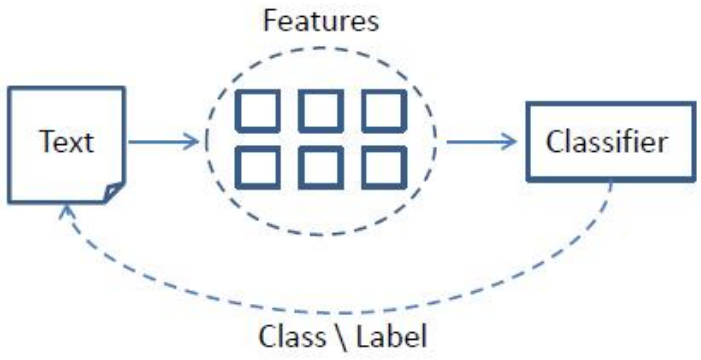

Figure 13: Text classification

In a research study, researchers underscore the benefits of automating security classification by classifying US security documents using NLP. The researchers also propose using text categorization to classify over a 100,000 declassified government records available on the Digital National Security Archive. [8]

Information Retrieval

Information Retrieval is the ability to find electronic records, context and documents related to a question. The simple case is that documents are characterized by keywords or an explanatory title, while the question being a word conjunction. This approach is used today by all the search engines on the internet, without delivering acceptable results. Ideally, the search engine would be able to make meaningful processing of the records and the question would be formulated in natural language. In this case it would be desirable for the search engine not to return texts that simply contain the keywords, but the meaning of which matches the query independently of the vocabulary used or the text.

Information Extraction

Information Extraction is used to automatically find meaningful information in unstructured text. One potential application can improve transparency and accuracy

in crime reporting. Often, police reports are written in haste, while crime witnesses' and victims' accounts can be incomplete due to embarrassment or fear of repercussions. Researchers at Claremont Graduate University found that NLP technology could comb through witness and police reports and related news articles to identify crucial elements such as weapons, vehicles, time, people, clothes, and locations with high precision.

Named Entity Resolution (NER)

This method can extract the names of persons, places, companies, and more, and classify them into predefined labels and link the named entities to a specific ontology. For instance, a text may contain references to the entity "Paris," which is both a famous person and a capital city. With the help of entity resolution, "Paris" can be resolved to the correct category, the person or the city.

Government agencies can extract named entities in social media to identify threat perpetrators of terrorist attacks, for instance, as well as their prospects. The more ontologies are defined in the NLP tool, the more effective the outcome.

Relationship extraction

Relationships are the grammatical and semantic connections between two entities in a piece of text. Relation Extraction (RE) is the task that establish semantic relations between entities. For instance, if a document mentions the Office of Management and Budget and the US federal government, relationship extraction identifies and creates a parent–agency relationship between them. These relations can be of different types. E.g "Paris is in France" states a "is in" relationship from Paris to France. This can be denoted using triples, ***is in(Paris, France)***.

Sentiment Analysis

Sentiment analysis decodes the meaning behind human language, allowing agencies to analyze and interpret citizen and business comments on social media platforms, websites, and other venues for public comment. Washington, DC's sentiment analysis program (GradeDC.gov), for example, examines citizens' feedback by analyzing their comments on social media platforms. The district was the first municipal government in the United States to adopt such an initiative.

Another study used sentiment analysis to examine the experiences of patients with various health care providers throughout the United States. The authors used an exhaustive dataset of more than 27 million tweets related to patient experience over a period of four years. A principal objective of the study was to examine the variation in such experiences across the country. The findings suggested a higher proportion of positive experiences for metropolitan areas compared to the nonmetropolitan areas.

More details on the concept, capabilities and application of the Sentiment Analysis field will be given in next chapter.

1.3.6 NLP capabilities across different domains

NLP capabilities have the potential to be used across a wide range of domains. Based on a recent study [9] of industry vertical, the NLP market is segmented into healthcare, retail, high tech & telecom, BFSI, automotive & transportation, manufacturing, advertising & media, and other(government)

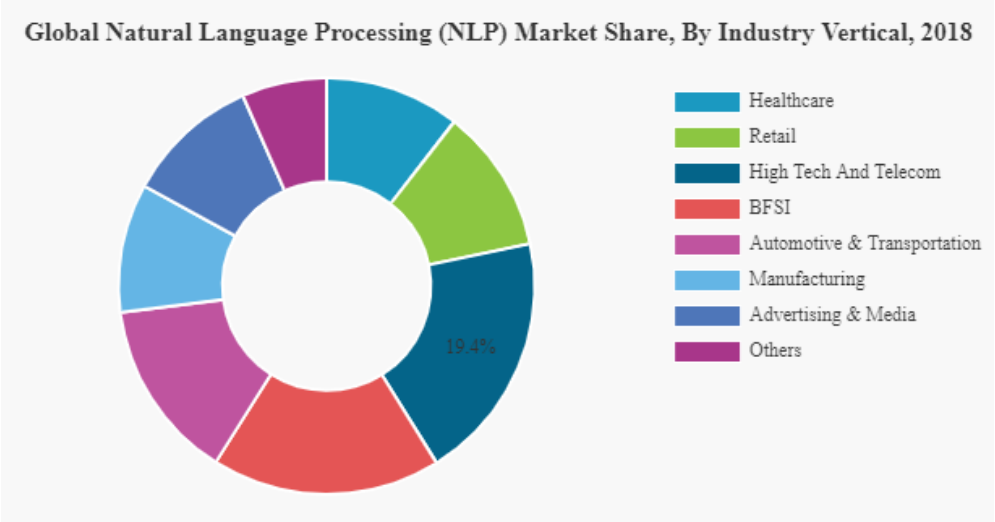

Figure 14: Global NLP market share by industry vertical, 2018

Healthcare

Recent studies in healthcare domain consider the study and application of Natural Language Processing (NLP) in healthcare as one of the most challenging issues in the field of medical information retrieval [10]. Since the medical texts and patients' records are representative of distinct language uses, it is necessary to design suitable NLP tools for their processing.

The US National Library of Medicine's Lister Hill National Center for Biomedical Communications uses NLP to "de-identify" clinical information in narrative medical reports, protecting patient privacy while preserving clinical knowledge.

Topic modeling has been used to deconstruct large biomedical datasets for medical surveillance. The National Center for Toxicological Research, for instance, used topic modeling on 10 years of reports extracted from the FDA's Adverse Event Reporting System (FAERS) in order to identify relevant drug groups from more than 60,000 drug adverse event pairs, i.e., pairs of drugs and adverse events in which the adverse reaction is caused by the drug.

Retail

A virtual assistant placed at the shop can identify and know the requirement of the customers and provide them quick information and promotional offers. Advanced NLP-powered chatbots serve the purpose of customer engagement, customer support, and in-house operations optimization.

Advertising and Media

NLP helps advertisers in identifying the potential buyers of their products by analyzing social media, emails, and purchase behaviors. Applications using NLP have already added value to many businesses, as they broaden the range of channels for ad placement. This enables the companies to optimally spend their ad budgets and target potential buyers from their social media platforms.

Automotive and Transportation

A recent study showed that Natural Language Processing (NLP) applications enable not only richer experience for passengers, but also a trust relationship between the passenger and the autonomous vehicle [11]. Moreover, in-car assistants, driven by natural language processing (NLP) and machine learning techniques, allow the vehicle's systems to respond to voice commands and infer what actions to take, without human intervention.

Governments

Government agencies around the world are accelerating efforts to abandon paper and modernize how they handle data. According to the National Archives and Records Administration, the US federal government has already digitized more than 235 million pages of government records and plans to reach 500 million pages by fiscal 2024 [12].

The use and creation of more unstructured data that are transferred into government services in both analog and digital formats present significant challenges for agency operations, rulemaking, policy analytics and customer service. NLP can provide the tools needed to identify patterns and glean insights from all this data, allowing government agencies to improve operations and policy making, identify potential risks, solve crimes, and improve public services.

1.3.7 Use Cases of NLP

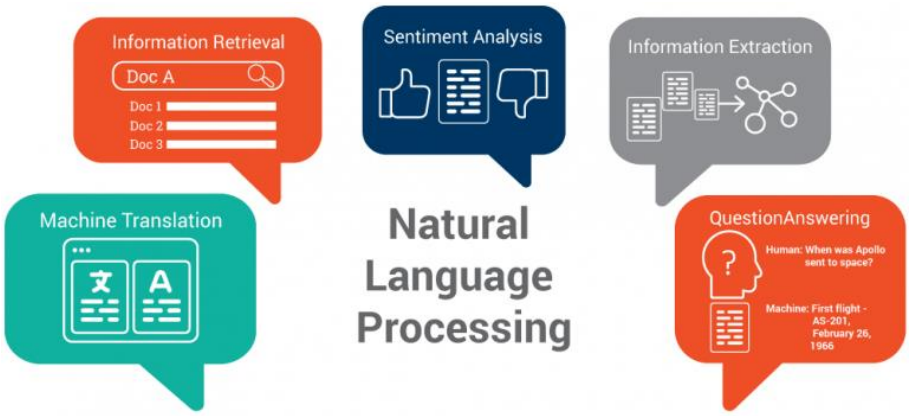

Figure 15: NLP areas of applications

NLP represents the automatic handling of natural human language like speech or text, and although the concept itself is fascinating, the real value behind this technology comes from the use cases.

NLP can help our world with lots of tasks and the fields of application just seem to increase daily. Some examples are being mentioned below:

NLP enables the recognition and prediction of diseases based on electronic health records and patient's own speech. This capability is being explored in health conditions that go from cardiovascular diseases to depression and even schizophrenia. For example, Amazon Comprehend Medical is a service that uses NLP to extract disease conditions, medications and treatment outcomes from patient notes, clinical trial reports and other electronic health records.

Moreover, using NLP and machine learning in healthcare to identify patients for a clinical trial is an exciting and moreover an essential use case. A few companies are now trying to resolve the challenges in this area using NLP engines for trial matching. With current advancements, NLP has the potential to automate trial matching and make it a seamless process.

An inventor at IBM developed a cognitive assistant that works like a personalized search engine by learning all about you and then remind you of a name, a song, or anything you can't remember the moment you need it to.

Companies like Yahoo and Google analyze, filter, and sort your NLP emails by analyzing text in emails flowing through their servers, even separating and blocking spam before they even enter your inbox. Google also recently introduced automatic correction in its systems making texts and emails more correct.

To help identifying fake news, the NLP Group at MIT developed a new system to determine if a source is accurate or politically biased, detecting if a news source can be trusted or not.

Amazon's Alexa and Apple's Siri are examples of intelligent voice driven interfaces that use NLP to respond to vocal prompts and do everything like find a particular shop, tell us the weather forecast, suggest the best route to the office or turn on the lights at home.

Having an insight into what is happening and what people are talking about can be very valuable to financial traders. NLP is being used to track news, reports, comments about possible mergers between companies, everything can be then incorporated into a trading algorithm to generate massive profits.

Researchers from Stanford and Cornell University with the use of NLP techniques identify antisocial behavior in three large online discussion communities by analyzing users who were banned from these communities, a task of high practical importance to community maintainers [13].

NLP is also being used in both the search and selection phases of talent recruitment, identifying the skills of potential hires and spotting prospects before they become active on the job market.

Powered by IBM Watson NLP technology, LegalMation developed a platform to automate routine litigation tasks and help legal teams save time, drive down costs and shift strategic focus.

NLP is particularly booming in the healthcare industry. This technology is improving care delivery, disease diagnosis and bringing costs down while healthcare organizations are going through a growing adoption of electronic health records. The fact that clinical documentation can be improved means that patients can be better understood and benefited through better healthcare. The goal should be to optimize their experience, and several organizations are already working on this.

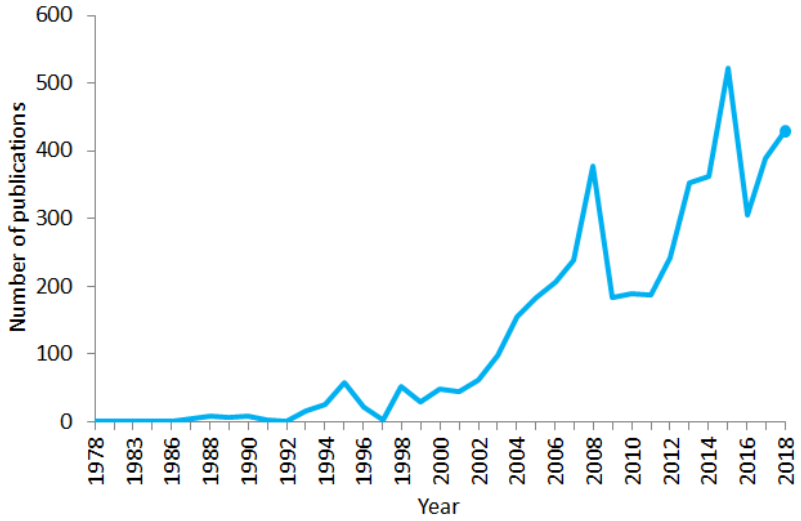

Figure 16: Number of NLP publications 1978-2018

Number of publications containing the sentence "natural language processing" in PubMed in the period 1978–2018. As of 2018, PubMed comprised more than 29 million citations for biomedical literature. [14]

1.4 Sentiment Analysis – An Introduction

The term Sentiment Analysis refers to the automatic identification and extraction of opinions, feelings, and dispositions from text documents. The main purpose of sentiment analysis is to characterize the polarity of a specific text and if the opinion expressed in it is interpreted as positive, negative, or neutral.

Social networks, blogs, and reviews sites have become popular platforms for people to express their opinions. Social network and micro-blogging platforms like Facebook and Twitter have millions of users who generates millions or billions of lines of textual information per day. This data contains opinions, sentiments, attitudes and emotions toward entities and aspects such as products, organizations, individuals, places, social

events, global problems etc. Many companies extract opinions for different purposes, e.g to know about product demand, influencing factors on the product, people choice etc. This process of extracting and classifying opinion on the different subject is known as **Sentiment Analysis or Opinion Mining** and is one of the most rapidly expanding areas of great interest from the research community.

The analysis of emotion in large data volume of internet users is gaining more and more ground in academia as well in business field. Academic researchers are finding great interest in the technical challenges that emotion analysis presents. Innovative businesses are engaged in the extraction of knowledge through user reviews on online stores and social media based the analysis of emotion. Finally, there is great consumer influence from the reviews that exist in the online world before someone decides to buy a service or product.

Therefore, sentiment analysis is a practice of gauging the sentiment expressed in a text, such as a twitter post or a review on TripAdvisor. Sentiment Analysis helps in estimating customer feedback on a brand and a product while adjusting sales and marketing strategy, but it is also capable of analyzing news and blogs assigning a value to the text (negative, positive or neutral) over social media platforms. As it stands now NLP algorithms can identify emotions such as happy, annoyed, angry, sad. In other words, Sentiment analysis is a text classification task. Through the combination of sentiment analysis and NLP, companies, organizations and governments will have all it takes to develop actionable strategies and implement data-driven decisions and policies.

1.5 Machine Translation – An Introduction

With most of the information around the world being made available in English, linguistic diversity around the world and the result of globalization requires the information be made available in local languages where English is not spoken or written. This requires huge amount of money being invested in translation services to make information available in local languages. European Union (EU) for instance, with its 27 member states and 23 official languages, is spending 1 billion on translation services, which is approximately 1% of its annual budget.

With the advent of inexpensive hardware and having lot of potential applications in future, governments and commercial vendors started encouraging Machine Translation research, a new branch in Computer Science with the goal of developing automatic language translation systems using computers. Thus, Machine Translation (MT) can be formally defined as the task of translating text given in one natural language to another automatically by making use of computers.

MT is an interesting and one of the difficult problems in the area of Artificial Intelligence (AI). Machine Translation research, not only popular among academic research community, its social, political and commercial applications surrounding it makes governments and industries show special interest towards developing high

quality machine translation systems. Though MT research has been active for past fifty years, fully automatic high-quality machine translation is still an elusive one to achieve.

Machine translation (MT) investigates the approaches to translating text from one natural language to another. It is a subfield of computational linguistics that draws ideas from linguistics, computer science, information theory, artificial intelligence, and statistics. For a long time, it had a bad reputation because it was perceived as low quality. Especially in the last two decades, we have been witnessing great progress in MT quality, which made it interesting also for the use in the translation industry. Its quality is still lower than human translation, but that does not mean it does not have good practical uses. In the past, translation agencies and other professional translators were the only actors in the translation industry, but, in recent years, we have been faced with the rapidly growing range of machine translation solutions entering the market and being of practical use. There is increasing pressure on the translation industry in terms of price, volume, and turnaround time. The emergence of commercial applications for MT is a welcome change in translation processes. In professional or official circumstances, human translation is inevitable, as humans are essential to making sure a translation is grammatically correct and carries the same meaning as the original text. Machine translation is appropriate in different circumstances, mainly for unofficial purposes or for providing content for a human translator to improve upon it. MT has proved to be able to speed up the whole translation process, but it cannot replace the human translator. The questions that researchers in the Machine Translation area are trying to answer today are: *How much can human translators benefit from using MT? How could MT be integrated efficiently into translation processes?* These questions will not be answered explicitly in this Thesis, but an effort will be made to show that MT is worth being part of the translation process, as its quality can be evaluated reliably. MT opens new opportunities for translators through using MT output only as a suggestion and, if necessary, post-editing it to the desired quality. It could be much faster than translation from scratch. This process is further discussed in the penultimate section of the chapter.

1.6 Thesis Contribution

Whereas the research field is very active, most of publications are limited to the domains of movie and product reviews in English. The object of the current Thesis is to evaluate the impact of Machine Translation on Sentiment Analysis. The goal of the evaluation is to compare the performance of Neural Machine Translation techniques and tools on original English corpora and on the corresponding corpora in German and Greek, translated employing Neural Machine Translation. The comparison examines the impact of two factors on Sentiment Analysis, the translation noise and the difference of sentiment lexicons in English and original languages. We have to mention that the English sentiment lexicon is well tested and more extensive than those in

other languages, which is likely to lead better performance in English. The comparison reveals equivalent performance rates of sentiment analysis on original and translated data, leading to a conclusion that the state-of-the-art Machine Translation systems can provide an alternative to the costly development of languages features to realize Sentiment Analysis in multilingual level.

1.7 Thesis Organization

This thesis is organized as follows. In the **first section** we provided an introduction in meanings such as Natural Language Processing (NLP), Sentiment Analysis and Machine Translation (MT). Main tasks and approaches of NLP subtasks were analyzed and the impact and the big significance of NLP in our societies also get noticed.

Although most of us use these three subtasks of Machine Learning in our everyday life, only a small amount can understand the impact of these three fields in our way of thinking and living. Every time we search a keyword in Search Machines, every time we read about a review in a hotel, a movie or a product we make use of these techniques and tools. Most of the tools, software, applications and devices that we use perform these Machine Learning tasks in order to provide high-level services and results and allow even near real-time analysis of various user generated data sources like online reviews, social posts and news.

Through, the sections of this Thesis we try to analyze these particular tasks (SA and MT), to provide recent works and to study the impact that Machine Translation has on the Sentiment Analysis.

In the **second section** we try to analyze the meaning, techniques and tools that used in Sentiment Analysis. Recent work in the field of Sentiment Analysis is also being presented and

In the **third section** we seek to analyze and mention the state-of-the-art methods and approaches for achieving Neural Machine Translation by using methods and tools from the area of Deep Learning.

Our proposed methodology is being described in the **fourth section** of this Thesis. We try to analyze the implementation of our approach and give short information about the collection of the processed datasets. Finally, we mention the measures that we take into consideration in order to measure the results of our method.

In the **fifth section** the results and the final implementation of our algorithms are being presented.

Finally, discussion about the results of our method and practices and the future work that we seek to do to will be introduced in the **final section**, Section 6, of this Thesis.

2 Sentiment Analysis

2.1 What is Sentiment Analysis (SA)?

One of the key areas of Natural Language Processing (NLP), a subset of artificial intelligence (AI) is Sentiment Analysis, the ability to understand emotional tones in speech and written context. It is an area that is the focus for several different functional applications and is widely utilized for online reviews and social media for a variety of applications, ranging from marketing to customer service.

Sentiment analysis or opinion mining is a notoriously difficult sub-field of Natural Language Processing and Data Science. At the most fundamental level, the task refers to the use of natural language processing, text analysis, and computational linguistics to identify, extract and automatically value the opinions and sentiments contained in source context.

A first separation of sentiment analysis is based on the exact meaning of the emotional state it is trying to identify. According to wikipedia SA can refer either to author's general emotional state when writing the examined context, or in the feeling that is deliberately transmitted by the author to the reader through the text, or to attitude - viewpoint - author's assessment of a topic. The first two cases can be classified into classes that express emotions perceived by man. That is, it is an attempt to recognize the emotions in the text such as joy, sadness and anger or situations like irony. However, classification can also be done in more general classes such as positive, negative and neutral feeling. In the latter case, where the attitude on a topic is examined, the classification is usually in two (positive attitude, negative attitude), three (positive, neutral, negative) or five (positive, rather positive, neutral, rather negative, negative) classes.

Another separation is based on the size of the text being examined. Thus, the emotion or its polarity may be sought in a whole document-based sentiment analysis, in a sentence-based sentiment analysis or even in individual phrases (feature / aspect-based sentiment analysis) when they refer to features of an entity in relation to which we seek for emotion. Following are some examples to give make the above separation clearer.

2.2 How does SA work?

There are two approaches to identify the sentiment of the people about the entity and most research on sentiment analysis field can be categorized into these two approaches: lexical based or machine-learning based. The machine learning method is further divided into two approaches, supervised learning and unsupervised learning, while the lexical based approach is divided into dictionary-based approach and corpus-based approach.

Machine Learning and SA

As for machine learning method, supervised learning requires labeled data to train algorithms, while unsupervised learning, does not require labeled data. The combination of labeled and unlabeled data yields semi-supervised learning.

The categories of machine learning are analyzed further below:

1) Supervised Learning: they appear on the system some input data and is able to learn how inputs are mapped to desired outputs each time. The next thing to do is to be able to predict future unknown exit entrances. Algorithms that fall into this category are Naïve Bayes, SVM and Maximum Entropy.

2) Unsupervised Learning: the algorithm creates one model for a sequence of inputs, with no known outputs. The most known algorithms are clustering algorithms, such as k-means and hierarchical clustering.

3) Semi-supervised Learning: Classifiers training is achieved by well-known learners and unknown outputs.

Most supervised and unsupervised approaches for sentiment analysis use words found as features. In order to apply these techniques to Internet data it is necessary to undergo appropriate preprocessing techniques of Natural Language Processing for the purpose of eliminating noise and unnecessary information that does not bring value to the result of the categorization. Then each text is represented as a vector of features, so that the classifier is able to identify and locate the most representative differences between texts belonging to different texts classes. Some of the most popular supervised learning algorithms are Naive Bayes, Maximum Entropy and Support Vector Machines (SVM). To be emphasized whereas the criteria used for the selection of these characteristics. They should, of course, be representative sample of the text while many times, depending on the classifier When applied, other features are required. For some classifiers (e.g., Naive Bayes), it is necessary the features to be binary, so that the classifier can judge whether an input has (or does not have) that particular attribute. However, there are many noteworthy techniques that automate the selection of these features and aim to find the most suitable combination for the model or classification purpose. The most popular in research community, mainly because of its simplicity and flexibility that provides in terms of its application is the TF - IDF (Term Frequency - Inverse Document Frequency) method, which will be discussed in more details in the next section. Other features such as syntactic features and frequency of words can also be used in the process of class labeling. The labels of a class can range from positive, negative and neutral to actual emotions like sad, happy or angry [15]. A classifier learns patterns from given features and then predicts sentiment of new instances based on their features. The data used for training directly affects the performance of the classifier. Therefore, data used to test classifiers is usually from the same domain as the data used for training. This characteristic is referred as domain specificity. Like domain specificity, the classifier can only be tested on the language on which it has been trained.

Based on the analysis of emotion in an extensive amount of texts, the algorithm is initially trained to be able to categorize new texts that resemble those that are trained. Unsupervised classifiers are used on Twitter and Facebook, with the aim to reduce dependency on data that have a known characteristic which is achieved by using different processes.

In addition, when classifiers are trained in a particular topic, it is possible not to perform equally well in another field that has not been trained. Moreover, classifiers do not perform well when posts or tweets include syntax and grammar, misspellings or abbreviations.

Lexical-Based SA

On the other hand, lexical based methods use a dictionary of words, where each word is associated with specific sentiment score such as positive (+1), negative (-1) and neutral (0). In other words, lexicon-based uses emotional dictionaries to give emotional rating to words and phrases and then composes them to get emotional score for the whole document. A simple implementation of the lexicon-based approach, in the case of simple sentences, is the addition of emotional ratings of words to the way the sentence's parse tree indicates. So, starting from the leaves, which are represent the words of the sentence, are successive additions of the scores to the final score at the root of the tree, which characterizes the whole sentence. Depending on the final rating the sentence is either negative or positive. Such approaches often contain negation handling. As the refusal reverses emotion of the word (cool - not cool), the emotional score of words written with a refusal can simply change the sign. Therefore, the intensity of the emotion is preserved but the polarity is reversed. One popular dictionary of emotional polarity is SentiWordNet [16], WordNet's version which gives emotional ratings to words on two levels, positive and negative.

There is an approach to use sentiment analysis with constructing a lexicon with information about which words and phrases are positive and which are negative. For example, SentiWordnet is a lexical resource for opinion mining. Each synset of WordNet, a publicly available thesaurus-like resource, is assigned one of three sentiment scores, positive, negative, or objective, [17] where these scores were automatically generated using a semi-supervised method described in Esuli and Sebastiani [18]. This lexicon can either compile manually or be acquired automatically. Also, lexicon can make use of specialized and generalized lexicons as well as the two together are able to eliminate the problem of word ambiguity, because a word has a different meaning each time depending on how it will be translated by the user.

Also, the use of words with the same meaning from generalized to specialized lexicon, increases the chances of some expression appearing poorly in the set training, to be recognized simultaneously in the control group. Therefore, the specialized lexicon is characterized by the frequency of occurrence of each word in either positive, negative or neutral messages.

2.3 Related Work

Sentiment Analysis is a relatively new field of research. Nasukawa and Yi [19] first introduced the term Sentiment Analysis, who used various developed standards to identify the emotion, while the term Opinion Mining made its first appearance by Dave, Lawrence and Pennock [20].

In 2003, Dave et al. [20] wanted to give the definition of Sentiment Analysis as the process of a multitude of data as a fundamental objective of categorizing according to the characteristics that they have also issued an opinion on the matter relating to these objectives. Sentiment analysis additionally employs various Natural Language Processing (NLP) and machine learning techniques that are implemented to classify a text, which contains emotions.

The first attempts made in the early 21st century by Turney [21] and Pang and Lee [22] who attempted to classify long texts into categories according to overall feeling they express. Turney has attempted to rank online reviews on cars, movies, banks and travel destinations using statistical methods and unsupervised learning based on point mutual information (Pointwise Mutual Information (PMI) index) between a phrase containing an adjective or adverb and words "Excellent" and "poor". While Pang, Lee και Vaithyanathan [22], in the same year, examined the application of supervised machine learning algorithms (Naive Bayes, Maximum Entropy Classification and SVMs) using various attributes to classify in problem of sentiment analysis for movie reviews.

The binary classification of one text in one of two categories (positive or negative) although it is the most common approach, it is not the only one. Work has also been done on multi-level emotional rankings such as in [23], where Pang and Lee examined the classification of critical films into one of 5 categories, ranging from one to five stars. More similar works in the field of sentiment analysis followed, where more complex machine learning models were being used (e.g. Hidden Markov Models - HMMs [24], or Conditional Random Fields - CRFs [25], models that take into account word order in the text) and different sets of features.

Yu and Hatzivassiloglou, also, considered reviews as bag of words and focused on classifying them as positive, negative, or neutral using classifiers and noticed that lexicons can be manually created as a list of predefined words or automatically created using machine learning techniques [26]. Taboada et al. [27], also, used lexicon-based method to perform sentiment classification.

For Zhang et. al [28], Sentiment Analysis is a kind of computational study of natural language text, with the main purpose of recognizing polarity of emotions, the degree of intensity and the issues that it applies to, so we have the ability to extract information about users' emotional state through the various texts they write daily on social media. This analysis has a great impact on the fields of management, finance, marketing, politics, and other social issues, as all users have the opportunity to express their aspects.

A lot of work has already been done, also, in the field of Sentiment analysis by using machine learning methods. There are several machine learning methods for sentiment categorization such as Maximum Entropy (MaxEnt) which is a feature-based model and doesn't takes independent assumptions as explained by [29]. Stochastic Gradient Descent (SGD) is another machine learning based algorithm that is capable of making the classifier learn even if it is based on non-differentiable loss function as explained by [30]. Moreover, Support Vector Machines (SVM) are one of the most famous supervised machine learning based classification algorithm [31].

Finally, recent years research is oriented towards the use of Neural Networks and Deep Learning techniques, such as Convolutional Neural Networks or Recurrent Neural Networks, which show remarkably results.

Support Vector Machines (SVM) and Artificial Neural Networks (ANN) are compared by Moraes et al. [32] for document level sentiment classification, which demonstrated that ANN produced competitive results to SVM's in most cases.

Johnson and Zhang [33] proposed a CNN variant named BoW-CNN that employs bag-of-word conversion in the convolution layer. They also designed a new model, called Seq-CNN, which keeps the sequential information of words by concatenating the one-hot vector of multiple words.

Yang et al. [34] proposed a hierarchical attention network for document level sentiment rating prediction of reviews. The model includes two levels of attention mechanisms: one at the word level and the other at the sentence level, which allow the model to pay more or less attention to individual words or sentences in constructing the representation of a document.

Zhou et al. [35] designed an attention-based LSTM network for cross-lingual sentiment classification at the document level. The model consists of two attention-based LSTMs for bilingual representation, and each LSTM is also hierarchically structured. In this setting, it effectively adapts the sentiment information from a resource-rich language (English) to a resource-poor language (Chinese) and helps improve the sentiment classification performance.

2.4 Text Classification

Text or document classification is a machine learning technique used to assigning text documents into one or more classes, among a predefined set of classes. A text classification system would successfully be able to classify each document to its correct class based on inherent properties of the text. Text Classification is a family of supervised machine learning algorithms that identify which category an item belongs to (such as whether an email is spam or not), based on labeled data (such as the email subject and message text). Some common use cases for text classification include credit card fraud detection, email spam detection, and sentiment analysis.

Text Classification takes a set of data with known labels and predetermined features and learns how to label new records, based on that information. Features are the

properties that we can use to make predictions. To build a classifier model, we have to explore and extract the features that most contribute to the classification.

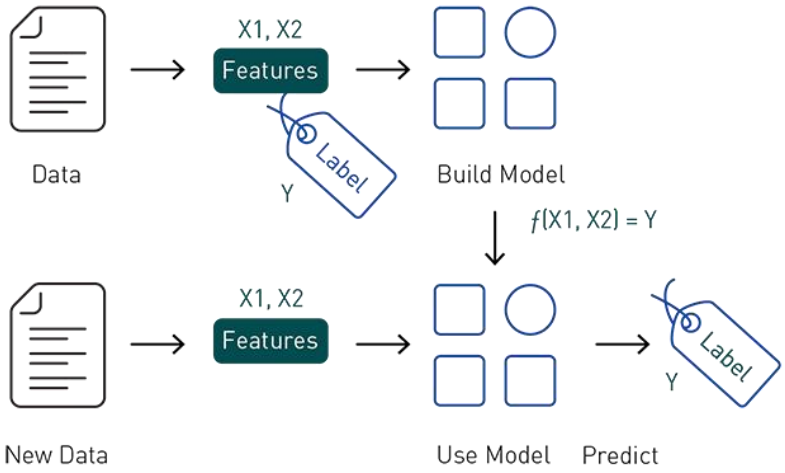

Figure 17: Text Classification Task

The most common type of text classification is sentiment analysis, whose goal is to identify the polarity of text content: the type of opinion it expresses. This can take the form of a binary like/dislike rating, or a more granular set of options, such as a star rating from 1 to 5. Examples of sentiment analysis include analyzing Twitter posts to determine if people liked a movie or extrapolating the general public's opinion of a new product from their reviews.

Much of the recent work on automatic document classification has involved supervised learning techniques such as classification trees, Naive Bayes, support vector machines (SVM), neural nets, and ensemble methods [36]. However, automatic Text Classification has become increasingly challenging over the last several years due to growth in corpus sizes and the number of domain and sub-domains that this task is been applicable. Moreover, Neural networks based multi-task learning has proven effective in many NLP problems. Thus, recent years many researchers seek to manage the Text Classification Tasks based on Deep Learning techniques by using Neural Networks, like CNN and RNN. [37]

2.5 TF-IDF (Term Frequency-Inverse Document Frequency)

TF-IDF systems are also called vector space models and have been proposed for the first time from Salton since 1971 [38]. According to this model, each term k_i in a document d_j is associated with a positive weight w_{ij} which expresses how important the term is for defining the semantics of the document and hence of its importance in the search system. The main idea behind this approach is that the words that occur more frequently in one document and less frequently in other documents should be given more importance as they are more useful for classification of its importance in the search system. There is also a weight to the terms of the query. If we represent the document as a vector $(w_{1j}, w_{2j}, \ldots, w_{tj})$ and the query as a vector $(w_{1q}, w_{2q}, \ldots, w_{tq})$

where *t* is the total number of terms in the system, then the relevance of the query and the document can be measured by the equation:

$$sim(d_j,q) = \frac{\sum_{i=1}^{t} w_{i,j} \, X \, w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \, X \, \sqrt{\sum_{i=1}^{t} w_{i,q}^2}}$$

The weight of term in document vector can be determined using TF × IDF method. The weight of term is measured how often the term j occurs in the document i (the term frequency $tf_{i,j}$) and in the whole document collection (the document frequency df_j (number of documents containing term j)). The weight of a term j in the document i is:

$$w_{i,j} = a(1 + \log(tf_{i,j}))log\frac{N}{df_i}, \text{ if } tf_{i,j} \geq 0$$

Where the term $tf_{i,j}$ is the frequency of term *i* in the document *j* (term frequency), df_i is the number of records containing the term *i* (document frequency) and N is the number of records in the collection.

Term Frequency is equal to the number of times a word occurs in a specific document.

$$TF(t) = \frac{Number\ of\ times\ term\ t\ appears\ in\ a\ document}{Total\ number\ of\ terms\ in\ the\ document}$$

Inverse Document Frequency for a specific word is equal to the total number of documents, divided by the number of documents that contain that specific word. The log of the whole term is calculated to reduce the impact of the division.

$$IDF(t) = log_e(\frac{Total\ number\ of\ documents}{Number\ of\ documents\ with\ term\ t\ in\ it})$$

2.6 Bag of Words (BOW)

The main two methods of sentiment analysis that mentioned before, lexicon-based method and machine learning based approach, both rely on the bag-of-words model (BOW). In this model [39], which is the most used in text classification, a document is represented as an unordered collection of words disregarding grammar and even word order. According to this model, during the training phase, a dictionary is built by dependent on training data and is then used to characterize between the positive and negative documents in the testing procedure.

For example, considering the above two sentences:

1) "George likes to watch movies. Mary likes movies too."

2) "George also likes to watch football games."

The vocabulary which is constructed based on BOW will be:

Dictionary = {1:" George", 2:"likes",3:"to", 4:" watch", 5:" movies", 6:" Mary",

7:" too", 8:"also", 9:" football", 10:" games"}

Hence, the feature vector of each document has “10 dimensionalities” based on the constructed vocabulary. The BOW model only considers if a known word occurs in a document or not. It does not care about meaning, context, and order in which they appear. According to the nature of natural language, one word can express the authors’ attitude clearly while a sequence of words cannot.

Next step after creating the vocabulary is to vectorize our document, as so our sentences, by counting how many times each word appears. Therefore, the 10-length vectors for each sentence will be as below:

"George likes to watch movies. Mary likes movies too." -> [1, 2, 1, 1, 2, 1, 1, 0, 0, 0]

"George also likes to watch football games." -> [1, 1, 1, 1, 0, 0, 0, 1, 1, 1]

It should be noticed that BOW completely ignores the context in which it is used. It only shows *what* words occur in the document, not *where* they occurred. Moreover, for a large document, the vector size can be huge resulting in a lot of computation and time. Removal of stopwords or other preprocessing steps may be executed in order to ignore words based on relevance to the specific use case.

2.7 Word Embeddings

The need to represent words in vectors was dictated by many NLP problems. These vectors are usually based on the method of word embeddings, but they can be also one-hot vectors with the unit indicating the word index in the vocabulary. Sparse vector representations of text, the so-called bag-of-words model have a long history in NLP. Dense vector representations of words or word embeddings have been used as early as 2001 as we have seen above. Proposed by Mikolov et al. [40] in 2013 Word Embeddings attracted academics interesting because of the quality of the vector representations they produce.

There are two main differences between single integer representation and word embeddings. With one-hot representation, a word is represented only with a single integer. With vector representation a word is represented by a vector of 50, 100, 200, and more dimensions. Hence, word embeddings capture a lot more information about words and encode the semantic affinity between relationships. The representation of words with one-hot vectors although it gives good results in document classification problems, it does not perform equally well in sentence classification problems, which are preferred word2vec or Glove vectors.

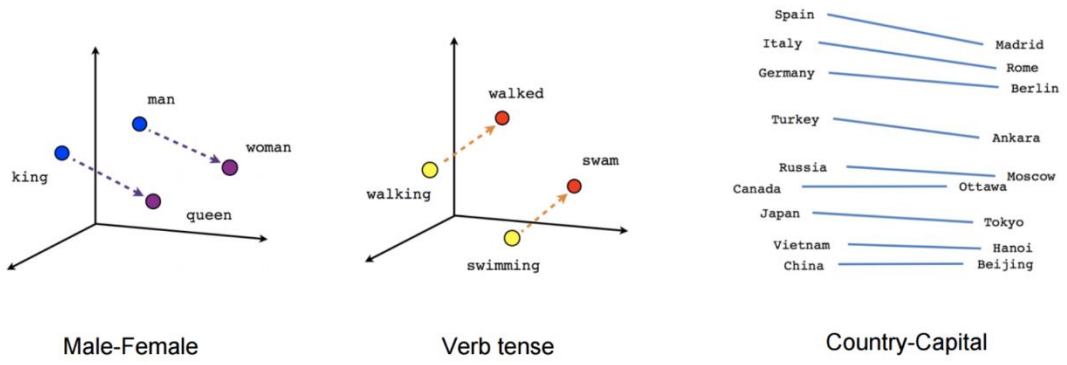

Figure 18: Relations captured by Word2Vec

Word embeddings, like Word2Vec and GloVe that will analyzed below, are often pre-trained on text corpus from co-occurrence statistics and combine the distributional hypothesis with artificial neural networks [41].

GloVe

The Glove algorithm[2] is an unsupervised learning algorithm for acquisition vector representations of words. Education is applied to a large board co-occurrence measurement from a corpus, and the resulting representations exhibit useful properties of words, such as proportion relations or semantic affinity relationships.

For example, it produces a vector that approaches the representation of vec ('Rome') as result of the action vec ('Paris') - vec ('France') + vec ('Italy') and represents words that are semantically close to corresponding near positions in the sense of euclidean distance or of the cosine distance.

Word2Vec

The Word2Vec model is a neural language model that is used to represent words in small dimensional vector spaces. The Word2Vec model is capable of discovering complex semantic and syntactic meaning relationships between words and convert them into linear properties of vector space. Generally, this is a window-based prediction language model that it is based on a simple neural network architecture.

The Word2vec algorithm uses a 3-layer Neural Networks, as shown in Figure 19, where neurons of the hidden plane are all linear. The input neurons are what the words are of the corpus vocabulary for education. The number of hidden neurons is equal to the desired dimension of the word vectors that will result. The output neurons are equal to the number of input neurons. Word2vec is like an autoencoder, encoding each word in a vector, but rather than training against the input words through reconstruction, word2vec trains words against other words that neighbor them in the input corpus.

[2] https://nlp.stanford.edu/projects/glove/

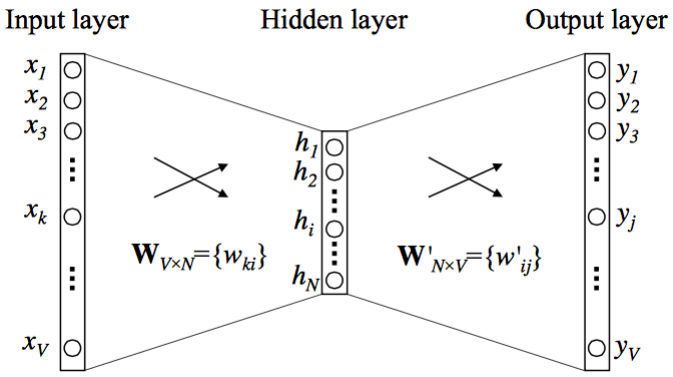

Figure 19: Word2Vec architecture

In order to achieve this encoding Word2Vec algorithm uses context to predict a target word, a method known as continuous bag-of-words (CBOW), or uses a word to predict a target context, which is called skip-gram. In the CBOW model, the context is represented by multiple words for a given goal. The prediction in the case of CBOW relates to the central data word of the surrounding words.

The skip-gram model reverses the use of the target word and context words. In this case, the target word is fed to the input and output level of the neuron is repeated many times to match the surrounding words. This model produces more accurate results on large datasets.

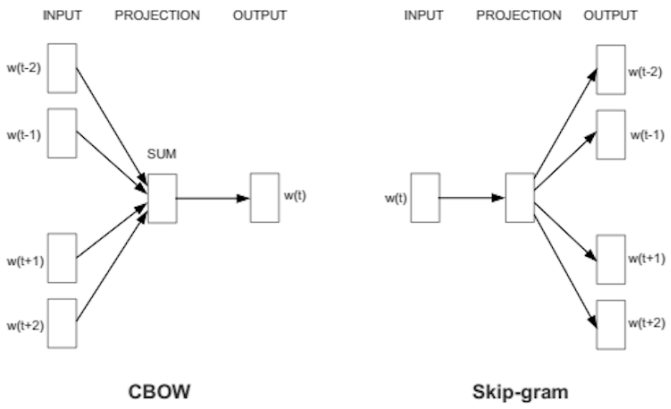

Figure 20: CBOW vs. Skip-gram model

3 Neural Machine Translation

3.1 What is Machine Translation (MT)?

Translation is an essential tool for communicating between businesses and companies with their clients, but also between organizations and countries. One of the usual ways of translating any document is to get it done by a freelance translator or a translation agency. However, in recent years, translation machines have become more and more popular. Mechanical translation has evolved dramatically in recent times. We live in a time when most students are looking for Google translate help with their foreign language assignments.

In recent years, translation engines have become more precise thanks to word and content training programs.

The reason for the above upgrade is the drastic changes in technology, as all industries are now turning to artificial intelligence and machine learning. This has made mechanical translation and automatic translation services some of the driving forces behind the development of this technology. Companies like Google and Microsoft are clearly showing a growing interest in automatic translation.

Most of us were introduced to machine translation when Google came up with the service of Google Translate. But the concept has been around since the middle of last century. Machine Translation was one of the first applications envisioned for computers and the first suggestions concerning MT where made by the Russian Smirnov-Troyansky and the Frenchman G.B. in 1930's [42].

However, the first serious discussions were begun in 1946 by the mathematical Warren Weaver and research work started as early as 1950's, primarily in the United States. These early systems relied on huge bilingual dictionaries, hand-coded rules, and universal principles underlying natural language.

First demonstrated by IBM in 1954 with a basic word-for-word translation system. The system had a pretty small vocabulary of only 250 words, and it could translate only 49 hand-picked Russian sentences to English. The number seems minuscule now, but the system is widely regarded as an important milestone in the progress of machine translation.

The idea of a machine for automatic translation has been around since the 1960s. Despite the initial enthusiasm, scientists quickly found the difficulty of the task due to the complexity of the problem. One way to reduce the complexity of such systems is

to limit the themes that the system can address, as well as the relevant vocabulary. Another possible limitation is only the vocabulary or the concepts with which a word can be banned by prohibiting metaphorical metaphors. The first successful attempt to do so was made by the company XEROX which defined a subset of English language with the words that its technical manuals had to be written. An automated translation system was then implemented by SYSTRAN, which automatically translated texts into other languages.

Finally, in 1981, a new system called the METEO System was deployed in Canada for translation of weather forecasts issued in French into English. It was quite a successful project which stayed in operation until 2001.

The world's first web translation tool, Babel Fish, was launched by the AltaVista search engine in 1997.

And then came the breakthrough we are all familiar with now – Google Translate. It has since changed the way we work and even learn with different languages. Google's research team has made a great impact in the field of Machine Translation in the last ten years and even today they introduce innovative approaches and techniques.

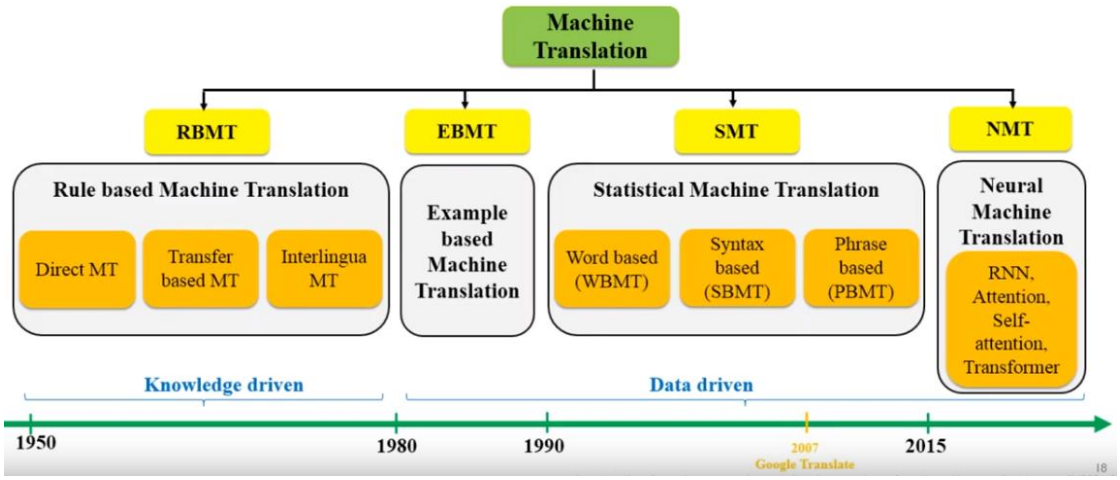

Figure 21: History of MT's Approaches

3.2 How does MT work?

One very simple but still useful representation of any automatic translation process is the following triangle which was introduced by the French researcher B. Vauquois in 1968 [43].

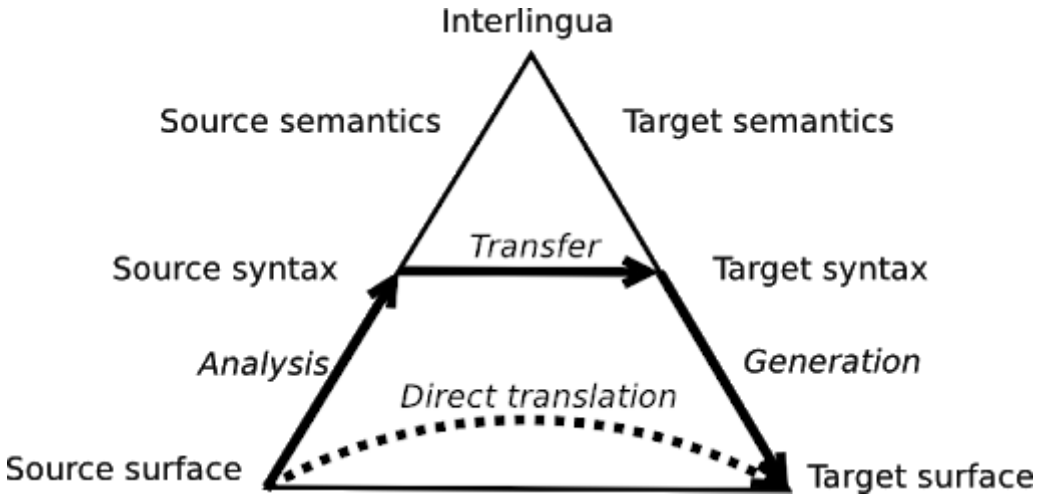

Figure 22: Vauquois' translation triangle

The triangle represents the process of transforming the source sentence into the target sentence in 3 different steps.

The left side of the triangle characterizes the source language, when the right side the target language. The different levels inside the triangle represent the depth of the analysis of the source sentence, for instance the syntactic or semantic analysis. We now know that we cannot separate the syntactic and semantic analysis of a given sentence, but still the theory is that you can dig deeper and deeper into the analysis of a given sentence. The first red arrow represents the analysis of the sentence in the source language. From the actual sentence, which is just a sequence of words, we can build an internal representation corresponding to how deep we can analyze the sentence.

For instance, on one level we can determine the parts of speech of each word (noun, verb, etc.), and on another we can connect words, e.g. which noun phrase is the subject of which verb.

When the analysis is finished, the sentence is "transferred" by a second process into a representation of equal or slightly less depth in the target language. Then, a third process called "generation" generates the actual target sentence from this internal representation, i.e. a meaningful sequence of words in the target language. The idea of using a triangle is that the higher/deeper you analyze the source language, the smaller/simpler the transfer phase. Ultimately, if we could convert a source language into a universal "interlingua" representation during this analysis, then we would not need to perform any transfer at all – and we would only need an analyzer and generator for each language to translate from any language to any language.

This is the general idea and explains intermediate representation, if any exists, and the mechanisms involved to go from one step to the next. More importantly, this model describes the nature of the resources that this mechanism uses. Let us illustrate how this idea works for the 3 different technologies using a very simple sentence: *"The smart mouse plays violin."*

3.2.1 Rule-Based MT

Rule-Based Machine Translation is a classical approach of machine translation (MT) systems based on linguistic information that includes morphological, syntactic, and semantic of both the source language (SL) and target language (TL). The main approach of rule-based machine translation systems is based on linking the structure of the given source sentence with the structure of the target sentence, keeping the meaning of source sentence unchanged.

There are three approaches being used for developing rule-based translation systems: direct translation that uses word-to-word translation, transfer-based approach that applies linguistic rules during the translation process, and interlingua-based translation that maps the SL to an abstract intermediate representation from which the TL is generated [44].

Rule-Based machine translation is the oldest approach and covers a wide variety of different technology. However, all rule-based engines generally share the following characteristics:

- The process strictly follows the Vauquois triangle and the analysis side is often very advanced, while the generation part is sometimes reduced to the minimal.
- All 3 steps of the process use a database of rules and lexical items on which the rules apply.
- These rules and lexical items are machine-readable and can be modified by linguist/lexicographer.

3.2.2 Statistical MT

Statistical machine translation (SMT) learns how to translate by analyzing existing human translations, known as bilingual text corpora. In contrast to the Rules Based Machine Translation (RBMT) approach that is usually word-based, most modern SMT systems are phrased-based and assemble translations using overlap phrases. SMT is a machine translation paradigm where translations are generated based on statistical models whose parameters are derived from the analysis of bilingual text corpora and its main idea comes from the information theory. Thus, SMT uses statistical analysis and predictive algorithms to define rules that are best suited for target sentence translation. Translation systems are trained on large quantities of parallel or monolingual data.

In this approach a document is translated according to the probability distribution *P(e/f)* that a string *e* in the target language is the translation of a string *f* in the source language.

The first ideas of Statistical Machine Translation were introduced by Warren Weaver as far back as 1949. He explained that language had an inherent logic that could be treated in the same way as any logical mathematical challenge. He contended that

logical deduction could be used to identify “conclusions” in the target language based on what already existed in the source language [45].

Statistical machine translation was re-introduced in 1991 by researchers at IBM's Thomas J. Watson Research Center and has contributed to the significant resurgence in interest in Machine Translation in recent years. With the advent of the cloud, and affordability of powerful computers, the theory of Statistical Machine Translation (SMT) became the most widely studied Machine Translation method.

3.2.3 Word-Based MT

When the Word-Based SMT was first proposed, the model translates words as unique units and inputs. The target language is generated word by word and we need to calculate how fluency is the concatenation of the individual words [46].

For example, considering the German sentence *“Auf diese Frage habe ich leider keine Antwort bekommen”* we can translate it based on the Word-Based SMT approach in the equivalent English sentence *“I did not unfortunately receive an answer to this question”*.

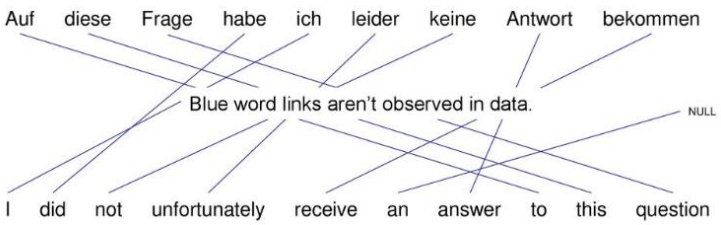

Figure 23: Example of Word-Based SMT, German to English

And the representation of the model can be shown in the below image. Where words are the inputs in the system and

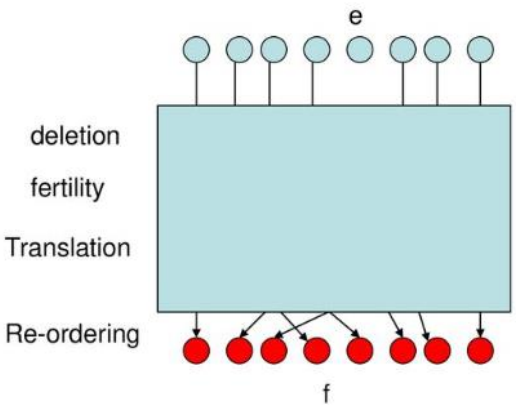

Figure 24: Word-Based SMT architecture

3.2.4 Phrase-Based MT

Phrase-Based Machine Translation is the simplest and most popular version of statistical machine translation and it aims to reduce the restrictions of word-based approach by translating whole sequences of words, where the lengths may differ. The

sequences of words are called phrases or blocks and are not linguistic phrases, but phrasemes found using statistical methods from corpora.

The same sentence that we used as an example in the Word-Based approach can now be translated as below, when using the phrases as fundamental inputs to our translation:

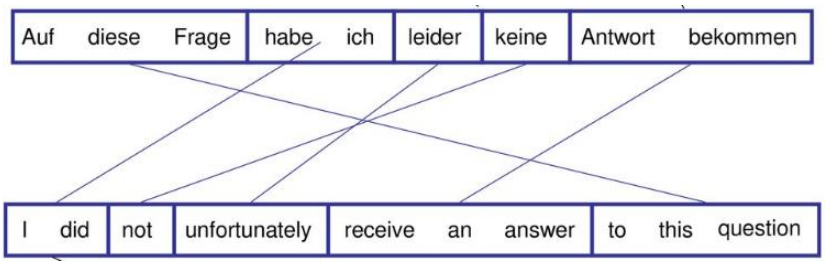

Figure 25: Example of Phrase-Based SMT, German to English

And the model that represents this functionality is as below:

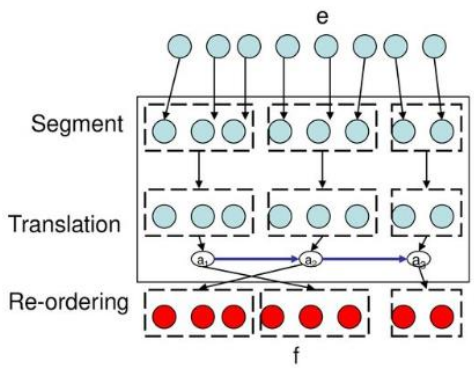

Figure 26: Phrase-Based SMT architecture

Technically-speaking, phrase-based machine translation does not follow the process defined by Vauquois. Not only is there no analysis or generation, but more importantly the transfer part is not deterministic. This is to say that the engine can generate multiple translations for one source sentence, and the strength of the approach resides in its ability to select the best one.

Thus, the model is based on 3 main resources:

- A phrase-table which produces translation option and their probabilities for "phrases" (sequences of words) on the source language
- A reordering table indicating how words can be reordered when transferred from source language to target language
- A language model which gives probability for each possible word sequence in the target language

3.2.5 Example-Based MT

Example-based translation is another corpus-based Machine Translation, which is also known as Memory based translation. This approach is based on finding analogous examples of the language pairs and the idea of "Translation by Analogy", first proposed by Makoto Nagao in 1984 [47]. In other words, examples of already existing translations are used as the basis for new translations.

Given the source sentence, sentences with similar sub-sentential components are extracted from the source side of the bilingual corpus, and their translations to the target language are then used to construct the complete translation of the sentence.

3.2.6 Hybrid MT

Method of MT characterized by using multiple approaches within a single MT system. While statistical methods still dominate research work in MT, most commercial MT systems were, from the beginning, only rule based. Recently, boundaries between the two approaches have narrowed, and hybrid approaches emerged, which try to gain benefit from both. We distinguish two groups of hybrid MT, those guided by rule-based MT and those guided by statistical approaches. Hybrid systems, guided by rule-based MT, use statistical MT to identify the set of appropriate translation candidates and/or to combine partial translations into the final sentence in the target language. Hybrid systems, guided by statistical MT, use rules at pre-/post-processing stages.

3.3 From MT to Neural Machine Translation (NMT)

Neural Machine Translation departs from existing phrase-based statistical approaches that use separately optimized engineered subcomponents [48]. Research output has consistently increased since 2014, growing at close to a 180% compound annual growth rate over five years. The below image shows the research papers were published on Arxiv.org that mentioned Neural Machine Translation in the title or abstract since January 1, 2014 to March 15, 2019 [49]. With the huge amount of research interest in recent years, there are various variants of NMT that are currently being investigated and developed in the industry.

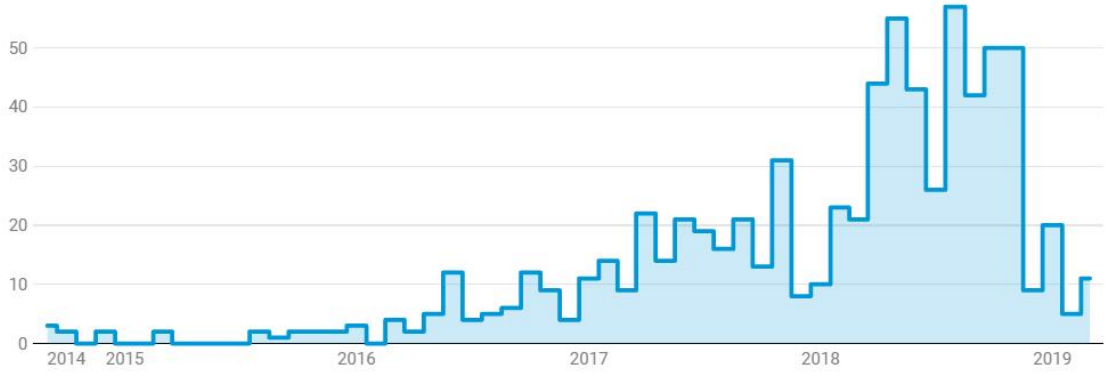

Figure 27: Neural Machine Translation Research Output

NMT has improved the imitation of professional translations over the years of its advancement. (Kalchbrenner and Blunsom 2013; Sutskever et al., 2014) [50, 51] inspired by the use and the great results of deep representational learning, proposed

a new approach for Machine Translation, the Neural Machine Translation, which is based exclusively on Neural Networks. The main approach and architect on using NNs on NLP tasks and especially in the task on Neural Machine Translation is based on the use of an Encoder-Decoder approach. Translation by its definition is a complex task, where in one hand we seek to understand and extract the meaning and the information from one language and in the other hand we want to translate and transform the extracted information in another language. This concept is represented as an Encoder-Decoder system with NNs. The encoder extracts a fixed-length representation from a variable-length input sentence, and the decoder generates a correct translation from its representation.

When applied in neural machine translation, natural language processing helps educate neural machine networks. This can be used by businesses to translate low texts, etc. Such impact content including emails, regulatory machine translation tools speed up communication with partners while enriching other business interactions. NMT is a relatively new approach to statistical MT based purely on Neural Networks.

In two years, neural networks surpassed everything that had appeared in the past 20 years of machine translation. Neural Machine Translation contains 50% fewer word order mistakes, 17% fewer lexical mistakes, and 19% fewer grammar mistakes. The neural networks even learned to harmonize gender and case in different languages.

The most noticeable improvements occurred in fields where direct translation was never used. Statistical machine translation methods always worked using English as the key source. Thus, if you translated from Russian to German, the machine first translated the text to English and then from English to German, which leads to a double loss of the meaning between the two translations.

Neural Machine Translation is not relied on this approach. Using a state-of-the-art encoder-decoder model it implements transfer learning, so the direct translation between languages with no common dictionary became possible. Most research works and applications make use of a bidirectional recurrent neural network (RNN), known as an encoder, in order to encode a source sentence for a second RNN, known as a decoder, that is used to predict words in the target language [52]. This state-of-the-art algorithm is an application of deep learning in which massive data sets of translated sentences are used to develop a model capable of translating between any two languages.

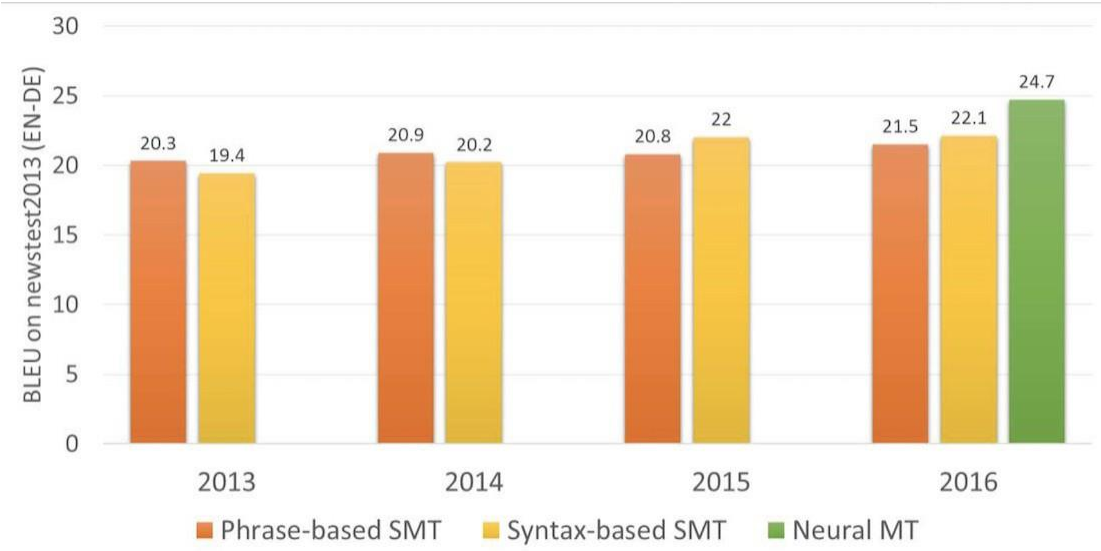

Figure 28: NMT vs. Phrase-based SMT vs. Syntax-based SMT

3.4 Related Work

Recent years research community has a great interest on Machine Translation applications and tools.

Christoph Tillmann and Fei Xia [53] introduced a phrase-based Unigram Model for Statistical Machine Translation a much simpler SMT phrase-based model were blocks - pairs of phrases have used us inputs in the translation model. Moreover, this study indicates that longer phrases which occur less frequently do not help much.

Also, many different suites of open-source tools for Machine Translation, like Moses [54], have been introduced in recent years and many corpora, like Europarl [55], have been collected in order to measure their acquisition and application as training data for Machine Translation systems and to generate sentence aligned text for statistical machine translation systems.

OpenNMT [56] ecosystem is another project that has been introduced in December 2016 by the Harvard NLP group and SYSTRAN and has since been used in several research and industry applications.

The encoder-decoder architecture for recurrent neural networks is displacing classical phrase-based statistical machine translation systems [52]. Cho et al., 2014 [57] first proposed a novel neural network model called RNN Encoder–Decoder that consists of two recurrent neural networks (RNN). The valuation of this model on the task of translating from English to French improves the translation performance. This approach has let us have huge potentials due to the internet, globalization and international politics. Translation is reasonable for language pairs with a large amount of resources and the research and work in order to include more “minor” languages, had already began.

A problem with the naive Encoder-Decoder model is that the encoder maps the input to a fixed-length internal representation from which the decoder must produce the entire output sequence.

Google's research team in their publication [58] introduced an improvement to the model that allows the decoder to "pay attention" to different words in the input sequence as it outputs each word in the output sequence. In the same paper the term of "Transformers" also being introduced. It applies a self-attention mechanism, in each step, which directly models relationships between all words in a sentence, regardless of their respective position. Transformer's high performance has demonstrated that feed forward neural networks can be as effective as recurrent neural networks when applied to sequence tasks, such as language modeling and translation.

Just few months ago, May 2019, Google's research team after conducting an evolution-based neural architecture search (NAS), using translation as a proxy for sequence tasks in general, found the Evolved Transformer [59], a new Transformer architecture that demonstrates promising improvements on a variety of natural language processing (NLP) tasks. In tests, the team compared the Evolved Transformer with the original Transformer on the English-German translation task used during the model search, and found that the former achieved better performance on both BLEU (an algorithm for evaluating the quality of machine-translated text) [60] and perplexity (a measurement of how well probability distribution predicts a sample) at all sizes. At larger sizes, the Evolved Transformer reached state-of-the-art performance with a BLEU score of 29.8, and on experiments involving translation with different language pairs and language modeling, they observed a performance improvement of nearly two perplexity.

One of the latest milestones in the field of Transformers and generally of NLP is the release of BERT, an event described as marking the beginning of a new era in NLP. BERT (Bidirectional Encoder Representations for Transformers) is a new transfer learning technique and is the first deeply bidirectional, unsupervised language representation, pre-trained using only a plain text corpus [61]. Soon after the release of their associated paper [62], the team also open-sourced the code of the model and made available for download versions of the model that were already pre-trained on massive datasets.

3.5 DNNs (Deep Neural Networks) in MT

Deep Neural Networks (DNNs) are powerful models that have achieved excellent performance on difficult learning tasks. Deep-learning models owe their accuracy to the huge amounts of data they are trained on. The more examples an AI system sees, the more likely it is to find answers to the questions it must answer. At its heart, deep learning is doing pattern matching—using complex math to map inputs to outputs based on statistics and similarities. And pattern-matching is different from understanding the meaning of words and sentences.

DNNs are have been proven to efficiently solve time series problems, also called sequence problems, in a variety of domains. NLP field is one of the main domains that have benefited from the application of deep learning techniques and tools. Text classification and Machine Translation, that we focus to examine and analyze in this Thesis, are prime examples of many-to-one and many-to-many sequence problems. In Text Classification we have an input sequence of words and we want to predict a single output tag, where contrary in Machine Translation a text sequence is an input and another text sequence is the output.

Despite the flexibility Deep Neural Networks bring to NLP, current AI is still far from understanding natural language in the way that humans do. Language models based on deep learning still suffer from some of the same fundamental problems that their rule-based predecessors did. When they become involved in tasks that require general knowledge about people and things, deep-learning language models often make easy errors. Therefore, many companies are still hiring thousands of human operators to steer the AI algorithms in the right direction.

Real natural language processing probably will not be possible until we crack the code of human-level AI, the kind of synthetic intelligence that really works like the human brain. But as we move toward that elusive goal, our discoveries are helping bridge the communication gap between humans and machines. DNNs models that are widely explored to handle various NLP tasks base on the implementation and use of two main types of Neural Networks, Convolutional Neural Networks (CNNs) and Recurrent Neural networks (RNNs). CNN is supposed to be good at extracting position-invariant features and RNN at modeling units in sequence. Yin et al. [63] have proven that RNNs perform well and robust in a broad range of tasks in which sequential information is clearly important. On the other hand, CNNs can work well for tasks where feature detection in text is more important, like sentiment classification since sentiment is usually determined by some key phrases. In addition, hidden size and batch size can make DNN performance vary dramatically and that the optimization of these two parameters is crucial to good performance of both CNNs and RNNs.

Sequences pose a challenge for DNNs because they require that the dimensionality of the inputs and outputs is known and fixed. Thus, Sutskever et al., 2014 [51] proposed the implementation of a new approach, the Sequence-to-sequence (Seq2seq) model. The seq2seq architecture is a type of many-to-many sequence modeling that have achieved a lot of success in a variety of tasks such as Text-Summarization, chatbot development, conversational modeling, and neural machine translation.

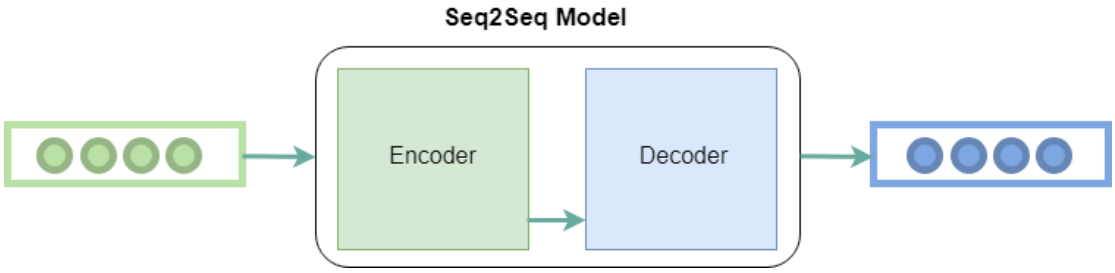

Figure 29: Representation of a Seq2Seq model

Seq2seq models that used and have been proven to solve based on the implementation of three main types of Recurrent Neural networks (RNNs). The simple vanilla (RNN), gated recurrent unit (GRU) and long short-term memory unit (LSTM). These 3 types will be analyzed further below.

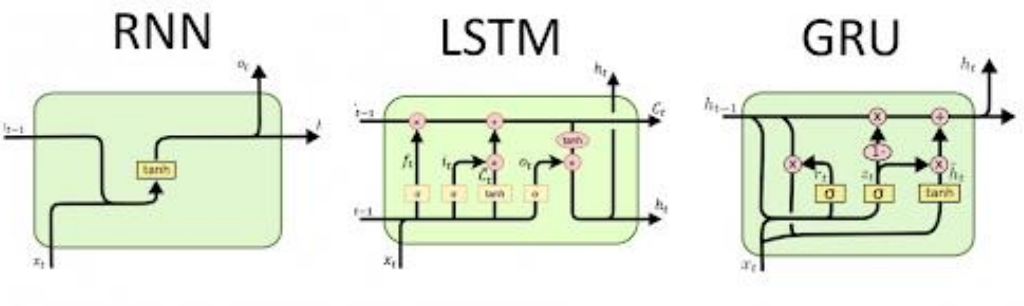

Figure 30: RNNs' models

3.5.1 Convolutional Neural Networks (CNNs)

Words in any document or text appear in sentences (sequences) and their meaning is compositional, formed by other neighbor words and sentences. Convolutions are one way to take advantage of this structure. The main approach is that certain sequences of words, or n-grams, usually have the same meaning regardless of their overall position in the sentence. Introducing a structural prior via the convolution operation allows us to model the interaction between neighboring words and consequently gives us a better way to represent the meaning of the sentences.

CNNs were responsible for major breakthroughs in Image Classification and are the core of most Computer Vision systems today, from Facebook's automated photo tagging to self-driving cars. Convolutional neural network is a type of network that is primarily used for 2D data classification, such as images. A convolutional network tries to find specific features in an image in the first layer. In the next layers, the initially detected features are joined to form bigger features. In this way, the whole image is detected.

Instead of image pixels, the input to most NLP tasks are sentences or documents represented as a matrix. Each row of the matrix corresponds to one token, typically a word, but it could be also a character. That is, each row is vector that represents a word. Typically, these vectors are word embeddings like word2vec or GloVe, which have mentioned in one of our previous sections, but they could also be one-hot

vectors that index the word into a vocabulary. For a 10-word sentence using a 100-dimensional embedding we would have a 10×100 matrix as our input. That will be our "image".

In vision, our filters slide over local patches of an image, but in NLP we typically use filters that slide over full rows of the matrix (words). Thus, the "width" of our filters is usually the same as the width of the input matrix. The height, or region size, may vary, but sliding windows over 2-5 words at a time is typical. Putting all the above together, a Convolutional Neural Network for NLP may look like in the below image.

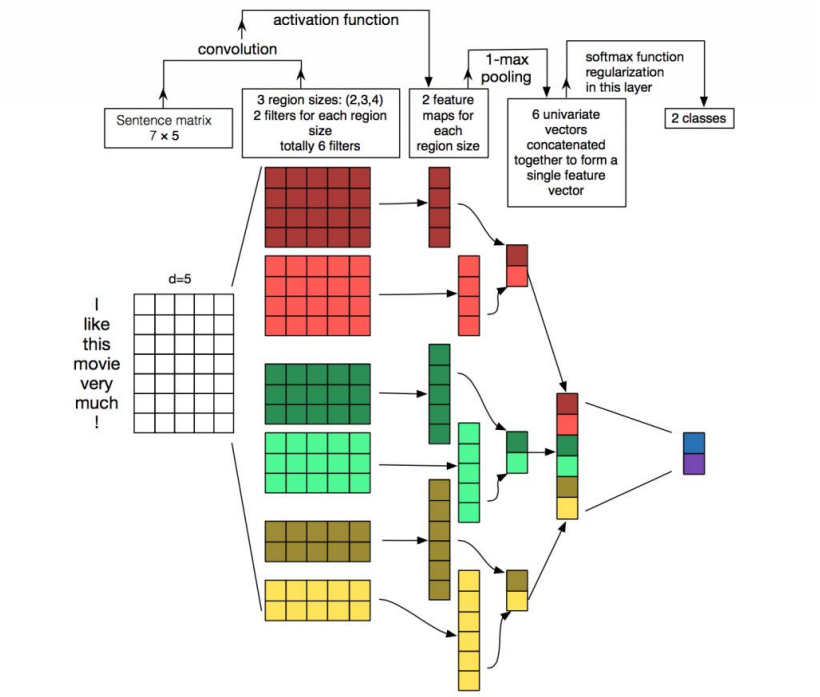

Figure 31: Example of a CNN

Here we depict three filter region sizes: 2, 3 and 4, each of which has 2 filters. Every filter performs convolution on the sentence matrix and generates (variable-length) feature maps. Then 1-max pooling is performed over each map, i.e., the largest number from each feature map is recorded. Thus, a univariate feature vector is generated from all six maps, and these 6 features are concatenated to form a feature vector for the penultimate layer. The final softmax layer then receives this feature vector as input and uses it to classify the sentence; here we assume binary classification and hence depict two possible output states. [64]

3.5.2 Recurrent Neural Networks (RNNs)

DNNs are powerful models that have achieved excellent performance on difficult learning tasks. In many practical applications of machine learning, the entry or exit space contains sequences. For example, sentences are often modeled as word sequences, where each word can be represented as a single vector (one hot-vector), which means that this vector has zero everywhere except one position has an ace and corresponds to the word of a dictionary we want to represent. Although DNNs work

well whenever large labeled training sets are available, they cannot be used to map sequences to sequences (seq2seq). Thus, a new approach to sequence learning that makes minimal assumptions on the sequence structure was introduced. One Recurrent Neural Network is a model that process a sequence of vectors $\{x_1, x_2, \ldots, x_n\}$ using a feedback formula of the form $h_t = f_\vartheta(h_{t-1}, x_t)$, where f is a function, while the same parameters ϑ are used in each step, which gives us the right to process sequences with a random number of vectors. The hidden vector h_t can be interpreted as a current summary of all vectors x up to that step and the feedback formula updates this summary based on next vector.

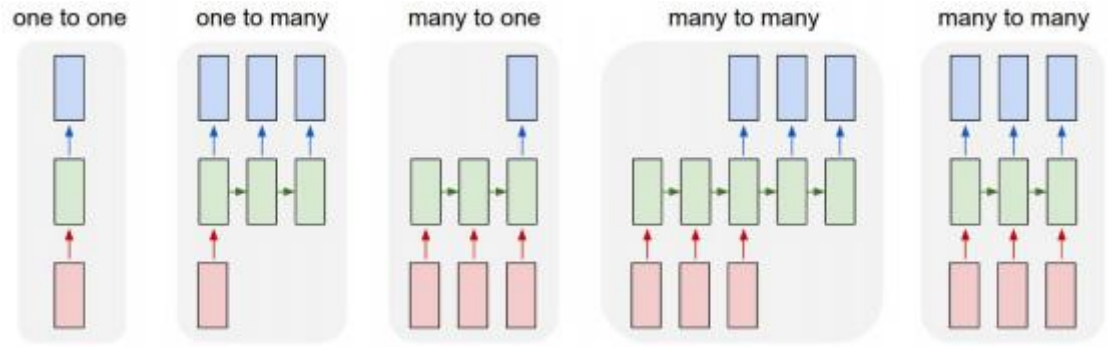

Figure 32: Representation of different Seq2Seq models for RNNs

As shown in the above diagram, a typical Recurrent Neural Network on the left can accept a vector as input (red), transform it through a hidden layer (green) and ultimately produce an output vector (blue). In these diagrams, the boxes represent vectors while the arrows indicate functional dependencies. Feedback networks allow us to process vector sequences, such as: 1) at the output, 2) at the input, or 3) and on both sides, either serially or in parallel, as shown in the other figures.

Also, we can notice that every prediction at time t is dependent on all previous predictions and the information learned from them. RNNs differ from simple feedforward networks, the notion that they are promoting their output again towards their input. Therefore, we can imagine that these networks have memory. Adding memory, however, in a network, has a purpose: There is information in the sequences input, and which are used by the feedback neural networks to perform tasks that simple promotional networks cannot.

The Encoder-Decoder architecture with Recurrent Neural Networks (RNN) has become an effective and standard approach for both Neural Machine Translation (NMT) and sequence-to-sequence (seq2seq) prediction in general. Machine Translation is similar to sequence-to-sequence (seq2seq) prediction in that our input is a sequence of words in our source language (e.g. German) and we want to output a sequence of words in our target language (e.g. English).

The key benefits of this approach are the ability to train a single end-to-end model directly on source and target sentences and the ability to handle variable length input and output sequences of text.

RNNs can solve our purpose of sequence-to-sequence (seq2seq) prediction to a great extent. But, in most of NLP's tasks, like Speech Recognition and Machine Translation, we want to process and generate large scales of written or spoken text. This text should be depending for example from the content of the last sentence. Now RNNs are great when it comes to short contexts, but in order to be able to build a story and remember it, we need our models to be able to understand and remember the context behind the sequences, just like a human brain. This is not possible with a simple RNN.

The reason behind this is the problem of vanishing/exploding gradients [65], which has led to the development of LSTMs and GRUs.

Recurrent Neural Networks, like most neural networks, have created for many years. The problem of vanishing/exploding gradients was introduced in the early 1990s and has been a major impact to the performance of RNNs. As a straight line expresses a change in the x-axis along with a change in y-axis, so the slope expresses a change in weights with respect to the change in error. If we cannot know the slope, we cannot adjust the weights in a direction that will reduce the error and, therefore, our network stops learning.

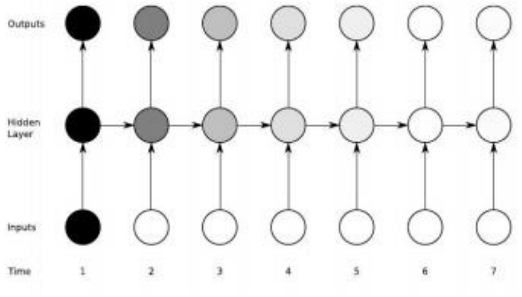

Figure 33: Vanishing gradient problem in a RNN

Recurrent neural networks are designed to establish connections between a final output and events, many steps before being shortened because it is quite difficult to know in advance how important it is to attribute to individuals' entrances. This is caused due to the fact that the information that flows through Neural Networks go through many stages of propagation. Because the levels and time steps of deep neural networks are interconnected by multiplying them, the derivatives are sensitive to vanishing or "exploding".

This issue can be resolved by applying a slightly tweaked version of RNNs – the Long Short-Term Memory Networks.

3.5.3 Long Short-Term Memory (LSTMs) Networks

RNNs have difficulties learning long-range dependencies – interactions between words that are several steps or even sentences apart. If a sequence is long enough,

they have a hard time carrying information from earlier time steps to later ones. That is the real problem behind the approach of a RNN model, because the meaning of an sentence is often determined by words that are not very close: “I was born in *Greece*, in 1990, and although I know 4 different languages my mother tongue is *Greek*”. It is unlikely that a simple RNN would be able to capture the information, that because I was born in Greece my mother language is Greek. The context is preserved from timestamp to timestamp, which can be very useful, but that might get lost in longer sentences, so a new DNN model called LSTM was introduced [66].

LSTMs have an edge over RNNs in many ways. This is because of their property of selectively remembering patterns for long durations of time, which make them capable of learning long-term dependencies. Their main characteristic is the ability to remember information for long time periods.

In addition to the context being passed as it is in RNNs, LSTMs have an additional pipeline of contexts called cell state, the horizontal line in the top of the below diagram. This pipeline helps LSTMs maintain the error that can be traced back through time and levels, based on back propagation learning, and can pass through the whole network and impact it. By maintaining a more stable error, we give the ability to RNNs to learn over many timestamps and therefore to open a channel to connect the outputs and results separately.

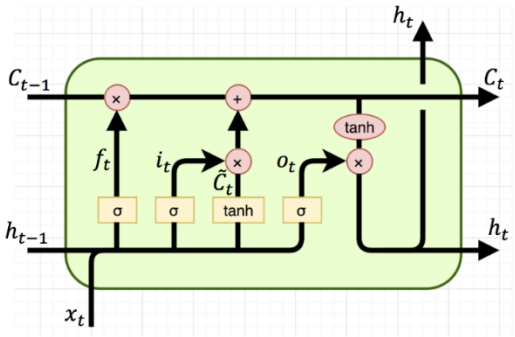

Figure 34: Representation of LSTM

The cell state is almost like a conveyor belt carrying linear context and interactions down to the whole network. LSTMs contain information that extends beyond its normal flow in a gated cell. The information can be stored, written, or read from a cell in the same way that the data can in a computer memory. The cell gets decisions about what to keep, what to read, what to write and what to delete, through gates that open and close. Gates are a way to optionally let information through. But unlike digital storage at computers, these gates are analog, implemented by multiplying by element of sigmoid functions, in the range of 0-1. The use of analog over digital gates has the advantage that they are different and therefore suitable for back propagation.

These gates operate based on the signals they receive, and similar to the neurons of a neural network, block or allow information to pass through based on importance, which they filter with their own set of weights. These weights, as well as the weights

that adjust the input and hidden states, change through the process of back propagation. Which means that the cells learn when to leave one information to pass on, when to block or delete it through iterative process of predictions, error retrieval and updating weights through the gradient descent.

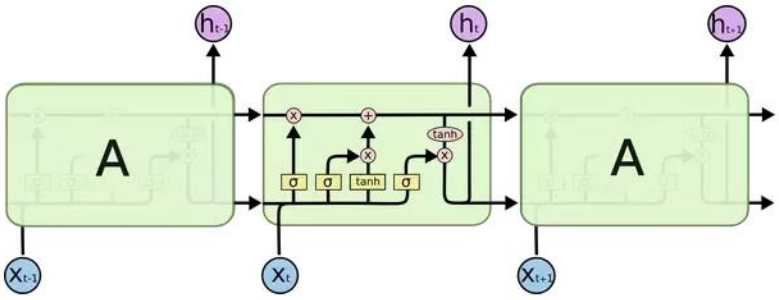

Figure 35: LSTM pipeline example

An LSTM has three of these gates, to protect and control the cell state and two activation functions.

Tanh activation

The tanh activation is used to help regulate the values flowing through the network. The tanh function squishes values to always be between -1 and 1.

When vectors are flowing through a neural network, it undergoes many transformations due to various math operations.

Sigmoid activation

Gates contains sigmoid activations. A sigmoid activation is similar to the tanh activation. Instead of squishing values between -1 and 1, it squishes values between 0 and 1. That is helpful to update or forget data because any number getting multiplied by 0 is 0, causing values to disappear or be “forgotten.” Any number multiplied by 1 is the same value therefore that value stays the same or is “kept.” The network can learn which data is not important therefore can be forgotten or which data is important to keep.

Forget gate

First, we have the forget gate. This gate decides what information should be thrown away or kept. Information from the previous hidden state and information from the current input is passed through the sigmoid function. Values come out between 0 and 1. The closer to 0 means to forget, and the closer to 1 means to keep.

Input Gate

To update the cell state, we have the input gate. First, we pass the previous hidden state and current input into a sigmoid function. That decides which values will be updated by transforming the values to be between 0 and 1. 0 means not important, and 1 means important. You also pass the hidden state and current input into the tanh function to squish values between -1 and 1 to help regulate the network. Then you multiply the tanh output with the sigmoid output. The sigmoid output will decide which information is important to keep from the tanh output.

Cell State

Now we should have enough information to calculate the cell state. First, the cell state gets pointwise multiplied by the forget vector. This has a possibility of dropping values in the cell state if it gets multiplied by values near 0. Then we take the output from the input gate and do a pointwise addition which updates the cell state to new values that the neural network finds relevant. That gives us our new cell state.

Output Gate

Last, we have the output gate. The output gate decides what the next hidden state should be. The hidden state contains information on previous inputs and is also used for predictions. Thus, at first, we should pass the previous hidden state and the current input into a sigmoid function. Then we pass the newly modified cell state to the tanh function. We multiply the tanh output with the sigmoid output to decide what information the hidden state should carry. The output is the hidden state. The new cell state and the new hidden is then carried over to the next time step.

3.5.4 GRUs (Gated Recurrent Unit)

Introduced in 2014, GRU (Gated Recurrent Unit) aims to solve the vanishing gradient problem which comes with a standard recurrent neural network [67]. A GRU is typically an LSTM without the output gate, which means that all the contents of the memory cell are recorded in the wider network over time. GRU's use the hidden state to transfer information.

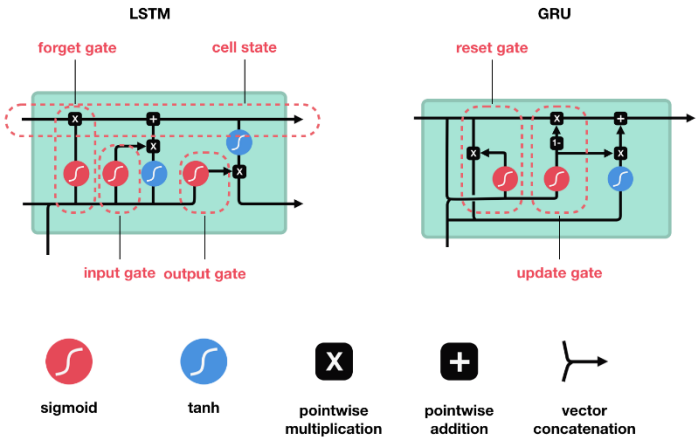

Figure 36: Differences between LSTM & GRU

Update Gate

The update gate acts like the forget and input gate of an LSTM. It decides what information to throw away and what new information to add.

Reset Gate

The reset gate is used to decide how much past information to forget.

GRU's has fewer tensor operations, therefore, they are a little speedier to train than LSTM's. Researchers and engineers usually try both to determine which one works better for their use case.

4 Proposed methodology

In the previous chapter, three of the most important and widely used Artificial Neural Networks (ANNs) for Machine Translation were analyzed. The key concepts and ideas on which these networks are based were, also, mentioned in order to have a better understanding for their usage and their advantages. Moreover, we mentioned and noticed the impact that Deep Learning in the modern areas of Sentiment Analysis and Machine Translation has. The present thesis work, as highlighted in Section 1, is the subject of study the impact that Machine Translation and more general the translation of a text has in the extraction of sentiment from it. The process of exploiting and utilizing insights and emotions that we can retrieve and analyze from this data played an important role in the field of Sentiment Analysis, which is presented in Section 2.

In this chapter we provide more information on how the integration and implementation of Deep Learning techniques has been achieved analysis and translation over the raw text. Therefore, is analyzed the methodology followed in order to perform Sentiment Analysis and Machine Translation based on Deep Learning, using data from IMDB movie reviews. In addition, in the past years, the performance of Machine Translation systems has steadily improved. Open access solutions (e.g. Google Translate, Bing Translator, Yandex Translate) offer more and more accurate translations for frequently and widely known used languages. These open services and APIs will also be utilized in order to perform Machine Translation and their insights and results will be exploited in terms of their effectiveness and accuracy.

A general definition of the problem is given will follow below and then a brief reference to key concepts of the main techniques, that are being applied in the pipeline and tools of the proposed system, is highlighted. Finally, all stages, models, techniques and the metrics that are being used are described in detail.

4.1 Implementation

Initially, in this thesis we want to create a Sentiment Analysis model through Deep Learning techniques that will be trained in movie reviews. Then, using Deep Neural Networks, like CNNs and RNNs, we try to perform Machine Translation in the already analyzed reviews and get the same reviews in a target language. Finally, by performing the same Deep Learning Techniques in the target language as we did in the source language of the reviews, we seek to define and observe the impact and the differences that the Machine Translation task introduced to the reviews that analyzed for their sentiment in the first step. Thus, we seek to perform a Multilingual Sentiment Analysis

task. The possible ways of performing multilingual sentiment analysis are machine translation (MT) and building language specific classification system. For example, in MT a classifier is trained using the dataset in the English language and for testing, the data instances are translated into English from another language. Whereas in language specific classification systems classifiers are trained and tested on the same language, this approach is called native classification.

Sentiment Analysis is one the most significant NLP fields for the business and research community. The amount of research work on sentiment analysis is growing explosively. However, the majority of research efforts are devoted to English-language data, while a great share of information is available in other languages. Based on this we seek to investigate the impact that Machine Translation will have in the Sentiment of movie reviews, written in English Language. Our approach and work bases on the translation of movie reviews for which we have already performed the Sentiment Analysis task. After the translation we perform again Sentiment Analysis on the translated texts with the same models and techniques that we used in the original reviews' language, in order to observe and record the differences between the performed Sentiment Analysis in the source language and in the target language.

Thus, as described above we seek to perform Sentiment Analysis and Machine Translation based on Deep Neural Networks. More specifically, we analyzed and created models for Simple Artificial Neural Networks, CNNs and types of RNNs, like LSTMs and GRUs.

As for the Sentiment Analysis task, for the purposes of this Thesis, we followed a Deep Learning approach in which the Text Classification model architecture consist of the following components connected in sequence:

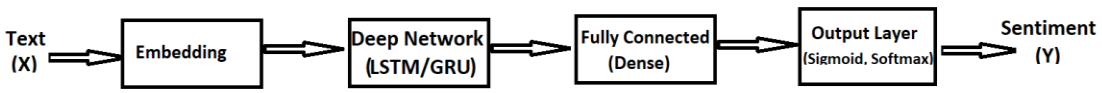

Figure 37: Deep Learning Text Classification model architecture

In order to be more specific and detailed:

- ***Preprocess***: Before performing any Deep Learning technique we must perform some steps of preprocessing in our raw data. The reviews that will be analyzed contain special characters, punctuations, html tags and generally whatever we can noticed in a raw text. For the purposes of this task we filter and remove punctuations, numbers, stopwords, normalize all Unicode characters to ASCII, normalize the case to lowercase and any other special character that will be recognized.
- ***Embedding***: The next step that we should do is to create a vocabulary. A part of preparing text for sentiment analysis involves defining and tailoring the vocabulary of words supported by the model. In order to achieve a better representation of the words that will used word embedding models, like Word2Vec and the pretrained GloVe. It has been shown in the literature [68],

that especially for small labelled datasets, it is beneficial to train a pretrain word embeddings on a large unlabelled corpora using an unsupervised task. Word Embedding is a representation of text where words that have the same meaning have a similar representation. In other words, it represents words in a coordinate system where related words, based on a corpus of relationships, are placed closer together. In the Deep Learning frameworks such as TensorFlow, Keras, that we use in our Thesis, this part is usually handled by an embedding layer which stores a lookup table to map the words represented by numeric indexes to their dense vector representations.

- ***Deep Network***: Deep network takes the sequence of embedding vectors as input and converts them to a compressed representation. The compressed representation effectively captures all the information in the sequence of words in the text. In our approach this deep network part, for the Sentiment Analysis task, is a type of an RNN like LSTM or GRU, or a CNN. The Dropout is added to overcome the tendency to overfit, a very common problem with RNN based networks.
- ***Fully Connected Layer:*** The fully connected layer, like Dense, Flatten and GlobalAveragePooling1D that are used in this Thesis, takes the deep representation from the CNN/LSTM/GRU and transforms it into the final output classes or class scores. This component is comprised of fully connected layers along with batch normalization and optionally dropout layers for regularization.
- ***Output Layer:*** Based on the problem at hand, this layer can have either **Sigmoid** for binary classification or **Softmax** for both binary and multi classification output. Our models contain sigmoid function in their Output Layers since we have a binary classification problem. Based on the binary nature of our problem, another parameter is defined. When we compile our trained neural model, we make use of ***loss='binary_crossentropy'*** in order to value and maximize the performance of our model.

Recent advances in Neural Machine Translation have proven that, if we have a strong sequential model, we are likely to achieve the best results, by maximizing the probability of correct translation, given an input sequence. These models make use of Recurrent Neural Networks, which encode the length of the variable input into unstable dimensions - vector and use its encoding to then decode the desired output sequence.

Thus, Neural Machine Translation (NMT) models are often based on the seq2seq architecture. In our implementation, the seq2seq architecture is an encoder-decoder architecture which consists of two LSTM networks: the encoder LSTM and the decoder LSTM. The input to the encoder LSTM is the sentence in the original language, while the input to the decoder LSTM is the sentence in the translated language with a start-of-sentence token. The output is the actual target sentence with an end-of-sentence token.

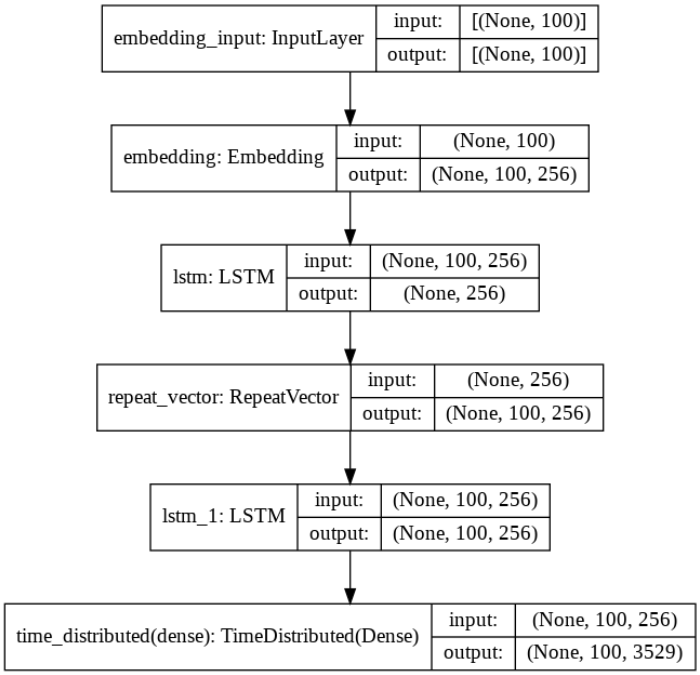

Figure 38: Encoder-Decoder architecture based on LSTMs

4.2 Data Collection

Sentiment Analysis

Implementation of Sentiment Analysis took place in the imdb_reviews dataset[3] which consists of 100,000 reviews with 5 attributes each:

1. ID: review's serial number

2. Type: indicating whether this criticism will be included in the training set or the test set.

3. review: this is about the body of criticism, that is, the user's opinion of the movie.

4. label: deciding whether the criticism is in favor of (pos), against the film (neg) or not unsupporting to positive or negative.

5. file: is the source (file) from which the review has derived.

The comments are categorized as either positive or negative. It is provided for this purpose a set of data from a .csv file. Finally, we select the best recorded parameter combination in order to implement the chosen algorithm.

Neural Machine Translation

[3] https://www.kaggle.com/utathya/imdb-review-dataset

To train our Machine Translation models we based on language datasets provided by http://www.manythings.org/anki/. These datasets contain several language translations pairs and are comprised of phrases and their counterparts in the format:

['i need to stop' 'ich muss aufhoren']

Moreover, these datasets were used in order to create the needed vocabularies and word embeddings for all the languages that we used for the purposes of this Thesis.

4.3 Evaluation Measures (P,R,F1, BLEU)

This section describes the measures that will be used to evaluate our Sentiment Analysis and Machine Translation systems. Measures of evaluating the efficiency of Machine Learning systems go beyond concepts of Precision, Recall and F1-measure. Suppose we have a request for information as otherwise stated a query *q* and *R* the total of relevant records according to this query.

Supposing that we implement a method S, which is checked for its efficiency, processes the information need and produces a total of records *A*. Also, assuming that $|A|$ be the number of objects in total *A* and that $|Ra|$ is the number of records in the intersection of the totals *R* and *A*, the evaluation measures of Recall and Precision can be defined as follows:

Precision, is the fraction of the found records (A) of which they are relative, that is:

$$p = \frac{|Ra|}{|A|}$$

Recall, is the fraction of the found documents (R) of which they are relative, that is:

$$r = \frac{|Ra|}{|R|}$$

F-Measure

The F-measure or F-score is one of the most commonly used measures in NLP and Machine Learning tasks, because it provides a way to combine both precision and recall into a single measure that captures both properties. The system is defined as the weighted harmonic mean of its precision and recall, that is:

$$F = \frac{1}{a\frac{1}{p} + (1-a)\frac{1}{r}}$$

, where *p* is the Precision and *r* the Recall of the system.

F-Measure might be a better measure to use if we need to seek a balance between Precision and Recall and there is an uneven class distribution (large number of Actual Negatives).

BLEU Score

In 2002, Kishore Papineni, et al. [60] proposed a new metric that will accelerate the MT R&D cycle by allowing researchers to rapidly home in on effective modeling ideas. The Bilingual Evaluation Understudy Score, or BLEU for short, is a metric for evaluating a generated sentence to a reference sentence.

The BLEU score is a number between zero and one that measures the similarity of the machine-translated text to a set of high-quality reference translations. A higher match degree indicates a higher degree of similarity with the reference translation, and higher score. Intelligibility and grammatical correctness are not taken into account. A value of 0 means that the machine-translated output has no overlap with the reference translation (low quality) while a value of 1 means there is perfect overlap with the reference translations (high quality).

The first step is to count the occurrences of each unigram in the reference and the candidate. Note that the BLEU metric is case-sensitive. The comparison is made regardless of word order.

The counting of matching n-grams is modified to ensure that it takes the occurrence of the words in the reference text into account, not rewarding a candidate translation that generates an abundance of reasonable words. The score is for comparing sentences, but a modified version that normalizes n-grams by their occurrence is also proposed for better scoring blocks of multiple sentences.

5 Experimental Results

Experiments were based on implementing Natural Language Processing tasks with TensorFlow 2.0[4] and Keras[5]. Moreover, the implementation of our code, for both Sentiment Analysis and Neural Machine Translation tasks, performed on Google Colab[6] online platform in order to make use of the GPU option that is available. As shown in the above images, where run times from a flat and simple Artificial Neural Network that is trained on GPU and on CPU, by using GPU is much faster.

```
Epoch 1/10
75000/75000 [==============================] - 493s 7ms/sample - loss: 0.4731 - accuracy: 0.7498 - val_loss: 0.4709 - val_accur
acy: 0.7490
Epoch 2/10
75000/75000 [==============================] - 502s 7ms/sample - loss: 0.3959 - accuracy: 0.7817 - val_loss: 0.4917 - val_accur
acy: 0.7570
Epoch 3/10
75000/75000 [==============================] - 471s 6ms/sample - loss: 0.3284 - accuracy: 0.8305 - val_loss: 0.5343 - val_accur
acy: 0.7420
Epoch 4/10
75000/75000 [==============================] - 489s 7ms/sample - loss: 0.2511 - accuracy: 0.8822 - val_loss: 0.7030 - val_accur
acy: 0.7407
Epoch 5/10
75000/75000 [==============================] - 517s 7ms/sample - loss: 0.1989 - accuracy: 0.9143 - val_loss: 0.8622 - val_accur
acy: 0.7339
Epoch 6/10
75000/75000 [==============================] - 469s 6ms/sample - loss: 0.1696 - accuracy: 0.9314 - val_loss: 1.0164 - val_accur
acy: 0.7325
Epoch 7/10
75000/75000 [==============================] - 473s 6ms/sample - loss: 0.1571 - accuracy: 0.9392 - val_loss: 1.1671 - val_accur
acy: 0.7364
Epoch 8/10
75000/75000 [==============================] - 481s 6ms/sample - loss: 0.1496 - accuracy: 0.9431 - val_loss: 1.1358 - val_accur
acy: 0.7278
Epoch 9/10
75000/75000 [==============================] - 490s 7ms/sample - loss: 0.1442 - accuracy: 0.9458 - val_loss: 1.3174 - val_accur
acy: 0.7353
Epoch 10/10
75000/75000 [==============================] - 471s 6ms/sample - loss: 0.1415 - accuracy: 0.9472 - val_loss: 1.2295 - val_accur
acy: 0.7247
25000/25000 [==============================] - ETA: 0s - loss: 1.2295 - accuracy: 0.72 - 22s 892us/sample - loss: 1.2295 - accu
racy: 0.7247
```

Figure 39: Train NN Model without using GPU

[4] https://www.tensorflow.org/
[5] https://keras.io/
[6] https://colab.research.google.com/

```
Train on 75000 samples, validate on 25000 samples
Epoch 1/10
75000/75000 [==============================] - 48s 642us/sample - loss: 0.5202 - accuracy: 0.7500 - val_loss: 0.5020 - val_accuracy: 0.7490
Epoch 2/10
75000/75000 [==============================] - 49s 647us/sample - loss: 0.4804 - accuracy: 0.7503 - val_loss: 0.5060 - val_accuracy: 0.7490
Epoch 3/10
75000/75000 [==============================] - 48s 643us/sample - loss: 0.4669 - accuracy: 0.7505 - val_loss: 0.5087 - val_accuracy: 0.7490
Epoch 4/10
75000/75000 [==============================] - 48s 635us/sample - loss: 0.4574 - accuracy: 0.7505 - val_loss: 0.5139 - val_accuracy: 0.7490
Epoch 5/10
75000/75000 [==============================] - 48s 633us/sample - loss: 0.4498 - accuracy: 0.7506 - val_loss: 0.5186 - val_accuracy: 0.7490
Epoch 6/10
75000/75000 [==============================] - 47s 629us/sample - loss: 0.4446 - accuracy: 0.7505 - val_loss: 0.5278 - val_accuracy: 0.7490
Epoch 7/10
75000/75000 [==============================] - 47s 627us/sample - loss: 0.4387 - accuracy: 0.7507 - val_loss: 0.5309 - val_accuracy: 0.7490
Epoch 8/10
75000/75000 [==============================] - 47s 626us/sample - loss: 0.4342 - accuracy: 0.7507 - val_loss: 0.5340 - val_accuracy: 0.7491
Epoch 9/10
75000/75000 [==============================] - 47s 628us/sample - loss: 0.4307 - accuracy: 0.7507 - val_loss: 0.5543 - val_accuracy: 0.7490
Epoch 10/10
75000/75000 [==============================] - 47s 633us/sample - loss: 0.4271 - accuracy: 0.7508 - val_loss: 0.5550 - val_accuracy: 0.7490
25000/25000 [==============================] - 11s 433us/sample - loss: 0.5550 - accuracy: 0.7490
```

Figure 40: Train NN Model using GPU

At first basic tools and techniques in order to clean our raw reviews were implemented. Raw reviews contain punctuations, brackets, and a few HTML tags as well, so a preprocess step is essential in order to provide our system with machine-readable clean data.

Afterwards, techniques like Tokenization and Padding of text were implemented, in order to produce data structures that could be used later from our Neural Networks.

Creating vocabularies based on Word Embeddings that allow us to map words into vectors was the next step. Therefore, words of similar semantics given vectors pointing in a similar direction, giving us a mathematical model for their meaning. These vector representations could then be fed into a deep neural network for classification.

As mentioned in other sections Text Classification and Machine Translation tasks are sequence problems. Thus, we decide to utilize sequence neural models, in order to deepen our understanding of sentiment and translation in text by not just looking at words in isolation, but also, how their meanings change when they qualify one another.

5.1 Data Preprocessing and Word Embeddings

5.1.1 Text Preprocessing

Basic techniques like filtering and removing punctuations, HTML tags, stopwords and transform all letters into lowercase were some of the first scripts of our code. NTLK library is one of the most usable libraries in NLP for text preprocessing. It provides us with packages and other libraries that help us to implement the above filters and removals. For example, by download and install to our system the ***nltk.download('punkt')*** and ***nltk.download(stopwords)*** libraries we are able to use them in order to remove stopwrods and punctuations from our given raw text.

Next, for converting labels and sentiments of a text in a machine-readable way, we need to convert our labels into digits. Since we only have two labels in the output, “positive” and “negative”, we can simply convert them into integers by replacing “positive” with digit 1 and “negative” with digit 0.

In addition, our dataset contains 50000 “unsup” labeled reviews, which are neither positive or negative, for that reason these records are not taken into consideration and are deleted from the final process.

5.1.2 Preparing the Embedding Layer based on Word2Vec model

As mentioned in Section 2.7, word embeddings are a cornerstone for performing Deep Learning (DL) techniques in NLP tasks. Discrete words are mapped to vectors of continuous numbers. This is useful when working with natural language problems with neural networks and deep learning models are, we require numbers as input. These can easily be understood, also, if we observe the different layers in our trained Neural Networks. As described, also, in Section 4, for preparing the data for Deep Learning we have to start our implementation by preparing the Embedding Layers and organized the vocabulary based on Word Embeddings models. By using Word Embeddings, we can have a distributed representation of words where different words that have a similar meaning (based on their usage) also have a similar representation. Moreover, the embeddings will be feed into the first layer of our DNN, which requires that the input data be integer encoded so that each word is represented by a unique integer.

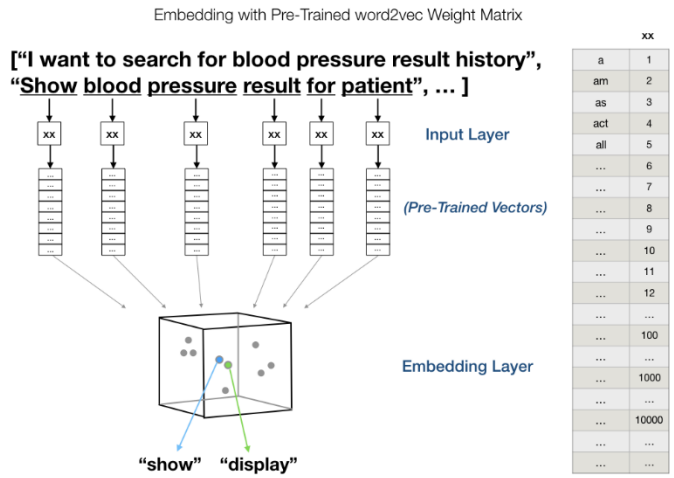

Figure 41: Embedding with Pre-Trained Word2Vec Weight Matrix

Keras library provides a convenient way to convert positive integer representations of words into a word embedding by an Embedding layer. As a first approach to our Word Embedding implementation, we utilize the Word2Vec model and more specific the CBOW approach, where the Embedding layer takes the integer-encoded vocabulary and looks up the embedding vector for each word-index. These vectors are learned as the model trains and add a dimension to the output array. To achieve this, we use the Gensim Python library for topic modeling. Gensim[7] offers an implementation of the word2vec algorithm, developed at Google for the fast training of word embedding representations from text documents.

[7] https://radimrehurek.com/gensim/models/word2vec.html

```
import gensim

embedding_dim=100

model = gensim.models.Word2Vec(sentences=review_lines, size=embedding_dim, window=4, workers=5, min_count=1)
words=list(model.wv.vocab)
```

Figure 42: Using gensim library in order to use Word2Vec

```
model.wv.most_similar('bad')
```

```
[('terrible', 0.7314393520355225),
 ('horrible', 0.717605710029602),
 ('awful', 0.7067137360572815),
 ('sucks', 0.6632874011993408),
 ('reak', 0.6631817817687988),
 ('good', 0.6618109345436096),
 ('lousy', 0.6558427810668945),
 ('atrocious', 0.6284555196762085),
 ('crappy', 0.6230224370956421),
 ('suck', 0.6091160774230957)]
```

Figure 43: The most similar words for word "bad"

In order to visualize our Word Embeddings, we will upload our .tsv files to the embedding projector. By using the Embedding Projector[8] provided by Tensorflow we can have a visual representation of our embeddings. Choosing a specific word its neighbor words can also be shown. By default, the Embedding Projector uses PCA technique for reducing the dimensionality of the data and virtualizing word embeddings and their distance.

The embeddings we have trained will now be displayed. Through this dashboard we can search for words to find their closest neighbors. For example, in the above image we searched for word "bad". Among to its neighbors are words like “terrible” and “horrible”.

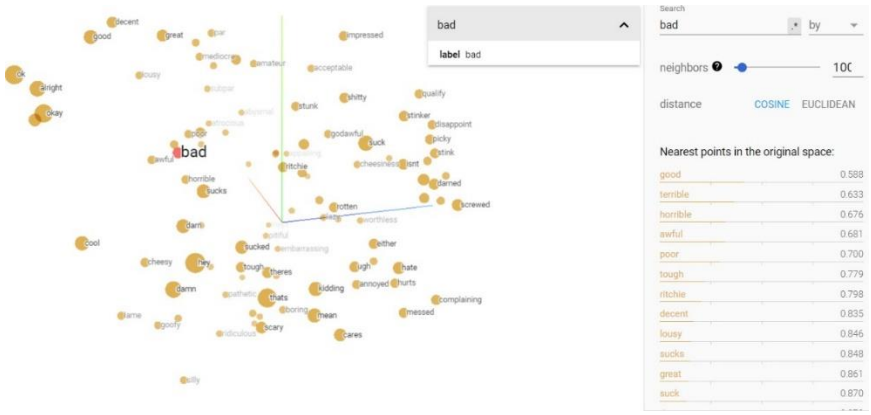

Figure 44: PCA representation of word "bad" and its neighbors using Word2Vec

[8] http://projector.tensorflow.org/

5.1.3 Preparing the Embedding Layer based on GloVe

Thus, we will use another pre-trained vectors from another popular model, GloVe, in order to create our word embeddings feature matrix. In the following script we load the GloVe word embeddings and create a vocabulary that will contain words as keys and their corresponding embedding list as values.

```
from numpy import array
from numpy import asarray
from numpy import zeros

embeddings_dictionary = dict()
glove_file = open('/content/drive/My Drive/NMT/glove.6B.100d.txt', encoding="utf8")

for line in glove_file:
    records = line.split()
    word = records[0]
    vector_dimensions = asarray(records[1:], dtype='float32')
    embeddings_dictionary [word] = vector_dimensions
glove_file.close()
```

Figure 45: Sample code for using GloVe

Finally, we will create an embedding matrix where each row number will correspond to the index of the word in the corpus. The matrix will have 100 columns where each column will contain the GloVe word embeddings for the words in our corpus.

```
embedding_matrix = zeros((vocab_size, 100))
for word, index in tokenizer.word_index.items():
    embedding_vector = embeddings_dictionary.get(word)
    if embedding_vector is not None:
        embedding_matrix[index] = embedding_vector
```

Figure 46: Creating embedding matrix sample code

Words' embedding matrixes that derived from the above approach will be used as weights in the embedding layers of our Neural Network models. These layers will have an input length of 100 and the output vector dimension will also be 100.

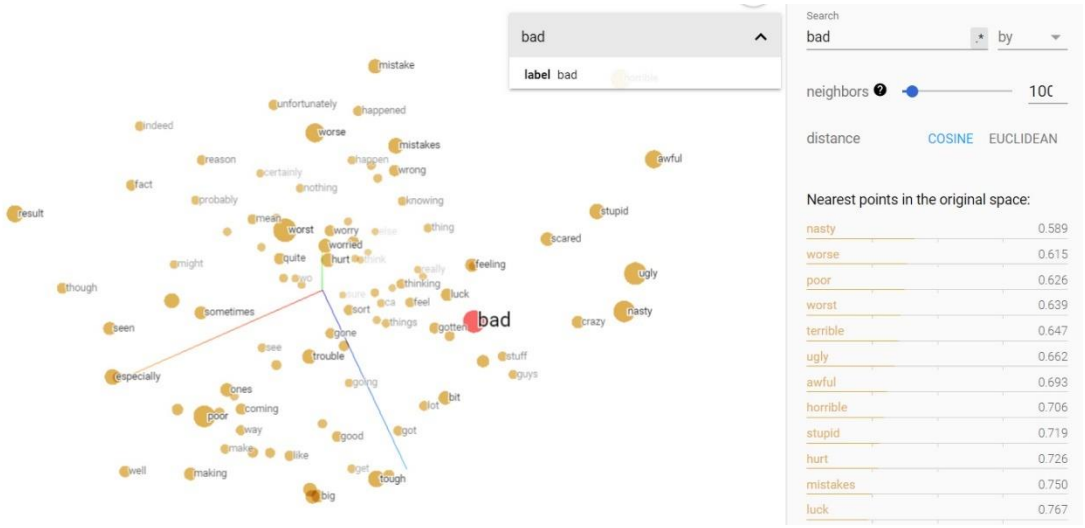

Figure 47: PCA representation of word "bad" and its neighbors using GloVe

Then, in order to create any Neural Network classifier that leverages pretrained embeddings, we can reuse the same models as we did in the Word2Vec trained embeddings method but pass in a custom initializer that initializes the embeddings with our pretrained embedding matrix.

5.1.4 Tokenization and Padding

The next step is to convert the word embedding into tokenized vector. Recall that the review documents are integer encoded prior to passing them to the Embedding layer. The integer maps to the index of a specific vector in the embedding layer. In the word-to-index vocabulary, each word in the corpus is used as a key, while a corresponding unique index is used as the value for the key. Therefore, it is important that we lay the vectors out in the Embedding layer such that the encoded words map to the correct vector.

Thus, we make use of Keras "Tokenizer" and convert the texts in train and test datasets to sequences so that they can be passed through embedding matrices.

```
tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(X) #by calling tokenizer.fit_on_texts will generate the word index and we'll initialize the tokenizer

word_index = tokenizer.word_index #this returns all words that the tokenizer saw when tokenizing the sentences

sequences = tokenizer.texts_to_sequences(review_lines)
X_train = tokenizer.texts_to_sequences(X_train)
X_test = tokenizer.texts_to_sequences(X_test)

# Adding 1 because of reserved 0 index
vocab_size = len(tokenizer.word_index) + 1
```

Figure 48: Keras Tokenizer sample code

***Tokenizer(num_words=5000)*:** initializes the tokenizer and sets the maximum number of words that will be passed in our vocabulary in 5000

tokenizer.fit_on_texts(X) function tokenizes the sentences of the whole dataset and generates the word index(vocabulary) that will be used to train and test our models.

tokenizer.word_index function returns all words that the tokenizer found when tokenizing the sentences.

tokenizer.text_to_sequences(X_train) function transforms each sentence into sequences. Represents, in a machine-readable way, every word with the corresponding word's vocabulary position.

```
max_len = 100

X_train_padded = pad_sequences(X_train, padding='post', maxlen=max_len)
X_test_padded = pad_sequences(X_test, padding='post', maxlen=max_len)
```

Figure 49: Sample code for performing padding using Keras

After creating the word-to-index vocabulary, we must process the length of the sentences. Each sentence has a different length, but as inputs to our Neural Network they have to be the same size. In order to achieve this, we use the ***pad_sequences*** function. By padding the inputs, we decide the maximum length of words in a sentence, then zero pads the rest, if the input length is shorter than the designated length. In the case where it exceeds the maximum length, then it will also truncate either from the beginning or from the end. We set the maximum size of each sentence to 100. Sentences with size greater than 100 will be truncated to 100 and those that have length less than 100, 0s will be added at the end of the sentence, as we have chosen *padding='post'*, until it reaches the max length.

5.2 Sentiment Analysis based on DNNs

Having carried out the preparation of our Embedding Layers we seek to perform Sentiment Analysis using Keras deep learning library. By utilize the trained Word2Vec & GloVe models we are able to use them as pre-trained embeddings to our Deep Neural Networks for the Sentiment Analysis tasks and observe the different results that may have.

We used three different types of deep neural networks to classify public sentiment about different movies. A simple Neural Network, a CNN and two different types of RNNs (LSTMs & GRUs). For, all the models that we trained we also had finally fit our neural network with early stopping and checkpoint so that we can save the best performing weights on validation accuracy.

Furthermore, our models were trained, and their results were recorded for two different sizes of the provided dataset. At first, we recorder the results for the whole dataset of negative and positive reviews, 50000 records (25000 positive and 25000 negative reviews). We split these records in 80% train set (40000 records) and 20% (10000 records) test set and performed our Neural Models.

Afterwards, due to the limitations of the open source providers' Machine Translation APIs we achieved to translate only 4251 total records, 2000 negative (from record 0 to 1999) and 2251 positive reviews (from record 12500 to 14751). Based on this, Sentiment Analysis task on translated data performed only on this amount of translated reviews. Thus, we utilize again the same models we have applied the same neural network models to this specific range of records in order to have a right

comparison between the Sentiment Analysis on source language and Sentiment Analysis on target languages.

Text Classification with Simple Neural Network

The first deep learning model that we developed is a simple deep neural network. In the script below, we create a Sequential() model. Next, we create our embedding layer. For text or sequence problems, the Embedding layer takes a 2D tensor of integers, of shape *(samples, sequence_length),* where each entry is a sequence of integers. In our approach the Embedding layer is initialized with the weights of embedding matrixes that we have created while we prepared the Embedding layer and will learn an embedding for all of the words in the training dataset during training of the model.

The embedding layer will have an input length of 100, the output vector dimension will also be 100. The vocabulary size for the dataset of 4251 records was of 32687 words. Since we are not training our own embeddings and using the GloVe or Word2Vec embedding, we set trainable to False and in the weights attribute we pass our own embedding matrix.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_4 (Embedding)      (None, 100, 100)          3268700
_________________________________________________________________
flatten_1 (Flatten)          (None, 10000)             0
_________________________________________________________________
dense_8 (Dense)              (None, 6)                 60006
_________________________________________________________________
dense_9 (Dense)              (None, 1)                 7
=================================================================
Total params: 3,328,713
Trainable params: 60,013
Non-trainable params: 3,268,700
```

Figure 50: Simple Artificial Neural Network architecture

Since there are 32687 words in our corpus and each word is represented as a 100-dimensional vector, the number of trainable parameters will be 32687x100 in the embedding layer. In the flattening layer, we simply multiply rows and column. Finally, in the first dense layer the number of parameters are 60000, from the flattening layer, and each of the 6 Dense Nodes has 10000 parameters plus 1 each of them for the bias parameter, and the second Dense layer has 7 nodes, 1 for each of the nodes of the previous layer and 1 for the bias parameter, for a total of 60013 parameters. Moreover, model's total params are 3,328,713, but trainable params = 60013. Since the model uses pre-trained word embedding it has very few trainable params and hence should train faster and only in 4s.

Respectively, we calculate the total number of parameters to be trained and in the case of most records in the whole examined dataset as well as for all the Neural Networks that we train below.

Text Classification with Convolutional Neural Network

As analyzed in a previous section (3.5.1) Convolutional Neural Networks (CNNs) have been found to work well with text data as well. Though text data is one-dimensional, due to the fact that they represented in word vectors, we can use 1D convolutional neural networks to extract features from our data.

In order to feed to a CNN, we have to not only feed each word vector to the model, but also in a sequence which matches the original review.

By adding GlobalMaxPooling1D layer the pooling layer will extract the maximum value from each filter, and the output dimension will be a just 1-dimensional vector with length as same as the number of filters we applied.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 100, 100)          10124600
_________________________________________________________________
conv1d (Conv1D)              (None, 96, 128)           64128
_________________________________________________________________
global_average_pooling1d (Gl (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 774
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 7
=================================================================
Total params: 10,189,509
Trainable params: 64,909
Non-trainable params: 10,124,600
```

Figure 51: CNN model architecture

In the above model we create a sequential model, followed by an embedding layer. This step is similar to what we had done earlier in the Simple Neural Network. Next, we create a one-dimensional convolutional layer with 128 features, or kernels. The kernel size is 5 and the activation function used is relu. We use relu in place of tahn function since they are very good alternatives of each other. Next, we add a global max pooling layer to reduce feature size. Finally, as before we add two Dense Layers, the first one using relu activation and the final, which also does the classification with sigmoid activation. Similarly, a softmax classifier could be used for a multiclass classification task.

Text Classification with LSTMs

Recurrent neural network is a type of neural networks that is proven to work well with sequence data. Since text is actually a sequence of words, a recurrent neural network is an automatic choice to solve text-related problems. In this section, we will use an LSTM (Long Short-Term Memory network) which is a variant of RNN, to solve sentiment classification problem.

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 100, 100)          10124600
_________________________________________________________________
lstm (LSTM)                  (None, 32)                17024
_________________________________________________________________
dense_4 (Dense)              (None, 6)                 198
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 7
=================================================================
Total params: 10,141,829
Trainable params: 17,229
Non-trainable params: 10,124,600
```

Figure 52: LSTM model architecture

As we can see in the above model all layers except the LSTM layer are the same with those we created in the two previous neural models. The difference in this model is that we create an LSTM layer with 32 nodes which provides us 17024 features. The rest of the code is same as it was for the CNN.

Text Classification with GRUs

As mentioned in previous sections GRUs eliminate the problem of exploding and diminishing gradient problem as effectively as the LSTM networks does with lesser computational overhead. The GRU also converges to the solution faster than LSTM.

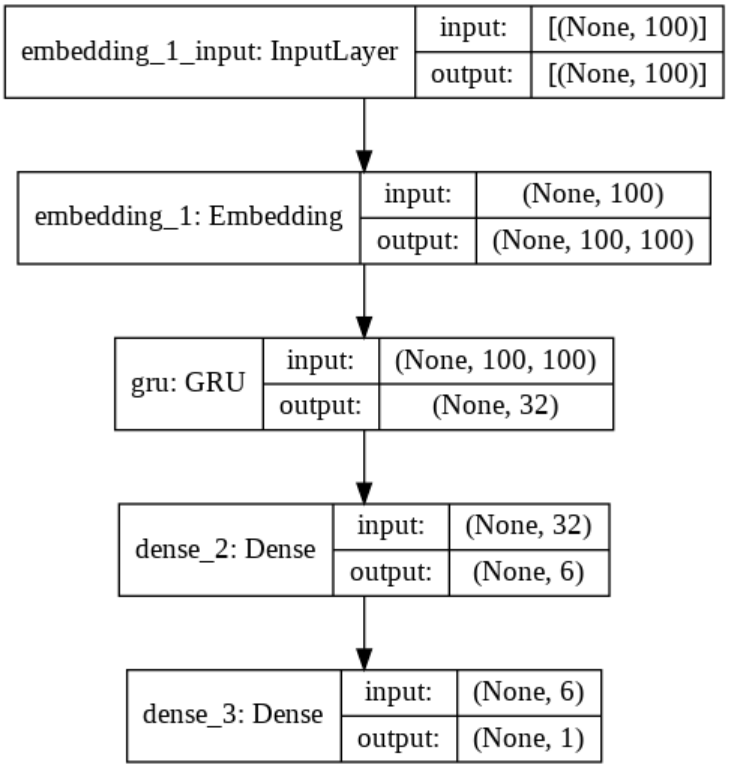

Figure 53: GRU architecture

We used only single layered GRUs even though it is possible to stack them, or to create a Bidirectional network. At the end of the hidden layer we get the representation of the entire sequence which can be used as input to linear model or classifier. We used

sigmoid function for binary classification in the last dense layer and relu activation function in the first Dense layer with 6 Nodes.

For all the above models we decide to train them in 10 epochs. Neural Networks seems to overfit in small datasets, so 10 epochs are a sufficient number to train our models. The above image represents the Accuracy and Loss curves of our neural models. As a general outcome we can understand that our models overfit in our data. In ANN for example we probably only need 3 or 4 epochs. As for both RNNs models we can observe that even from the 1 epoch we achieve the best val_accuracy, while CNN achieves this in the 2nd epoch. At the end of every training sample, we can see that there is a little bit overfitting. Moreover, in LSTM and GRU the accuracy and loss show peak performance and then show a decrease during training and testing. This, also, shows that these particular models may be need a little tuning and optimizing. Change the number of their nodes in GRU and LSTM hidden layers or change the learning rate may be some changes that we can perform in our future work.

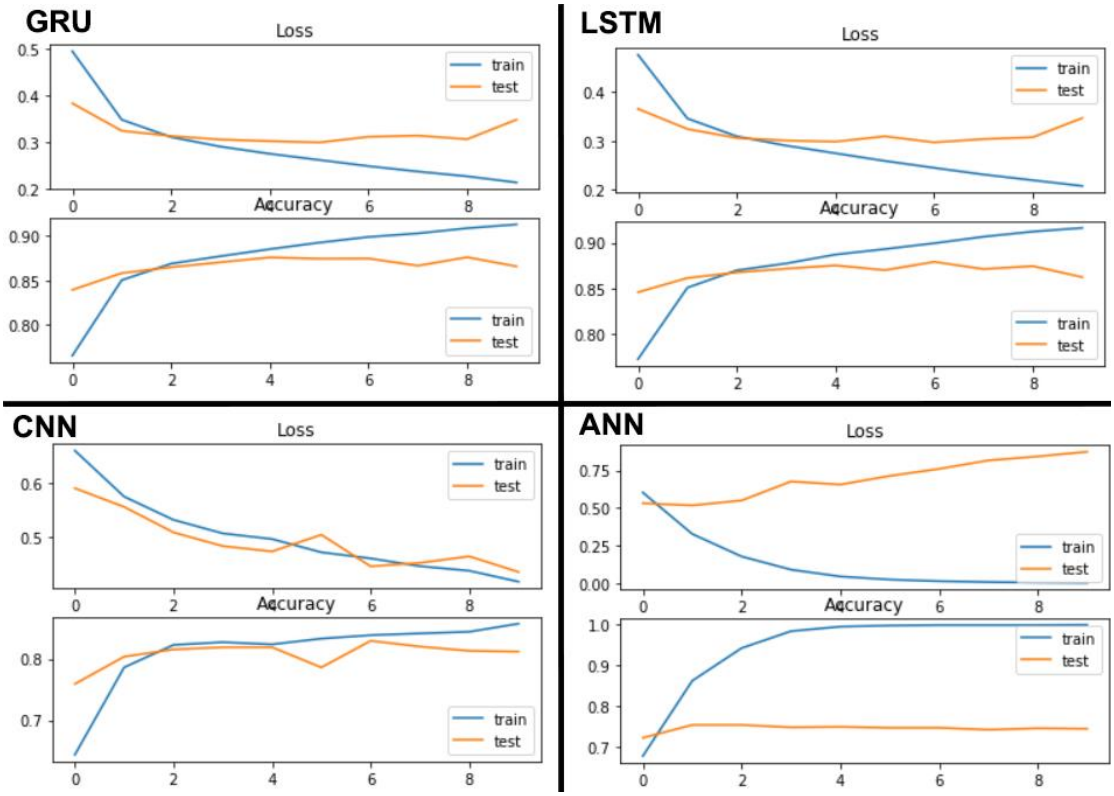

Figure 54: Accuracy and Loss curves

Finally, while losses decreasing, it is decreasing in a very small way for ANN. The reason why this is happening of course is just because we are working on sub words, because we are training on things that it is very hard to pull semantics and meaning out of them. So, when we start putting this together with CNN and RNNs (LSTM and GRU) and learning things where sequences are important, then we see an improvement.

Scenario 1 (50000 records – 25000 positive & 25000 negative) using Word2Vec

	Simple ANN	CNN	LSTM	GRU
Accuracy:	0.763700	0.873600	0.865700	**0.874000**
Precision:	0.717413	0.845634	0.888147	**0.891771**
Recall:	0.861476	**0.910617**	0.833367	0.848129
F1 score:	0.782872	**0.876923**	0.859885	0.869403

(Table 3: Results of the 4 ANN on 50000 movie reviews using Word2Vec)

Scenario 2 (4251 records – 2000 positive & 2251 negative) using Word2Vec

	Simple ANN	CNN	LSTM	GRU
Accuracy:	0.743831	0.811986	0.824912	**0.848414**
Precision:	0.717195	**0.847222**	0.778252	0.803456
Recall:	0.773171	0.743902	0.890244	**0.907317**
F1 score:	0.744131	0.792208	0.830489	**0.852234**

(Table 4: Results of the 4 ANN on 4251 movie reviews using Word2Vec)

As shown in the above tables, the results show that GRU outperforms of the other three types, CNN, LSTM, and the Simple Neural Network especially when the dataset is very small, like with 4251 records. In the whole dataset of 50000 records CNN and GRU have almost the same accuracy, with GRU be a slit better. However, the computation time with CNN model seems to be faster than RNNs. While both RNNs models had an average time of 70s per epoch, CNN had average time of 57s in order to perform a training. Moreover, CNN has better Recall and F1-Score something that we expected because of their ability to extract local and position-invariant features.

In the case that our examined dataset contains only few records (4251), all of our developed neural models had a negative effect on their performance and accuracy. All four Neural models had a decrease of 3-4% on their accuracy and their metrics. GRU, also in that case, outperforms the other models, but has about the same results with LSTM model. Due to the limited number of train records, CNN has not the ability to learn as fast as RNNs because its complexity. Moreover, on this dataset each one of the models overfits on training data. The models had their best validation accuracy among their first 3 epochs, and after that, they fail to generalize so validation accuracy slowly decreases, while training accuracy increases. Our models overfit in training data, while they perform good in testing data.

Scenario 1 (50000 records – 25000 positive & 25000 negative) using GloVe

	Simple ANN	CNN	LSTM	GRU
Accuracy:	0.750500	0.865500	0.852800	**0.874700**
Precision:	0.743056	0.849379	0.825981	**0.893436**
Recall:	0.757331	0.884934	**0.889788**	0.847725
F1 score:	0.750125	0.866792	0.856698	**0.869980**

(Table 5: Results of the 4 ANN on 50000 movie reviews using Glove)

Scenario 2 (4251 records – 2000 positive & 2251 negative) using GloVe

	Simple ANN	CNN	LSTM	GRU
Accuracy:	0.714454	0.835488	0.837838	**0.844888**
Precision:	0.694639	0.822967	0.804933	**0.837379**
Recall:	0.726829	0.839024	**0.875610**	0.841463
F1 score:	0.710369	0.830918	0.838785	**0.839416**

(Table 6: Results of the 4 ANN on 4251 movie reviews using Glove)

By using GloVe pretrained Word Embeddings we notice again that GRU model has the best performance and accuracy. Decreasing the size of the examined dataset affects the performance of the model in this case too. We can observe a significant loss about 3-4% in the accuracy and F1-Score between the outcomes of performing Sentiment Analysis on the whole dataset than in the smaller one, with only 4251 records. The deep neural networks (DNN) based methods usually need a large-scale corpus due to the large number of parameters, it is hard to train a network that generalizes well with limited data.

Finally, we can observe that we have about the same performances and accuracies by using these two different pretrained embedding models. The overall results show that GRU, which is a variant of RNN, outperformed all the other models by a significant margin, the CNN, the LSTM and the simple neural network. In addition, a general trend was observed that a greater number of dimensions in word embeddings and gated units resulted in better performance.

5.3 Neural Machine Translation

Initially we tried to implement the Neural Machine Translation task by using custom made Neural Networks. Thus, we created sequential Encoder-Decoder implementations of Neural Machine Translation using Tensorflow Keras. The model that we trained and checked for their results were based on LSTM. In this architecture, the input sequence is encoded by a front-end model called the encoder then decoded word by word by a backend model called the decoder. Moreover, we select to train this model for 30 epochs, because we also did not used any pretrained word embedding model, as in Sentiment Analysis task, for the Embedding layer, so our model had to learn embeddings while fitting on the training data.

A 100-dimensions embedding size selected in order to represent words in a bigger dimensionality. This, also, selected due to the large length of sentences that we want to translate when we fit our model to the imdb_reviews dataset. Our train corpora for this Neural Machine Translation model has very little sentences according to these that are recorded in the examined dataset. For example, the maximum length of

English sentences of the train corpora is 5 words long, while the minimum length of reviews in the examined dataset is 32 words long. So, we make an assumption that if we pad and create larger sequences in the train corpora, may be then we can fit the train model in the IMDB dataset. As an activation function for the last Dense layer we choose to use the *"softmax"* function because we consider the translation problem as multi-class classification.

In addition, the model is trained using the efficient Adam approach to stochastic gradient descent and minimizes the categorical loss function because, as said above, we consider the translation problem as multi-class classification.

```
model.compile(loss='categorical_crossentropy',optimizer='adam',me
trics=['accuracy'])
```

```
model = Sequential()
model.add(Embedding(src_vocab, n_units, input_length=src_timesteps, mask_zero=True))
model.add(LSTM(n_units))
model.add(RepeatVector(tar_timesteps))
model.add(LSTM(n_units, return_sequences=True))
model.add(TimeDistributed(Dense(tar_vocab, activation='softmax')))
return model
```

Figure 55: Sample python code to create the NMT model

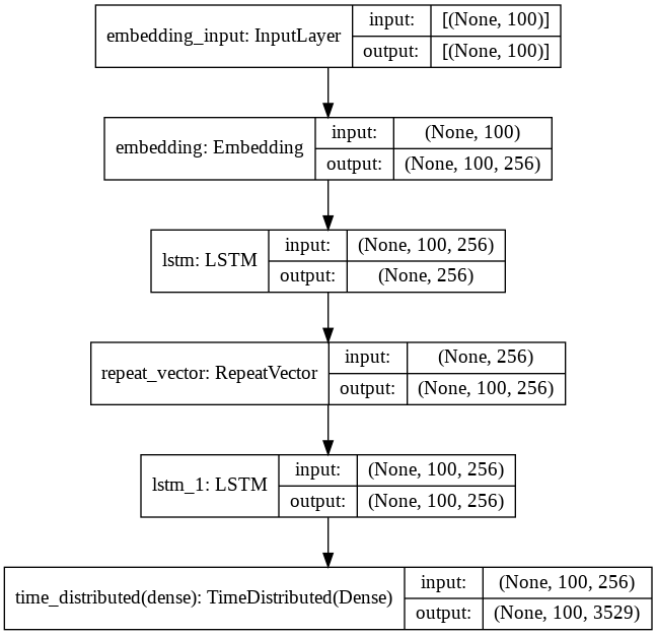

Figure 56: NMT LSTM model architecture

The above model translates the input English sentence into the corresponding German sentence with a Bleu Score of 0.648859 on the test set.

```
src=[['tom stabbed me' 'tom hat mich gestochen'
 'ccby france attribution tatoebaorg ck felixjp']], target=[['tom stabbed me' 'tom hat mich gestochen'
 'ccby france attribution tatoebaorg ck felixjp']], predicted=[tom hat mich]
src=[['get lost' 'mach dich vom acker'
 'ccby france attribution tatoebaorg dorenda sudajaengi']], target=[['get lost' 'mach dich vom acker'
 'ccby france attribution tatoebaorg dorenda sudajaengi']], predicted=[verzieh dich]
src=[['drink it down' 'trinken sie es'
 'ccby france attribution tatoebaorg cm tamy']], target=[['drink it down' 'trinken sie es'
 'ccby france attribution tatoebaorg cm tamy']], predicted=[spul es herunter]
src=[['she dumped him' 'sie gab ihm den laufpass'
 'ccby france attribution tatoebaorg ck zaghawa']], target=[['she dumped him' 'sie gab ihm den laufpass'
 'ccby france attribution tatoebaorg ck zaghawa']], predicted=[sie hat mir den laufpass]
src=[['show it to tom' 'zeig ihn tom'
 'ccby france attribution tatoebaorg ck raggione']], target=[['show it to tom' 'zeig ihn tom'
 'ccby france attribution tatoebaorg ck raggione']], predicted=[zeig sie tom]
src=[['i saw a dog' 'ich habe einen hund gesehen'
 'ccby france attribution tatoebaorg ck muiriel']], target=[['i saw a dog' 'ich habe einen hund gesehen'
 'ccby france attribution tatoebaorg ck muiriel']], predicted=[ich habe einen hund hund]
src=[['please say yes' 'sag bitte ja'
 'ccby france attribution tatoebaorg ck pfirsichbaeumchen']], target=[['please say yes' 'sag bitte ja'
 'ccby france attribution tatoebaorg ck pfirsichbaeumchen']], predicted=[hey mal]
Bleu-1: 0.939758
Bleu-2: 0.808313
Bleu-3: 0.738791
Bleu-4: 0.648859
```

Figure 57: BLEU score of our model in the test data

Looking at the results of the test set, do see readable translations, which is not an easy task. For example, we see “please say yes” correctly translated to “sag bitte yes“. But, we also see some poor translations and a good case that the model could suffer from further tuning, such as “she dumped him” translated as “sie gab ihm den laufpass” instead of the expected “sie ließ ihn fallen “.

A BLEU-4 score of about 0.6488 was achieved, providing a baseline skill to improve upon with further improvements to the model.

The main concept for the sequential Encoder-Decoder implementation that we created is that the encoder represents the input text corpus, English text in our situation, in the form of embedding vectors and trains the model, while the decoder translates and predicts the input embedding vectors into one-hot vectors representing German words in the dictionary.

Neural Machine Translation with LSTMs in imdb_reviews dataset

By performing the above model to the imdb_reviews dataset we had very poor results. Our neural trained model was uncapable to translate and manage long length sentences like these that are included in our examined dataset.

Unfortunately, the open-source datasets that we had at our disposal contain small length language pair sentences. Therefore, our NMT model are capable to translate only small length sentences like “Are you still at home?” or “Is the weather good?” from English to any other language (German and Greek for the purposes of our Thesis). This indicates that larger datasets and bigger corporas with longer language pairs sentences would be more suitable in order to fit and evaluate our models in big datasets with long sentences like the one that we used in our experiments.

```
input_seq = eng_tokenizer.texts_to_sequences(['I go to work'])
print(input_seq)
# pad sequences with 0 values
input_seq = pad_sequences(input_seq, maxlen=20, padding='post')
# pad sequence with 0 values
print(input_seq)
translation1 = predict_sequence(model2, ger_tokenizer, input_seq)

print('-')
print('Input: I go to work')
print('Response:', translation1)
```

```
[[2, 13, 17, 93]]
[[ 2 13 17 93  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]]
-
Input: I go to work
Response: ich gehe arbeiten
```

Figure 58: Example of a custom translation from English to German using our trained model

```
Input: seeing vote average pretty low fact clerk video store thought ok much expectations renting film contrary enjoyed lot charming movie
Response: mach sie mein sicher
```

Figure 59: Example of a false translation on IMDB dataset using our trained model

5.4 Machine Translation using open source APIs

Therefore, we were led to the implementation of Machine Translation based on open source services and libraries. Faced the problem that is being described above we tried to translate our text based on different APIs and services provided by major companies, like Google, Microsoft and Yandex. All these three companies provide their own Machine Translation APIs. Google provides its Google Cloud Translator API through the Google Cloud Platform (GCP). Microsoft provides its own Translator Text API through Azure platform and Yandex, also, provides its own Yandex-Translate API.

However, these APIs are available for use under some prerequisites. Due to limitations and different pricing levels according to the number of characters that can be translated we were not able to translate the whole dataset. Thus, we utilized these APIs only on 4251 total records of our initial dataset. We selected 2000 negative and 2251 positive reviews in order to perform translate based on these open source APIs. Google for example translates for free only 1M characters and then there is a price for 20$ per million characters and supports 104 languages. Yandex also has a free translation over 1M characters but has a lower price, 10$ per million characters, and supports 90 languages. Microsoft API allows to translate 2M characters but supports only 60 languages.

Metrics	**Google**	**Yandex**	**Microsoft**
Free translated characters offered	1M	1M	**2M**
Price per 1 million characters	20$	15$	**10$**

Number of supported languages	**104**	90	70

(Table 7: Google, Yandex & Microsoft SLA's compare)

Google Cloud Translation API

Google Cloud Translator API[9] makes use of two different models in order to perform custom translation. We can specify which model to utilize for translation by using the model query parameter. By specifying *"base"* we are able to make use of the Phrase-Based Machine Translation model, and with the use of *"nmt"* configuration we can use the Neural Machine Translation model. If we specify the NMT model in our request and the requested language translation pair is not supported for the NMT model, then the PBMT model is used. For the purposes of this Thesis we make use of the NMT model option, which is provided both for English-Greek and English-German language pairs.

The Phrase-Based Machine Translation (PBMT) methods try to break an input sentence into words and phrases to be translated largely independently, whereas Neural Machine Translation (NMT) considers the entire input sentence as a unit of translation. The GNMT network can undertake interlingual machine translation by encoding the semantics of the sentence, rather than by memorizing phrase-to-phrase translations [69]. The Google Neural Machine Translation uses Recurrent Neural Networks to directly learn the mapping between sentence in one language to sentence in another language. The overall architecture of the model uses an "encoder - attention - decoder" structure as shown in the above image. [70]

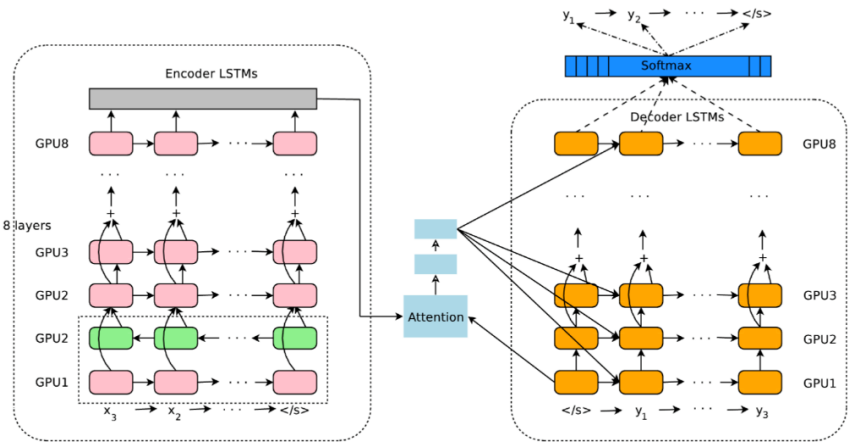

Figure 60: GNMT's "encoder - attention - decoder" approach

For the purposes of this Thesis we utilize Google Cloud Translator API, as well two other open sources translate APIs, in order to translate English movie reviews from the initial dataset to German and Greek reviews. Concerning Google's API we had to

[9] https://cloud.google.com/translate/docs/quickstarts

create a project in Google Cloud Platform (GCP) and enable Cloud Translation API. We utilize the Advance settings of this API in order to perform Google Neural Machine Translation to our source text.

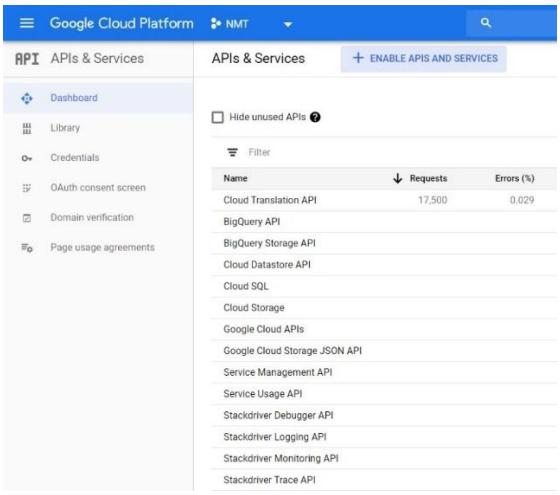

Figure 61: Enable Cloud Translation API in GCP

The utilization of the below code had as a result to perform English to German translation on our selected range of reviews. As shown in the image below, target language option is specified as “de” and each translated sentence is saved in a general txt file which includes all the German translated reviews. In order to use google TranslationServiceClient() method we have to import first the translate library from google.cloud module and provide our environment with a Google certificate, as shown in Image 62.

```
from google.cloud import translate
import os

os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = os.path.join(os.curdir, 'C:\\Users\\_
```

Figure 62: Google Translate requirements

```
# The encoding needs to be utf-8 (Unicode)
# because all languages are supported by
# Google Cloud and ASCII supports only English.
def google_translate_text(text):
    translate_client = translate.TranslationServiceClient()
    parent = translate_client.location_path("nmt-unipi", "global")
    translated = list()
    response = translate_client.translate_text(parent=parent,
        contents=[data],
        mime_type="text/plain",  # mime types: text/plain, text/html
        target_language_code="de",)

  # Display the translation for each input text provided
    for translation in response.translations:
        #print(u"Translated text: {}".format(translation.translated_text))
        f.write(translation.translated_text+'\n')
        return translation.translated_text
        #print(translated['translated_text'])
```

Figure 63: Sample python code for Google Cloud Translator API

Microsoft Translator API

Microsoft Translator[10] is a cloud-based machine translation service. The core service is the Translator Text API, which powers a number of Microsoft products and services, and is used by thousands of businesses worldwide in their applications and workflows, which allows their content to reach a global audience. Microsoft Translator offers neural network (LSTMs) based translation that enables a new decade of translation quality improvement. The V3 Translator API is neural by default and statistical systems are only available when no neural system exists.

The following figure depicts the various steps neural network translations go through to translate a sentence. Because of this approach, the translation will take into context the full sentence, versus only a few words sliding window that SMT technology uses and will produce more fluid and human-translated looking translations. NMT provides better translations than SMT not only from a raw translation quality scoring standpoint but also because they will sound more fluent and human. The key reason for this fluidity is that NMT uses the full context of a sentence to translate words. SMT only took the immediate context of a few words before and after each word.

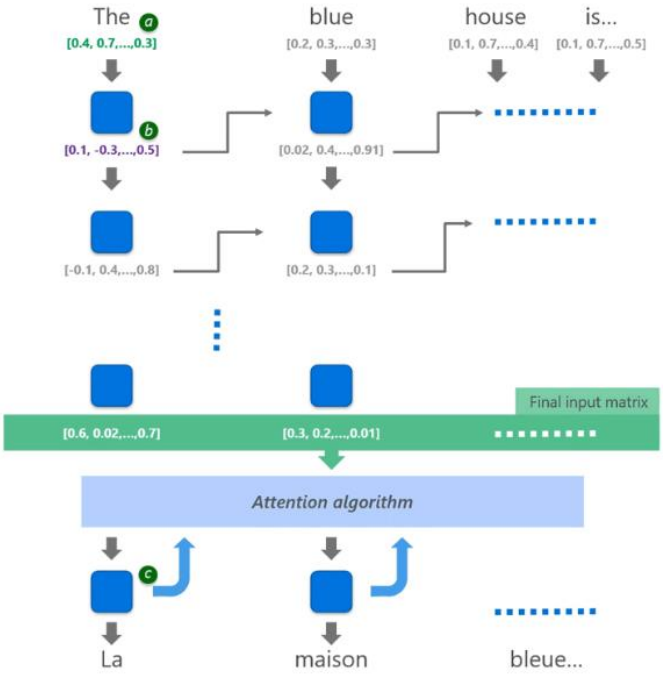

Figure 64: Microsoft Translator Example approach

In the example depicted above, the context-aware 1000-dimension model of "the" will encode that the noun (house) is a feminine word in French (la maison). This will allow the appropriate translation for "the" to be "la" and not "le" (singular, male) or "les" (plural) once it reaches the decoder (translation) layer.

The attention algorithm will also calculate, based on the word(s) previously translated (in this case "the"), that the next word to be translated should be the subject ("house") and not an adjective ("blue"). In can achieve this because the system learned that

[10] https://azure.microsoft.com/en-us/services/cognitive-services/translator-text-api/

English and French invert the order of these words in sentences. It would have also calculated that if the adjective were to be “big” instead of a color, that it should not invert them (“the big house” => “la grande maison”).

According to the instructions of Microsoft about the enabling of its Translator API, we had to create a project in the Azure Portal and enable the Translator Text service.

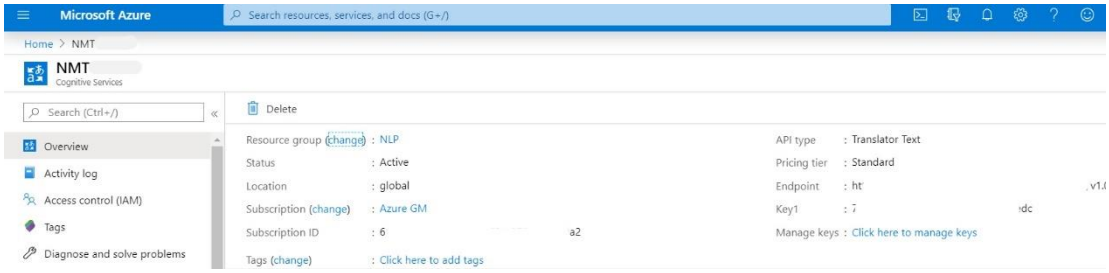

Figure 65: Microsoft Azure Portal and Translator Text

As shown in the image below, Microsoft API is more complicated than the one implemented for using Google Cloud Translation API. Using the subscription key provided in Azure Portal we are able to establish a client connection and send a POST request to Microsoft’s service. The response, in our situation the translated sentence, is in JSON format, so we have to dump this in Python code and convert it in a list in order to be able to fetch the translated text and save it in the corresponding txt file. Microsoft’s API service was the fastest of the three APIs tested, so we implement translation in German and Greek in parallel.

```
def translate (content):
    headers = { 'Ocp-Apim-Subscription-Key': subscriptionKey,
                'Content-type': 'application/json',
                'X-ClientTraceId': str(uuid.uuid4())
              }
    conn = http.client.HTTPSConnection(host)
    conn.request("POST", path + params, content, headers)
    response = conn.getresponse()
    return response.read()
```

```
output_file_de = "C:\\Users\\                                    \\trans-microsoft-de.txt"
output_file_el = "C:\\Users\\g                                   \\trans-microsoft-el.txt"
f=open(output_file_de, 'a+', encoding="utf-8")
w=open(output_file_el, 'a+', encoding="utf-8")

def microsoft_translate_text(data):
    requestBody = [{ 'Text' : data,}]
    content = json.dumps(requestBody, ensure_ascii=False).encode('utf-8')
    result = translate(content)
    output = json.dumps(json.loads(result), indent=4, ensure_ascii=False, separators=(',', ': '))
    print (output)
    decoded_hand = json.loads(output)
    de_text = decoded_hand[0]['translations'][0]['text']
    el_text = decoded_hand[0]['translations'][1]['text']
    print (de_text)
    print (el_text)
    f.write(de_text+'\n')
    w.write(el_text+'\n')
    return de_text, el_text
```

Figure 66: Sample python code for Microsoft Translate API

Yandex-Translate API

Yandex.Translate[11] differs from the above-mentioned APIs as it makes use of a hybrid model of machine translation that includes both neural network (deep learning) and statistical approaches.

As soon as the user enters text to translate, Yandex.Translate sends this text to both systems: the neural network and the statistical translator.

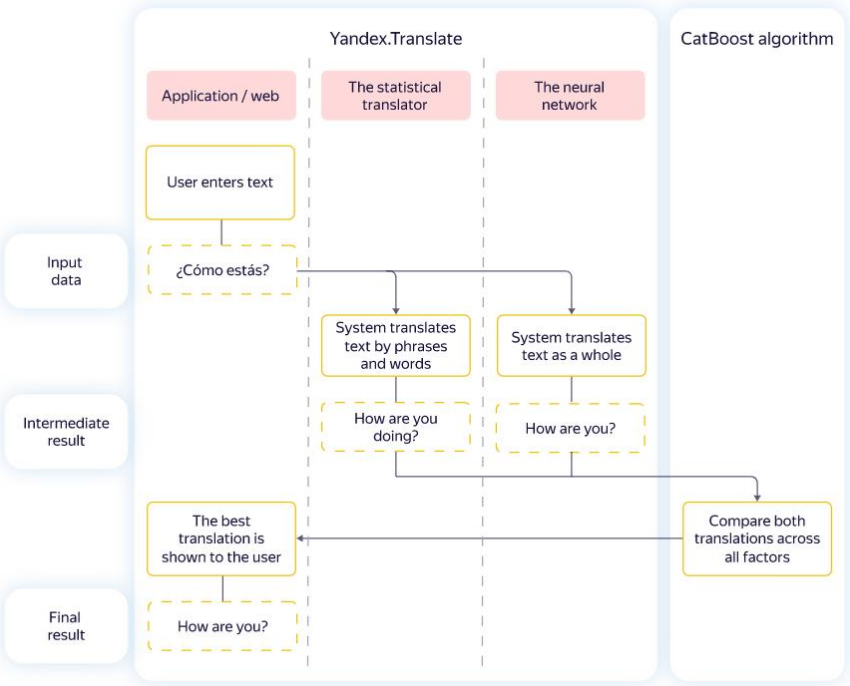

Figure 67: Yandex.Translate approach

[11] https://tech.yandex.com/translate/

The results obtained from both systems are evaluated by an algorithm based on the CatBoost[12] machine learning method. The algorithm analyzes dozens of factors, from sentence length (short phrases and rare words are better translated by the statistical model) to syntax. The two translations are compared across all factors, and the best one is shown to the user.

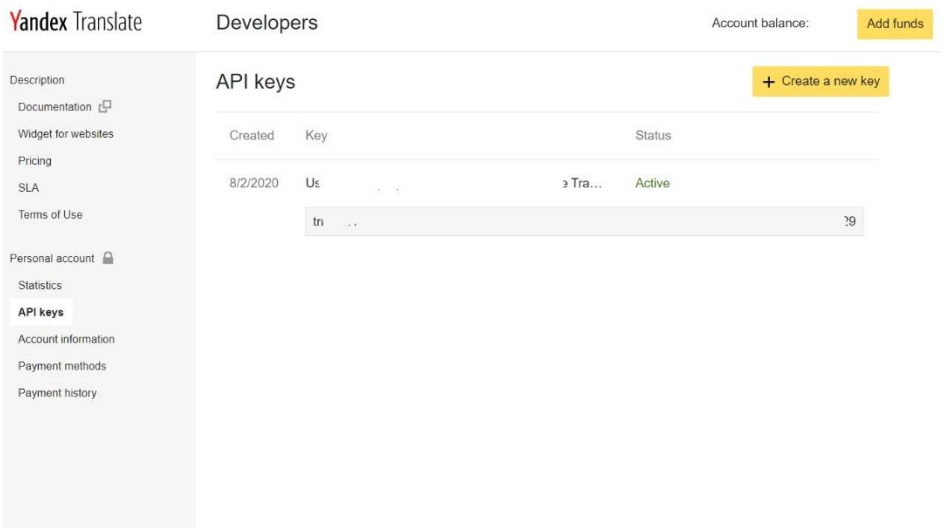

Figure 68: Yandex Translate developer dashboard

Yandex's Translate API was the easiest to implement. We all needed to install and import yandex.Translater module and its Translater() class. By, using the API key provided in Yandex Translate portal we are able to implement our translations with the below Python code.

```
tr = Translater()
tr.set_key('[redacted]
tr.set_from_lang('en')
tr.set_to_lang('el')

def yandex_translate_text(data):
    translated = list()
    tr.set_text(data)
    result =tr.translate()
    f.write(result+'\n')
    return result
```

Figure 69: Sample python code for Yandex.Translate

Finally, Machine Translation task in IMDB dataset from English to German and Greek was performed with the above three mentioned open source APIs. Google's API was the most expensive one according to providers SLAs and most slow in overall translation. Moreover, the translated files that created from the APIs and contain Greek characters had almost twice the size of German files. The records that produced from each of these implementations were 4251 for both German and Greek translated

[12] https://catboost.ai/docs/

reviews. Next, in our final step of our approach we seek to perform Sentiment Analysis in the translated text corporas and observe outcomes and different according to these that we had in Sentiment Analysis task in source language (English).

5.5. Sentiment Analysis on Translated Reviews

By utilizing the same models as we did in the first step of Sentiment Analysis in source language we came through major outcomes.

Scenario 1 (4251 records – 2251 positive & 2000 negative) using GloVe

		SA after Google Translator		SA after Yandex Translate		SA after Microsoft Translator	
	ANN(EN)	ANN(GR)	ANN (DE)	ANN (GR)	ANN (DE)	ANN (GR)	ANN (DE)
Accuracy:	**0.714454**	0.662750	**0.623972**	0.542891	0.542891	0.569918	0.568743
Precision:	**0.694639**	0.610018	0.592593	0.523918	0.521921	0.548889	0.547884
Recall:	**0.726829**	0.831707	0.702439	0.560976	0.609756	0.602439	0.600000
F1 score:	**0.710369**	0.703818	**0.642857**	0.541814	0.562430	0.574419	0.572759
	CNN(EN)	CNN(GR)	CNN (DE)	CNN (GR)	CNN (DE)	CNN (GR)	CNN (DE)
Accuracy:	**0.835488**	0.662750	0.658049	0.647474	**0.705053**	0.702703	0.701528
Precision:	**0.822967**	0.610018	0.633110	0.599278	0.673961	0.647834	0.630435
Recall:	**0.839024**	0.831707	0.690244	0.809756	0.751220	0.839024	0.919512
F1 score:	**0.830918**	0.703818	0.660443	0.688797	0.710496	0.731137	**0.748016**
	LSTM(EN)	LSTM(GR)	LSTM(DE)	LSTM(GR)	LSTM(DE)	LSTM(GR)	LSTM(DE)
Accuracy:	**0.837838**	0.685076	0.669800	0.529965	**0.747356**	0.662750	0.729730
Precision:	**0.804933**	0.730519	0.638116	0.507143	0.734940	0.621302	0.707373
Recall:	**0.875610**	0.548780	0.726829	0.865854	0.743902	0.768293	0.748780
F1 score:	**0.838785**	0.626741	0.679590	0.639640	**0.739394**	0.687023	0.727488
	GRU(EN)	GRU(GR)	GRU(DE)	GRU(GR)	GRU(DE)	GRU(GR)	GRU(DE)
Accuracy:	**0.844888**	0.701528	0.669800	0.645123	**0.703878**	0.694477	0.696827
Precision:	**0.837379**	0.648855	0.707395	0.599631	0.732353	0.680288	0.700000
Recall:	**0.841463**	0.829268	0.536585	0.792683	0.607317	0.690244	0.648780
F1 score:	**0.839416**	**0.728051**	0.610264	**0.682773**	0.664000	0.685230	0.673418

(Table 8: Results of SA models over translated data with using GloVe)

According to the results from this scenario we can move on to the following observations. All four Neural models outperform when used in German texts. CNN and both RNNs models outperform in German translated texts that have been produced by Yandex Translate API, while Simple ANN outperforms also in German text, but this time the one that derives from Google Translator.

Scenario 2 (4251 records – 2251 positive & 2000 negative) using Word2Vec

		SA after Google Translator		SA after Yandex Translate		SA after Microsoft Translator	
	ANN (EN)	ANN (GR)	ANN (DE)	ANN (GR)	ANN (DE)	ANN (GR)	ANN (DE)
Accuracy:	**0.743831**	0.611046	**0.641598**	0.613396	0.542891	0.608696	0.601645
Precision:	**0.717195**	0.584946	0.595978	0.590604	0.521921	0.594132	0.567878
Recall:	**0.773171**	0.663415	**0.795122**	0.643902	0.609756	0.592683	0.724390
F1 score:	**0.744131**	0.621714	**0.681296**	0.616103	0.562430	0.593407	0.636656
	CNN (EN)	CNN (GR)	CNN (DE)	CNN (GR)	CNN (DE)	CNN (GR)	CNN (DE)
Accuracy:	**0.811986**	0.713278	0.686251	**0.741481**	0.682726	0.706228	0.674501
Precision:	**0.847222**	0.744118	0.657837	0.705628	0.628205	0.643369	0.607780
Recall:	**0.743902**	0.617073	0.726829	0.795122	0.836585	0.875610	**0.914634**
F1 score:	**0.792208**	0.674667	0.690614	**0.747706**	0.717573	0.741736	0.730282
	LSTM (EN)	LSTM (GR)	LSTM (DE)	LSTM (GR)	LSTM (DE)	LSTM (GR)	LSTM (DE)
Accuracy:	**0.824912**	0.672150	0.666275	0.715629	0.605170	**0.733255**	0.672150
Precision:	**0.778252**	0.619744	0.612903	0.825581	0.560656	0.674952	0.672823
Recall:	**0.890244**	0.826829	0.834146	0.519512	0.834146	**0.860976**	0.621951
F1 score:	**0.830489**	0.708464	0.706612	0.637725	0.670588	**0.756699**	0.646388
	GRU (EN)	GRU (GR)	GRU (DE)	GRU (GR)	GRU (DE)	GRU (GR)	GRU (DE)
Accuracy:	**0.848414**	0.703878	0.709753	0.727380	0.714454	**0.737955**	0.688602
Precision:	**0.803456**	0.723164	0.696386	0.739247	0.670757	0.694387	0.635009
Recall:	**0.907317**	0.624390	0.704878	0.670732	0.800000	**0.814634**	0.831707
F1 score:	**0.852234**	0.670157	0.700606	0.703325	0.729700	**0.749719**	0.720169

(Table 9: Results of SA models over translated data with using Word2Vec)

In the case of performing Sentiment Analysis on translated data by utilizing also Word2Vec, all four models have decreased their performance and their accuracy in the reviews after implementing the translation for about 7-14%. Moreover, both RNNs perform better in Greek translations than in German and their best accuracy and F1-Score appear in Greek reviews that have been produced by Microsoft's Translator. In addition, CNN model has the best accuracy and F1-Score, also in Greek reviews, but this time these that have translated from English base on Yandex Translate API. Performing Sentiment Analysis with our trained models in Google's translated texts seem to be less effective.

Overall Outcome

From the above we can understand that we have significant loss about the sentiments expressed in reviews during the translation task, while using both pretrained models as inputs to our Embedding Layers. The above major impact may be caused for three reasons. At first, we must mention that we performed the translations in processed text, without using punctuations and with lowercase texts. This maybe caused some loss in the final meaning of the text. A translation in the whole dataset may provide slit better results.

In addition, we can observe that models using pretrained GloVe model perform better when used in German texts. However, models using Word2Vec outperform when used in Greek texts. If you observe the above results from API providers' perspective, we can assume that Google's API is the less effective, while Microsoft's Translator API produces texts in which our Sentiment Analysis RNNs models outperform, in the case that they use Word2Vec. In the other hand, CNN model outperforms when used in Greek texts that derive from Yandex Translate.

Moreover, using pretrained Word Embeddings in languages with less usability than English has negative effect to our models, because we do not train their Embedding Layers, instead we use the pre-trained models, like GloVe and Word2Vec. We can assume if we see the below image, where not all Greek words are representing as vectors using GloVe embeddings.

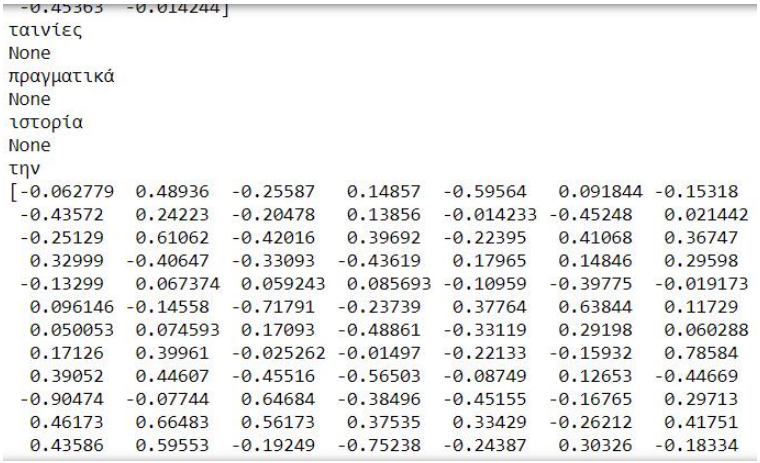

Figure 70: Representation example of Greek words using Glove

Especially using GloVe we even have loss of 18% of the accuracy of our model. This is happening because these pretrained models are not compatible and widely tested with other languages except English. Most of the research, models and applications focus on training and creating models based on English language. Thus, they fail to generalize, represent and support other languages. That led us to perform Sentiment Analysis without using pretrained models in target languages and to the assumption that may have better results if we leave the model to train and learn the embeddings by each own. So, we implemented one more scenario (Scenario 3) in which we trained our models without pretrained Word Embedding models. In order to perform this scenario we eliminated the two below settings (*weights=[embedding_matrix], trainable=False*) from the Embedding layers of our Neural Network models.

Scenario 3 (4251 records – 2251 positive & 2000 negative) without using pretrained Word Embeddings

	SA after Google Translator		SA after Yandex Translate		SA after Microsoft Translator	
	ANN (GR)	ANN (DE)	ANN (GR)	ANN (DE)	ANN (GR)	ANN (DE)
Accuracy:	0.844888	0.839013	0.861340	0.860165	**0.864865**	0.861340
Precision:	0.802174	0.816705	0.834862	0.819780	0.829978	0.833333
Recall:	0.900000	0.858537	0.887805	0.909756	0.904878	0.890244
F1 score:	0.848276	0.837099	0.860520	0.862428	**0.865811**	0.860849
	CNN (GR)	CNN (DE)	CNN (GR)	CNN (DE)	CNN (GR)	CNN (DE)
Accuracy:	0.849589	0.843713	0.860165	**0.869565**	0.855464	**0.869565**
Precision:	0.821918	0.807095	0.837587	0.843678	0.831409	0.846868
Recall:	0.878049	0.887805	0.880488	0.895122	0.878049	0.890244
F1 score:	0.849057	0.845528	0.858502	**0.868639**	0.854093	0.868014
	LSTM(GR)	LSTM(DE)	LSTM(GR)	LSTM(DE)	LSTM(GR)	LSTM(DE)
Accuracy:	0.814336	0.753231	**0.866040**	0.831962	0.840188	0.822562
Precision:	0.831579	0.755102	0.879487	0.823245	0.812785	0.767010
Recall:	0.770732	0.721951	0.836585	0.829268	0.868293	0.907317
F1 score:	0.800000	0.738155	**0.857500**	0.826245	0.839623	0.831285
	GRU(GR)	GRU(DE)	GRU(GR)	GRU(DE)	GRU(GR)	GRU(DE)
Accuracy:	0.813161	0.836663	**0.855464**	0.837838	0.815511	0.755582
Precision:	0.783296	0.791398	0.891008	0.770916	0.758691	0.795322
Recall:	0.846341	0.897561	0.797561	0.943902	0.904878	0.663415
F1 score:	0.813599	0.841143	**0.841699**	0.838684	0.825362	0.723404

(Table 10: Results of SA models over translated data without using pretrained data)

According to our final assumption and Scenario we can understand that pretrained Word Embeddings were responsible for the significant loss of sentiment and low performances of our models. If we compare Table 10 with Table 5 & 6, which contain results from Sentiment Analysis performed in source language (English Language), we can understand that the utilization of our models in the two target languages have similar performances and accuracies according them in English. This indicates that we had minor losses from Machine Translation task based on open source APIs and that our models are mature enough to self-trained and create the needed embeddings. The most effective translated texts derive from Yandex Translate API, where both RNNs models outperform, while CNN has similar accuracy in texts derive both from Yandex Translate API and Microsoft's Translator API.

Finally, as we also noticed and in the Sentiment Analysis task in English texts Deep Neural Network models fail to generalize when trained in small datasets. They seem to overfit in training data, while they cannot generalize what they learned in the small tests sets that we provide to them. Working on a bigger dataset may provide better outcomes. Another way, to hence the performance of the models is to increase the number of epochs in which they would be trained. For example, increasing the number of epochs on 20 can cause better results by 3-4%, as shown in the table below,

where results for the utilization of CNN model with 10 epochs and 20 epochs in Greek corpora after Google Translation are presented. But, for our comparisons we had to use 10 epochs option similar to the Sentiment Analysis performed in English dataset.

CNN (Greek) - 10 epochs	CNN (Greek) - 20 epochs
Accuracy: 0.713278	Accuracy: **0.753231**
Precision: 0.744118	Precision: 0.727273
Recall: 0.617073	Recall: **0.780488**
F1 score: 0.674667	F1 score: **0.752941**

(Table 11: Outcomes of CNN trained in different number of epochs)

6 Conclusion & Future Work

6.1 Conclusion

The comparison on using the same Neural Network models in source and target languages reveals a significant loss in the sentiment and emotion of our data. Performance rates of Sentiment Analysis on original and translated data, leading to a conclusion that the state-of-the-art Machine Translation systems can provide an alternative to the costly development of languages features to realize Sentiment Analysis in multilingual level. Our trained models when used in target languages and self-trained for creating the word embeddings, instead of using pretrained ones, had similar results and performances with the models used in source language.

However, there is a lot of work to do according to different language pair corporas and pretrained Word Embeddings models. So far research and business communities have focused on implementation of tools and techniques based on English languages. Few studies and tools target on less significant languages like Greek. Pretrained Word Embeddings models are not suitable for use in other languages than English and that is an issue that needs to be examined further. Recent years and due to the increasing interest in Multilingual Tools and Classifiers a few researchers try to create such models [71], but a lot of work to this direction should be done.

We must notice that our approach so far does not rely on a specific domain. It is not domain agnostic. Different datasets from various domains may have different results according to our Machine Translation and Sentiment Analysis implementations.

6.2 Future Work

In our future work we will seek to complete Machine Translation based on the approach of transfer learning and by using Google's open source and state-of-the-art techniques and tools, like BERT and Transformers. These two techniques are the fundamentals of transfer learning. In this Thesis we presented context-free models such as Word2Vec or GloVe in order to generate a single word embedding representation for each word in the vocabulary. Using BERT or other contextual models, like OpenAI and ELMo, we will be able to generate a representation of each word that is based on other words in the sentence. In other words, we can generate vector representations for sentences. The use of these contextual models instead of context-free models will be an area of our futured work.

Furthermore, we must notice that there is an additional extension of RNNs models, which are called Bi-directional RNNs that allows the Recurrent Neural Network model

to analyze the textual data in both the directions, that is, from start to end and from end to start. They present each training sequence forwards and backwards to two separate recurrent nets, both of which are connected to the same output layer [72]. This allows the model to capture the context of data properly and make accurate classifications.

Moreover, as (Conneau et al. 2018) mention [73] so far Deep Neural Networks learn based on bilingual dictionaries or parallel corpora, like these that we used for the purposes of this Thesis. The new approach of character-based translation showed encouraging results and achieved aligning monolingual word embedding spaces in an unsupervised way. Literature and practical work on this field and approach will also be examined.

Another major problem with automated Machine Translation systems is their lack of interoperability. If one implements a system that translates and builds sentiment classifiers between English and German, it is not easy to add one more or perform Sentiment Analysis with the same classifiers in another language. The reason is that there is no one-to-one correspondence between the words of the vocabulary of the languages and the grammatical and syntactic rules are different. Thus, the support of a new language must be implemented almost from the beginning with the use of the gained experience of the previous languages. Recently, the researchers at Google AI Team built a more enhanced system for neural machine translation (NMT) [74], where a single massively multilingual neural machine translation (NMT) model were build, which handles 103 languages.

The implementation of a Universal Machine Translation system and of a Multilingual Sentiment Classifier will be the main goals of our future work. Answer to questions like, *"How can we build a Cross-Lingual Machine Translation system based on Multilingual Classifiers and Universal Word Embeddings?"* will be examined and discussed. Future work will be in the direction of extending the list of languages further and evaluating the performance on data from multilingual social media platforms and other sources.

References

[1] https://www.zdnet.com/article/by-2025-nearly-30-percent-of-data-generated-will-be-real-time-idc-says/ [Accessed: 23 December 2019]

[2] https://www.internetworldstats.com/stats7.htm [Accessed: 22 November 2019]

[3] M. Cox, D. Ellsworth. Application-controlled demand paging for out-of-core visualization. VIS '97: Proceedings of the 8th conference on Visualization '97. pp. 235–ff. October 1997.

[4] D Reinsel, J. Gantz, J. Rydning. The Digitization of the World From Edge to Core An IDC White Paper – #US44413318. November 2018. https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf [Accessed: 10 January 2020]

[5] https://www.nltk.org/api/nltk.tokenize.html?highlight=stopwords#module-nltk.tokenize.texttiling [Accessed: 20 January 2020]

[6] J. Choi, J. Tetreault, A. Stent. It Depends: Dependency Parser Comparison Using A Web-based Evaluation Tool. 2015.

[7] E. S. Tellez, D. Moctezuma, S. Miranda-Jimenez, M. Graff. An Automated Text Categorization Framework based on Hyperparameter Optimization. April 2017. https://arxiv.org/pdf/1704.01975.pdf [Accessed: 13 January 2020]

[8] J. D. Brown. Developing an automatic document classification system—A review of current literature and future directions. Defense Research and Development Canada. January 2010.

[9] https://www.fortunebusinessinsights.com/industry-reports/natural-language-processing-nlp-market-101933 [Accessed: 20 January 2020]

[10] N. Sager, M. Lyman, NT Nhan, L. Tick. Medical Language Processing: Applications to Patient Data Representation and Automatic Encoding. Methods Inf Med 1995. pp. 140-146. 1995. DOI: 10.1055/s-0038-1634579

[11] https://www.autonomousvehicleinternational.com/features/natural-language-processing-enhances-autonomous-vehicles-experience.html [Accessed: 20 December 2019]

[12] https://www2.deloitte.com/us/en/insights/industry/public-sector/government-trends/2020/ai-augmented-government.html [Accessed: 20 January 2020]

[13] J. Cheng, C. Danescu-Niculescu-Mizil, J. Leskovec. Antisocial Behavior in Online Discussion Communities. https://arxiv.org/pdf/1504.00680v1.pdf%20 [Accessed: 20 Decmber 2019]

[14] https://opendatascience.com/an-introduction-to-natural-language-processing-nlp/ [Accessed: 10 January 2020]

[15] A. Esuli, F. Sebastiani. SentiWordNet: A Publicly Available Lexical Resource for Opinion Mining. Proceedings of LREC. Vol. 6. 2006.

[16] B. Pang, L. Lee. Opinion mining and sentiment analysis, Journal Foundations and Trends in Information Retrieval. Vol. 2. pp. 1–135. 2008.

[17] A. Esuli, F. Sebastiani. Determining term subjectivity and term orientation for opinion mining. In Proceedings of the European Chapter of the Association for Computational Linguistics (EACL), 2006.

[19] T. Nasukawa, J. Yi. Sentiment analysis: Capturing favorability using natural language processing, Conference: Proceedings of the 2nd International Conference on Knowledge Capture (K-CAP 2003). USA. October 2003. DOI: 10.1145/945645.945658

[20] K. Dave, S. Lawrence, D. M. Pennock. Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews. October 2003. DOI: 10.1145/775152.775226

[21] P. D. Turney. Thumbs up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. Proceedings of the 40th annual meeting on association for computational linguistics. 2002.

[22] B. Pang, L. Lee, S. Vaithyanathan. Thumbs up? Sentiment classification using Machine learning Techniques. In Proceedings of the conference on empirical methods in natural language processing. Association for Computational Linguistics. pp. 79–86. 2002.

[23] B. Pang, L. Lee. Seeing Stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In Proceedings of the 43rd annual meeting on association for computational linguistics. pp. 115-124. 2005.

[24] S. Rustamov, E. Mustafayev, M.A. Clements. Sentiment analysis using Neuro-Fuzzy and Hidden Markov models of text. In Proceedings of IEEE 2013. pp.1-6. 2013.

[25] T. Nakagawa, K. Inui, S. Kurohashi. Dependency tree-based sentiment classification using CRFs with hidden variables. In Proceedings of the 2010 annual conference of the north american chapter of the association for computational linguistics. pp. 786-794, 2010.

[26] H. Yu, V. Hatzivassiloglou. Towards answering opinion questions: separating facts from opinions and identifying the polarity of opinion sentences. In: Proceedings of the 2003 conference on empirical methods in natural language processing, EMNLP '03. Association for Computational Linguistics. Stroudsburg. pp 129–136. 2003.

[27] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, M. Stede. Lexicon-Based Methods for Sentiment Analysis. May 2011. https://doi.org/10.1162/COLI_a_00049

[28] L. Zhang, R. Ghosh, M. Dekhil, M. Hsu, B. Liu. Combining Lexicon-based and Learning-based Methods for Twitter Sentiment Analysis. 2011. https://www.hpl.hp.com/techreports/2011/HPL-2011-89.pdf [Accessed: 20 January 2020]

[29] M. Cheong, V. C. S. Lee. A microblogging-based approach to terrorism informatics: Exploration and chronicling civilian sentiment and response to terrorism events via Twitter. Inf. Syst. Front.. vol. 13, no. 1. pp. 45–59. 2011.

[30] A. Bifet, E. Frank. Sentiment knowledge discovery in Twitter streaming data. Lecture Notes in Computer Science. vol. 6332 LNAI. pp. 1–15. 2010.

[31] J. Khairnar, M. Kinikar. Machine Learning Algorithms for Opinion Mining and Sentiment Classification. Int. J. Sci. Res. Publ., vol. 3, no. 6. pp. 1–6. 2013.

[32] R. Moraes, J.F. Valiati, W.P. Neto. Document-level sentiment classification: an empirical comparison between SVM and ANN. Expert Systems with Applications. 2013.

[33] R. Johnson, T. Zhang. Effective use of word order for text categorization with convolutional neural networks. In Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2015). 2015.

[34] Z. Yang, D. Yang, C. Dyer, X. He, A.J. Smola, E.H. Hovy. Hierarchical attention networks for document classification. In Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2016). 2016.

[35] X. Zhou, X. Wan, J. Xiao. Attention-based LSTM network for cross-lingual sentiment classification. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2016). 2016.

[36] K. Kowsari, D. E. Brown, M. Heidarysafa, K. Jafari Meimandi, M. S. Gerber, L. E. Barnes. HDLTex: Hierarchical Deep Learning for Text Classification. 2017.

[37] P. Liu, X. Qiu, X. Huang. Recurrent Neural Network for Text Classification with Multi-Task Learning. 2016.

[38] G. Shalton. The SMART Retrieval System – Experiments in Automatic Document Processing. 1971.

[39] Z. Yin, J. Rong, Z. Zhi-Hua. Understanding Bag-of-Words Model: A Statistical Framework. International Journal of Machine Learning and Cybernetics. 2010.

[40] T. Mikolov, K. Chen, G. Corrado, J. Dean. Efficient Estimation of Word Representations in Vector Space. In Proceedings of Workshop at ICLR. 2013.

[41] O. Levy, Y. Goldberg. Neural word embedding as implicit matrix factorization. Advances in Neural Information Processing Systems 27, pp. 2177–2185. 2014.

[42] J. Hutchins. The history of machine translation in a nutshell. http://hutchinsweb.me.uk/Nutshell-2005.pdf [Accessed: 15 January 2020]

[43] B. Vauquois. A Survey of Formal Grammars and Algorithms for Recognition and Translation. 1968.

[44] S. Tripathi, J. K. Sarkhel. Approaches to machine translation. Ann. Libr. Inf. Stud., vol. 57. pp. 388–393. December 2010.

[45] W. Weaver. Recent Contributions to the Mathematical Theory of Communication. 1949. http://www.panarchy.org/weaver/communication.html [Accessed: 20 January 2020]

[46] J. Zhang, S. Liu, M. Li, M. Zhou, C. Zong. Beyond Word-based Language Model in Statistical Machine Translation. 2015. https://arxiv.org/pdf/1502.01446.pdf [Accessed: 10 January 2020]

[47] M. Nagao. A framework of a mechanical translation between Japanese and English by analogy principle. 1984.

[48] W. Krzysztof, M. Krzysztof. Neural-based machine translation for medical text domain. Based on European Medicines Agency leaflet texts. 2015. http://dblp.uni-trier.de/db/journals/corr/corr1509.html#WolkM15a [Accessed: 10 January 2020]

[49] https://slator.com/technology/corporates-going-all-in-on-neural-machine-translation-research/ [Accessed: 20 January 2020]

[50] N. Kalchbrenner, P. Blunsom. Recurrent Continuous Translation Models, Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. pp. 1700–1709. October 2013.

[51] I. Sutskever, O. Vinyals, Q. V. Le. Sequence to Sequence Learning with Neural Networks. 2014. https://arxiv.org/pdf/1409.3215.pdf [Accessed: 20 December 2019]

[52] Rojas R.. Neural networks: a systematic introduction. Springer. p. 336. ISBN 978-3-540-60505-8. 1996.

[53] C. Tillmann, F. Xia. A Phrase-Based Unigram Model for Statistical Machine Translation. IBM T.J. Watson, Research Center Yorktown Heights. 2003. https://www.aclweb.org/anthology/N03-2036 [Accessed: 20 December 2019]

[54] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, E. Herbst. Moses: Open Source Toolkit for Statistical Machine Translation. June 2007. https://www.aclweb.org/anthology/P07-2045/ [Accessed: 10 December 2019]

[55] P. Koehn. Europarl: A Parallel Corpus for Statistical Machine Translation School of Informatics. June 2006. https://www.aclweb.org/anthology/N06-1003/ [Accessed: 13 December 2019]

[56] G. Klein, Y. Kim, Y. Deng, J. Senellart, A. M. Rush, OpenNMT: Open-Source Toolkit for Neural Machine Translation, 2017.

[57] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. 2014. https://arxiv.org/pdf/1406.1078.pdf [Accessed: 10 January 2020]

[58] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin. Attention Is All You Need. 2017. https://arxiv.org/abs/1706.03762 [Accessed: 20 November 2019]

[59] D. R. So, C. Liang, Q. V. Le. The Evolved Transformer. 2019. https://arxiv.org/pdf/1901.11117.pdf [Accessed: 13 January 2020]

[60] K. Papineni, S. Roukos, T. Ward, W.J. Zhu. BLEU: a Method for Automatic Evaluation of Machine Translation, Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL). pp. 311-318. Philadelphia. July 2002. https://www.aclweb.org/anthology/P02-1040.pdf [Accessed: 13 January 2020]

[61] https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html [Accessed: 3 February 2020]

[62] J. Devlin, M.W. Chang, K. Lee, K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2019. https://arxiv.org/pdf/1810.04805.pdf [Accessed: 13 January 2020]

[63] W. Yin, K. Kann, M. Yu, H. Schutze. Comparative Study of CNN and RNN for Natural Language Processing. 2017. https://arxiv.org/pdf/1702.01923.pdf [Accessed: 10 January 2020]

[64] Y. Zhang, B. Wallace. A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. 2015. https://arxiv.org/pdf/1510.03820.pdf [Accessed: 10 January 2020]

[65] S. Hochreiter. Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, TU Munich. 1991.

[66] S. Hochreiter and Schmidhuber. LSTM CAN SOLVE HARD LONG TIME LAG PROBLEMS. 1997.

[67] K. Cho, B. Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. 2014. https://arxiv.org/pdf/1406.1078.pdf [Accessed: 13 January 2020]

[68] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa. Natural Language Processing (almost) from Scratch. 2011. https://arxiv.org/pdf/1103.0398.pdf [Accessed: 20 January 2020]

[69] M. Schuster, M. Johnson, N. Thorat. Zero-Shot Translation with Google's Multilingual Neural Machine Translation System. Google Research Blog. https://ai.googleblog.com/2016/11/zero-shot-translation-with-googles.html [Accessed: 13 January 2020]

[70] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. October 2016. https://arxiv.org/abs/1609.08144 [Accessed: 13 January 2020]

[71] X. Chen, C. Cardie. Unsupervised Multilingual Word Embeddings. 2018, https://arxiv.org/pdf/1808.08933v2.pdf [Accessed: 20 January 2020]

[72] M. Schuster, K. K. Paliwal. Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing. 45:2673–2681, 1997.

[73] A. Conneau, G. Lample, M.A. Ranzato, L. Denoyer, H. Jégou. Word Translation Without Parallel Data. 2018. https://arxiv.org/pdf/1710.04087.pdf [Accessed: 20 January 2020]

[74] Google AI team. Massively Multilingual Neural Machine Translation in the Wild: Findings and Challenges. July 2019. https://arxiv.org/pdf/1907.05019.pdf [Accessed: 13 January 2020]