UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Psicologia Generale

Corso di Laurea Triennale in Scienze Psicologiche Cognitive e Psicobiologiche

Elaborato finale

The use of Artificial Intelligence to solve Natural Language Processing problems: the example of BERT in a company project

Relatore
Prof. Marco Zorzi

Laureanda: **Giulia Migliore**
Matricola: **2011021**

Anno accademico 2022-23

Contents

Abstract

Il Natural Language Processing (NLP), o Elaborazione del Linguaggio Naturale, è un campo dell'Intelligenza Artificiale (AI) che negli ultimi anni sta progredendo molto rapidamente. Basti pensare ai recenti sviluppi in ambito dei software per la generazione di testi, come ChatGPT di OpenAI, rilasciato nel novembre 2022.

Lo scopo di questo lavoro è spiegare cos'è il Natural Language Processing, quali sono le sue applicazioni più utilizzate e fornire una panoramica dei principali modelli di apprendimento automatico utilizzati per risolvere compiti di elaborazione del linguaggio naturale. In particolare, si vuole approfondire il funzionamento di modelli quali le Reti Neurali Convoluzionali (CNN), le Reti Neurali Ricorrenti (RNN), tra cui le reti Long-Short Term Memory (LSTM), e i più recenti Transformers, come BERT e GPT. Inoltre, verrà illustrato anche un esempio dell'utilizzo di BERT in un progetto aziendale, svolto durante il mio tirocinio formativo presso Ixly B.V., ad Utrecht, in Olanda. Insieme al team di Data Science di Ixly, ho implementato la loro "Interview App", ovvero un software che registra la conversazione durante un colloquio di lavoro e restituisce un report con le seguenti caratteristiche: i principali argomenti che sono stati toccati, in termini di parole chiave riguardo competenze, motivatori e fattori di personalità; le parole più utilizzate; il numero e il tipo di domande che sono state poste e la corrispondenza di stile linguistico tra il candidato e l'intervistatore. Al momento la "Interview App" è disponibile solo in olandese, quindi è stato mio compito renderla adatta alla lingua italiana. Ciò significa che è stato necessario trovare un corpus italiano di testo parlato, normalizzarlo e addestrare il mio modello con questa nuova raccolta di dati. In seguito sono stati applicati alla lingua italiana tutti i filtri sopra menzionati e infine è stato creato l'algoritmo in grado di analizzare le conversazioni in lingua italiana. Questo algoritmo utilizza BERT per trovare le diverse tipologie di domande all'interno di una conversazione.

Per raccogliere le informazioni per questo lavoro, è stata effettuata una ricerca accurata su Google Scholar. Sono stati consultati anche alcuni dei principali siti web utilizzati in ambito di machine learning e intelligenza artificiale, come GitHub e Hugging Face, per ottenere i codici per eseguire BERT sul mio computer, il codice API di Azure per trascrivere il linguaggio parlato e il sito ufficiale di OpenAI, per ottenere informazioni sui loro prodotti GPT. È stato utilizzato Python come principale linguaggio di programmazione nell'ambiente

di sviluppo di Visual Studio Code. Come dataset principale per la lingua italiana è stato adottato il corpus italiano KIParla, raccolto nel 2019 da una collaborazione tra l'Università di Bologna e quella di Torino.

Preface

This work is aimed to discuss the broad domain of Natural Language Processing (NLP) and the effort that has been made within the field of Artificial Intelligence in order to solve NLP problems.

The first two chapters will come in the form of a meta-analysis of the AI models that have been created so far to solve NLP problems. The third chapter, instead, will talk about a company project which concerns the practical application of an AI model, that is called BERT, which I have been working on during my internship at Ixly B.V.

Chapter 1

Natural Language Processing (NLP)

Natural Language Processing (NLP) emerged in the 1950s as a discipline within Artificial Intelligence (AI), since its goal is to perform language-related tasks like a human. It is not casual if the word 'processing' was chosen [14]. In fact, it would be inappropriate to talk about language understanding. Machines do not understand language, instead, they can only learn how to process all its components to extract relevant information from some given text.

Even though there is a strong AI component, NLP has been influenced by several other disciplines. The most relevant ones are: Linguistics, which helps to explain language structure; Computer Science, which develops efficient models for machine learning to process data; Cognitive Psychology, which provides a link between human and machine performance.

Furthermore, NLP is divided into two different categories: Natural Language Processing and Natural Language Generation. Language processing tasks can be compared with the function of a reader or a listener, whereas language generation tasks aim to perform as a writer or a speaker.

1.1 Levels of language in Linguistics

NLP programs work with different levels of spoken and written language, in order to derive the most complete meaning from it. In [14] Liddy offers an overview of these levels and a brief explanation of the different meanings they convey.
Phonetics focuses on decoding the variations of sound waves in spoken language. It is used to detect fluctuations in intonation, stressed syllables, and pronunciation.

Morphology deals with the components of words, which are the smallest units of meaning, so-called morphemes. Taking as an example the word ‘replacing’, three morphemes can be found, that is, re-, -place-, and -ing. Words differ in meaning based on the morphemes they are composed of. In the previous example, the prefix re- conveys the idea that an action was taken more than once.

Lexical concerns the meaning of each word or a group of words, that is, the vocabulary of a language. ‘Dog’, ‘traffic jam’, and ‘look forward to’ are all examples of lexical features that have a specific meaning.

The syntactic level is used to analyze the grammar and dependency relationships between words within a sentence. In fact, the order of words can affect the meaning of a sentence. For example, ‘The lion eats the zebra.’ and ‘The zebra eats the lion.’ contain the same words, so they have the same lexicon; they only differ in their syntactic structure, which is the one conveying a different meaning.

Thanks to semantics, disambiguation of polysemous words is possible, since at this level the meaning of the entire sentence is processed. The word ‘mouse’, for example, means both the animal or a part of a computer, and its meaning can only be disambiguated by considering the meaning of the full sentence.

Discourse, instead, deals with the meaning of the whole text as a concatenation of its sentences.

Finally, pragmatics focuses on analyzing the context in which the speech is pronounced, in order to recognize hidden meanings that are not explicitly encoded in the sentence. This level requires some extra information, such as the aims and intentions of the speaker.

1.2 Introduction to machine learning

The ultimate goal of NLP is to derive meaning from a given text, in terms of its semantics. Basic NLP tasks consist of decoding verbs, conjunctions, or nouns, and the grammatical relationships between different words and clauses within a sentence. Although, the more accurate we want the model to be, the more rules we have to add, with the risk of them becoming unmanageable and interacting in an unstable way. It was at this point that machine-learning models started to be broadly used, as they are based on probability. These

algorithms are trained with wide corpora of text in many different languages, in order to be able to generalize and make predictions about new data in the future.
In general, there are two main methods to train algorithms, that is, supervised and unsupervised learning. In supervised learning the training data is labeled, so the model is already given the correct answers, while in unsupervised learning the model has to discern the correct patterns by itself. An issue that can occur in both learning strategies is called over-fitting and happens when the model becomes perfect at extracting patterns from the training dataset, but cannot make accurate predictions with unseen data. This happens because it also learns to recognize random noise inside the training dataset. In fact, instead of grasping the essential features, it goes into too much detail that is not relevant. Techniques such as cross-validation can be used to reduce the risk of over-fitting. Through cross-validation, training data are randomly split into training and test sets, to verify the model's prediction while the model is still learning. The sequence of training, test, and validation sets is then reiterated several times until all the dataset has been learned.
Machine learning models can be classified into discriminative or generative models. Discriminative models are used for classification problems, in which they have to assign a certain category to the input data. Differently, generative models are used to create new data based on the patterns they have learned in the training phase.

The branch of NLP that uses the machine learning methods explained above is known as statistical NLP [18]. In the past few years, this field has been substantially growing due to: the large amount of text that is now available on the Internet, which can be used to train models; great progress in the electronic engineering of computers' components, which lead to more capacious memory and faster CPU. Moreover, statistical approaches perform better than rule-based models, because the training is based on copious data from real life, so it is easier to identify the patterns when actual cases are involved.

1.2.1 NLP tasks

Machine learning models allow us to decode human language by means of a variety of tasks, each with different goals. A brief overview of the main NLP tasks is presented below.

Word Segmentation. Word segmentation [23] is the first phase for most of the advanced NLP tasks, because it enables the detection of single words from a string. The detected words are called tokens.

Named Entity Recognition (NER). The aim of NER [32] is to recognize and categorize proper names within a sentence. It is very useful for machine translation, because it avoids literal translations of proper nouns.

Part Of Speech (POS) Tagging. Parts Of Speech (POS) Tagging consists of labeling each word based on its grammatical role in the sentence. Some grammatical categories are verbs, conjunctions, nouns, determiners, adjectives, and pronouns.

Parsing. Parsing tasks [12] consist of analyzing the syntactic structure of a sentence by building a parse tree, which highlights the syntactic role and relations of each word (Fig.1).

Word Sense Disambiguation (WSD). A large number of words are polysemous, which means that they have more than one unique meaning. Models that perform Word Sense Disambiguation [1] are able to give the most appropriate meaning to a word based on the context of the sentence.

Speech Recognition. The goal of Speech Recognition [15] is to convert an audio speech into a written transcripted text. It is the first step for a machine to analyze a spoken message.

Machine Translation (MT). Machine Translation tasks [19] require a machine-learning model to translate some text from a given language to another.

Sentence Completion. Sentence Completion refers to the ability of completing an unfinished sentence by choosing the word that has the highest probability of being the next one (1.2.2).

Information Retrieval (IR). Information Retrieval consists of collecting information from one or more texts and retrieving those information later on. It is usually part of the Question Answering process.

Question Answering. A model that performs Question Answering is able to retrieve information from some previous text in order to answer specific questions about it.

1.2.2 N-grams

A popular approach to natural language processing is n-grams based techniques, which are widely used for text analysis and sentence completion tasks.
N-grams are groups of 'n' words within a sentence and can be gathered by the syntactic relations they have between each other, instead of by the original sequence in the sentence. This type of n-grams are called syntactic n-grams (sn-grams) and are represented with syntactic trees that show the dependency between words. Sidorov et al. in [24] provide an example in which a sentence from the novel "Dracula" by Bram Stoker is analyzed: "I can even now remember the hour from which I dedicated myself to this great enterprise.". The analysis is made by using the Stanford parser that takes the sentence as an input and returns a dependency tree and the syntactic relations between words as an output.

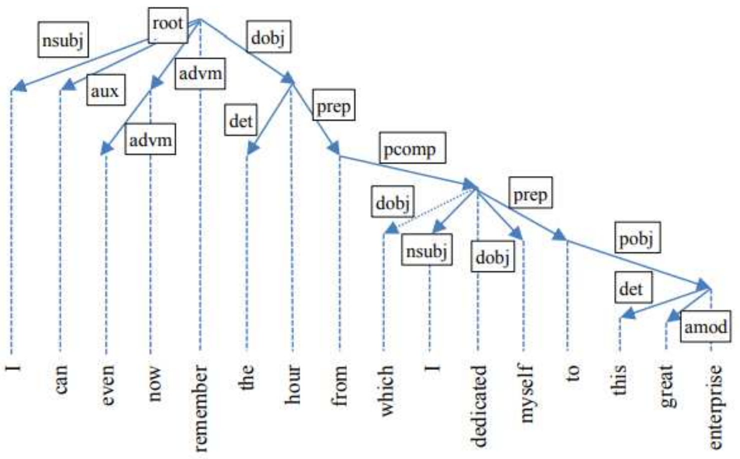

Figure 1.1: Syntactic tree with tags of syntactic relations (Sidorov et al., 2014, *Expert Systems with Applications)*

Thus, the sentence is fragmented into groups, that is, n-grams, which can contain single or multiple words.

As mentioned above, n-grams can be used to predict word probabilities in sentence completion tasks. For example, in the sentence “I really appreciate your” the majority of people would agree that “help” has a high probability to be the next word. To reach this performance, the model has been trained with a wide corpus of sentences, so it could calculate the number of occurrences of every word inside that database. Once it has been trained, the model is able to calculate the probability of each word to be the next one in the given sentence, so it can subsequently choose the word with the highest probability. The word probability is calculated as it follows [24].

$$P = \frac{count\ (w_2 w_1)}{count\ (w_2)} \tag{1.1}$$

This formula can be transferred to the example above.

$$P(help) = \frac{count\ (your\ help)}{count\ (your)} \tag{1.2}$$

In 1.2 the probability that the word “help” follows “your” is the result of the occurrences of “your help” divided by the occurrences of “your” in the training corpus.
In this way the model is capable of predicting sequential words in incomplete texts. Sentence completion is one of the main tasks for text generation models in machine learning.

Chapter 2

Solving NLP problems using Artificial Intelligence: main models

The recent success of Deep Learning largely contributed to developing advanced NLP techniques. This success is due to two main factors: the access to big datasets providing a large quantity of labeled data; the use of GPUs (Graphic Processing Units), which are video cards initially used for gaming. They are an affordable and accessible tool to supply parallel elaboration of information that is faster and more powerful. In fact, instead of calculating the synaptic weights for each neuron of a Deep Neural Network (DNN), GPUs allow machine-learning models to make calculations in parallel. Thus, elaboration times are shortened and the network is able to more easily process large quantities of data.
In this chapter an overview of the most well known deep learning models is provided.

2.1 Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNN) are an example of supervised deep learning networks. As explained in chapter 1.2, this means they receive labeled data as an input during the training phase. Nevertheless, they are trained with a particular method, which is called end-to-end learning. In fact, in the classic supervised learning it is necessary to preprocess the original raw data in order to extract the relevant features that the network has to learn to extract. Differently, with end-to-end learning raw data can be already given to the network, because it is the model itself that extracts the main features to produce the correct output.
CNNs make use of hierarchical information processing, as happens in the human brain. For example, in the visual cortex there are various layers of neurons with receptive fields that become more and more complex (Fig. 2.2).

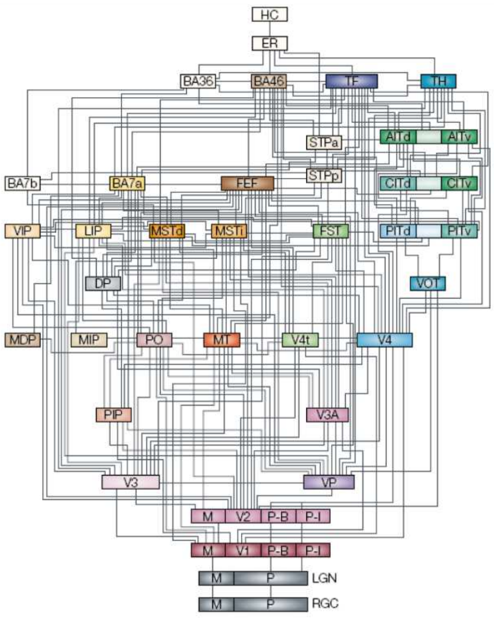

Figure 2.1: Hierarchical organization of the visual cortex (Felleman & Van Essen, 1991, *Cerebral Cortex*)

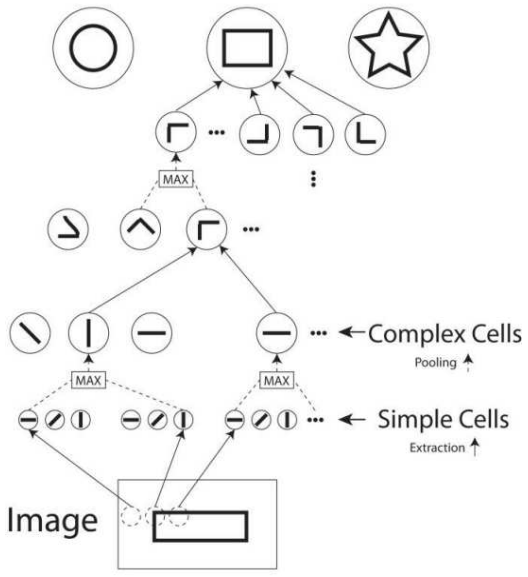

Figure 2.2: Increasing level of complexity within visual receptive fields (Riesenhuber & Poggio, 1999, *Nature Neuroscience*)

Because of the similarity between the visual cortex and the architecture of CNNs, these networks were first used to process images for machine vision tasks. Later on, they were applied to a variety of other problems, including NLP. However, image examples will be used to explain CNNs' architecture.
In [3] Bhatt et al. provide a clear explanation of CNN's architecture.

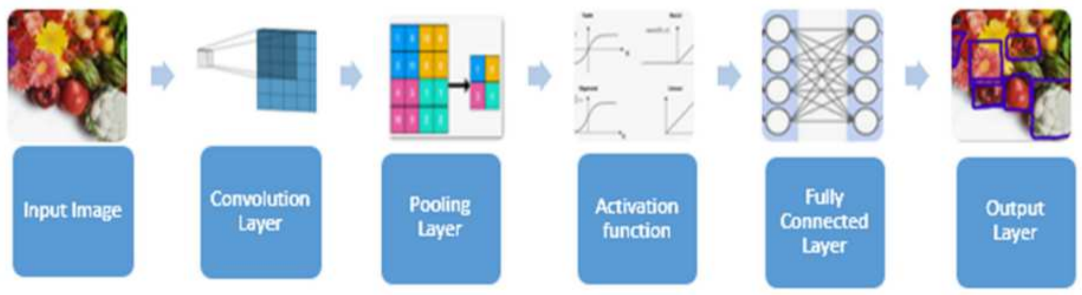

Figure 2.3: CNN components (Bhatt et al., 2021, *Electronics*)

Convolutional layer

CNNs are deep networks with at least one convolutional layer, where the hidden neurons are not fully connected with the previous layer, but with a limited number of previous neurons. This means that each neuron has a local receptive field that decodes one specific feature, which is called filter or kernel. The amount of neurons defines how many features will be represented in each layer. Each filter is applied to the entire image by means of a convolutional operation, which is the operation between the input matrix and the filter matrix. Different filters generate different maps, so called feature maps, of the same image by emphasizing a particular feature.

In CNNs the following hyper parameters can be set. The number of hidden neurons indicates how many filters need to be used in each layer. The kernel dimension specifies the dimension of the receptive field, whereas the stride is the dimension of the filter's moving pace over the image. Padding is used to maintain a stable dimension of the image. In fact, the filters come in the form of a matrix and the target pixel is scanned by the central value of the matrix. In this way, the pixels that are located at the borders of the image can not be processed, otherwise the external values of the filter's matrix would exceed the image size. To overcome the problem, a canvas with additional pixels with a value of 0 is set around the image, so the filter can go over the borders too (Fig. 2.4).

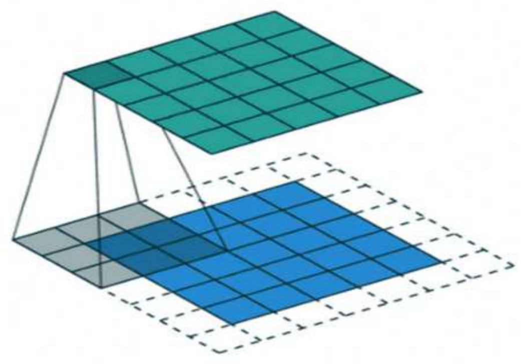

Figure 2.4: Padding (Bhatt et al., 2021, *Electronics*)

Pooling layer

Following the convolutional layer there is a pooling layer (Fig. 2.3) that reduces the image dimensionality and emphasizes the salient features. Thus, the amount of parameters are diminished and the risk of overfitting can be controlled. To do so, in the pooling layer the max pooling operation is utilized as activation function, which only keeps the highest value, so the strongest feature.

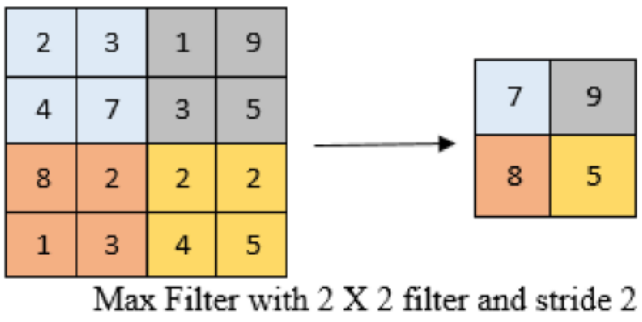

Figure 2.5: Max pooling (Bhatt et al., 2021, *Electronics*)

A similar process happens in the human brain, where neurons compete with each other and the one with the highest activation wins.

There can be more than one convolutional and pooling layer based on how complex the network is (Fig. 2.6).

Fully connected layer

The last layer of the network is fully connected to the output (Fig. 2.3). Here the softmax function is utilized, which returns values that already represent the probability rate. For example, for a value of 0.6 there is a 60% probability that the correspondent output is correct.

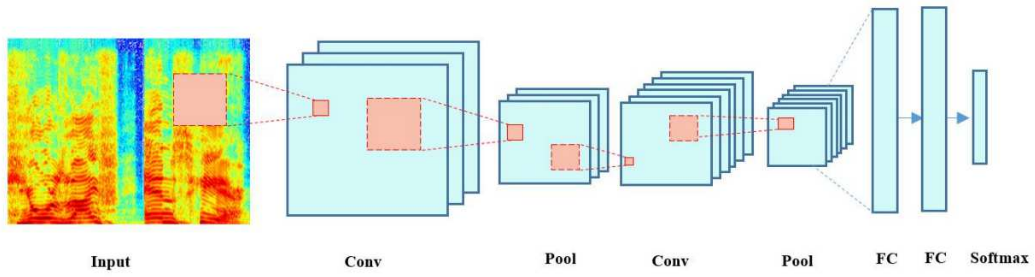

Figure 2.6: CNN typical structure (Musaev et al., 2019, *Proceedings of the 2019 3rd International Symposium on Computer Science and Intelligent Control*)

2.1.1 CNN for Natural Language Processing

It has been demonstrated that CNNs perform well in NLP tasks when the external noise is low. In fact, speech varies substantially based on the speaker's intonation, dialect, tone of voice, and pronunciation. CNNs performance has been studied in several NLP tasks, with a major success in speech recognition (1.2.1).

Ahmadi et al. in [2] have proposed an innovative way to process phonemes by using speech images. In order to give an audio track as an input to the CNN (Fig. 2.8), the speech signal is first filtered to reduce noise and interruptions, and then converted into a spectrogram, which is the graphic representation of the sound intensity over time. The spectrogram is split into multiple sections to give to the model separately (Fig. 2.7).

The CNN is then ready to extract the relevant features from the spectrogram images.

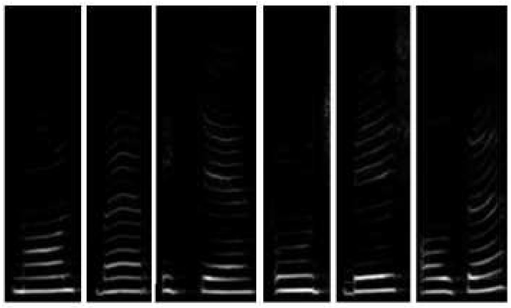

Figure 2.7: Speech spectrograms (Musaev et al., 2019, *Proceedings of the 2019 3rd International Symposium on Computer Science and Intelligent Control*)

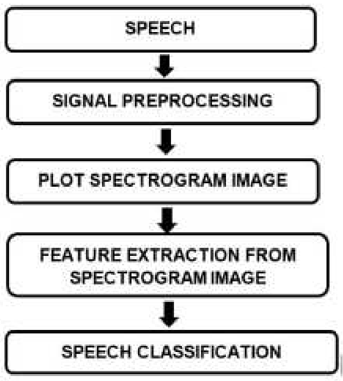

Figure 2.8: Speech recognition process (Musaev et al., 2019, *Proceedings of the 2019 3rd International Symposium on Computer Science and Intelligent Control*)

2.2 Word2Vec

Machine learning models for natural language processing take words as input, but can not process them in their raw form of letters, therefore they have to be preprocessed first. There are different ways in which words can be translated into suitable inputs for a neural network, however the most common technique is called word embedding, which refers to transforming words into spatial numeric vectors by means of special algorithms. The algorithm is trained with wide language corpora that allow the algorithm to create a space full of vectors that decode different words. The vectors that are closer to each other refer to words with similar meanings. Since vectors are numerical, operations between word vectors are possible. An example of proportions is provided in [16] and displayed below.

$$\frac{man}{woman} = \frac{king}{x} \qquad (2.1)$$
$$x = \frac{king \cdot woman}{man}$$

The result of the equation will be a vector that is very close to the one of the word "queen" (2.2)

$$\frac{man}{woman} = \frac{king}{queen} \qquad (2.2)$$

One of the most successful vector algorithms is called Word2Vec, which has been implemented by Mikolov in [16] and is used in a variety of neural networks that perform NLP tasks.

2.3 Recurrent Networks

Recurrent Networks are Deep Learning networks that have a temporal structure. This means that the input arrives sequentially and the output depends both on the current and the previous inputs. Some examples of recurrent network tasks on NLP are provided. For speech recognition these networks can convert a spectrogram into a sequence of phonemes and words that are subsequently processed. Sentence completion is performed by predicting the next word based on the previous ones, which underlies text generation tasks. An interesting example is NETtalk developed by Sejnowski and Rosenberg in [22]. It performs text to speech tasks, so it has been trained to read English words. To read each letter, NETtalk takes seven letters as an input: the previous three, the target letter, and the subsequent three (Fig. 2.9). Eventually, it returns the phoneme of the target letter as an output.

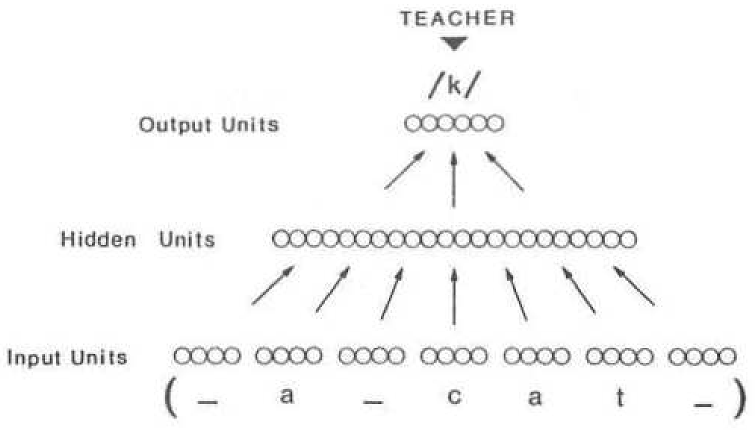

Figure 2.9: NETtalk structure (Sejnowski & Rosenberg, 1987, *Complex systems*)

In this chapter the most well known recurrent networks will be illustrated, together with their specific applications in NLP tasks.

2.3.1 Simple Recurrent Networks (SRN)

Simple Recurrent Networks (SRN) have an internal state, or short memory, which makes them applicable to sequential problems, such as speech prediction, classification, and generation.

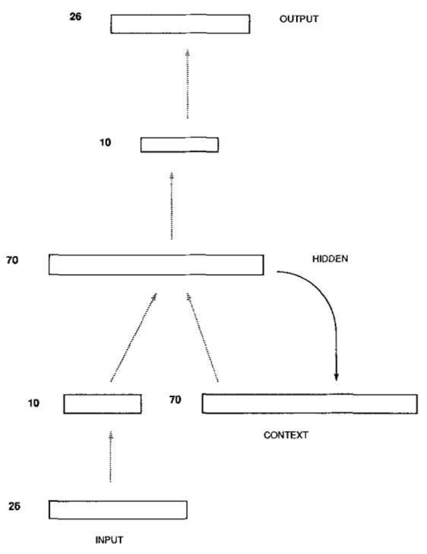

Figure 2.10: SRN architecture (Elman, 1991, *Machine learning*)

These networks have a partially recurrent architecture that is presented in Fig. 2.10. The base is a classic feedforward structure, which connects the input layer with the hidden and the output layers. What makes the difference is that a context layer, identified as C, is added at the same level of the input layer. At the beginning, each context unit (c_1, c_2, c_3, …) has value 0, because there is no history yet. As soon as the network is activated, they take the value of the correspondent hidden neuron. In this way, the following state will be not only influenced by the inputs, but also by the context, which represents the value of the previous interaction.

The most famous example of a SRN is Elman's network, described in [7]. This network was trained with a corpus of English words and it was designed to perform word predicting tasks, receiving a letter as input and predicting the next one until a word was formed. This particular instance is called self-supervised learning, because the input and target are of the same character, in fact they are both letters.

2.3.2 Long-Short Term Memory (LSTM) Networks

RNNs can be considered as short term memory networks, because they can not learn long term dependencies. An effective solution for analyzing data with long-range dependencies is to use Long-Short Term Memory (LSTM) networks, since they can keep information in their memory.
LSTM networks have the standard input and output layer and one or more hidden layers, which are made of LSTM units with a more complex structure. In fact, they include gates, which define if the information has to be maintained in the temporary memory cells and for how long it has to be propagated. Memory cells are the structures that keep the information over time. They are organized into blocks, which contain one or more memory cells. The memory cells of the same block share an input and an output gate.

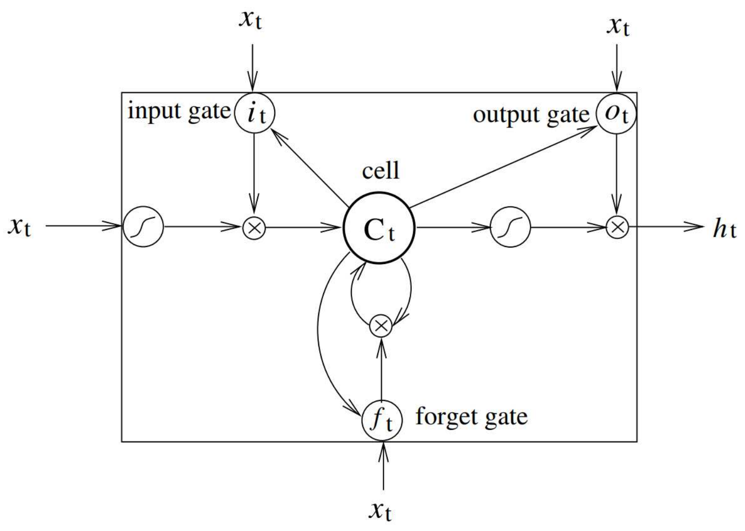

Figure 2.11: LSTM memory cell (Huang et al., 2015, *arXiv preprint arXiv*)

Input gates let the input enter the memory cell. The current value contained in the memory cell can exit from the output gate. Moreover, there is an additional gate, so called forget gate, which can reset the current value of the memory cell. Each gate has its own set of synaptic weights. Throughout the training phase the network learns itself which information enters the gates.

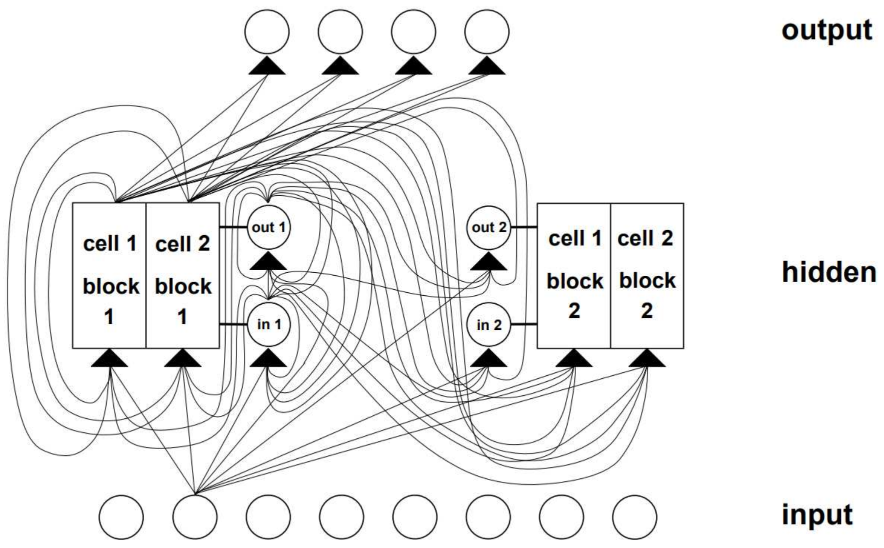

Figure 2.12: Example of a LSTM network with 8 input units, 4 output units, and 2 memory blocks of size 2. There are 2 input gates and 2 output gates. (Hochreiter & Schmidhuber, 1997, *Neural computation*)

An extension of LSTM networks is the bidirectional LSTM (bi-LSTM) networks [9]. This means that the input sequence is read twice, once forwards from left to right and once backwards from right to left. By doing so, the network has access both to past inputs, via the forward states, and to future inputs, via the backwards states. This extension comes out to be very useful in NLP tasks, such as speech recognition, transcription, and POS tagging.

2.3.2.1 LSTM Networks for Natural Language Processing

Speech Recognition and Transcription

Natural Speech Recognizer is a model presented by Soltau, Liao and Sak in [28] and it is designed to solve transcription tasks. It takes acoustic data as input and returns a written transcription of it. This model has a deep LSTM architecture with multiple LSTM layers, using bi-LSTM, and a training corpus that contains both acoustic and written words.

POS Tagging

In [20] a bi-LSTM model has been used to perform POS tagging tasks. The model takes word embeddings as input and returns the appropriate POS tags as output (Fig. 2.13).

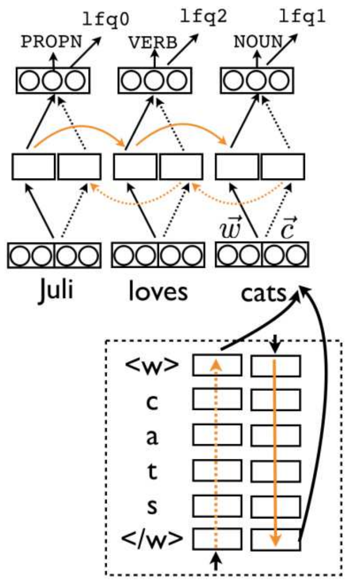

Figure 2.13: Example of a bi-LSTM model assigning tags to the sentence “Juli loves cats” (Plank et al., 2016, *arXiv preprint arXiv*)

2.3.4 Transformers

The Transformer model that has been implemented in [29] and is a sequential model that has shown great efficacy in NLP tasks over recent years, since it can decode long-range dependencies between words. LSTM Networks (Ch. 2.3.2) are also utilized to deal with long-range relations, but even if their memory is longer than a simple recurrent network, it is not enough to compute calculations within extended texts that contain more than a few sentences.

As explained above (Ch. 2.3), RNNs make references by the position of words. For example, there are models that look at the word before to predict the next one in a text. Differently, transformers make references by context, which is closer to what humans do as well. For example, transformer based models can produce some text and cite what they have already generated a few lines before. Referencing by context is possible thanks to Self-Attention functions, where the model makes a query (Q) of vectors and looks for similar ones in the

past, retrieving the most similar keys (K) and the related values (V) and finally produces the output. In this process, queries, keys, values and outputs are all in the form of vectors. Thus, Self-Attention can be defined as a similarity measure, which retrieves the most similar word from a set of words in the past, which means that it doesn't consider the original order of the words. However, in human language the order of words plays an important role and should not be neglected. Multi-Head Attention overcomes this limit by performing multiple attention functions while considering the position of words as well.
Transformers' architecture can be shown as it follows.

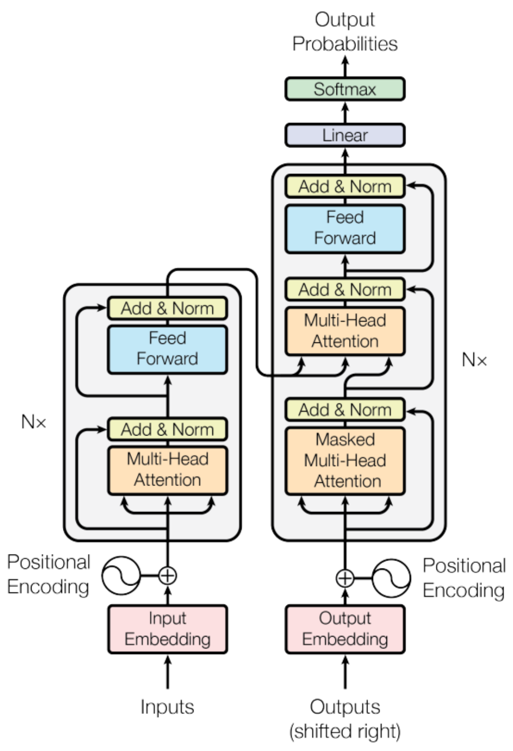

Figure 2.14: Transformer architecture. On the left side there is the encoders block and on the right side there is the decoders block. (Vaswani et al., 2017, *Advances in neural information processing systems*)

The input is first transformed into a vector by using a word embedding algorithm, e.g. Word2Vec (Ch. 2.2). A positional encoding is given, which provides information about the position of the word taken as an input in the text. The input subsequently moves through $N \times$ blocks of encoders and decoders, which utilize attention functions to process the incoming data. Ecoders process all the inputs at the same time and give information to the decoders, which produce sequential outputs.

The initialization of transformers models is divided in two parts: pre-training and fine tuning. First the model is pre-trained with wide corpora of text that can be found online, for example from Wikipedia, newspapers, articles, blogs, and websites. Once the model has learnt general sequences of words from the corpora, it is fine tuned to perform a specific task.
Transformers are suitable models for unsupervised learning, since they can process such big amounts of data during the training phase.

Transformers are the state-of-the-art approach to perform a variety of NLP tasks, such as machine translation, text generation, question answering, and sentiment analysis. Two examples of transformers are presented below, which are currently widely used by companies that make use of artificial intelligence.

2.3.4.1 BERT

BERT refers to Bidirectional Encoder Representations from Transformers and has been implemented at Google AI Language in 2018 [6]. Before this point, an individual model had to be implemented to solve one single task, but BERT revolutionized the NLP world with its ability to solve more than 11 language tasks, such as sentiment analysis, text generation, text prediction, question answering, and summarization. BERT is at the base of many online services, such as Google Translate, voice assistants like Alexa or Siri, and Google Search. For example, since November 2020 it helps Google Search to detect what the user is looking for by using context elaboration and not only keywords anymore. In this way, the results of a research on Google are more accurate and relevant. BERT's success is mainly due to its training dataset of 3.3 billion words taken from Wikipedia and Google's BooksCorpus. These data are unlabeled, in fact BERT is trained with the unsupervised learning method. It's architecture is a multi-layer bidirectional Transformer and is released in the `tensor2tensor` library[1]. The only difference between BERT and a classic Transformer is that BERT doesn't need to use a decoder.

[1] https://github.com/tensorflow/tensor2tensor

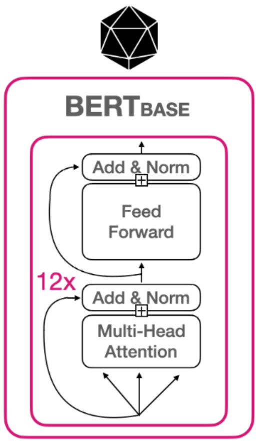

Figure 2.15: BERTbase architecture (Muller, 2022, *Hugging Face*)

BERT's performances in many NLP tasks are incredibly high, outdoing even human performance. For example, the ranking on question answering tasks that has been made in 2018 places BERT at the first place, with a score of 93.16, and the human performance at the second place, with a score of 91.221. In general language understanding tasks BERT obtained an average score of 82.1, whereas OpenAI GPT obtained 75.1.

The biggest advantage that BERT offers is that it is open-source. This means that developers in companies can easily get the code for free on GitHub[2] to build their own BERT model and train it for their specific tasks. Nevertheless, there are thousands of already pre-trained BERT models for different specific tasks, in case companies don't want to fine-tune it.

2.3.4.2 GPT

The GPT (Generative Pre-Training) model was released for the first time in 2018 by OpenAI to solve language generation tasks. Differently from BERT, it is not open-source, since its performance is so high that there is an elevated risk of malicious use. In fact, the output texts are considered credible by humans.

For GPT the OpenAI developers used a multi-layer Transformer decoder, which has been trained by using the unsupervised learning method with the BooksCorpus dataset.

[2] https://github.com/google-research/bert

In 2019 GPT-2 was released to create coherent paragraphs of text. It can support in prose, with tasks like grammar assistance and autocompletion-assisted writing, in poem, with tasks like literary art and poetry generation, and in programming, with code autocompletion tasks. GPT-2 has been trained with the WebText dataset containing 40GB of text, which has not been publicly released.

OpenAI’s work proceeds with the release of GPT-3 in 2020, ChatGPT in November 2022, and the latest version GPT-4 in March 2023. ChatGPT has been trained with Reinforcement Learning from Human Feedback (RLHF). Now GPT-4 provides more accurate help than ChatGPT, thanks to more advanced general knowledge and problem solving skills.

Chapter 3

Applying BERT for a company product: the Ixly example

In this chapter an example of a BERT application in a company project will be provided. In the summer of 2022 I had the chance to do my internship at Ixly B.V., in Utrecht, Netherlands. Together with Ixly's Data Science team, I had to work on "In2dialog"[3], which is an interview app that records any job interview and returns a report with an objective analysis of some specific features of the speech.

3.1 Ixly B.V.

Ixly B.V. was founded in 2005 by Diddo van Zand[4] and provides an e-assessment platform for personnel selection inside organizations. The "Assessment Platform" offers a test toolkit[5] with a variety of personality and intelligence tests that can be used to recruit new employees. In the past two years they have been working on "In2dialog", which is a software that records any job interview and returns a report with the following features: the main topics that have been touched, in terms of competencies, motivators and personality keywords, a wordcloud of the most used words, the number and type of questions that have been asked, and the language style matching between the candidate and the interviewer.

[3] https://in2dialog.com/
[4] https://nl.linkedin.com/in/diddovanzand
[5] https://www.ixly.com/test-toolkit-tests

3.2 Project overview: goals and procedure

Since the “In2dialog” app is currently available in Dutch and English, my job was to make it suitable for the Italian language. This meant that I had to find an Italian corpus of spoken text, normalize it and train my model with this new dataset. Then I had to apply all the filters mentioned above to the Italian language and finally create the pipeline that computes the language analysis for Italian conversations. This pipeline utilizes a modification of BERT to find question types within a conversation.

Python has been used as the main programming language with Visual Studio Code as its development environment. Microsoft Azure API[6] has been utilized to transcribe the input conversation from speech into text. We adopted the Italian corpus KIParla [4] as the main dataset for the Italian language, which has been collected and released in 2019 together by the University of Bologna and Turin. I downloaded the html file containing all the conversations[7], created a python data frame that connected each speaker with the utterances they pronounced, and finally stored the data frame into a csv file, called ‘`KIParla_Corpus.csv`’. Through this csv file I could work separately on each feature of the speech that I had to analyze.

3.3 Question Type Analysis

One of the task that the “In2dialog” app does is Question Type Analysis: when the app recognizes a question in the speech, it marks it with a type label. In particular, we wanted to detect if the questions were ‘who’, ‘what’, ‘how’, or ‘why’ questions.

[6] https://azure.microsoft.com/en-gb/products/cognitive-services/

[7] http://130.136.148.2/bonito/run.cgi/wordlist?corpname=KIP;usesubcorp=;wlattr=doc.full_conversation;wlminfreq=1;wlmaxfreq=0;wlpat=.%2A;wlmaxitems=1000;wlsort=;ref_corpname=;ref_usesubcorp=;wlcache=;simple_n=1.0;wltype=simple;wlnums=docf;include_nonwords=1;blcache=;wlpage=1;usengrams=0;ngrams_n=2;ngrams_max_n=2;nest_ngrams=0;complement_subc=0

The company already collected sample questions from Dutch interviews, matched them with a question type label, and stored them into a csv file. Thus, I used `googletrans`[8], which is the python library used for the Google Translate API, to automatically translate it into Italian and store the data into another csv file, named ‘`question_types_ita.csv`’. Then I used Sentence-BERT [21], which is an adaptation of the traditional BERT network that calculates cosine similarity not between words, but between sentences. I imported the Sentence Transformer[9] library and used the ‘`distiluse-base-multilingual-cased-v1`’[10] model, which is a multilingual model for cosine similarity trained on 15 different languages. To install it and run it on any Python environment it is sufficient to write the following lines of code.

```
import sentence_transformers
from sentence_transformers import SentenceTransformer, util
model=SentenceTransformer('distiluse-base-multilingual-cased-v1')
```

Given an input question, the Transformer calculates the similarity between the input question and all the other questions from the ‘`question_types_ita.csv`’ file, and finds the question within the csv file with the highest similarity score. Then, it takes the corresponding question type of that question and it labels the input question with that type.

[8] https://py-googletrans.readthedocs.io/en/latest/
[9] https://www.sbert.net/docs/pretrained_models.html
[10] https://huggingface.co/sentence-transformers/distiluse-base-multilingual-cased-v1

Chapter 4

Conclusions

The aim of this paper was to highlight the advancement of Natural Language Processing techniques in recent years and to summarize the main machine learning algorithms that have been implemented to solve such tasks up to the ‘state-of-the-art’ models increasingly used by companies for different purposes. Natural Language Processing is a field that a lot of companies in the technology sector have interest in developing and investing in, thus the pace of progress is rapid. It is likely that in a few years most advanced technologies mentioned in this paper will already be outdated. Within the Hype Cycle of Artificial Intelligence, nowadays we collocate ourselves in a ‘hype’ peak, with OpenAI’s ChatGPT and GPT-4, which can perform in a way that nobody could ever imagine.

References

[1] Agirre, E., & Edmonds, P. (Eds.). (2007). *Word sense disambiguation: Algorithms and applications* (Vol. 33). Springer Science & Business Media.

[2] Ahmadi, M., Bailey, N. J., & Hoyle, B. S. (1996, October). Phoneme recognition using speech image (spectrogram). In *Proceedings of Third International Conference on Signal Processing (ICSP' 96)* (Vol. 1, pp. 675-677). IEEE.

[3] Bhatt, D., Patel, C., Talsania, H., Patel, J., Vaghela, R., Pandya, S., ... & Ghayvat, H. (2021). CNN variants for computer vision: history, architecture, application, challenges and future scope. *Electronics*, *10*(20), 2470.

[4] Caterina, M., Silvia, B., Goria, E., Cerruti, M., & Francesco, S. (2019). KIParla corpus: a new resource for spoken Italian. In *CEUR WORKSHOP PROCEEDINGS* (pp. 1-7). SunSITE Central Europe.

[5] Church, K. W. (2017). Word2Vec. *Natural Language Engineering*, 23(1), 155-162.

[6] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv*:1810.04805.

[7] Elman, J. L. (1991). Distributed representations, simple recurrent networks, and grammatical structure. *Machine learning*, 7, 195-225.

[8] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.

[9] Huang, Z., Xu, W., & Yu, K. (2015). Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv*:1508.01991.

[10] Hugging Face (2022, December 16) *Gpt2*. Retrieved by https://huggingface.co/gpt2

[11] Hugging Face (2022, March 2). *BERT 101 State Of The Art NLP Model Explained.* Retrieved by https://huggingface.co/blog/bert-101

[12] Jaf, S., & Calder, C. (2019). Deep learning for natural language parsing. *IEEE Access*, *7*, 131363-131373.

[13] Koutnik, J., Greff, K., Gomez, F., & Schmidhuber, J. (2014, June). A clockwork rnn. In *International conference on machine learning* (pp. 1863-1871). PMLR.

[14] Liddy, E. D. (2001). *Natural language processing*.

[15] Lu, X., Li, S., & Fujimoto, M. (2020). Automatic speech recognition. *Speech-to-Speech Translation*, 21-38.

[16] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

[17] Musaev, M., Khujayorov, I., & Ochilov, M. (2019, September). Image approach to speech recognition on CNN. In *Proceedings of the 2019 3rd International Symposium on Computer Science and Intelligent Control* (pp. 1-6).

[18] Nadkarni, P. M., Ohno-Machado, L., & Chapman, W. W. (2011). Natural language processing: an introduction. *Journal of the American Medical Informatics Association*, *18*(5), 544-551.

[19] Nakazawa, T., Yu, K., Kawahara, D., & Kurohashi, S. (2006). Example-based machine translation based on deeper NLP. In *Proceedings of the Third International Workshop on Spoken Language Translation: Evaluation Campaign*.

[20] Plank, B., Søgaard, A., & Goldberg, Y. (2016). Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. *arXiv preprint arXiv*:1604.05529.

[21] Reimers, N., & Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv*:1908.10084.

[22] Sejnowski, T. J., & Rosenberg, C. R. (1987). Parallel networks that learn to pronounce English text. *Complex systems*, *1*(1), 145-168.

[23] Shao, Y., Hardmeier, C., & Nivre, J. (2018). Universal word segmentation: Implementation and interpretation. *Transactions of the Association for Computational Linguistics*, *6*, 421-435.

[24] Sidorov, G., Velasquez, F., Stamatatos, E., Gelbukh, A., & Chanona-Hernández, L. (2014). Syntactic n-grams as machine learning features for natural language processing. *Expert Systems with Applications*, *41*(3), 853-860.

[25] Solaiman, I., Clark, J., & Brundage, M. (2019). GPT-2: 1.5 B Release. *OpenAI*. Available online at https://openai. com/blog/gpt-2-1-5b-release/, checked on, 11(13), 2019.

[26] Solaiman, I., Brundage, M., Clark, J., Askell, A., Herbert-Voss, A., Wu, J., ... & Wang, J. (2019). Release strategies and the social impacts of language models. *arXiv preprint arXiv*:1908.09203.

[27] Solaiman, I., & Dennison, C. (2021). Process for adapting language models to society (palms) with values-targeted datasets. *Advances in Neural Information Processing Systems*, 34, 5861-5873.

[28] Soltau, H., Liao, H., & Sak, H. (2016). Neural speech recognizer: Acoustic-to-word LSTM model for large vocabulary speech recognition. *arXiv preprint arXiv*:1610.09975.

[29] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

[30] Wang, C., Li, M., & Smola, A. J. (2019). Language models with transformers. *arXiv preprint arXiv*:1904.09408.

[31] Wang, W., & Gang, J. (2018, July). Application of convolutional neural network in natural language processing. In *2018 international conference on information Systems and computer aided education (ICISCAE)* (pp. 64-70). IEEE.

[32] Webster, J. J., & Kit, C. (1992). Tokenization as the initial phase in NLP. In *COLING 1992 volume 4: The 14th international conference on computational linguistics*.

[33] Yin, W., Kann, K., Yu, M., & Schütze, H. (2017). Comparative study of CNN and RNN for natural language processing. *arXiv preprint arXiv:1702.01923*.

[34] Zweig, G., Platt, J. C., Meek, C., Burges, C. J., Yessenalina, A., & Liu, Q. (2012, July). Computational approaches to sentence completion. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 601-610).

Acknowledgments

I would like to express my gratitude to Ixly B.V., especially to its Data Science team, that let me work with them on their new project and trained me on the use of different machine-learning models to solve NLP tasks.
Thanks should also go to my family, that made it possible for me to go to the Netherlands to do my internship, which helped me deepen the knowledge about my thesis topic.