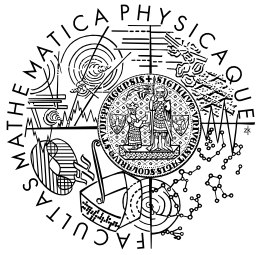


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Bc. Petra Vysušilová

Czech NLP with Contextualized Embeddings

Institute of Formal and Applied Linguistics

Supervisor of the master thesis:	RNDr. Milan Straka, Ph.D.
Study programme:	Computer science
Study branch:	Artificial intelligence

Prague 2021

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ..... signature of the author

Dedication.

Title: Czech NLP with Contextualized Embeddings

Author: Bc. Petra Vysušilová

Institute: Institute of Formal and Applied Linguistics

Supervisor: RNDr. Milan Straka, Ph.D., Institute of Formal and Applied Linguistics

Abstract: Recently, several methods for unsupervised pre-training of contextualized word embeddings have been proposed, most importantly the BERT model (Devlin et al., 2018). Such contextualized representations have been extremely useful as additional features in many NLP tasks like morphological or syntactic analysis, entity recognition or text classification.

Most of the evaluation have been carried out on English. However, several of the released models have been pre-trained on many languages including Czech, like multilingual BERT or XLM-RoBERTa (Conneau et al, 2019). Therefore, the goal of this thesis is to perform experiments quantifying improvements of employing pre-trained contextualized representation in Czech natural language processing.

Keywords: key words

Contents

Introduction

This work aims to improve selected NLP tasks for Czech with the use of recently published state-of-the-art (SOTA) techniques.

In famous Deep Learning book (Goodfellow et al., 2016), NLP is defined as "...the use of human languages, such as English or French, by a computer.". NLP brings a variety of problems to solve from oral–written language conversion, machine translation, syntax analysis used for automatic grammar correction as well as it serve as a base for further linguistic processing. Semantic analysis includes in addition to already mentioned machine translation other tasks like sentiment analysis, natural language text generation or recognition of homonymy or polysemy of given words. It could solve sophisticated assignments as answering questions about the input text document.

This work applies the most successful NLP methods (transfer learning of multilingual bidirectional language model) of recent years to Czech NLP task. Such multilingual models are trained on a big bunch of data in different languages. In the case of applied models, one of training languages is Czech, although it forms only a small part of the data. Selected tasks are tagging, lemmatization and sentiment analysis. Tagging and lemmatization represent syntax analysis, in contrast with sentiment analysis which is a part of semantic task. Task were chosen from both semantics and syntax to show how pre-trained multilingual language models can help with different types of NLP tasks. This work builds on previous work on tagging and lemmatization contextualized embeddings (Straka et al., 2019a) and aims to state new SOTA results at least for syntactic tasks.

The work is divided into six chapters. First chapter presents theoretical background in NLP and used Artificial Intelligence (AI) methods. This quite general chapter is followed by a thorough description of implemented tasks, each in its own separate chapter (Lemmatization and POS tagging, Sentiment analysis, Language modelling). Description of each task includes its definition, previous work and state-of-the-art results, methods applied in this work and their results. Implementation details like code overview, third-party libraries, technical problems and their solution can be found in chapter 5. For personal trying and exploration of presented models serves user documentation of attached models in chapter 6 and text is closed by a discussion about results (chapter 7) and a conclusion with future work proposals.

1. Theory

This chapter is divided into three parts. The first part introduces basic concepts of linguistics and natural language processing. With the focus of this work in mind, deep learning basics are presented in the second part. The third part of this chapter offers a more detailed explanation of methods directly relevant to this work (especially the BERT model).

1.1 Linguistics and NLP

Natural language processing can be described as a science at the border of linguistics and computer science. However, according to (Wilks, 2005), NLP itself is not a scientific research subject. It is instead a collection of problems, which can be examined. These tasks are drawn from the general linguistics field, and the goal is to solve (or *process*) them by computers. General linguistics studies and describes the language. Its focus can be divided into the following sub-fields: phonetics, phonology, morphology, syntax, semantics, and pragmatics. This work focuses on morphology (lemmatization, POS tagging) and semantics (sentiment analysis). A more detailed description of tasks can be found in the following subsections and dedicated chapters for each task. Apart from the introduction of tasks, this chapter also includes a brief NLP history overview and a presentation of possible data sources for NLP training.

1.1.1 Morphology

Morphology studies an internal structure of words. Morphology is used for many tasks, which can be divided into generative and analytical ones. Generative tasks focus on the generation of word forms for a given grammatical category. These forms can be useful for machine translation, full-text searching, labeling documents by their keywords, or spell-check. The result of morphological analysis is a set of possible lemmas and tags for a given word form. This set can be further narrowed by lemmatization (selection of the one "right" lemma), morphological tagging (selection of one correct tag in the given context), or partitional disambiguation (removes all tags and lemmas, which can be reliably removed).

Lemmatization task

Lemmatization task consist of finding a *lemma*. Lemma is one chosen form of a word, selected to represent the whole set of all possible word's forms (such set is called lexeme). A convention chooses word form used as lemma – it is nominative of singular for a noun, an infinitive for a verb, etc.

For example, lexeme for a Czech word *jablko* is *jablko, jablka, jablku, jablkem, jablek, jablky, jablkům* and the lemma is *jablko*.

Part-of-speech tagging

Part-of-speech tagging classifies word into one of the part-of-speech categories (like a noun, pronoun, or verb) (Hladká). Part of tagging can also be a determi-

nation of grammatical categories (e.g., case, number, or tense).

1.1.2 Semantics

Semantics deals with word meaning. This is a more challenging study than morphology, even for humans, let alone for computers. Word meaning can be subjective, change during historical periods, a sentence is not a simple sum of meanings of its words, and many other problems come with learning computers to truly understand a human language.

Semantic analysis can be useful for various tasks, from natural language text generation to recognizing homonymy or polysemy of given words. It could solve sophisticated assignments as answering questions about the input text document or help in high-quality translation. Another possible task is to find in the text so-called named entities - like persons, months, or cites - or linking these entities to some knowledge base.

Sentiment analysis

An input of sentiment analysis is a text, and the output is a classification into one of the categories. In this work, categories are positive, negative, and in some datasets also neutral, but it is common to use labels like abusive or ironic, too. As a part of sentiment analysis can be involved so-called subjectivity(Montoyo et al., 2012). The subjectivity prediction goal is to classify if the opinion (both positive or negative) is objective or the author is personally interested and has strong emotions about his claims. For example, the following text could be recognized as objective: "The sound of this notebook is clear.", "The base is not stable enough." or even "An internet connection in this area is bad." in contrast with "I hate the way the new touchpad works.". The subjectivity of the claim does not depend on its sentiment. This part of sentiment analysis was not included in this work, and it is mentioned just for completeness.

1.1.3 Language data

Data of many kinds can serve as an input into natural language processing, and this data can be categorized by a form or by a source. As for form, we can work with corpora or datasets of various sizes, containing data from many sources. A corpus is a large collection of texts, aiming to be a representative sample of a language. This is not entirely possible, mainly because the selection of examples in the corpus is limited compared to language diversity. Despite these limitations, corpora are a valuable source of language information and are widely used in NLP. The most famous linguistic corpora are co-called Brown corpus (Francis and Kucera, 1979) – first electronic corpus mixed from newspaper articles and fiction literature, and PennTreebank (Marcus et al., 1993), which is the first syntactically annotated corpus but has quite a domain limited source - articles from Wall Street Journal. The internal structure of corpora can be of many types. One of corpora's type is a treebank. A treebank is a parsed corpus with many possible types of annotations and uses trees to represent dependencies. An example of such trees can be seen in figure 1.1 and an example of a treebank is presented in picture 1.2. The question arises as to what is the difference between

a corpus and a "simple" dataset. Sometimes these terms can be interchangeable in the sense that the usage of both can be the same. Both types of data can be used for the same task, but the difference can be seen in a purpose of a collection. A corpus's idea is to collect a somehow representative sample of a language with annotations on many levels, which allows the performance of various analysis upon this data. Dataset is typically created on a restricted domain (like tweets on US Airlines pages [1] or movie reviews [2]) Moreover, they are annotated for one type of task (in this case, sentiment).

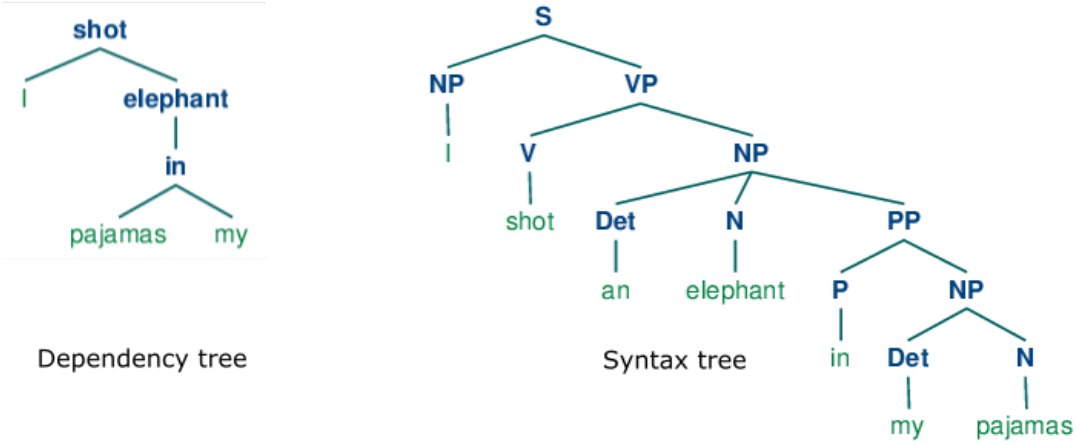


Figure 1.1: An example of the syntax and dependency tree. The dependency tree, as the name indicates, describes dependencies between words. Such dependencies are of various types; for example, an elephant in the example text is a direct object of the shooting action. A root of such a tree is typically a predicate of the sentence. On the other hand, the syntax tree represents the sentence's syntactic structure according to the grammar. The root of the tree is *sentence*, which is split into noun and verb phrase. These can be further divided into phrases compound from particular instances of parts of speech (e.g., nouns, adverbs, verbs, prepositions, etc.).
Source: Bird et al. (2009)

Data in both dataset and corpora can come from many written or oral sources. For machine translation, documents with many language versions are appropriate. An example of the use of such multilingual documents was a project Eurotra (Oakley et al., 1995). Linguistic data can, however, differ in quality and length. A relatively new source of data are social networks like Twitter, Reddit, or Facebook. Data from some social networks (Facebook, Twitter) are very different from traditional sources like scientific papers, newspaper articles, or books. These data are short snippets of text full of odd characters, newlines, and ends of lines. They contain pictures, emojis, a mixture of different languages, slang expressions, and grammatical errors. Furthermore, they are very short; sometimes, they consist only of one sentence, few hashtags, and a link or a picture. Because people share their opinions and emotions and are ready to make their choices according to

[1] https://www.kaggle.com/crowdflower/twitter-airline-sentiment
[2] https://www.kaggle.com/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews

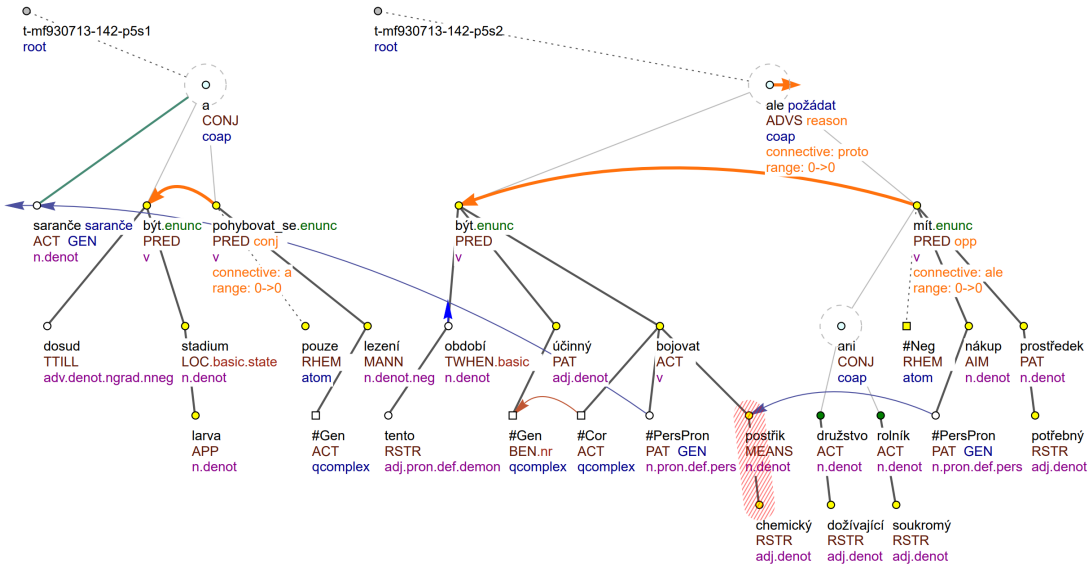


Figure 1.2: Prague dependency treebank example PDT35 for the sentences: *Grasshoppers are still in the larvae stadium, crawling only. At this time of the year, it is efficient to fight them using chemicals, but neither the ailing cooperatives nor private farmers can afford them.* czech: *Sarančata jsou doposud ve stadiu larev a pohybují se pouze lezením. V tomto období je účinné bojovat proti nim chemickými postřiky, ale dožívající družstva ani soukromí rolníci nemají na jejich nákup potřebné prostředky.* This treebank contains dependency trees, but is is just one of many possibilities. This example is from Prague dependency treebank, which offers different layers of annotations. Red strips over words *chemický* and *postřik* marks multiword phrase, conjunction between *rolník* and *družstvo* is expressed as by one type of nodes, blue lines denotes coreference etc.
Praque dependency treebank data are used for tagging and lemmatization tasks.

incoming influences, social networks are an important source of information. The big problem while analyzing this data is their amount. Only a twitter uses produce about 12 TB of data per day [3]. It is impossible to process all the data manually, so this is one reason for the rasing industry importance of natural language processing.

1.1.4 Historical Development

The historical development of computer linguistics was significantly affected by machine translation (Wilks, 2005). Many essential milestones in machine translation history also improved other parts of NLP, so I will briefly present its historical development. First attempts to translation formally started in 1933 by patents for machine translations (mechanical multilingual dictionaries) (Hutchins) followed by a big boom of machine translation in the 50s and 60s and then continued by a slowdown after ALPAC report in 1966 (Hutchins, 1996).

The first solutions to machine translation tasks were based on bilingual dictionaries and sets of rules. This method translated an individual word or a small group of words with a subsequent improvement of syntax and morphology. The resulting translation was not good and required lots of work of human work of expert linguists. This approach was replaced around the year 1990 by a statistical translation (Brown et al., 1990). A statistical machine translation's central idea is a probability of a translated sentence, given the original sentence. This includes also a probability of resulting sentence in the target language. This probability distribution is called a language model, and although it is one of the most aged ideas in NLP, it is an integral part of current best NLP solutions. Words probabilities were originally computed using frequencies of words or their sequences (n-grams) in large language corpus (Jurafsky and Manning, 2012) (for more information about probabilistic language models, see subsection 1.2.7). These methods were quite successful, but they suffered from the curse of dimensionality (Goodfellow et al., 2016, p.450). The next stage of machine translation (and other NLP tasks) starts with the second wave of neural networks' popularity. There are too many possible words or n-grams of words, and there is no way to share learned information between similar words or sequences in statistical methods. A solution to this problem could be a paper (Bengio et al., 2003), where authors presented neural trained word embeddings (see 1.2.3). Neural language models obtained even better results (a neural network learns probabilities of words) (Schwenk et al., 2006). Current natural language processing is built on deep recurrent neural networks and encoder-decoder architecture, firstly published in (Cho et al., 2014), (Sutskever et al., 2014) and (Wu et al., 2016). In current times, neural networks are applied to machine translation and almost any other linguistic tasks. State-of-the-art result for machine translation, sentiment analysis, and many others hold by methods based on deep learning [4]. For that reason, the following subsection presents deep learning basics, their usage, and improvements in NLP in recent years.

[3] https://bigdatashowcase.com/how-much-big-data-companies-make-on-internet/
[4] http://nlpprogress.com/

1.2 Deep Learning

1.2.1 Deep Learning History

The history of learning algorithms inspired by a human brain started in the 1940s under the name *cybernetics* (Goodfellow et al., 2016), (McCulloch et al.). The first such architecture was a perceptron (Rosenblatt, 1958). Perceptron, found in 1958, is the most straightforward neural network with just one layer serving for binary classification (see Fig. 1.3). A perceptron's input is a vector of features describing an input example, and output is classification into class 0 or 1.

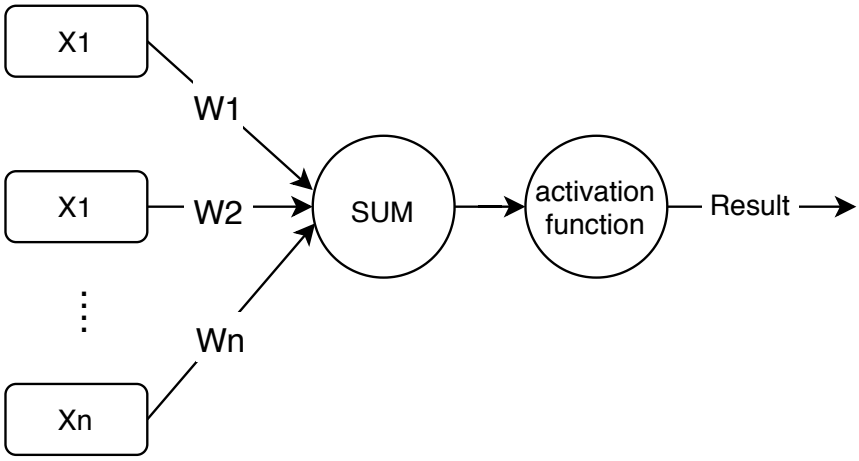


Figure 1.3: A one-layer perceptron architecture. The result is formed by the application of the activation function on a weighted sum of inputs. Weights are updated during training till it returns satisfactory results.

There also exists a multi-class version for the general classification. The problem of the perceptron was the inability to classify data that are not linearly separable (Minsky and Papert, 2017) (see Fig. 1.4), which led to a lack of interest in deep networks for some period.

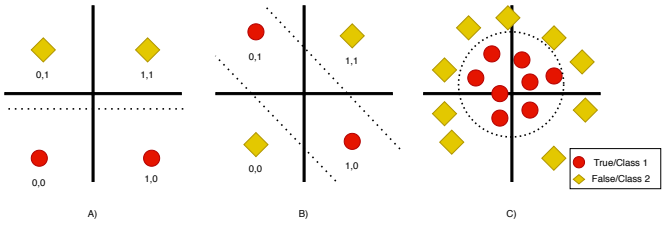


Figure 1.4: This picture illustrates XOR problem. Perception can find the correct solution only if the data are linearly separable. It means that they can be divided by a hyperplane. An example of such two-dimensional data can be seen in picture A). The dotted line shows a possible border for separation. Picture B) shows XOR problem. XOR is a logical operation on two boolean variables, which returns true if one variable is True (1) and the other one is False (0), and returns False otherwise. Such data cannot be separated by one hyperplane. Linearly non-separable data can be, for example, separated by a circle (pic. C)

The era of *deep* neural networks started around the year 2007 (Goodfellow et al., 2016) with bigger datasets and greater computational resources. These two

new features opened the possibility of neural network learning without expertly handcrafted parameters tuning with good results. Many ideas which are currently frequently used are quite old - like backpropagation (Rumelhart et al., 1986) or even encoder-decoder architecture (Allen, 1987), (Forcada and Ñeco, 1997). However, they became popular only after the development in other computer science areas reached a level where they can be trained in a reasonable time.

The basic type of DNN is a multilayer perceptron (see picture 1.5). It is combined with neurons; every neuron has an *activation function*, which is applied to its input. Input to every, but the first layer is a weighted combination of (possibly selection of) neurons from a previous layer. Different NN types differ by number and shape of layers, activation functions, and connections between neurons. (see Fig. 1.5).

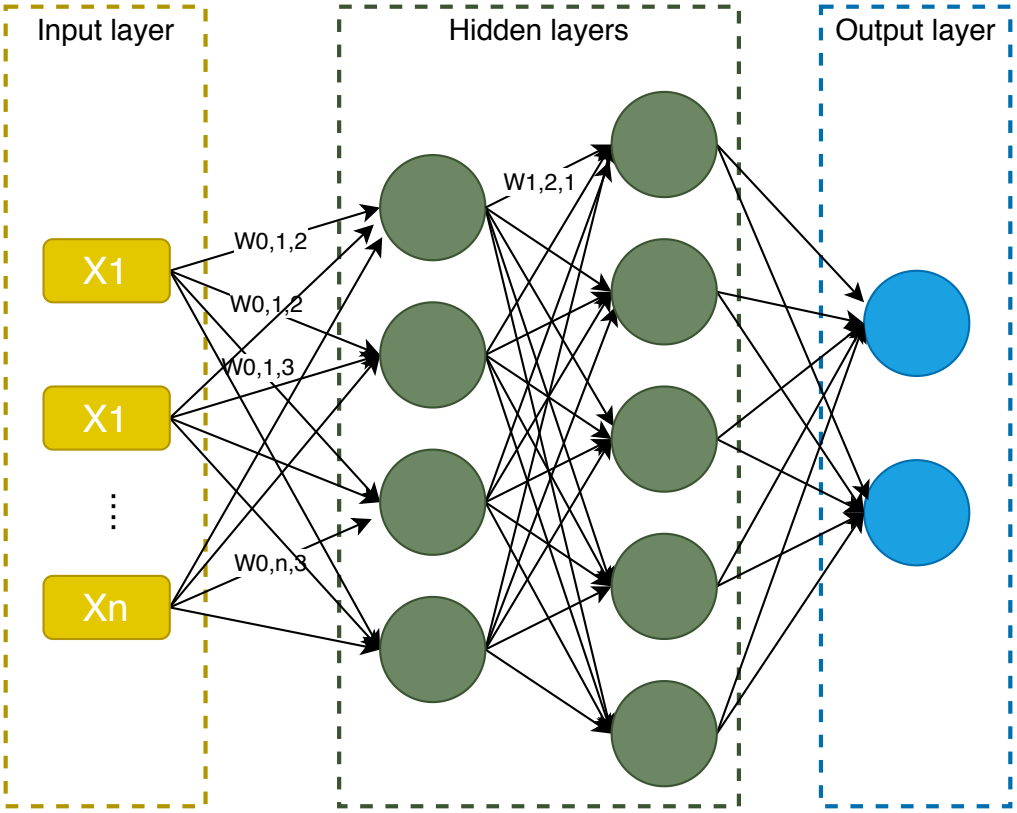


Figure 1.5: Multilayer perceptron (or feed-forward neural network) is formed input and output layer and a variable number of hidden layers with different sizes. In every layer, the chosen application function is applied to a weighted sum of inputs from the previous layer.

The following subsections present some of the key ideas for (not only) NLP and specifically for BERT. In NLP, the same as in other machine learning methods, it is necessary to decide how to encode the input to make it processable by computers, especially which input features are important for the given task and should be included. These questions are in NLP addressed, among others, by two techniques: word embeddings (see section 1.2.3) and attention mechanism (see section 1.2.4). Word embeddings deal with the representation of words and their meaning, while the attention mechanism determines which parts of the text

are relevant for a given task. Many machine learning methods are applied to data with no defined order between samples, like images or descriptions of petals for each sample flower[5]. Language data are, however, different because their nature is sequential. Word and sentence ordering is an essential part of the text, and lack of it can make text absolutely nonsense. This problem can be more or less satisfactorily handled by Recurrent Neural Networks (section 1.2.5) and Transformers architecture (section 1.2.6). Combining all these methods led to a BERT models family, which are used for this work (see section 1.3).

1.2.2 Machine Learning and Regularization

According to (Mitchell, 1997), machine learning is: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." For the techniques examined in this work, experiences are language data – every single experience is a text. To provide some measure, it is needed to know the ML algorithm's correct output. Many deep learning methods are constructed as *supervised*, which means that every experience has a corresponding label with a correct response (so-called gold data). Metrics used in this case reflects the portion of correctly predicted labels. (Russell et al., 1995) Such data structure is handy for machine learning, but it is hard to obtain them in the required amount because they are usually created manually by humans [6]. Opposite to this approach are *unsupervised* methods, where no labeled data exists, and metrics are based on different result features (e.g., the compactness of resulting groups). As for tasks, it is possible to distinguish them by the desired outcome between two basic categories: classification and regression. The classification consists of sorting data into one of the predefined classes (e.g., noun, adjective, verb); meanwhile, regression's goal is to predict a numerical result (e.g., expected number of borrowed books in a school library this year).

To be precise supervised machine learning goal is not to predict all labels correctly in an example data (it would be enough to memorize them), but to predict correctly all possible inputs from the distribution example data are taken from (find some general features for correct performance). During training, there can appear a problem called *overfitting*. As showed in Figure 1.6, overfitting problem is that the result prediction function practically memorized all training data examples and can minimize the error on them very nicely, but probably will not perform well on previously unseen data. A regularization is a tool for preventing such issues. Well known regularization techniques like lasso regression (Tibshirani, 1996) or ridge regression (Hoerl and Kennard, 1970), which are used for linear regression, works by adding some new members into the sum for the loss function, which should be minimized. Another classification regularization method, which is also used in this work, is label smoothing. Label smoothing (Szegedy et al., 2015) is an idea applicable to every classification problem, therefore is not limited to an NLP only. In any classification task, training data contains labels of the correct classes. In binary classification or one-hot encoded

[5]https://archive.ics.uci.edu/ml/datasets/iris

[6]Obviously, if it were already possible to create labels by computers, there would not be necessary to learn it.

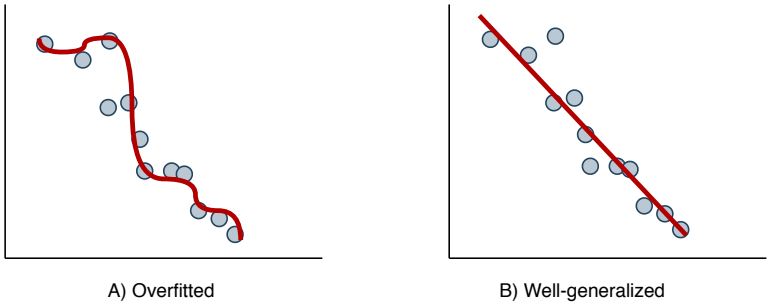


Figure 1.6: Figure A) presents overfitting scenario. Figure B) illustrates possible well-generalized solution.

labels, correct class is denoted by 1 and incorrect class(es) by 0 [7]. Instead of this, label smoothing applies following formula:

$$y_{new} = (1 - \epsilon) * y + \alpha/K$$

, where K is number of classes. Label smoothing is used as a cure for overfitting and overconfidence in the case of use of a softmax as output activation function. Loss function for softmax classification is:

$$loss = -\sum_{i=1}^{n}\sum_{y=1}^{K} p(y \mid x_i) log\, q_\theta(y_i \mid x_i)$$

and after substitution of label smoothing[8]:

$$loss_{ls} = -\sum_{i=1}^{n}\sum_{y=1}^{K}[(1 - \epsilon)p(y \mid x_i) + \epsilon u(y \mid x_i)] log\, q_\theta(y \mid x_i)$$

, which gaves after multiplication [9]:

$$loss_{ls} = \sum_{i=1}^{n}(1 - \epsilon)[-\sum_{y=1}^{K} p(y \mid x_i) log\, q\theta(y \mid x_i)] + \epsilon[-\sum_{y=1}^{K} u(y \mid x_i) log\, q\theta(y \mid x_i)] \tag{1.1}$$

From equation 1.1 can be seen, then if the network is very confidential about some prediction, the second part of the loss function is very large, so label smoothing works as a regularization of overconfidence.

1.2.3 Embeddings

Good performance of NLP models relies on a text representation. What is hypothetically deserved is to teaching computers to understand the semantics of the language. Once the computer has a good representation of what given text *means*, it should be easy to answer questions, translate it into another language,

[7] One-hot encoding transforms each label into a vector of size K, where n is a number of all possible classes. Than such vector is zero at all position expect the c-th position, where c is the correct class.

[8] Taken from: https://leimao.github.io/blog/Label-Smoothing/

[9] see footnote 8.

etc. Because NN can work only with a numerical representation of inputs, the second requirement upon such language representation is to be numerical. The straightforward way is to represent input words in one-hot encoding. In one-hot representation, a single word is represented by a vector, and one is only at the position of the respective word; all other positions are zeros. Such vector is long as a count of distinct words in a dictionary, therefore it could be quite large. The most important problem of one-hot representation is that every two words similarly distant from each other. It can be an advantage in some areas, but it is not a correct assumption in linguistics. Words can have similar meanings or be opposite to each other, generally spoken, a distance between them is not uniform.

When compared to one-hot encoding, embeddings (Bengio et al., 2003), (Ling et al., 2016) are better solution for language data. Embedding is also a vector representing an input word, but in contrast to one-hot encoding, its size does not depend on the vocabulary size. These embeddings are not handcrafted by humans, NN learns them. They can be learned for every specific task from scratch, or it is possible to use embeddings trained for usage in many tasks like in the following cases.

Non-contextualized embeddings

Before contextualized embeddings appeared, pretrained embeddings were created mainly by Word2Vec (Mikolov et al., 2013), (Turian et al., 2010), (Pennington et al., 2014), specifically by its two variants: CBOW and SkipGram model. CBOW model's objective is to predict a masked word from its context, and SkipGram does precisely the opposite - predicting the context of the given word. These predicting objectives serves just as a tool for forcing a network to learn a useful word representation. Embeddings are then imputed into a network as an embedding layer with size $number\_of\_words \times embedding\_size$. We still need to bridge the gap between text and numbers, so it is possible to use one-hot encoding or simple word numbering as an input into this first embedding layer. Word2Vec-like embeddings use just a few context words, and they do not explicitly use information about statistics in the whole dataset. GloVe (Pennington et al., 2014) embeddings, on the other hand, uses information about frequencies of pairs of words in a whole dataset and are designed to project word vectors into meaningful vector space.

Embeddings of previously described types use a context of the word – it is, in fact, the way how they are meant to work. Similar words are supposed to appear frequently in a similar context. The problem of such embeddings is that an input word embedding in the first layer of the neural network is computed independently on neighbor words, so the same word always has the same embedding regardless of the context. In the case of homonyms, non-contextualized embeddings are a mixture of all the (possibly very different) meanings, which can lead to poor results. The problem with same word embedding in a different context is a problem in the case of homonyms, but also in other cases, contextualized embeddings seem to give better results (Straka et al., 2019c), (Liu et al., 2020).

Contextualized embeddings

To solve the problem of different word meanings, contextualized embeddings were invented in recent years, namely BERT (Devlin et al., 2019), ELMo (Peters et al., 2018) and XLNet (Yang et al., 2019a). Their comparison can be found in a subsection 1.3.4.

As embeddings are trained after the input is encoded into one-hot vectors, it is impossible to use pretrained embedding for the encoding of previously unseen words. This problem is solved by embeddings of characters or subwords so that the whole word embedding can be later compounded of them. Section 1.2.7 describes the possibilities of involving and raining such embeddings in more detail. Embeddings are currently the best option of an input representation, but another part of the problem is recognizing which parts of the input are valid for the given task. This problem is addressed by the attention mechanism, described in the following subsection.

1.2.4 Attention mechanism

Attention mechanism (Bahdanau et al.) is widely used in NLP as a tool for extracting relevant information from word sequences. For example, when generating a sentence translation, each word in the target language corresponds to just a few words in the source sentence, not to a whole sentence. Attention gives weights to the words, which represents this connections (see picture 1.7).

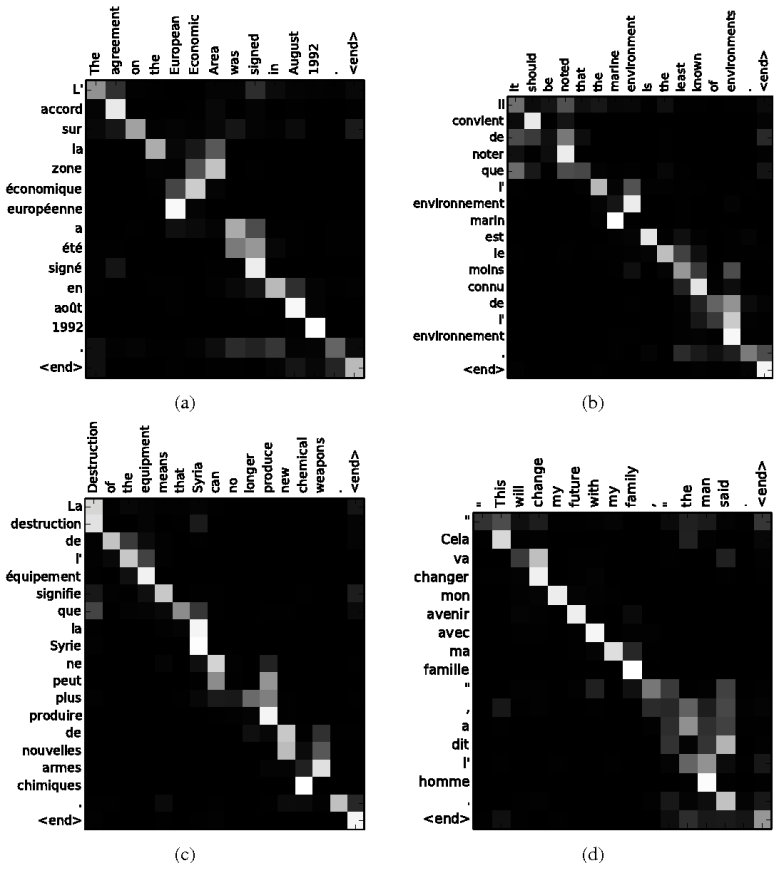


Figure 1.7: Figure 3 of (Bahdanau et al.)

When the task is a question answering, attention can help a model focus on a relevant part of the text, where the answer is located (dos Santos et al., 2016). For example see picture 1.8

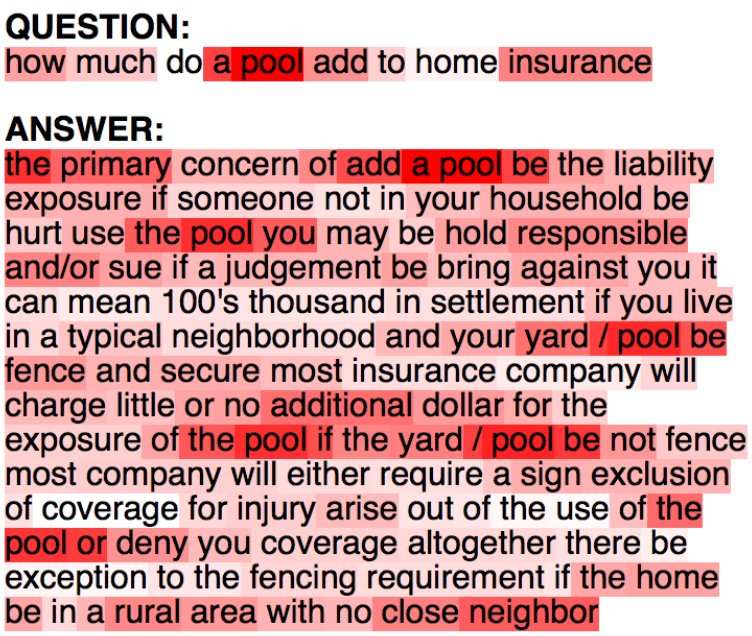


Figure 1.8: Figure 3 of (dos Santos et al., 2016)

The same idea can be applied to computer vision, where it imitates human behavior. Humans also focus on (or *attend to*) just a few parts of their visual input when they are, for example, recognizing things in pictures. Modification to an attention concept, called self-attention (Cheng et al., 2016), deals with relationships inside one part of the text (e.g., a sentence). This variant of attention do not connect one part of the text (like a question) to another distinct part of the text (like an answer) but only models relationships inside one part. To offer more technical details, attention has three inputs – key, value, and query, all vectors. The output is a weighted sum of values, and the weights are computed with respect to key-query pair [10]. For more explanation, see picture 1.9 With transformers architecture, an important part of BERT and other language models used in this work, it also published a new approach to self-attention - *multihead attention* (Vaswani et al., 2017). Multihead attention also tries to model relationships between the words in the same sequence. As it is multiheaded, it can, for one word, pay attention to more words (or their parts) (see figure 1.10.

1.2.5 Recurrent Neural Networks

Text sequences can be very long, and related words often have a long distance in between. This fact places challenging demands on neural networks because such data structure differs from most other NN applications, where input samples are independent, and order does not matter. Text size can also lead to vanishing/exploding gradients because information should be carried for many steps,

[10]This video series helped me a lot with an understanding of the self-attention concept: https://youtube.com/playlist?list=PL75e0qA87dlG-za8eLI6t0_Pbxafk-cxb

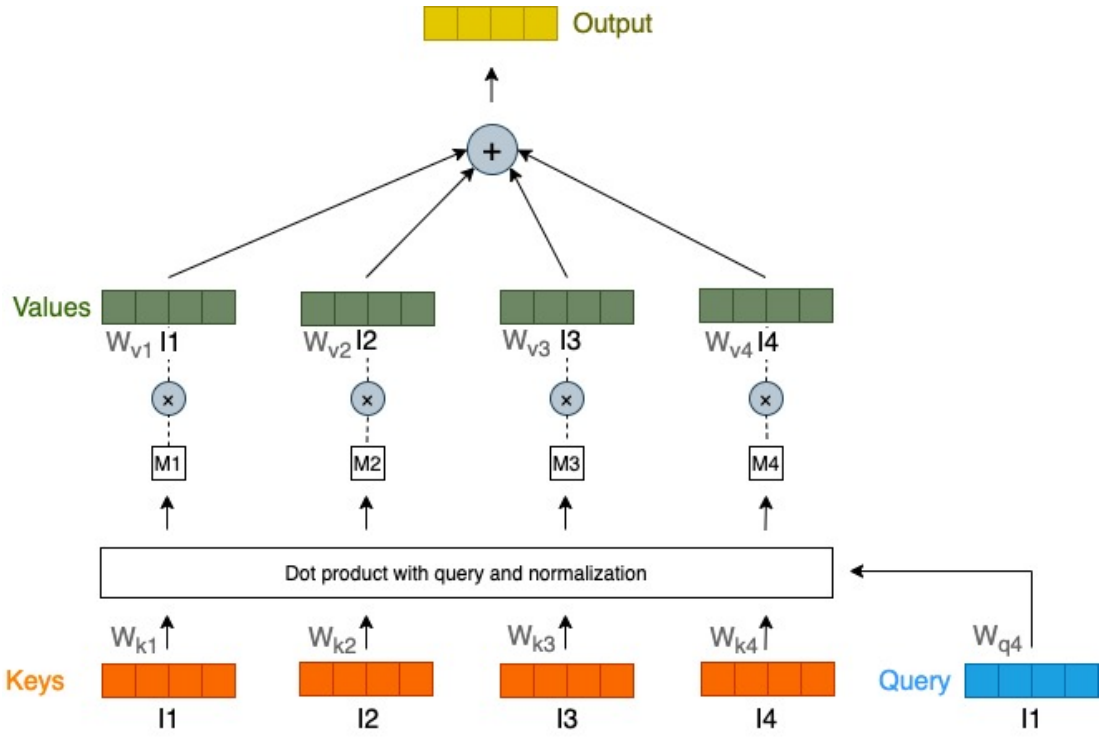


Figure 1.9: Self-attention mechanism scheme for one concrete query vector. The result is an embedding, which is improved by the context of the word. This picture illustrates the result for the embedding of the first word (l1) in a four-word long text. Keys, values, and a query are all multiplied by their respective weights (W_k, W_v, and W_q) before any other operation with them. These weights are trained during learning. Dot products between every word and a query are computed. A result is a number for every input word, so four numbers at the end. These numbers are normalized, so the sum of them is equal to 1. These words serve as a weight (M_x), which indicates the relationship between the query and every other word. The resulting better embedding for the query is then obtained as a sum of the word embeddings weighted by these obtained weights.

leading to many multiplications of very small or big numbers in neural networks. The construction of such networks, which can capture natural language structure, requires solving two problems:

- input should be understood by the network as a sequence,
- there must be a possibility to use information from other parts of the sentence (and do not forget them).

The first problem was solved by simple Recurrent Neural Networks (RNN). To represent an ordering and a continuity of input words, basic RNN adds a connection between the output for one word and an input for the next word (see picture 1.11).

The latter problem needs a more complicated approach. There are three attempts to solve this *short memory* of RNN cells – Gated Recurrent Unit (GRU) (Cho et al., 2014), Long Short Term Memory (LSTM) (Hochreiter and Urgen

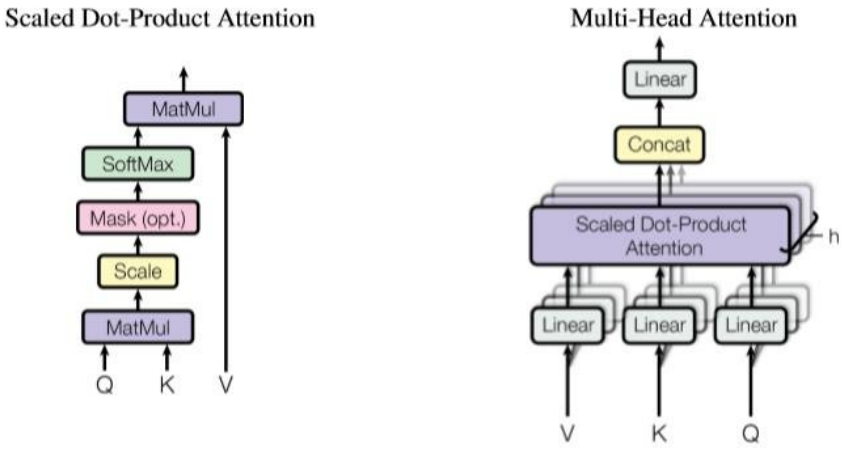


Figure 1.10: Figure 2 from (Vaswani et al., 2017).

Schmidhuber, 1997) and Transformers architecture (Vaswani et al., 2017) with an attention mechanism.

Both LSTM and GRU uses an idea of gating. Term *gate* refers to a weight (multiplication factor) for previous informations and new input. Gate determines which information from previous words should be remembered and which should be forgotten. For regulating the memory, the gate is formed by the sigmoid activation function, ranging from 0 to 1 and presenting a portion of remembered information. Previous information is encoded in a cell state and a hidden state, both passed from cell to cell (with the application of gates). Comparison of both architectures can be found in figure 1.12.

LSTM

LSTM cell uses input, output and forget gate. Every gate is composed by sigmoid function with actual input and previous hidden state as inputs. *Forget gate* filters information from previous cell state. Input gate decides which parts of input will affect the results. Output gate then selects which part of the result will be actually part of an output. Source: *(Varsamopoulos et al., 2018)*

GRU

GRU also uses gates: reset gate and update gate. The reset gate is responsible for how much of the previous state will take in the new state. Update gate serves as a weight for a combination of previous and current states, which form new output.

1.2.6 Transformers

A Tranformers architecture was proposed in 2017, in a paper Attention Is All You Need (Vaswani et al., 2017) and essentially depends on a self-attention mechanism (see subsection 1.2.4). In the example of sentence processing, attention is able

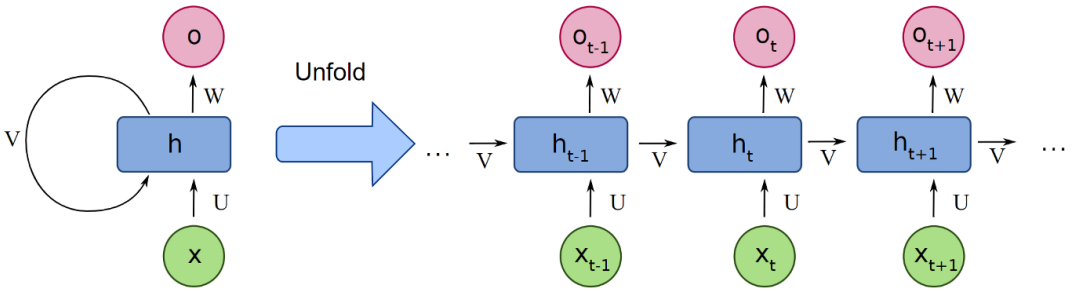


Figure 1.11: Basic Recurent neural network architecture. It is composed by one rnn cell which recurrently uses informations from previous seen input. For better illustration of working in the time, RNN can be visualised as a chain of cells connected by a result of previous cell. source: *Picture from https://medium.com/deeplearningbrasilia/deep-learning-recurrent-neural-networks-f9482a24d010.*

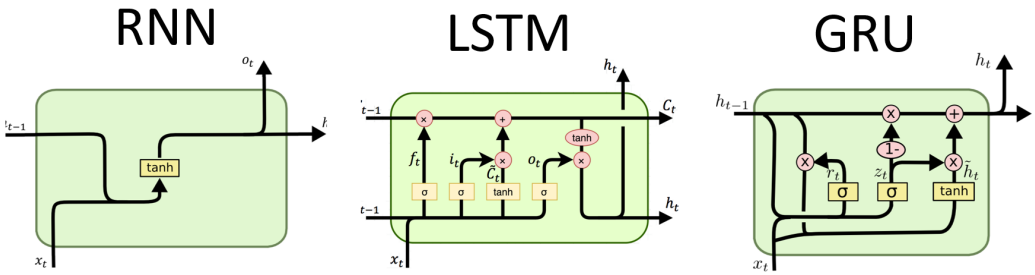


Figure 1.12: Comparison of LSTM and GRU architecture *http://dprogrammer.org/rnn-lstm-gru.*

to tell us which other words are important while determining something about the currently processed word. Tranformers uses encoder-decoder architecture, which was simultaneously published in 2014 by (Cho et al., 2014), (Sutskever et al., 2014) and (Wu et al., 2016). This architecture's basic idea is the following: encoder and decoder are connected by some middle layer of fixed size, aiming to be a good representation of the input. The encoder reads its input and tries to learn such weights that the encoder's final representation of the input contains all important information. This representation serves as an input into the decoder, which tries to produce the best results. It was first used for machine translation, so the decoder's output, in this case, is a sentence in the target language with the same meaning as an original input (see figures 1.13 and 1.14).

A self-attention mechanism is supposed to select the most important words to be focused on and used in many places – for the input of every encoder layer, for the input of every decoder layer (although masked), and also between encoder and decoder. On the decoder side, the self-attention layer is masked so the decoder can "see" just previous words (there are $-inf$ values in the positions on the right side). For a representation of word position in the sentence (as it is not an

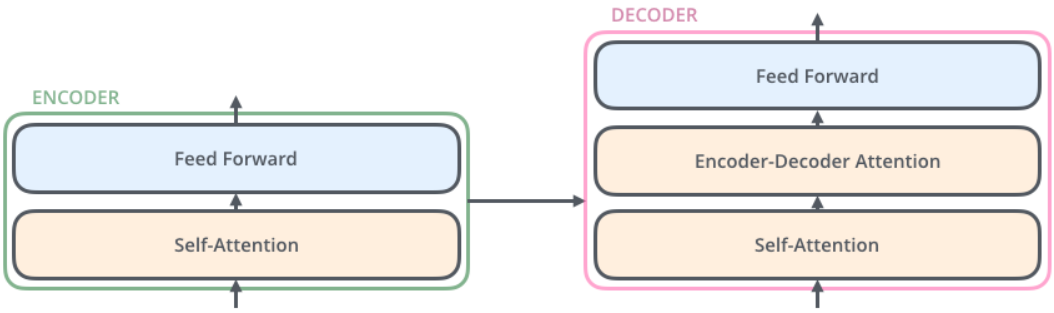


Figure 1.13: This picture describes design of one transformers layer. source: http://jalammar.github.io/illustrated-transformer

RNN cell and it can process all words simultaneously), transformers use position embeddings, which are trained to represent the sentence's ordering.

1.2.7 Transfer learning

Transfer learning is a very important idea because it allows, as the name suggests, to transfer learned knowledge between different tasks. Reusing the knowledge can lead to lower training times with fewer demands on technical resources (GPU, CPU) and training data size. It even allows to successfully apply automatic processing into domains where labeled data are not available by transferring the knowledge from another domain with enough training examples. In addition, usage of underlying common knowledge between different tasks can also improve the results of learning algorithms on each task. One of the first big successes of transfer learning comes from the computer vision field by using models pretrained on ImageNet (currently also on other datasets). ImageNet (Russakovsky et al., 2015) is a big dataset of pictures. Every picture is labeled by one of one thousand classes. Many large deep models were trained on this dataset and then applied to different computer vision tasks with great success (Huh et al.).

Following the taxonomy in figure 1.15, currently most important transfer learning applications in NLP falls into *sequential transfer learning* category. Machine learning models are first trained on some tasks, and the trained result is then used in many different tasks. For natural language processing, transfer learning is currently mainly represented by contextualized embeddings obtained from pretrained language models. Contextualization of embeddings is not important only because it solves the problem of homonyms but also because they are supposed to store knowledge independent of any language (Feijo and Moreira, 2020), (Hewitt and Liang, 2020). Contextualized embeddings can be obtained by supervised or unsupervised learning (Liu et al., 2020). This work focuses on unsupervised learning, as it is currently a promising field according to recent results and because unsupervised learning does not depend on large manually created datasets[11]. Supervised methods use machine translation, which is the classic

[11]This is a slight terminology inaccuracy. In section 1.2.2, unsupervised learning was defined as a task where data does not contain correct answer. BERT and derived models, however, use pretraining tasks where the input does not need the manual annotation, but in training itself,

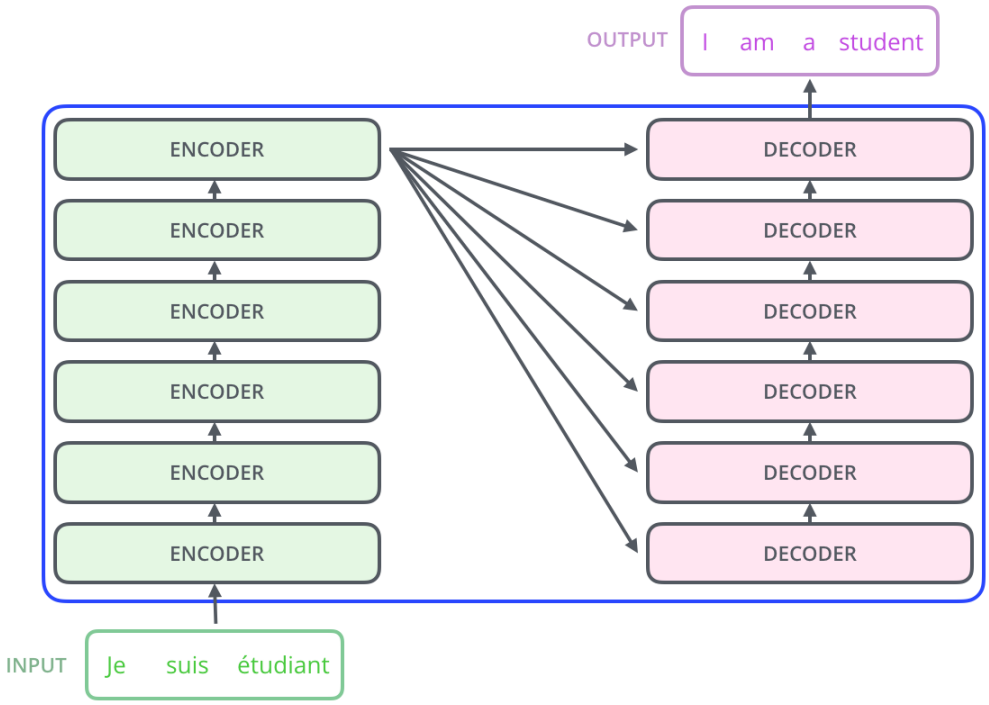


Figure 1.14: In transformers, encoder and decoder parts are both composed by many of block of respective types. The input goes first through a series of encoders and than the output of encoder part is put into every decoder in the decoder part. source: http://jalammar.github.io/illustrated-transformer/

NLP task, but also natural language inference or other tasks with the potential to capture general knowledge about language.

Unsupervised learning tries to learn a language model – a probability distribution over a sequence of tokens given by the following equation:

$$p(t_1, t_2, ..., t_N) = \prod_{i=1}^{N} p(t_i | t_1, t_2, ..., t_{i-1})$$

(Liu et al., 2020). To reduce this problem, it is possible to consider only fixed size sequences of $n-1$ previous words for every word probability called *n-grams* (Bengio et al., 2003). To achieve the goal of learning a language model, one can use many different tasks, which are believed to force the network to learn useful knowledge about language. First attempts were made with autoencoders (read the text, encode and decode back) (Dai and Le, 2015) and machine translation (Ramachandran et al., 2017), but later papers come with better architectures and various new objectives which are described later in section 1.3. There are generally two ways to transfer learned knowledge (Feijo and Moreira, 2020): extract some representation from the model and use it in another model statically or modify a model by changing the head and *finetune* the whole newly created model for a specific task. There is also a possibility to combine both approaches and, at first, take static features as an input for training, and then when the head starts to perform well, fine-tune the whole model with this better head, so the original

individual experiences are provided with the correct answer.

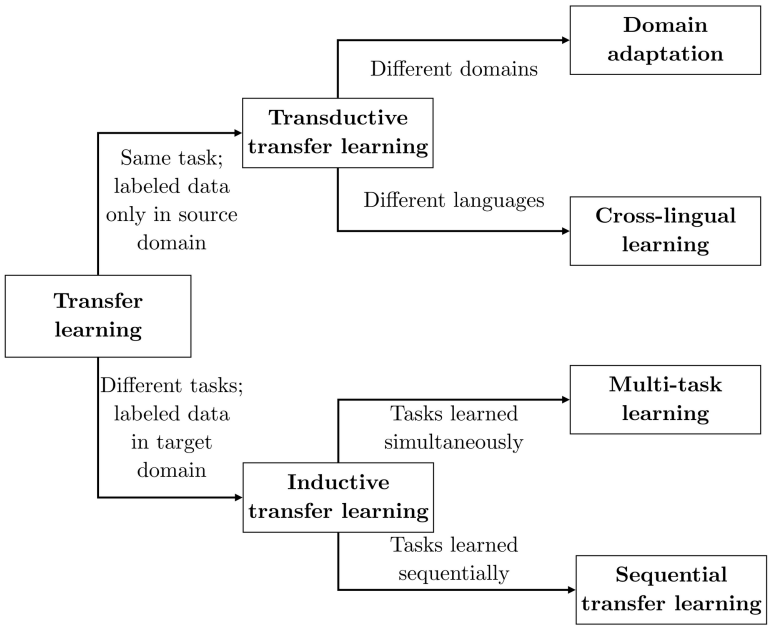


Figure 1.15: Figure from (Ruder et al., 2019) offers possible taxonomy for transfer learning. Following definiton in (Pan and Yang, 2009), transfer learning's goal is to improve the performance on task T_1 from domain D_1 by learning knowledge on task T_0 from domain D_0. Domain is defined as $D = \chi, P(X)$, where $X \in \chi$, χ is a feature space and $P(X)$ is a marginal probability distribution over the feature space. Transfer learning allows the use of trained models on tasks with different sets of labels or different input data's nature. Input data can vary in the source they come from (wikipedia text versus a novel or a social network posts), they can learn from different features (e.g. different languages) or the distribution of classes is different than it was in the training data (so some highly presented classes in training data are rare in this new task and others are quite common but previously not seen too many times).

weights converge more efficiently to the wanted solution. More detail is offered in 1.3.3.

1.3 BERT and its descendants

BERT-like models belong into a family of contextualized embeddings together with e.g., Contextualize Word Vectors (CoVe) (McCann et al., 2017), (Peters et al., 2017), ELMo (Peters et al., 2018), Flair (Akbik et al., 2018), OpenAI GPT , and series of Generative Pretrained Transformers (GPT) models (Radfort et al., 2018), (Radford Alec et al., 2019), (Brown et al., 2020). These models are important steps in NLP progression, which led to BERT family of models. BERT, representing a very effective way of contextual embeddings, shown better ability to capture the language knowledge and meant a turning point in the NLP history

First attempts to contextual embeddings appeared with two models: CoVe (McCann et al., 2017), which uses supervised machine translation and unsuper-

vised language modeling (Peters et al., 2017). Both these models are used for extracting embeddings. These embeddings concatenated to the non-contextual embeddings (i.e., GloVe) for the target NLP tasks. CoVe uses machine translation task (it needs parallel bilingual dataset) and biLSTM encoder-decoder architecture. biLSTM are bidirectional LSTMs, which are described in more details in Figure CoVe therefore uses supervised learning, in contrast to following presented models, which took the path of unsupervised learning as learning from the raw text has a big potential due to easy access to a big amount of unlabelled text data (in opposite to labeled datasets, where is almost always not enough data). (Peters et al., 2017) uses unsupervised method – language modeling (see section Uses concatenation of forward and backward RNN (similarly to CoVe, but uses both GRU and LSTM depending on the task). Both models uses last layer as an embedding representation.

ELMo is third in the series of biLSTM architecture models and builds on (Peters et al., 2017), but it uses deeper representation of words. Embeddings are created from weighted combination of all network layers (in the case of original ELMo only 2 layers). This deeper combination was led by the assumption that different layers of the network are capturing different (but useful) knowledge. Experiments with weights shown, that lower layers tends to capture syntactic information and therefore are more important for syntactic task, while higher layers are important for semantic tasks.

Flair uses also unsupervised LM, but the smallest unit of an input is a character, not a word. In the probabilistic description of language modelling, Flair models the probability of the n-th character given the previous characters. An output is again a word embedding, but this time combined from a representation of its characters. The authors chose this approach to eliminate problems with unknown words.

GPT by OpenAI (actually in version 3) also uses language modelling for pretraining, but the difference is this time in the architecture. Instead of LSTM-based RNN network, GPT variants use decoder part of transformer together with the attention mechanism. GTP also presented deeper architecture than all previously presented methods. GTP 2 version proposes 4 models sizes with the smallest one having 12 layers and the deepest one with 48 layers. Each layer is a transformer decoder with the self-attention as described in section GTP is deeper than ELMo and others, but only takes into account a left context of the word, as the text is processed sequentially in only one direction.

On the top of all this contextual embedding models stands BERT with the depth comparable to GPT 2, but using different part of transformers. It has modified pretraining tasks compare to other classical language modelling and has a context from both sides.

1.3.1 BERT

This work mainly relies on Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019) and its variants, so this section describes

and explains the main features of this model. BERT is a pre-trained language model that is fine-tuned for many other tasks, so it is an example of transfer learning (see 1.2.7). For the purpose of training a language model, original BERT is trained on two tasks – next sentence prediction and masked language modeling. BERT is proceeding both sides context simultaneously, and due to architecture, in *every layer* every transformer block potentially has information from every other block (and thus from every other word). That is why BERT is called **deeply** bidirectional or non-directional (because there is no right-to-left or left-to-right direction of processing). Result BERT model can be used for almost every NLP task just by changing the classification head. BERT is very successful in solving NLP tasks, although it was overcome in many tasks by later derived models. The second part of its popularity is the transfer learning feature of BERT. The authors published pre-trained models for English and later also for Chinese and multilingual model and other monolingual models were also published by other authors [12]. Training of the language model requires a considerable amount of data and computational time, but it is needed to be done just once. When the model is trained, it can be used for many tasks by changing the classification head and training only newly added layers and/or training all layers (but in this case only for a few episodes and with smaller data) and still performing on the state-of-the-art level.

The core of BERT algorithm is based on these three features

- two unsupervised task for pretraining,
- input embeddings,
- transformers encoder architecture,

which will be described in the following sections.

Input embeddings

BERT uses the concatenation of three types of embeddings as an input representation – token embeddings, position embeddings, and segment embeddings.

Token embeddings The BERT model input can be one or two sequences (not necessarily two sentences, but e.g., also paragraphs) [14]. All words are split into tokens and converted into embeddings with the use of a pretrained embeddings model. One word can be tokenized into more tokens because BERT uses WordPiece embeddings (Wu et al., 2016). WordPiece pretrained embedding algorithm was originaly created for the task of Google voice search for Asian languages and is designed to minimize unknown word tokens. WordPiece model was not pretrained as a part of BERT paper experiments, but represents a quite interesting solution, so the idea will be briefly explained here.

[12] Some published models, which can be used in popular python library Tranformers from HugginFace[13], can be found here: `https://huggingface.co/models`

[14] Different terminology is used here than in the original paper. In (Devlin et al., 2019), *sentence* is a term for a whole part of input (fist or second), while a term *sequence* is used for whole bert input compound from one or two *sentences*.

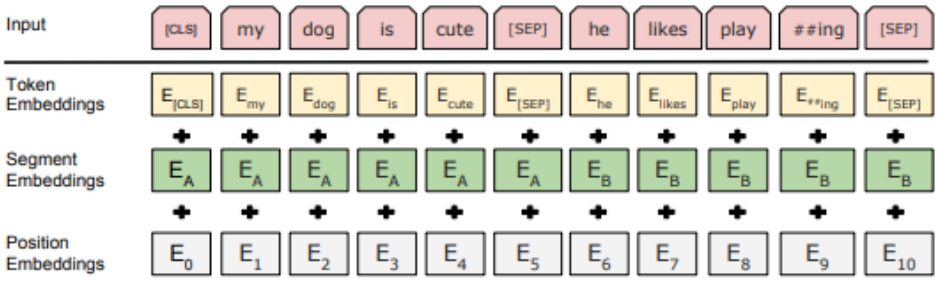


Figure 2: BERT input representation. The input embeddings is the sum of the token embeddings, the segmentation embeddings and the position embeddings.

Figure 1.16: Input is representing using tree kinds of embeddings for every input word. Every sequence is also decorated by beggining and ending marker (CLS and SEP token), which are also encoded using a combination of all three embedding types. Source: *(Devlin et al., 2019)*.

It is impossible to prepare embeddings for every possible word in a language because this would cause an intractably long embedding size. Every word which is not a part of the selected embedding set is encoded in the same way (as an unknown word). This situation is not desired because we lost information about words. WordPiece deals with this problem in the following manner: In the first iteration of training, the model creates embeddings only for characters. In every other iteration, some existing model words are concatenated together in a way that causes the highest likelihood of input text. As a result of this method, some words will be embedded as one word, and some will be split into more tokens, as can be seen in figure 1.17.

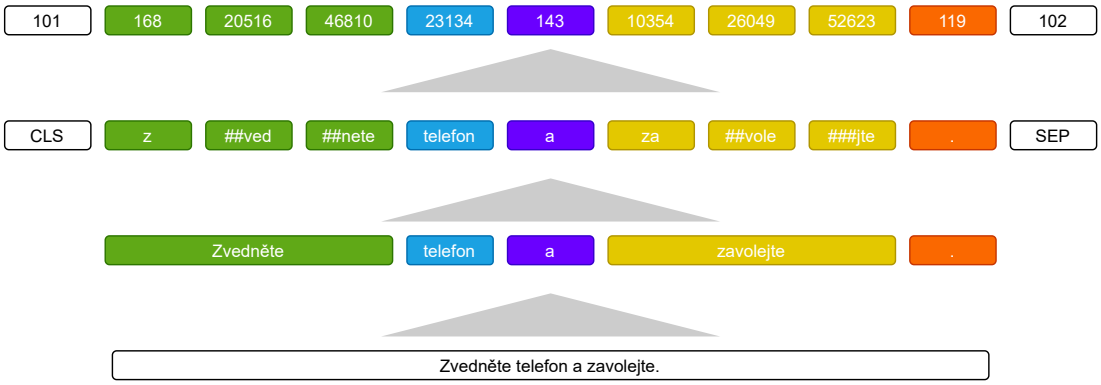


Figure 1.17: This figure illustrates a transformation of one input sentence (from PDT3) to suit bert input expectations. The sentence is divided into words and then into tokens from wordpiece tokenizer vocabulary. Accents may be removed depending on the used model. The sentence is decorated with special CLS and SEP tokens to mark the beginning and the end of the sentence. All tokens are then converted into numbers.

Three other tokens are added after this step – CLS, END, SEP. CLS token is added at the beginning of the input and is used as the first sequence embedding for

classification tasks (as sentence analysis). SEP token separates both sequences, and END token is appended at the end. Whole input transformation can be seen on figure 1.16.

Position embeddings All input tokens are processed simultaneously. That is the reason why BERT is often called *undirectional* rather than *bidirectional*. This feature causes an absence of information about the order of tokens. However, the nature of the language is sequential. A bunch of words without an order has no language meaning, and capturing this problem led to recurrent neural networks at first. In BERT is no recurrent cell, and instead, position embeddings solve this problem. They have the same shape as token embeddings (and as segment embeddings), and they are learned the same way as other embedding layers. The original BERT's maximum input size is 512, so this embedding layer should represent positions from 1 to 512. This learning is different from original transformers, where the position was also encoded as embeddings, but embeddings were fixed, not learned.

Segment embeddings These embeddings indicate whether the token belongs to the first or second part of the input. It has the same dimension as position and token embeddings, and they are also learned. Because BERT input can consist of at most two parts, segment embeddings encode whether the token belongs to the first or second part.

Pretraining tasks

BERT is pretrained on two unsupervised tasks – Next Sentence Prediction (NSP) and Masked language modeling (MLM). These two tasks were chosen because BERT's authors believe they can force language models to learn general and valuable knowledge about language.

Next Sentence Prediction The input of the BERT model for this task is two sentences, A and B. In 50% of cases, sentence B is the sentence that follows sentence A in the source text. Otherwise, it is a random sentence from the text. A goal of the task is whether sentence B is following or random, i.e., binary classification. The motivation for this task is a need to represent relationships between sentences, not only between words. Experiments in (Devlin et al., 2019) showed its usefulness for text tasks as question answering. Sentence-level classification with BERT, as in this case, can be performed by using the last hidden representation of CLS token (the first token of every input example). Authors assumed that this token could work as a summary of the whole sentence, although later work has shown better approaches (i.e., (Liu et al., 2019)).

Masked Language Modeling Masked Language Modeling or in other words Cloze task (Taylor, 1953) consist of prediction of some missing words in the text. In BERT's case, its implementing in the following way: 15% of tokens in each sequence is chosen. For each of these chosen tokens, there is 80% chance to be replaced by a MASK token, 10% chance to be replaced by a random token or it will remain unchanged with 10% probability. This way of masking ensures that the

	BERT Base	BERT Large
L	12	24
H	768	1024
A	12	16
Total Parameters	110M	340M

Table 1.1: Difference between base and large version of BERT model, as published in (Devlin et al., 2019).

model will try to predict tokens not only in MASK token presence. For backpropagation, only predictions of the MASK token are taken into account. Prediction is made by a softmax function whose input is the last hidden representation of the respective token, and softmax layer outputs a probability distribution for predicting every possible word.

Architecture

BERT adapts encoder part of architecture from original Transformers paper (Vaswani et al., 2017) (see section 1.2.6). BERT uses its encoder architecture for each layer, so L encoder layers are finished with one fully connected layer for a specific task. The original paper presents two sizes of architecture, base and large, in the dependence on the number of layers (L), size of the hidden layer (H) and a number of heads in multi-head self-attention (A). An output of before the classification head is a vector of size H for each of the input words. Original paper proposes two main architecture hyperparameters version, base and large (see table 1.1).

1.3.2 Derived models

After BERT, many other models built on similar architecture appeared. They all aims to improve original BERT paper in (at least) one of this three methods:

- A more efficiency – original BERT models was trained for about 4 days on 4/16 TPU for base and large version respectively and is quite memory intensive. Many methods for shortening the training time, memory consumption or inference time while preserving results were succesfully implemented - some of them even outperformed SOTA results stated by BERT.

- An extension of applicability – BERT model works nice for tasks requiring sentence or token classification, but is unable of language generation. Original model also does not offer a possibility of connecting a knowledge out of the processed text. Both of these problems were explored with well-performing adjusted model architectures as result.

- A results improvements – Larger models, longer training times, more data, better pretraining objectives and evolved architecture can lead to a big improvement. Some models only demonstrate scalability with more data, while others win over BERT with more creative pretraining tasks or with combination of many changes. There is another reason behind these models. It is not completely clear why BERT should work so well so many

authors offers a deep study of individual BERT component's impact on the performance, theoretical explanations and possible improvements.

Most famous and important BERT's derivates will be presented in more detail in following paragraphs.

XLNet (Yang et al., 2019b) solves to theoretical BERT's problems:

- MASK token, used in BERT's MLM objective, never occurs in real texts, therefore training data are substantially different from desired practical use, and
- BERT uses assumption of independence between MASK tokens, given the unmask words, which does not hold. There definitely could be strong relationship between two masked tokens in the sequence, even if they are not near to each other. And it is desired for the model to learn such relationships.

XLNet presents three basic differences from BERT: presents new training objective (permutation language modeling), uses Transforemers-XL (Dai et al., 2019) architecture instead of original Transformers encoders (from this come the name) and uses two-stream self-attention. The latter is a consequence of used objective, which will be briefly explained now. XLNet uses permutations of input sequence's tokens to model the probability of each token given the rest of the sequence (bidirectionally). To put it simply, few of every possible permutations are selected and than the model learns the probability of given word depending on the previous words in the permuted sentence. As any of words is masked, we need to hide the content (=segment and token embeddings) of the chosen token from the network, but keep position information. Two-stream self-attention process these two kinds of information separately, so it is possible to mask out content information. XLNet presents new state-of-the-art results over previous BERT achievements – not only with best presented models (trained on more data than BERT and with four times more training time) but also with comparable setting (both model and training data sizes).

ERNIE (= Enhanced Language Representation with Informative Entities) (Zhang et al., 2019) enriches BERT with knowledge graphs. This contains knowledge presentation and a selection of objective, which will be able to work with knowledge as well as language information. As for encoding, ERNIE finds all named entities, links them into the knowledge graph, encodes knowledge graph using knowledge embeddings which forms the input into ERNIE. BERT architecture with transformer encoders is supplemented with knowledge encoders (K-encoders) stacked on the top of text encoders (T-encoders) and the output of the whole model are embeddings for words and for entities. Pretraining objective for language is same as for BERT (NSP and MLM) and authors also presents MLM-like objective for knowledge: masking of entities. This model outperforms BERT on knowledge-based tasks (entity typing and relation classification and achieves results comparable to BERT on other NLP tasks.

RoBERTa)= robustly optimized BERT approach) (Liu et al., 2019) fall into the third category of BERT family. Great results are result of thorough exploration of choices made for original BERT model. RoBERTa uses larger models with more training data (145 GB of uncompressed text more than BERT, which has only 16 GB) and longer training time, but also offers at least partial explanation of influence of architecture, training settings and objectives on BERT's success. Main differences from BERT are:

- removing NSP objective, which surprisingly increases the performance,
- FULL-SENTENCES sampling from the dataset: Every input sample contains of full sentences sampled sequentially from data till the maximum size (512) is reached. Samples may cross boundaries of documents,
- bigger batch size (2K comparing to 256 of BERT)
- Byte-level tokens encodings instead of WordPiece, with larger vocabulary size
- dynamic masking: masks are not selected in advance before pretraining but created always before data enters the model.

Resulting architecture is better than BERT even trained on the same amount of data for the same time.

UNiLM (= Unified Language Model (Dong et al., 2019) enables BERT to generate text. Model architecture coresponds to $BERT_{large}$ but it combined bidirectional BERT training with objectives strenghtening usage of a left context, which is important for language generation and the model also uses NSP. Training objectives are selected by this rule: 1/3 of time, MLM objective (same as BERT) is used, 1/3 of time model uses sequence-to-sequence language modeling (using previous sentence and all words form left context) and unidirectional language modeling (righ-to-left and left-to-right) objectives are used with 1/6 probability. Results are comparable to BERT, with no significant improvement, but the model is able to reach new state-of-the art on several language generation tasks.

ELECTRA (= Efficiently Learning an Encoder that Classifies Token Replacements Accurately) (Clark et al., 2020). ELECTRA presents both more efficiency in training time and improvement on pretraining task. Problem of BERT's MLM task as identified by ELECTRA is its waste of training data. As BERT masks 15% of tokens, almost 85% of training data are unused. To solve this problem, ELECTRA proposes new training task, which is defined over all input tokens – selected tokens are replaced by their alternatives (generated by additional small network) and the goal for each token is to decide whether it is original or replaced token. Model has therefore two parts – generator and discriminator (loosely on spored by Generative Adversial Networks) It significantly decreases training time, because data are used more efficiently. Even with shorter training time, ELELCTRA outperforms both XLNet and RoBERTa.

T5 (= "Text-to-Text Transfer Transformer") (Raffel et al., 2019) is another approach to langauge generating and also another deep study of various parameters' influence.The training objective is similar to BERT, they choose 15% of tokens and replace them by the mask token. One change is presented – it showed to be useful to select whole spans of text of length 3 and replace them with one mask token rather than randomly select only words. Final architecture implements encoder-decoder pattern with 5 different model sizes. Only the two largest models achieved results comparable to BERT and in the case of the largest one (even outperformed previous state-of-the-art. T5 model uses text-to-text input format. Text-to-text format means that every input is in the text form and also every output. This is natural for some tasks like question answering or translation, but here also tasks with another output type are converted into text and the format is unified for all tasks. Unified format allows same architecture and training for all tasks and also easy multitask training.

BART (= Bidirectional and Auto-Regressive Transformers) (Lewis et al., 2019) is an improvement over BERT, which offers generalization upon previous BERT-like models in terms of training objective and architecture. BART presents good result in both classification and text generation tasks. BART, similarly to T5, implements encoder-decoder architecture, in this case uniting bidirectional encoder (like BERT) with left-to-right decoder (like in GPT). Main benefit of BART architecture is that it allows arbitrary noising of the input text. Encoder first process the noised input and than decoder tries to reproduce original (not damaged) text. Among many studied possible noises, sentence shuffling and text infilling provided to be the best, although for different task the performance of various pretraining objectives varied. First objective simply change the position of sentences. Text infilling replace randomly chosen span of text and replaces it by MASK token. Size of the span is chosen from Poison distribution with λ equal to 3 (in contrast to T5, where the size was fixed) and can be zero. Size is comparable to BERT (10% more parameters). It reaches the performance of BERT and RoBERTa for comparable tasks and presents new state-of-the-art on language generation tasks.

Compression of BERT In addition to models mentioned above, there is a wide range of compression efforts. In the (Ganesh et al., 2020), authors offers a overview of possible compression methods:

- data quantization: using less bits to represent weights
- various types of prunning: removing less important weights or components (encoder layers, attention heads)
- knowledge distillation: involves the large model as a teacher and a smaller model (possibly with completely different architecture) as a student. Student model can learn to imitate different settings of the teacher model (i.e., encoder outputs or output logits)
- architecture compression: sharing some weights across the model or decrease the vocabulary size of embeddings.

This study showed a possibility of reducing BERT to the quarter of original size while preserving performance. One of the famous smaller models is ALBERT (=A Lite BERT) (Lan et al., 2019). ALBERT uses to reduction techniques – reduces the size of embedding matrix by approximating it with two smaller matrices, and parameters sharing across the layers. In addition to this, ALBERT presents a modification to NSP objective: sentence-order prediction (SOP).

1.3.3 How to use language models

For using BERT-like models in new tasks, there are three possible ways (Liu et al., 2020):

- feature-based: one data pass through BERT to generate embeddings
- fine-tuning: add new classification head(s) and train the whole model
- adapter methods: adding task-specific layers between BERT layers and train them for target tasks with other BERT layers frozen.

. In the first case, the model is used only once to generate embeddings for all input data. These embeddings are stored and later fed to any other NLP model like regular embeddings. Technically this can be achieved by stacking the second model on the top of BERT and freeze (disable training) on BERT layers, but whit would be time and memory more consuming. This method is valid, the results will more likely be improved by using BERT embeddings over some other types, even the study in the BERT paper shows that using concatenation of last four layers can achieve comparable performance to fine-tuned BERT, but this method still seem to loose some potential of language models compared to learning the model on specific task.

During the training, selecting the right learning rate strategy is the essential thing. (Howard and Ruder, 2018)?? and (Sun et al.) propose selecting different learning rates for different BERT layers during finetuning.

Another possibility is a further pretraining on the domain specific data (Sun et al.) with original BERT objectives.

1.3.4 Why is BERT so special?

For deep learning architectures, it is quite common that why they work (especially why they work so well compared to other possibilities) is not visible at first sight. The authors offer the following explanation.

Responses to questions why and how BERT works are, however, not so simple. There are many papers on how BERT works, what BERT learns and knows, how BERT architecture can be simplified and so on. There appeared new term "bertology" for this area of research.

2. Task 1: Lemmatization and part-of-speech tagging

Czech morphology developement is dated from 1989 and in description of words uses 15-places morphological tags (Hana and Zeman, 2005). A more detailed description of each position can be seen in table 2.

This work aims to improve previously published SOTA results for contextualized embeddings in czech lemmatization and tagging (Straka et al., 2019b).

(Horsmann and Zesch) (Plank et al.) (Plisson et al.) (Straka et al., 2019d) (Straka et al., 2019c) (Toutanova et al., 2003) (Wang et al., 2015)

Position	Name	Description
1	POS	Part of speech
2	SubPOS	Detailed part of speech
3	Gender	Gender
4	Number	Number
5	Case	Case
6	PossGender	Possessor's gender
7	PossNumber	Possessor's number
8	Person	Person
9	Tense	Tense
10	Grade	Degree of comparison
11	Negation	Negation
12	Voice	Voice
13	Reserve1	Reserve
14	Reserve2	Reserve
15	Var	Variant, style

Table 2.1: Table from (Hana and Zeman, 2005) describes all 15 positions of Czech morphological tagging.

Experiment	Without Dictionary			With Dictionary		
	Tags	Lemmas	Both	Tags	Lemmas	Both
StrakaB	97.94%	98.75%	97.31%	98.05%	98.98%	97.65%
emb (lr 0.0001)	97.80%	98.70%	97.17%	97.95%	98.93%	97.55%
baseline	97.04 %	98.56 %	96.41%	97.31 %	98.83 %	96.90%
StrakaC	97.67%	98.63%	97.02%	97.91%	98.94%	97.51%

Table 2.2: Straka2019B is the best solution from (Straka et al., 2019b) paper. Straka2019C is a comparable solution (BERT embeddings only), which was transformed into TF2 in this work.

3. Task 2: Sentiment Analysis

3.1 Task definition

In this work, sentiment analysis tasks consist of classification of czech texts into three categories – positive, neutral and negative. Although it is possible to predict also a subjectivity part of sentiment analysis or other characteristics (e.g. irony) (it is not included in this work, mainly because the lack of labelled data in Czech. This work primary tries to improve existing tasks and show the ability of contextualized embeddings to improve results, so tasks with existing data and results were selected.

citace: (Çano and Bojar, 2019) (Hercig et al., 2018) (Li et al.) (Libovický et al.) presents state-of-the art results in three czech NLP tasks including sentiment analysis. They use only CSFD dataset with resulting accuracy 80.8 .1 which is still not so good as Brychcín (81.5.3). The second mentioned paper, however, uses quite complicated method for classification incorporating the fact of which movie is reviewed.

Bachelor thesis which aplies bert on sentiment analysis in czech: results are accuracy 84 with tfidf and about 81 with bert and they used mall dataset.

3.2 Datasets

There are three available czech datasets - user reviews from MALL.cz, film reviews from csfd.cz and facebook.

3.2.1 Facebook

Length: 9752
From: ?
Labels: neutral, positive, negative

3.2.2 csfd

Length: 91304
From:
Labels: negative, positive ?

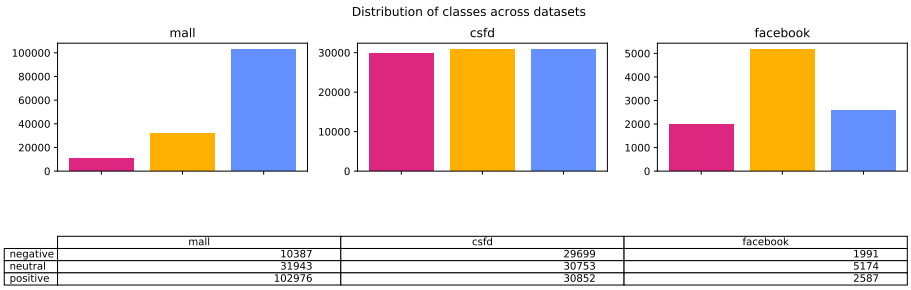
3.2.3 mallcz

Length: 145306
From:
Labels: negative, neutral, positive

3.2.4 imdb

Length: 25000
From: Labels: negative, positive

On addition to previously mentioned datasets, there also exist dataset of anotated articles from aktualne.cz



	mall	csfd	facebook
negative	10387	29699	1991
neutral	31943	30753	5174
positive	102976	30852	2587

Figure 3.1:

As can be seen in figure 3.1 distribution of labels differs among datasets. Moreover, Figure 3.2 shows, that the resulting dataset is highly unbalanced, which causes divergence and stuck in training. Due to the big part of labels being positive, many learning strategies just ended with predicting only *positive* class, i.e. 55% accuracy, so unfortunately learned nothing. This was the the case for following experiments settings:
learning rate of $2e-3$ for 10 epochs. This ended with predicting only one class (positive). Some progress showed the followign approach:"2:5e-5,1:2e-5, unfreezed" for facebook data only (which are balanced). This results are above 71 for accuracy (Test accuracy: 0.726 F1 metrics: 0.7077061630227595 weighted).

3.2.5 Experiments

I tried another approach: Because the facebook dataset is balanced, I pretrained the model first on facebook and after reaching a decent accuracy, i further trained the model on the join data: for ten epochs with learning rate 5e-5 and an effecctive batch size 48. [[6240 914 2783] [1236 4447 529] [1050 273 19020]] Test accuracy: 0.814068837005371 F1 metrics: 0.8087903170140932

The pretraining model was obtained as: without freezing, batch size 48, 2:5e-5,4:2e-5

Than I trained for 3 more epochs: [[6599 995 2343] [1246 4520 446] [1470 269 18604]] Test accuracy: 0.8145072892688808 F1 metrics: 0.8119431050948791 see 3.3

My baseline is tf-idf method, which resulted in [[4310 102 3015] [685 1016 779] [842 48 20227]] precision recall f1-score support

0 0.74 0.58 0.65 7427 1 0.87 0.41 0.56 2480 2 0.84 0.96 0.90 21117

accuracy 0.82 31024 macro avg 0.82 0.65 0.70 31024 weighted avg 0.82 0.82 0.81 31024

0.8236526560082517 acc,

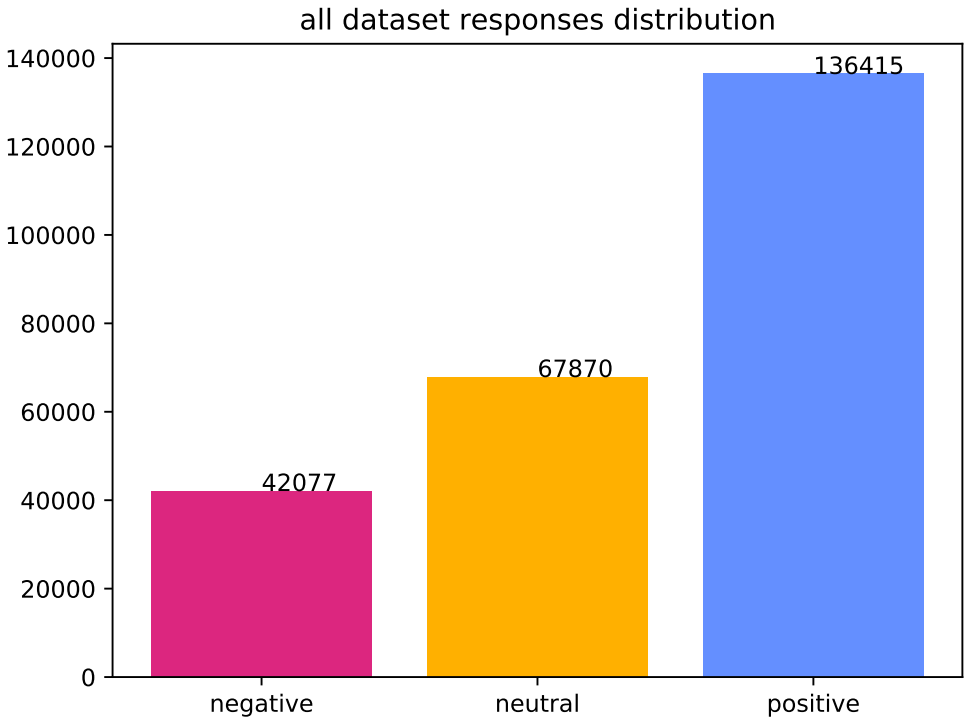


Figure 3.2:

Because BERT model was trained on multilingual data, it is naturally not so good in language minoritly presented in the Bert's training data. When transfering the learned knowledge to czech sentiment task, we actually want to improve model in two ways: teach it something more specific about given task, i.e., sentiment, and improve its knowledge about selected language (czech in this case). By using czech sentiment dataset, both thing are incorporated into training. To obtained better results and following the (Putra et al.), I also selected english sentiment dataset. The idea behind is that BERT is quite good in english and maybe can learn faster useful knowledge about the given task from data in more familiar language.

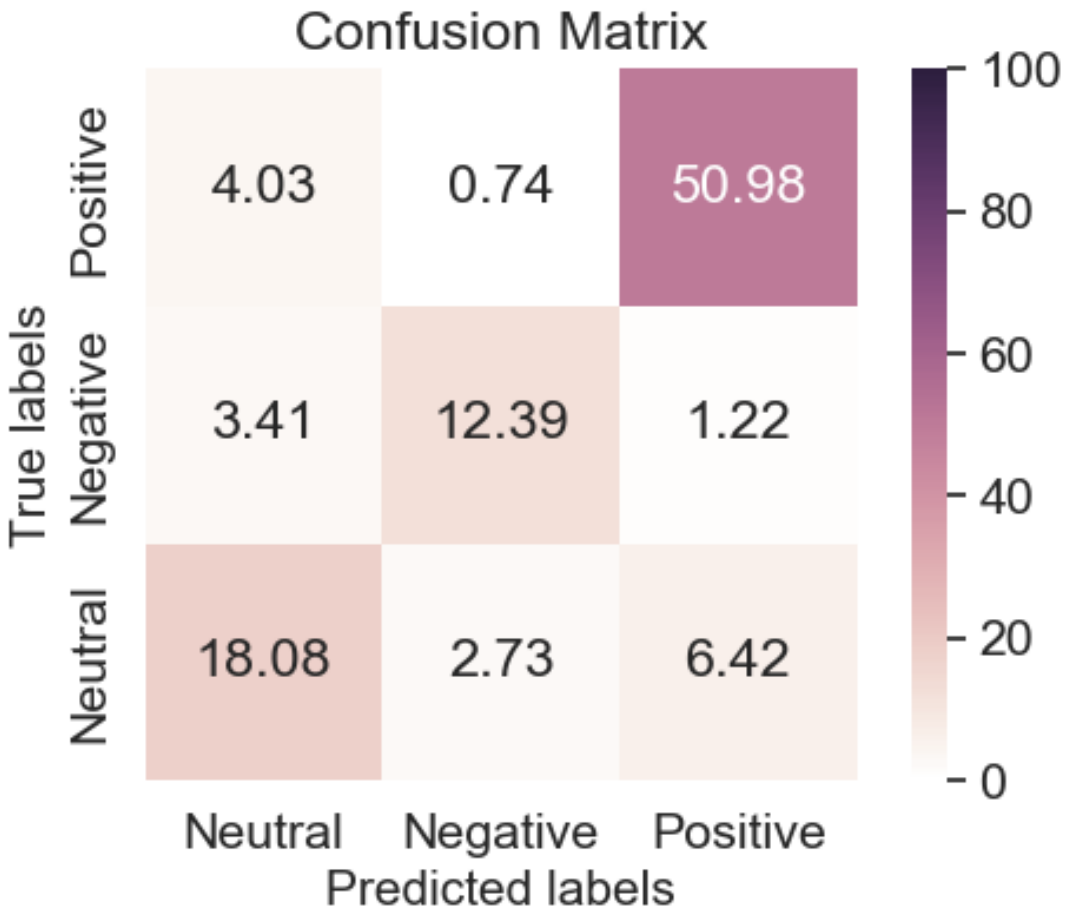


Figure 3.3: Normalized confusion matrix

4. Task 3: Language Modeling

5. Implementation analysis

This chapter describes an implementation of all language models and other related code. Following a logical code structure, this chapter is divided into three sections. First section covers lemmatization and POS tagging task, which shares same neural network model.

The practical part of this work is focused on these tasks:

- POS tagging
  input: a word
  output: part-of-speech tags – as noun, pronoun, punctuation mark etc.
- lemmatization
  input: a word
  output: lemma – a base form of a given words, meaning for example nominative of singular for nouns or infinitive for verbs.
- sentiment analysis
  input: a sentence or a sequence of sentences
  output: prevailing sentiment of the input from categories: neutral, positive, negative.

. This chapter describes implemented linguistic models. As mentioned before, this work implements models for three Czech NLP tasks: tagging, lemmatization and sentiment analysis. Common model for first two tasks develops on the paper by (Straka et al., 2019a) focused on application of contextual embeddings produced by language models as Bert or Flair. Third task, sentiment analysis, is performed by adding only one fully-connected layer at the top of bert model. So in the opposite to previous tasks, no sophisticated handcrafted pipeline is built and model relies only on the network powers of language structure representation.

doplnit diskuzi o ruzných možnostech definice

5.1 POS tagging and lemmatization model

The model for this part is build upon a model (and a code) for previous work on Czech NLP processing with contextual embeddings (Straka et al., 2019a). Data preparation pipeline - tokenization and sentence segmentation is taken over from the paper as well as base structure of lemmatizer and tagger network.

5.1.1 Dataset and preprocessing

Dataset for these tasks is based on Prague Dependency Treebank (PDT), specifically version 3.5, which is PDT edition from year 2018. Dataset is divided into three parts - train, dtest and etest. Second one is used as development set while the third one as a test set for validation. Data consists of sentences with lemmas and tags.Input sentences are preprocessed as follows:

- white space deletion
- mapping characters – all unknown dev and test characters are then mapped into one *unk* token.

- mapping words from train into integers – all unknown dev and test words are then mapped into one *unk* token.

5.1.2 Network

Lemmatization as well as tagging are considered to be classification tasks (same as in (Straka et al., 2019a)), i.e. lemmatization classification target are lemma generating rules and tagging target is tag, of course. Both tasks share one network.

Script allows training of these model variants:

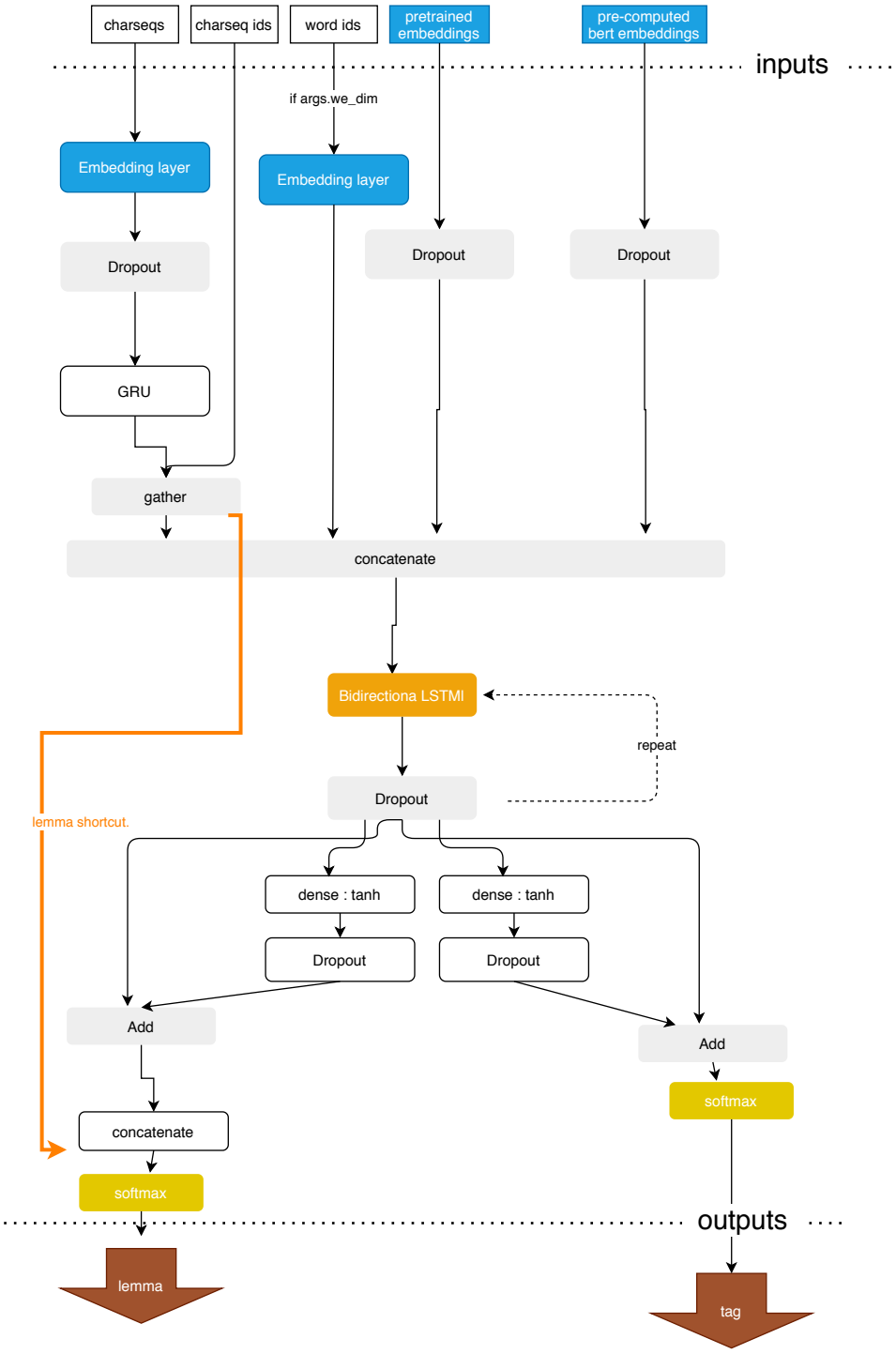
- baseline model – original implementation of network as described in figure **Embeddings:** Model contains potentially embeddings of four types – pretrained word2vec embeddings , word embeddings trained with network, bert embeddings (just in some models as described later) and character-level embeddings.

- label smoothing – baseline model with label smoothing

- bert embeddings – model can take precomputed bert embeddings as an input, otherwise embeddings are computed at the beginning and also saved for further use. In this case, data are tokenized by BERT tokenizer. Same principle is used for unknown words as before. Embeddings for word segments are averaged for each original input word, as BERT word segments and sentence words does not correspond to each other one to one. These embeddings serves as additional input to the network as in figure

- BERT finetunning – In contrast to the BERT embeddings model, in this case BERT embeddings are also trained during network training, i.e. whole BERT network is chained to baseline network instead of embedding numbers only, gradients are backpropagated throw whole new network and weights are updated also in the BERT part of network.

All classification can be done with or without use of a morphological dictionary MorFlex . If the option with dictionary is used, generated tags and lemmas are chosen just from the dictionary. So it is selected lemma and/or tag with maximal likelihood, but just from those presented in the dictionary.



Metric used for evaluation of the model is accuracy.

5.1.3 Python implementation

Model is implemented in Python (specifcally python version 3.6.9). All dependencies and used libraries are listed in the requirements.txt file, but I should specifically mention library Transformers , which contains pretrained bert models and tools for their usage as tokenizer. (In this model, I used

uncased version of multilingual models .
Code for all models is available as an attachement of the work.

6. User documentation

7. Discussion

7.1 Experiments

This section offers a description of all performed experiments. For tagging and lemmatization exist two main experiment settings: with (morphodita) pipeline on top of bert embedding and simple version with bert plus simple classification head. Both networks can finetune (or can freeze bert layers.

Table 7.1:

lr 3e-05	1	2	3	4
lemmasRaw	98.60	98.59	98.58	98.58
lemmasDict	98.86	98.87	98.88	98.90
TagsRaw	97.52	97.42	97.54	97.50
TagsDict	97.71	97.65	97.77	97.73

Table 7.2:

lr 2e-05	1	2	3	4
lemmasRaw	98.63	98.62	98.63	98.61
lemmasDict	98.90	98.89	98.91	98.90
TagsRaw	97.59	97.59	97.68	97.68
TagsDict	97.79	97.79	97.85	97.88

Table 7.3:

lr 5e-05	1	2	3	4
lemmasRaw	98.53	98.49	98.48	98.47
lemmasDict	98.81	98.80	98.80	98.80
TagsRaw	97.21	97.14	97.19	97.14
TagsDict	97.46	97.39	97.45	97.40

Conclusion

Bibliography

[PDT35] : *PDT 3.5 Main page.* – URL `https://ufal.mff.cuni.cz/pdt3.5`

[Akbik et al. 2018] Akbik, Alan ; Blythe, Duncan ; Vollgraf, Roland: Contextual String Embeddings for Sequence Labeling. In: *Proc. 27th Int. Conf. Comput. Linguist.* (2018)

[Allen 1987] Allen, Robert B.: Several Studies on Natural Language · and Back-Propagation. 1987. – Forschungsbericht

[Bahdanau et al.] Bahdanau, Dzmitry ; Cho, Kyunghyun ; Bengio, Yoshua: NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE. – Forschungsbericht

[Bengio et al. 2003] Bengio, Yoshua ; Ducharme, Réjean ; Vincent, Pascal ; Jauvin, Christian ; Ca, Jauvinc@iro U. ; Kandola, Jaz ; Hofmann, Thomas ; Poggio, Tomaso ; Shawe-Taylor, John: A Neural Probabilistic Language Model. 2003. – Forschungsbericht. – 1137–1155 S

[Bird et al. 2009] Bird, Steven ; Klein, Ewan ; Loper, Edward: *Natural Language Processing with Python.* 1st. O'Reilly Media, Inc., 2009. – ISBN 0596516495

[Brown et al. 1990] Brown, Peter F. ; Cocke, John ; Della Pietra, Stephen A. ; Della Pietra, Vincent J. ; Jelinek, Fredrick ; Lafferty, John D. ; Mercer, Robert L. ; Roossin, Paul S.: A STATISTICAL APPROACH TO MACHINE TRANSLATION. 1990. – Forschungsbericht

[Brown et al. 2020] Brown, Tom B. ; Mann, Benjamin ; Ryder, Nick ; Subbiah, Melanie ; Kaplan, Jared ; Dhariwal, Prafulla ; Neelakantan, Arvind ; Shyam, Pranav ; Sastry, Girish ; Askell, Amanda ; Agarwal, Sandhini ; Herbert-Voss, Ariel ; Krueger, Gretchen ; Henighan, Tom ; Child, Rewon ; Ramesh, Aditya ; Ziegler, Daniel M. ; Wu, Jeffrey ; Winter, Clemens ; Hesse, Christopher ; Chen, Mark ; Sigler, Eric ; Litwin, Mateusz ; Gray, Scott ; Chess, Benjamin ; Clark, Jack ; Berner, Christopher ; McCandlish, Sam ; Radford, Alec ; Sutskever, Ilya ; Amodei, Dario: *Language models are few-shot learners.* 2020

[Çano and Bojar 2019] Çano, Erion ; Bojar, Ondřej: Sentiment Analysis of Czech Texts: An Algorithmic Survey. In: *ICAART 2019 - Proc. 11th Int. Conf. Agents Artif. Intell.* 2 (2019), jan, S. 973–979. – URL `http://arxiv.org/abs/1901.02780http://dx.doi.org/10.5220/0007695709730979`

[Cheng et al. 2016] Cheng, Jianpeng ; Dong, Li ; Lapata, Mirella: Long Short-Term Memory-Networks for Machine Reading. URL `https://arxiv.org/abs/1601.06733`, 2016. – Forschungsbericht

[Cho et al. 2014] Cho, Kyunghyun ; Van Merriënboer, Bart ; Gulcehre, Caglar ; Bahdanau, Dzmitry ; Bougares, Fethi ; Schwenk, Holger ; Bengio, Yoshua: Learning phrase representations using RNN encoder-decoder for

statistical machine translation. In: *EMNLP 2014 - 2014 Conf. Empir. Methods Nat. Lang. Process. Proc. Conf.*, Association for Computational Linguistics (ACL), jun 2014, S. 1724–1734. – URL https://arxiv.org/abs/1406.1078v3. – ISBN 9781937284961

[Clark et al. 2020] Clark, Kevin ; Luong, Minh-Thang ; Le, Quoc V. ; Manning, Christopher D.: ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. (2020), mar. – URL http://arxiv.org/abs/2003.10555

[Dai and Le 2015] Dai, Andrew M. ; Le, Quoc V.: Semi-supervised Sequence Learning. URL http://ai.stanford.edu/amaas/data/sentiment/index.html, 2015. – Forschungsbericht

[Dai et al. 2019] Dai, Zihang ; Yang, Zhilin ; Yang, Yiming ; Carbonell, Jaime ; Le, Quoc V. ; Salakhutdinov, Ruslan: Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. In: *ACL 2019 - 57th Annu. Meet. Assoc. Comput. Linguist. Proc. Conf.* (2019), jan, S. 2978–2988. – URL http://arxiv.org/abs/1901.02860

[Devlin et al. 2019] Devlin, Jacob ; Chang, Ming W. ; Lee, Kenton ; Toutanova, Kristina: BERT: Pre-training of deep bidirectional transformers for language understanding. In: *NAACL HLT 2019 - 2019 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. - Proc. Conf.* Bd. 1, Association for Computational Linguistics (ACL), 2019, S. 4171–4186. – ISBN 9781950737130

[Dong et al. 2019] Dong, Li ; Yang, Nan ; Wang, Wenhui ; Wei, Furu ; Liu, Xiaodong ; Wang, Yu ; Gao, Jianfeng ; Zhou, Ming ; Hon, Hsiao-Wuen: Unified Language Model Pre-training for Natural Language Understanding and Generation. In: *arXiv* (2019), may. – URL http://arxiv.org/abs/1905.03197

[Feijo and Moreira 2020] Feijo, Diego de V. ; Moreira, Viviane P.: Mono vs Multilingual Transformer-based Models: a Comparison across Several Language Tasks. (2020), jul. – URL http://arxiv.org/abs/2007.09757

[Forcada and Ñeco 1997] Forcada, Mikel L. ; Ñeco, Ramón P.: Recursive hetero-Associative memories for translation. In: *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)* Bd. 1240 LNCS, Springer Verlag, 1997, S. 453–462. – URL https://link.springer.com/chapter/10.1007/BFb0032504. – ISBN 3540630473

[Francis and Kucera 1979] Francis, W. N. ; Kucera, H.: Brown Corpus Manual / Department of Linguistics, Brown University, Providence, Rhode Island, US. URL http://icame.uib.no/brown/bcm.html, 1979. – Forschungsbericht

[Ganesh et al. 2020] Ganesh, Prakhar ; Chen, Yao ; Lou, Xin ; Khan, Mohammad A. ; Yang, Yin ; Chen, Deming ; Winslett, Marianne ; Sajjad, Hassan ; Nakov, Preslav: Compressing Large-Scale Transformer-Based Models: A Case Study on BERT. In: *arXiv* (2020), feb. – URL http://arxiv.org/abs/2002.11985

[Goodfellow et al. 2016] Goodfellow, Ian ; Bengio, Yoshua ; Courville, Aaron: *Deep Learning*. MIT Press, 2016. – http://www.deeplearningbook.org

[Hana and Zeman 2005] Hana, Jiří ; Zeman, Daniel: Manual for Morphological Annotation Revision for the Prague Dependency Treebank 2.0 UFAL. 2005. – Forschungsbericht

[Hercig et al. 2018] Hercig, Tomáš ; Krejzl, Peter ; Král, Pavel: Stance and sentiment in Czech. In: *Comput. y Sist.* 22 (2018), Nr. 3, S. 787–794. – URL http://nlp.kiv.zcu.cz/research/sentiment{#}stance.. – ISSN 20079737

[Hewitt and Liang 2020] Hewitt, John ; Liang, Percy: Designing and interpreting probes with control tasks. In: *EMNLP-IJCNLP 2019 - 2019 Conf. Empir. Methods Nat. Lang. Process. 9th Int. Jt. Conf. Nat. Lang. Process. Proc. Conf.*, Association for Computational Linguistics, 2020, S. 2733–2743. – ISBN 9781950737901

[Hladká] Hladká: Part of Speech Tags for Automatic Tagging and Syntactic Structures 1. – Forschungsbericht

[Hochreiter and Urgen Schmidhuber 1997] Hochreiter, Sepp ; Urgen Schmidhuber, J J.: Long short-term memory. URL http://www7.informatik.tu-muenchen.de/{~}hochreithttp://www.idsia.ch/{~}juergen, 1997 (8). – Forschungsbericht. – 1735–1780 S

[Hoerl and Kennard 1970] Hoerl, Arthur E. ; Kennard, Robert W.: Ridge Regression: Biased Estimation for Nonorthogonal Problems. In: *Technometrics* 12 (1970), Nr. 1, S. 55–67. – ISSN 15372723

[Horsmann and Zesch] Horsmann, Tobias ; Zesch, Torsten: Do LSTMs really work so well for PoS tagging?-A replication study. – Forschungsbericht. – 727–736 S

[Howard and Ruder 2018] Howard, Jeremy ; Ruder, Sebastian: Universal language model fine-tuning for text classification. In: *ACL 2018 - 56th Annu. Meet. Assoc. Comput. Linguist. Proc. Conf. (Long Pap.* Bd. 1, 2018

[Huh et al.] Huh, Minyoung ; Agrawal, Pulkit ; Efros, Alexei A.: What makes ImageNet good for transfer learning? – Forschungsbericht

[Hutchins] Hutchins, John: Two precursors of machine translation: Artsrouni and Trojanskij. – Forschungsbericht

[Hutchins 1996] Hutchins, John: ALPAC: the (in)famous report. The MIT Press, 1996. – Forschungsbericht. – 131–135 S. – URL https://books.google.com/books?hl=cs{&}lr={&}id=yx3lEVJMBmMC{&}oi=fnd{&}pg=PA131{&}dq=alpac+report{&}ots=se2vhONMHp{&}sig=ByL2IgJLxRwF3f6n9bqOPFx88r4

[Jurafsky and Manning 2012] Jurafsky, Dan ; Manning, Christopher: Natural language processing. In: *Instructor* 212 (2012), Nr. 998, S. 3482

[Lan et al. 2019] Lan, Zhenzhong ; Chen, Mingda ; Goodman, Sebastian ; Gimpel, Kevin ; Sharma, Piyush ; Soricut, Radu: ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In: *arXiv* (2019), sep. – URL http://arxiv.org/abs/1909.11942

[Lewis et al. 2019] Lewis, Mike ; Liu, Yinhan ; Goyal, Naman ; Ghazvininejad, Marjan ; Mohamed, Abdelrahman ; Levy, Omer ; Stoyanov, Ves ; Zettlemoyer, Luke: BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In: *arXiv* (2019), oct. – URL http://arxiv.org/abs/1910.13461

[Li et al.] Li, Xin ; Bing, Lidong ; Zhang, Wenxuan ; Lam, Wai: Exploiting BERT for End-to-End Aspect-based Sentiment Analysis *. – Forschungsbericht. – 34–41 S

[Libovický et al.] Libovický, Jindřich ; Rosa, Rudolf ; Helcl, Jindřich ; Popel, Martin: Solving Three Czech NLP Tasks End-to-End with Neural Models. URL https://www.yelp.com/dataset/. – Forschungsbericht. – ISBN 11234/12839

[Ling et al. 2016] Ling, Wang ; Luís, Tiago ; Marujo, Luís ; Fernandez, Ramón ; Amir, Astudillo S. ; Dyer, Chris ; Black, Alan W. ; Trancoso, Isabel: Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation. 2016. – Forschungsbericht

[Liu et al. 2020] Liu, Qi ; Kusner, Matt J. ; Blunsom, Phil: A Survey on Contextual Embeddings. In: *arXiv* (2020), mar. – URL http://arxiv.org/abs/2003.07278

[Liu et al. 2019] Liu, Yinhan ; Ott, Myle ; Goyal, Naman ; Du, Jingfei ; Joshi, Mandar ; Chen, Danqi ; Levy, Omer ; Lewis, Mike ; Zettlemoyer, Luke ; Stoyanov, Veselin: RoBERTa: A Robustly Optimized BERT Pretraining Approach. In: *arXiv* (2019), jul. – URL http://arxiv.org/abs/1907.11692

[Marcus et al. 1993] Marcus, Mitchell ; Santorini, Beatrice ; Marcinkiewicz, Mary A.: Building a Large Annotated Corpus of English: The Penn Treebank. In: *Tech. Reports* (1993), oct. – URL https://repository.upenn.edu/cis{_}reports/237

[McCann et al. 2017] McCann, Bryan ; Bradbury, James ; Xiong, Caiming ; Socher, Richard: Learned in translation: Contextualized word vectors. In: *Adv. Neural Inf. Process. Syst.* Bd. 2017-Decem, 2017. – ISSN 10495258

[McCulloch et al.] McCulloch, WS ; Biophysics, W Pitts T. bulletin of mathematical ; 1943, Undefined: A logical calculus of the ideas immanent in nervous activity. URL https://link.springer.com/article/10.1007{%}252FBF02478259. – Forschungsbericht

[Mikolov et al. 2013] Mikolov, Tomas ; Chen, Kai ; Corrado, Greg ; Dean, Jeffrey: Distributed Representations of Words and Phrases and their Compositionality. URL http://papers.nips.cc/paper/

5021-distributed-representations-of-words-and-phrases-and, 2013. – Forschungsbericht

[Minsky and Papert 2017] Minsky, M ; Papert, SA: Perceptrons: An introduction to computational geometry. (2017). – URL https://www.google.com/books?hl=cs{&}lr={&}id=PLQ5DwAAQBAJ{&}oi=fnd{&}pg=PR5{&}dq=perceptrons+an+introduction+to+computational+geometry{&}ots=zzCzAMspY0{&}sig=x0HLubdLNN3Irw4cZUlmdyHK8BM

[Mitchell 1997] Mitchell, Tom M.: *Machine Learning*. McGraw-Hil. New York : McGraw-Hill, 1997. – 99 S

[Montoyo et al. 2012] Montoyo, Andrés ; Martínez-Barco, Patricio ; Balahur, Alexandra: Subjectivity and sentiment analysis: An overview of the current state of the area and envisaged developments. In: *Decis. Support Syst.* Bd. 53, nov 2012, S. 675–679. – ISSN 01679236

[Oakley et al. 1995] Oakley, Brian et al.: *Final Evaluation Of The Results Of Eurotra: A Specific Programme Concerning the preparation of the development of an operational EUROTRA system for Machine Translation*. 1995

[Pan and Yang 2009] Pan, Sinno J. ; Yang, Qiang: A Survey on Transfer Learning. (2009). – URL http://socrates.acadiau.ca/courses/comp/dsilver/NIPS95

[Pennington et al. 2014] Pennington, Jeffrey ; Socher, Richard ; Manning, Christopher D.: GloVe: Global Vectors for Word Representation. URL http://nlp., 2014. – Forschungsbericht

[Peters et al. 2017] Peters, Matthew E. ; Ammar, Waleed ; Bhagavatula, Chandra ; Power, Russell: Semi-supervised sequence tagging with bidirectional language models. In: *ACL 2017 - 55th Annu. Meet. Assoc. Comput. Linguist. Proc. Conf. (Long Pap.* Bd. 1, 2017

[Peters et al. 2018] Peters, Matthew E. ; Neumann, Mark ; Iyyer, Mohit ; Gardner, Matt ; Clark, Christopher ; Lee, Kenton ; Zettlemoyer, Luke: Deep contextualized word representations. In: *NAACL HLT 2018 - 2018 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. - Proc. Conf.* Bd. 1, Association for Computational Linguistics (ACL), feb 2018, S. 2227–2237. – URL http://allennlp.org/elmo. – ISBN 9781948087278

[Plank et al.] Plank, Barbara ; Søgaard, Anders ; Goldberg, Yoav: Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss. URL https://github.com/clab/cnn. – Forschungsbericht

[Plisson et al.] Plisson, Joël ; Lavrac, Nada ; Mladenic, Dunja: A Rule based Approach to Word Lemmatization. – Forschungsbericht

[Putra et al.] Putra, Ilham F. ; Purwarianti, Ayu ; Ai-Vlb, U-Coe: Improving Indonesian Text Classification Using Multilingual Language Model. URL https://www.yelp.com/dataset. – Forschungsbericht

[Radford Alec et al. 2019] Radford Alec ; Wu Jeffrey ; Child Rewon ; Luan David ; Amodei Dario ; Sutskever Ilya: Language Models are Unsupervised Multitask Learners — Enhanced Reader. In: *OpenAI Blog* 1 (2019), Nr. 8

[Radfort et al. 2018] Radfort, Alec ; Narasimhan, Karthik ; Salimans, Tim ; Sutskever, Ilya: Improving Language Understanding by Generative Pre-Training. In: *OpenAI* (2018)

[Raffel et al. 2019] Raffel, Colin ; Shazeer, Noam ; Roberts, Adam ; Lee, Katherine ; Narang, Sharan ; Matena, Michael ; Zhou, Yanqi ; Li, Wei ; Liu, Peter J.: Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. In: *arXiv* 21 (2019), oct, S. 1–67. – URL `http://arxiv.org/abs/1910.10683`

[Ramachandran et al. 2017] Ramachandran, Prajit ; Liu, Peter J. ; Le, Quoc V.: Unsupervised pretraining for sequence to sequence learning. In: *EMNLP 2017 - Conf. Empir. Methods Nat. Lang. Process. Proc.*, Association for Computational Linguistics (ACL), 2017, S. 383–391. – ISBN 9781945626838

[Rosenblatt 1958] Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. In: *Psychol. Rev.* 65 (1958), nov, Nr. 6, S. 386–408. – ISSN 0033295X

[Ruder et al. 2019] Ruder, Sebastian ; Peters, Matthew E. ; Swayamdipta, Swabha ; Wolf, Thomas: Neural Transfer Learning for Natural Language Processing. In: *Proc. 2019 Conf. North Am. Chapter Assoc. Comput. Linguist. Tutorials*, 2019, S. 15–18

[Rumelhart et al. 1986] Rumelhart, DE ; Hinton, GE ; Nature, RJ W. ; 1986, Undefined: Learning representations by back-propagating errors. URL `https://www.nature.com/articles/323533a0`, 1986. – Forschungsbericht

[Russakovsky et al. 2015] Russakovsky, Olga ; Deng, Jia ; Su, Hao ; Krause, Jonathan ; Satheesh, Sanjeev ; Ma, Sean ; Huang, Zhiheng ; Karpathy, Andrej ; Khosla, Aditya ; Bernstein, Michael ; Berg, Alexander C. ; Fei-Fei, Li: ImageNet Large Scale Visual Recognition Challenge. In: *Int. J. Comput. Vis.* 115 (2015), dec, Nr. 3, S. 211–252. – URL `https://link.springer.com/article/10.1007/s11263-015-0816-y`. – ISSN 15731405

[Russell et al. 1995] Russell, Stuart J. ; Norvig, Peter ; Canny, John F. ; Malik, Jitendra M. ; Edwards, Douglas D.: Artificial Intelligence A Modern Approach. 1995. – Forschungsbericht. – 529 S. – ISBN 0131038052

[dos Santos et al. 2016] Santos, Cicero dos ; Tan, Ming ; Xiang, Bing ; Zhou, Bowen: Attentive Pooling Networks. (2016), feb. – URL `http://arxiv.org/abs/1602.03609`

[Schwenk et al. 2006] Schwenk, Holger ; Dchelotte, Daniel ; Gauvain, Jean-Luc: Continuous Space Language Models for Statistical Machine Translation. 2006. – Forschungsbericht. – 723–730 S

[Straka et al. 2019a] Straka, Milan ; Straková, Jana ; Hajič, Jan: Czech Text Processing with Contextual Embeddings: POS Tagging, Lemmatization, Parsing and NER. In: *International Conference on Text, Speech, and Dialogue* Springer (Veranst.), 2019, S. 137–150

[Straka et al. 2019b] Straka, Milan ; Straková, Jana ; Hajič, Jan: Czech Text Processing with Contextual Embeddings: POS Tagging, Lemmatization, Parsing and NER. In: *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)* 11697 LNAI (2019), sep, S. 137–150. – URL http://arxiv.org/abs/1909.03544

[Straka et al. 2019c] Straka, Milan ; Straková, Jana ; Hajič, Jan: Evaluating Contextualized Embeddings on 54 Languages in POS Tagging, Lemmatization and Dependency Parsing. (2019), aug. – URL http://arxiv.org/abs/1908.07448

[Straka et al. 2019d] Straka, Milan ; Straková, Jana ; Hajič, Jan: Regularization with Morphological Categories, Corpora Merging. URL https://github.com/google-research/, 2019. – Forschungsbericht. – 95–103 S

[Sun et al.] Sun, Chi ; Qiu, Xipeng ; Xu, Yige ; Huang, Xuanjing: How to Fine-Tune BERT for Text Classification? URL https://github. – Forschungsbericht

[Sutskever et al. 2014] Sutskever, Ilya ; Vinyals, Oriol ; Le, Quoc V.: Sequence to Sequence Learning with Neural Networks. In: *Adv. Neural Inf. Process. Syst.* 4 (2014), sep, Nr. January, S. 3104–3112. – URL http://arxiv.org/abs/1409.3215

[Szegedy et al. 2015] Szegedy, Christian ; Liu, Wei ; Jia, Yangqing ; Sermanet, Pierre ; Reed, Scott ; Anguelov, Dragomir ; Erhan, Dumitru ; Vanhoucke, Vincent ; Rabinovich, Andrew: Going Deeper with Convolutions. URL https://www.cv-foundation.org/openaccess/content{_}cvpr{_}2015/html/Szegedy{_}Going{_}Deeper{_}With{_}2015{_}CVPR{_}paper.html, 2015. – Forschungsbericht

[Taylor 1953] Taylor, Wilson L.: "Cloze Procedure": A New Tool for Measuring Readability. In: *Journal. Q.* 30 (1953), sep, Nr. 4, S. 415–433. – URL http://journals.sagepub.com/doi/10.1177/107769905303000401. – ISSN 0022-5533

[Tibshirani 1996] Tibshirani, Robert: Regression Shrinkage and Selection Via the Lasso. In: *J. R. Stat. Soc. Ser. B* 58 (1996), jan, Nr. 1, S. 267–288. – URL https://rss.onlinelibrary.wiley.com/doi/full/10.1111/j.2517-6161.1996.tb02080.xhttps://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.2517-6161.1996.tb02080.xhttps://rss.onlinelibrary.wiley.com/doi/10.1111/j.2517-6161.1996.tb02080.x. – ISSN 0035-9246

[Toutanova et al. 2003] Toutanova, Kristina ; Klein, Dan ; Manning, Christopher D. ; Singer, Yoram: Feature-rich part-of-speech tagging with a

cyclic dependency network. In: *Proc. 2003 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. - NAACL '03* Bd. 1. Morristown, NJ, USA : Association for Computational Linguistics (ACL), 2003, S. 173–180. – URL http://portal.acm.org/citation.cfm?doid=1073445.1073478

[Turian et al. 2010] Turian, Joseph ; Ratinov, Lev ; Bengio, Yoshua: Word representations: A simple and general method for semi-supervised learning. Association for Computational Linguistics, 2010. – Forschungsbericht. – 11–16 S. – URL http://metaoptimize.

[Varsamopoulos et al. 2018] Varsamopoulos, Savvas ; Bertels, Koen ; Almudever, Carmen: *Designing neural network based decoders for surface codes*. 11 2018

[Vaswani et al. 2017] Vaswani, Ashish ; Brain, Google ; Shazeer, Noam ; Parmar, Niki ; Uszkoreit, Jakob ; Jones, Llion ; Gomez, Aidan N. ; Kaiser, Łukasz ; Polosukhin, Illia: Attention Is All You Need. (2017)

[Wang et al. 2015] Wang, Peilu ; Qian, Yao ; Soong, Frank K. ; He, Lei ; Zhao, Hai: Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Recurrent Neural Network. (2015), oct. – URL http://arxiv.org/abs/1510.06168

[Wilks 2005] Wilks, Yorick: The History of Natural Language Processing and Machine Translation. 2005. – Forschungsbericht

[Wu et al. 2016] Wu, Yonghui ; Schuster, Mike ; Chen, Zhifeng ; Le, Quoc V. ; Norouzi, Mohammad ; Macherey, Wolfgang ; Krikun, Maxim ; Cao, Yuan ; Gao, Qin ; Macherey, Klaus ; Klingner, Jeff ; Shah, Apurva ; Johnson, Melvin ; Liu, Xiaobing ; Kaiser, Łukasz ; Gouws, Stephan ; Kato, Yoshikiyo ; Kudo, Taku ; Kazawa, Hideto ; Stevens, Keith ; Kurian, George ; Patil, Nishant ; Wang, Wei ; Young, Cliff ; Smith, Jason ; Riesa, Jason ; Rudnick, Alex ; Vinyals, Oriol ; Corrado, Greg ; Hughes, Macduff ; Dean, Jeffrey: Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. (2016), sep. – URL http://arxiv.org/abs/1609.08144

[Yang et al. 2019a] Yang, Zhilin ; Dai, Zihang ; Yang, Yiming ; Carbonell, Jaime ; Salakhutdinov, Ruslan ; Le, Quoc V.: XLNet: Generalized Autoregressive Pretraining for Language Understanding. In: *arXiv* (2019), jun. – URL http://arxiv.org/abs/1906.08237

[Yang et al. 2019b] Yang, Zhilin ; Dai, Zihang ; Yang, Yiming ; Carbonell, Jaime ; Salakhutdinov, Ruslan ; Le, Quoc V.: XLNet: Generalized Autoregressive Pretraining for Language Understanding. In: *arXiv* (2019), jun. – URL http://arxiv.org/abs/1906.08237

[Zhang et al. 2019] Zhang, Zhengyan ; Han, Xu ; Liu, Zhiyuan ; Jiang, Xin ; Sun, Maosong ; Liu, Qun: ERNIE: Enhanced Language Representation with Informative Entities. In: *ACL 2019 - 57th Annu. Meet. Assoc. Comput. Linguist. Proc. Conf.* (2019), may, S. 1441–1451. – URL http://arxiv.org/abs/1905.07129

List of Figures

List of Tables

A. Attachments

A.1 First Attachment