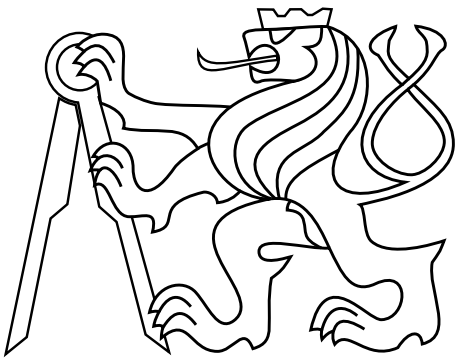
Master's Thesis

Context-Aware Prediction of Inflectional Word-Forms

Bc. Van Duy Ta
Supervisor: Ing. Pichl Jan

May 2021



Department of Computer Science
Faculty of Electrical Engineering
Czech Technical University in Prague

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Ta** Jméno: **Van Duy** Osobní číslo: **456906**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Specializace: **Datové vědy**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Kontextově informovaná predikce slovních tvarů

Název diplomové práce anglicky:

Context-Aware Prediction of Inflectional Word-forms

Pokyny pro vypracování:

1. Prozkoumejte aktuální metody pro predikci slovních tvarů. 2. Seznamte se s architekturou Transformerů a jejím využitím v oblasti NLP. 3. Na základě předchozího výzkumu navrhněte vhodný model pro predikci tvarů slov. 4. Vyberte nebo připravte českou datovou sadu pro experimenty s navrhovaným modelem. 5. Implementujte navrhovaný model a experimentálně vyhodnoťte na vybraných datech. 6. Diskutujte o svých výsledcích a případně porovnejte navrhovaný model s dříve zkoumanými metodami.

Seznam doporučené literatury:

[1] VASWANI, Ashish, et al. Attention is all you need. In: Advances in neural information processing systems. 2017. p. 5998-6008. [2] DEVLIN, Jacob, et al. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018. [3] STRAKA, Milan; STRAKOVÁ, Jana. MorphoDiTa: Morphological dictionary and tagger. 2014. [4] AHARONI, Roee; GOLDBERG, Yoav. Morphological inflection generation with hard monotonic attention. arXiv preprint arXiv:1611.01487, 2016. [5] FARUQUI, Manaal, et al. Morphological inflection generation using character sequence to sequence learning. arXiv preprint arXiv:1512.06110, 2015.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Jan Pichl, velká data a cloud computing CIIRC

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **12.02.2021** Termín odevzdání diplomové práce: **21.05.2021**

Platnost zadání diplomové práce: **30.09.2022**

Ing. Jan Pichl podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. Datum převzetí zadání

Podpis studenta

© ČVUT v Praze, Design: ČVUT v Praze, VIC

Abstract

Standard word form prediction is the task of predicting the target word-form given its base-form and morpho-syntactic tags. The typical use cases include automatic text correction, machine translation, and corpora enrichment.

The aim of our thesis is to create an infection prediction model that predicts the target inflection from the input base-form and the sentence context instead of relying on other additional inputs like morpho-syntactic tags.

In our work, we constructed a Czech dataset for the training, validation, and evaluation of the context-aware prediction of inflectional forms. Furthermore, we proposed two approaches for the prediction task. Finally, we performed both automatic and human evaluations of the proposed models.

Keywords: Inflection, Morphology, Transformers, Encoder-Decoder, Sequence-to-Sequence, Natural Language Processing, Neural Networks.

Abstrakt

Standardní formulace úlohy predikce slovních tvarů je predikce cílového tvaru slova, když je zadaný základní tvar a jeho morfo-syntaktické značky. Typické využití predikce slovních tvarů zahrnuje automatickou opravu textu, strojový překlad a obohacování korpusů.

Cílem naší práce je vytvořit model, který predikuje slovní tvar ze zadaného základního tvaru a kontextu věty namísto spoléhání se na další vstupy, jako například na morfo-syntaktické značky.

V naší práci jsme zkonstruovali českou datovou sadu pro trénink, validaci a vyhodnocení kontextově informované predikce inflexních forem. Dále jsme navrhli dva přístupy řešící zkoumaný problém. Na závěr jsme provedli automatické i lidské hodnocení navrhovaných přístupů.

Klíčová slova: Inflekce, Tvarosloví, Transformers, Encoder-Decoder, Sequence-to-Sequence, Zpracování přirozeného jazyka, Neuronové sítě

Author statement for graduate thesis:

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, date ....

signature

Acknowledgements

I would like to express my gratitude to my primary supervisor, Ing. Jan Pichl, who guided me throughout this project. Secondly, I would like to thank Ing. Jan Šedivý, CSc. and the whole Alquist team for being a great inspiration and support. Finally, I would also like to thank my friends and family who supported me and offered deep insight into the study.

Contents

Chapter 1

Introduction

Inflectional word-form prediction is a task in Natural Language Processing (NLP) [7] where we predict a target word-form given the base-form. The usages of inflectional word-form prediction include automatic text correction, machine translation, and corpora enrichment.

In the traditional formulation of the task, the model has access to the base-form of the target word and its morpho-syntactic tags. The example input in the Czech language considering the traditional approach could look somehow like this:

"Praha <Noun><F><Sg><Locative>".

Where string *"Praha"* is the Czech base form of the word Prague, and the accompanying morpho-syntactic tags describe part-of-speech, gender, number, and case.

While the task itself is not thoroughly explored, mainly because of the weakly inflected nature of English, there were several approaches applied to the inflection prediction task. The first approaches use traditional n-gram models over fully inflected word-forms [16]. The latter methods try to utilize morpho-syntactic features to predict the target form. Some of the systems do so by first predicting the morpho-syntactic features and inputting the predicted features with base-form to rule-based generators [16]. The others learn inflectional paradigms from publicly available inflection tables and directly predict the target word-forms using the learned paradigms on base-form and morpho-syntactic tags [26].

In the most recent approaches, authors try to move away from using hand-crafted rules, features, or transducers and prefer to utilize encoder-decoder architectures [36; 6]. The first attempt uses the character sequence-to-sequence LSTM encoder-decoder model [15]. Others try to build on this idea by adding hard monotonic attention [2]. However, all of the previously mentioned approaches do not take into consideration the context of the sentence.

Our thesis aims to leverage the sentence context for word-form prediction instead of using morpho-syntactic tags. To achieve the goal, we first need to design a suitable model. Afterward, we prepare a Czech dataset suitable for the training and experiments. Finally, we evaluate the proposed model and discuss the results. As an illustration, we aim to predict the correct inflection of base-form *"Praha"* given only the context of the sentence:

"Byl jsem v #Praha".

Which in translation means:

"I was in #Prague".

With such an approach, we aim to make word-form prediction more user-friendly by avoiding non-intuitive morpho-syntactic tags and expecting that word-form can be derived from the sentence context. Another reason for choosing a context-aware approach is

the recent success of transformer architectures in the NLP field. Transformers surpassed RNNs in context and long-term dependency encodings, and multiple Transformer based models achieved state-of-the-art results in various NLP tasks [37]. Moreover, the Huggingface Transformers library makes the usage of Transformers user-friendly [38].

The rest of the thesis is organized as follows. In Chapter 2, literature on inflection prediction methods is discussed, as well as their application in NLP.

Chapter 3 and Chapter 4 are devoted to the explanation of theoretical concepts used in the thesis. With Chapter 3 focusing on the linguistic terms and Chapter 4 focusing on the technical terms and algorithms .

In Chapter 5 we formulate the problem of context-aware inflectional word-form prediction. Moreover, we provide its use cases and motivation for selecting the topic.

Chapter 6 is devoted to the description, implementation, and evaluation of the proposed approaches.

Finally, **??** concludes the thesis and discusses some possible directions to be investigated in the future.

Chapter 2

Related Work

2.1 Modeling Inflection and Word-Formation in SMT

One of the main tasks where inflection and word-form models are used is statistical machine translation (SMT) [20]. The issue is word-form data sparsity which especially shows when translating to a morphologically rich target language. Considering that, Fraser et al. showed that first translating to a base-form representation and then inflecting to the correct word-form improves SMT systems' performance [16].

The authors experimented with several models using surface forms and linguistic features to predict target word forms. The procedure was first to translate English words to German stems and then to generate the correct inflected word form. The translation was conducted from English to German, and they used the 2009 ACL Workshop on Machine Translation dataset[1]. They also enriched the stems with handcrafted inflectional markup. For example, nouns were marked with gender and number; prepositions were marked with the case of their object, etc.

They presented in total five solutions for the inflection problem. The first two approaches are n-gram models over fully inflected surface forms. The inputs of the first n-gram model are pure surface forms of stems without additional linguistic features. The second approach additionally has access to the case, number, and gender if the features are present.

The rest of the models are linguistic feature prediction systems. Predicted linguistic features and stem of the word are used as an input to SMOR inflection generator [32]. The models are standard language models where tokens are represented as part-of-speech (POS) tags and linguistic features. The first model uses a single-joint prediction of all features, while the second one predicts the features separately. The last one differs only in the usage of conditional random fields (CRFs) [21] for each linguistic feature.

[1] http://www.statmt.org/wmt09/translation-task.html

2.2 Inflection Generation as Discriminative String Transduction

In the second analyzed approach [26], the authors were inspired by the works of Durrett & Denero [14], and Ahlberg et al. [19]. The common topic of the previously mentioned works is the generation of inflected forms based on a large number of training inflections tables from the Wiktionary dataset. While the latter two are more focused on generating all forms in the inflection table given the base form [26] focus on the correct inflection form provided the base form and abstract inflectional tag.

The main idea is that different lemmas tend to have similar inflectional patterns called paradigms. These paradigms can be learned from the inflectional tables present in Wiktionary and later used to generate inflection forms from unseen data. The example of an inflectional table can be seen in Figure 2.1.

The argument for using Wiktionary inflectional tables is that the tables are publicly available for multiple languages and many words, while on the other hand, it is challenging to create handcrafted morphological generators from scratch.

infinitive		atmen
present participle		atmend
past participle		geatmet
auxiliary		haben
	indicative	
present	ich atme	wir **atmen**
	du atmest	ihr atmet
	er atmet	sie **atmen**
preterite	ich atmete	wir atmeten
	du atmetest	ihr atmetet
	er atmete	sie atmeten

Figure 2.1: A partial inflection table for the German verb atmen “to breathe” in Wiktionary [26]

Their approach can be divided into three following subtasks.

2.2.1 Table alignment

The first step is to align related inflected forms in a table. One of the ways how to do so is to align all related inflected forms to a base form. The authors use an EM-driven many-to-many aligner where they look for small multi-character operations. Whereas Durrett & Denero use paradigm-aware, position-dependent edit distance [14], and Ahlberg et al. use finite-state-automata for multiple longest common subsequence alignment [19].

2.2.2 Rule extraction

The second step is to extract the rule given aligned table. Considering this, Durrett & Denero extract a rule for each changed span which creates vertical rules [14]. On the other hand, Ahlberg et al. take a different approach where they replace each unchanged span with a variable [19]. As a consequence, the latter approach makes a single rule for a whole table. While these resulting rules are larger than vertical rules and easier to interpret, they come with the downside of being less flexible.

With these approaches in mind, Nicolai et al. instead choose an even more flexible strategy where they extract a rule for each atomic, multi-character transformation. We can see the differences of described methods in Figure 2.2

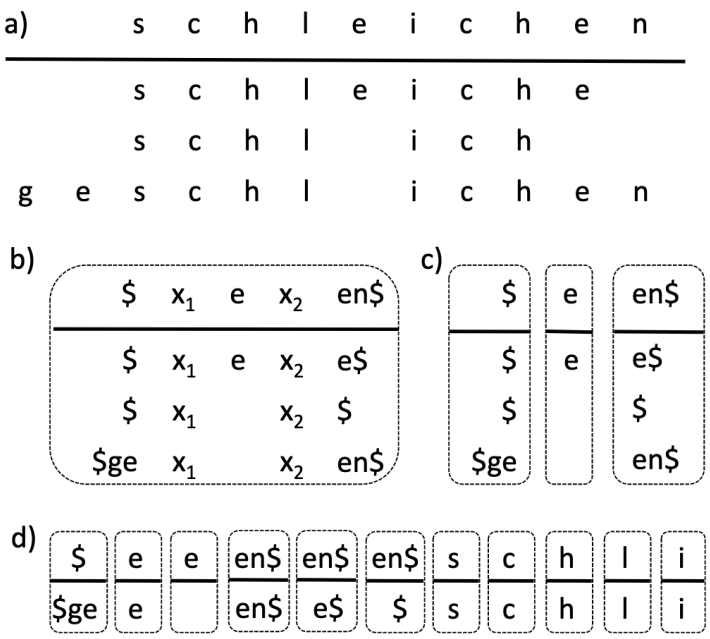


Figure 2.2: Competing strategies for rule extraction: (a) an aligned table; (b) a table-level rule; (c) vertical rules; (d) atomic rules. Dollar sign ($) is a word boundary marker. [26]

2.2.3 Rule selection

The last in the inflection generation pipeline is to pick the rules to use. Ahlberg et al. are using the longest suffix match against an index that associates rules with base-forms. Additionally, they use the corpus frequency of possible inflection options for reranking [19]. Durrett & Denero are using a semi-Markov model with features characterizing n-gram character context [14]. Similarly to Durrett & Denero, the authors used the semi-Markov model for rule selection. However, they also took inspiration from Ahberg et al. and used the corpus for reevaluation.

2.3 Morphological Inflection Generation Using Character Sequence to Sequence Learning

Unlike in the previous works, Faruqui et al. model the inflection generation task as a character sequence to sequence learning problem [15]. The comparison of their approach with the previously discussed methods can be seen in Figure 2.3

The authors present a language-independent neural encoder-decoder model that can be trained in both a supervised and semi-supervised manner. Similarly to the previous approach, the model is trained on Wiktionary data, precisely on pairs of root form and target inflected form. The authors also improved the performance with unlabeled data by integrating a character language model trained on the target language's vocabulary.

The supervised model is LSTM encoder-decoder model [18]. The loss function is negative log-likelihood, and the optimizer is AdaDelta [41]. They evaluate two different settings of the model. The first one is called Factored Model, where they train a single model for each inflection type. The second is the Joint Model, where the information of how the lemma inflects across all other inflection types is shared during the training by having the same encoder across all inflection models.

They also present semi-supervised models where they use the target language's language model to improve the predictions. The first setting is that they use the same supervised model as described before. However, in the end, they use beam search and rerank the possible outputs using the trained language model and other features. In the second semi-supervised setting, they change the loss function by accounting for the probability of observing output word given the history based on the learned language model. The last setting is an ensemble of two previously described settings.

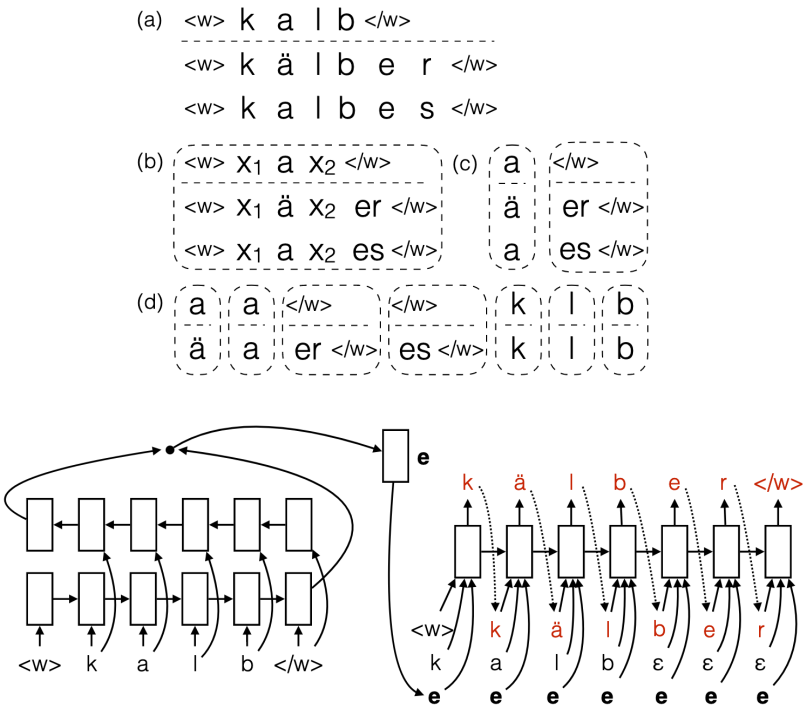


Figure 2.3: Comparison of strategies for rule extraction with character sequence to sequence model [15]: (a) an aligned table; (b) a table-level rule [19]; (c) vertical rules [14]; (d) atomic rules. [26]

2.4 Morphological Inflection Generation with Hard Monotonic Attention

Similarly to the previous approach [2] also use the encoder-decoder model. However, the authors propose solving the requirement of extensive training sets by adding a dedicated control mechanism to the encoder-decoder neural network. The suggested control mechanism tries to model natural monotonic alignment between the input and output character sequence. The authors work with the assumption that an almost monotonic alignment between the input and output character sequence is commonly present in the inflection task, especially in the case of languages with concatenative morphology.

The model consists of the bi-directional encoder RNN, the dedicated control mechanism, and the decoder. In each step of decoding, the model either makes a prediction based on a current input state or shifts the control mechanism pointer by "step" action so that the model attends to the next input state in the following decoding step. An illustrated example can be seen in Figure 2.4

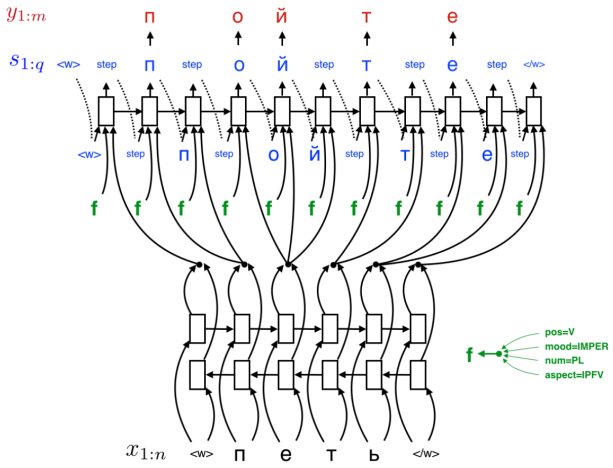


Figure 2.4: The hard attention network architecture. [2]

The hard monotonic attention model performs on par or even better than previous neural and non-neural approaches while learning using fewer training examples. Another interesting fact is that the model enables decoding in linear time with respect to the output sequence. All thanks to the fact that with hard monotonic attention decoder attends to a single input at a time, and the model therefore not require compressing the input sequence to a single fixed-sized vector.

Chapter 3

Linguistic concepts

3.1 Morphology

The term morphology originates from greek "morphe", meaning form, and "logia", meaning the study of [3]. Morphology is a part of is part of multiple scientific fields, such as linguistics, biology. However, no matter what the scientific field is, morphology is still the study of shapes, processes, or forms.

In linguistics, morphology refers to a discipline that focuses on words, their internal structure, and how they are formed. We can distinguish between two types of morphologies: inflectional and derivational. Derivational morphology is a morphology that creates new lexemes. That means it focuses on how to form new words given the existing ones. On the other hand, an inflectional morphology does not create a new lexeme but produces grammatical variants of the same lexeme instead.

The basic unit of morphology is a morpheme. It is the smallest unit having both expression and meaning. It is an abstract, systemic unit. It is determined based on syntagmatic and paradigmatic relations to adjacent units. A specific realization of a morpheme sign is a morph. If a morpheme has several different realizations, they are allomorphs. Thus, we can define a morpheme as a set of formally similar morphs with the same meaning, appearing in different word forms. According to the function, divide morphemes into lexical and grammatical.

The basic lexical morpheme carrying the meaning of the word is the root. According to the position, we divide non-root morphemes or affixes into pre-fixes (in front of the root), interfixes (between two roots), and suffixes (behind the root). The derivative morpheme located at the end of a word- form (after all grammatical suffixes) is called a postfix [29].

If we take a closer look at English inflection, we can observe that English is a weakly inflected language. Regarding that, most of the inflection in English is done with affixes. For example, plural number and third-person singular are expressed by suffix -s (as in "dog" → "dog-s"). The most common regular English inflection rules can be summarized in the Table 3.1. There are, of course, words that do not follow these rules. However, even if we count these irregularities like vowel alternations, invariant words, irregular adjectives, verbs, or adverbs, we can still see that one English word does not have many grammatical variants. We can see the comparison with the Czech words in Figure 3.1.

Now let us consider the Czech language instead. In Czech, a noun can have more than seven distinct variants only in the singular number. That clearly shows that Czech is a highly inflected language. In the linguistics of Czech, morphology is traditionally un-

Table 3.1: English inflection rules

Affix	Grammatical category	Mark	Part of speech
-s	Number	plural	nouns
-'s/'/s	Case	genitive	nouns and pronouns
-self	Case	reflexive	pronoun
-ing	Aspect	progressive	verbs
-en/-ed	Aspect	perfect non-progressive	verbs
-ed	Tense	past (simple)	verbs
-s	Person, number, aspect, tense	3rd person singular present	verbs
-er	Degree of comparison	comparative	adjectives
-est	Degree of comparison	superlative	adjectives

derstood only as inflected morphology, while the so-called derivative morphology (word formation) falls into lexicology [1].

English (1)	**Czech (12)**
nice	krásný, krásného, krásnému, krásném, krásným, krásná, krásné, krásnou, krásní, krásných, krásnými, krásnýma
English (5)	**Czech (19)**
break, breaks, broke, broken, breaking	zlomit, zlomím, zlomíš, zlomí, zlomíme, zlomíte, zlom, zlomme, zlomte, zlomil, zlomila, zlomilo, zlomili, zlomily, zlomen, zlomena, zlomeno, zlomeny, zlomeni

Figure 3.1: Comparison of English and Czech inflection [27]

As we mentioned earlier, inflectional morphology deals with those morphs that express grammatical meanings. On the other hand, lexicology is of interest to those that express lexical, factual meanings. Although the line between inflective morphology and word formation is not clear, the focus of our word-form approach will be predicting word-forms to express grammatical meanings and not the formation of new words.

3.2 Stemming and Lemmatization

In our inflectional word-form prediction task, we will need a base form of a word to serve as a model input together with a contextual sentence. There are two methods that identify a canonical word-form given a set of related word-forms of one word: stemming and lemmatization [33].

Stemming is a heuristic that cuts off the ends of the various word-forms of a word until reaching a common sequence of characters that serves as a base form. The resulting stem does not necessarily need to carry meaning and does not have to equal to a root. Moreover, a stemmer does not work with a knowledge of context when deriving the stem of a single word. Consequently, it cannot discriminate between words that have different meanings depending on the part of speech. However, stemmers are typically easier to implement and run faster than lemmatizers, and the reduced accuracy may not matter for some applications.

In our work, however, we will rely on lemmas instead of stems as base forms. Unlike stemming, lemmatization always returns a meaningful base form that is a valid dictionary form of a word. To be able to do so, lemmatizers use vocabularies and morphological analysis of words. As a consequence, lemmatization is naturally more computationally expensive compared to stemming.

Chapter 4

Technical concepts

4.1 Recurent Neural Networks

Traditional language models [34] for sequential data are based on predicting the current word on n previous words. While this assumption was proven to work well in practice, it does not show how the word is conditioned in reality. Moreover, as a consequence of this assumption, the memory requirements grow significantly as the number of previous words that a current prediction depends on grows.

Recurrent neural networks (RNNs) were introduced as a model with a theoretical capability to take all previous information into account. RNNs are architecturally suitable for NLP problems since we assume that text or speech has sequential nature. The simple RNN model is illustrated in figure 4.1.

In RNN, the output does not depend only on current input but also on the previous outputs. Moreover, an important design fact is that weights are shared through different time steps. Furthermore, compared to the traditional language models, the memory requirements do not grow with the number of previous words but only depend on the number of predicted words.

The following equations hold for the calculation of the hidden state and the output.

$$\mathbf{h}_t = \tanh(U\mathbf{x}_t + V\mathbf{h}_{t-1})$$

$$\mathbf{o}_t = softmax(W\mathbf{h}_t)$$

RNNs have problems with learning long-term dependencies [28] because of the vanishing gradient [17]. The problem of vanishing gradient appears similarly like in the case of standard feed-forward networks as the unfolded RNN is the same as feed-forward network Figure 4.1. That is why RNN architectures like GRUs and LSTMs, which were specially designed to persist the long-term dependencies, were introduced.

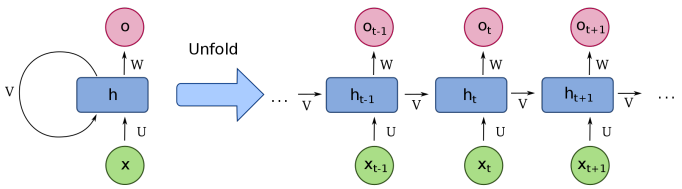


Figure 4.1: Recurrent neural network diagram [10]

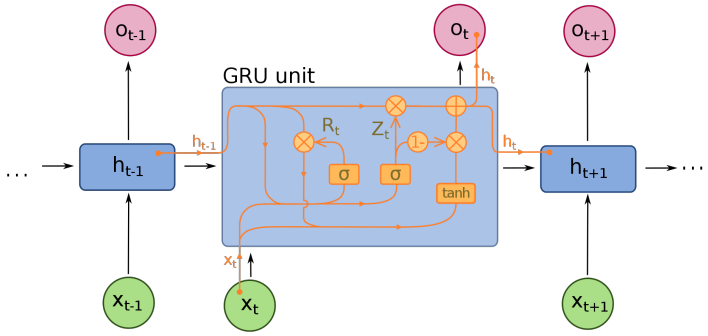


Figure 4.2: Gated recurrent unit [12]

4.1.1 Gated Recurrent Units

Gated Recurrent Units (GRUs), which were introduced in 2014 by Chun et al. [8] are a simplification of LSTMs. Compared to only the hyperbolic tangent nonlinearity present in the simple RNN unit, GRUs add two gates that influence how a neuron deals with current input and previous information. We can see an illustration in figure Figure 4.2.

The two gates are called update gate and reset gate, and they have following almost similar equations.

$$\mathbf{z}_t = \sigma(W^z \mathbf{x}_t + U^z \mathbf{h}_{t-1})$$
$$\mathbf{r}_t = \sigma(W^r \mathbf{x}_t + U^r \mathbf{h}_{t-1})$$

The following equations hold for the calculation of new memory content and final memory content.

$$\tilde{\mathbf{h}}_t = \sigma(W \mathbf{x}_t + \mathbf{r} \circ U \mathbf{h}_{t-1})$$
$$\mathbf{h}_t = \mathbf{z}_t \circ \mathbf{h}_{t-1} + (\mathbf{1} - \mathbf{z}_t) \circ \tilde{\mathbf{h}}_t$$

Based on the equations, we can see that the gates differ in weight matrices, but more importantly, they differ in usage.

The reset gate controls how much past information should be forgotten. Intuitively if all values in r get close to zero, we ignore the data from the previous hidden state because it would be irrelevant in the future.

The update gate controls the amount of past information that needs to be passed to the output. We can see that if all values in z get close to one, we copy the information from the previous step, and the vanishing gradient has no space to cause trouble. Furthermore, we can observe that GRUs are a generalization of RNNs because if r is 1 and z is 0 then the h_t equation for GRUs simplifies to the h_t equation for RNNs.

LSTM

Long short-term memory [18] (LSTMs) were introduced already in 1997 and are the predecessors of GRUs. LSTMs introduce three gates instead of two, which were introduced in GRUs. We refer to these gates as input, forget and output gates. Moreover, they also have one additional state, which is called a cell state. The following equations holds for the LSTMs.

$$\mathbf{i}_t = \sigma(W^i \mathbf{x}_t + U^i \mathbf{h}_{t-1})$$
$$\mathbf{f}_t = \sigma(W^f \mathbf{x}_t + U^f \mathbf{h}_{t-1})$$

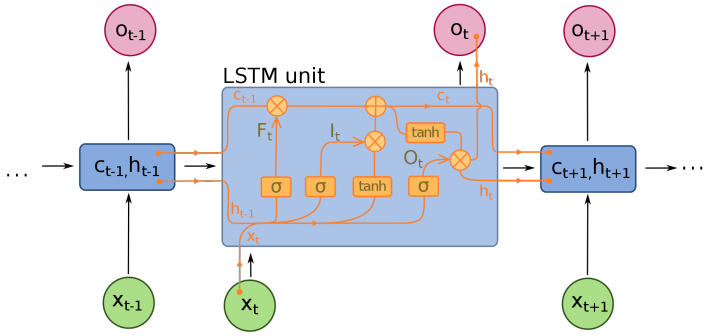


Figure 4.3: Long short-term memory unit [11]

$$\mathbf{o}_t = \sigma(W^o \mathbf{x}_t + U^o \mathbf{h}_{t-1})$$
$$\tilde{\mathbf{c}}_t = \tanh(W^c \mathbf{x}_t + U^c \mathbf{h}_{t-1})$$
$$\mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \tilde{\mathbf{c}}_t$$
$$\mathbf{h}_t = \mathbf{o}_t \circ \tanh(\mathbf{c}_t)$$

The forget gate controls the influence of the previous results on the current predictions. The input gate tells us how much we should care about the current state. This is a more robust mechanism than the update gate in GRUs as we have two different means to control the memory cell instead of a single update gate.

Another difference between LSTMs and GRUs is that LSTMs have an additional cell state. The reason is that LSTMs also control how much information they want to expose to the output. In GRUs, the output is equal to the final memory state, whereas, in LSTMs, the output is controlled by the output gate, which decides what information is relevant for the current state.

4.2 Encoder-Decoder architecture

Many NLP tasks like machine translation, text summarization, and even our inflection generation can be modeled as a sequence-to-sequence problem. The Encoder-Decoder architecture was introduced in [36; 6] as an alternative to DNNs, which despite their power and flexibility, struggle with sequence learning.

The main challenge for DNNs is that the dimensionality of the inputs and the outputs is fixed and must be known a-priori. However, as described before, many NLP problems are best represented with sequences whose lengths are not known a-priori. All these factors were the motivation behind the origin of Encoder-Decoder as a general approach to sequence learning.

The model consists of two parts, as the name suggests — an encoder and a decoder. The encoder takes a variable-length sequence as an input and outputs a fixed-length vector. The decoder takes the encoded fixed-length vector and decodes it into a variable-length sequence. In the original architecture, both the encoder and the decoder are RNNs trained to maximize the conditional probability of the target sequence given an input sequence.

The encoder RNN sequentially processes the input sequence. After processing the last input sequence item, the last hidden state returns a vector that captures the input sequence representation. This vector is referred to as the context.

The decoder is also an RNN that predicts one output symbol at a time given the previously predicted symbol, hidden state, and the context vector from the encoder. The architecture works on the same principles when RNN units are replaced with LSTM or GRU ones.

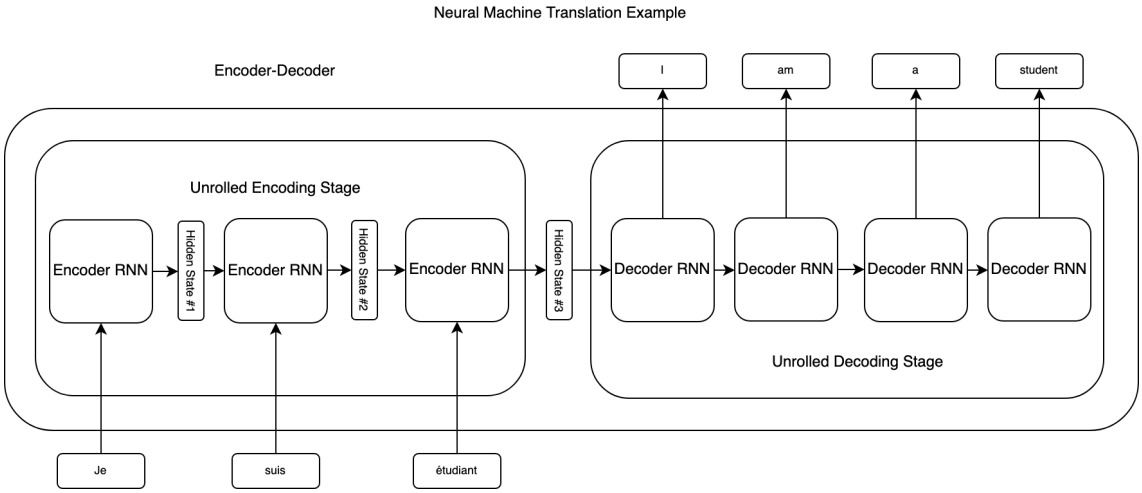


Figure 4.4: Encoder-Decoder architecture

4.3 Attention

Encoder-Decoder architecture tries to solve sequence by mapping arbitrary long input sequence to a fixed size context vector and then decoding to the arbitrary long input sequence. Although this architecture serves as a universal approach to sequence learning, it has natural drawbacks. One problem is that all the input sequence information must be captured in a fixed size vector no matter how long the sequence is. Another problem is that RNNs struggle to capture long-term dependencies. Both of these problems are difficult to solve with the increasing length of input sequences, and the attention mechanism came as a proposed solution in [4; 24].

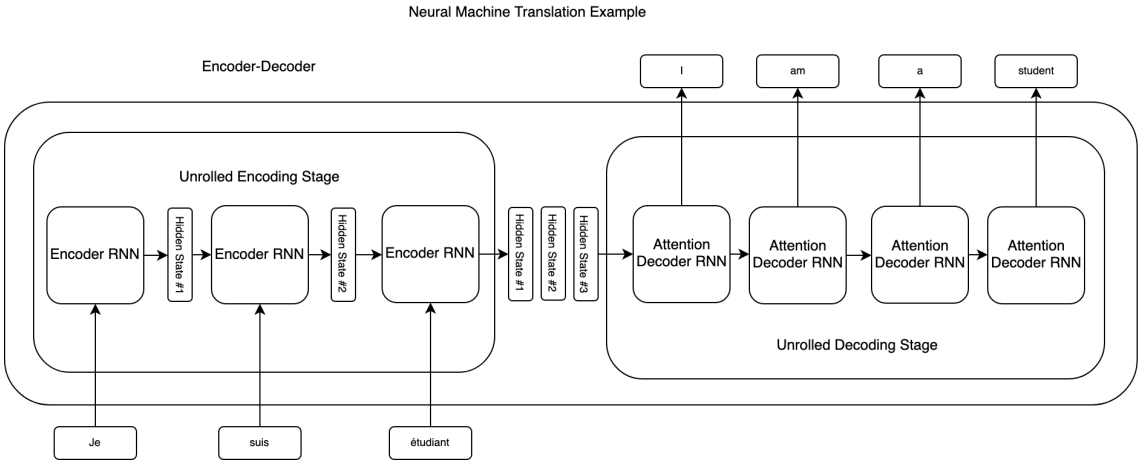


Figure 4.5: attention architecture

The main idea of the attention mechanism is to encode the input sequence into multiple vectors and let the decoder choose what is relevant instead of encoding the input sequence into one fixed-length vector. We can separate the attention models into two types—global and local. These types differ only in the way the context vector is derived. Specifically, the difference is considering either all or only some hidden states of the encoder.

Other than that, both types share the same architecture. Both calculate attentional hidden state from hidden state h_t and context vector c_t.

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_c[\mathbf{c}_t; \mathbf{h}_t])$$

The final output symbol is predicted based on the conditional probability

$$p(y_t|y_{<t}, x) = softmax(\mathbf{W}_s\tilde{\mathbf{h}}_t)$$

4.3.1 Global attention

As we previously stated, the global attentional [24] model considers all the hidden states of the encoder when deriving the context vector c_t. Firstly, it assigns a score to each of the hidden states. The scores are then normalized by using the softmax function so that they represent a probabilistic distribution. The normalized score for each source hidden state serves as a weight. The resulting context vector is computed as a weighted average over all the source hidden states.

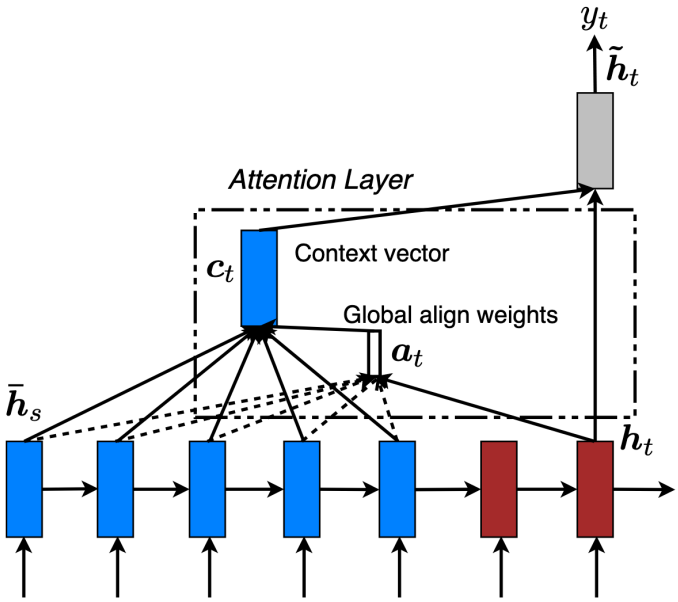


Figure 4.6: Global attention [24]

Local attention

An obvious drawback of global attention is that it has to attend to all inputs in the source sequence no matter the length. The local attentional mechanism [24] aims to tackle this problem by focusing only on a subset of the source positions during the target generation.

The model is inspired by the work of Xu et al. [40] where they used so-called soft and hard attentional models. The soft attention model considers all input hidden states and assigns weights to them. The previously mentioned global attention can be considered as a soft attention model. On the other hand, the hard attention considers only some hidden source states when generating a context vector. However, the problem with the hard attentional model is that it is non-differentiable and requires using more advanced techniques during the training.

Instead of choosing between soft and hard attention, the authors go somewhere in a middle way. Their local mechanism focuses on a fixed-size context window. By using this approach, they avoid the expensive computation of the global version. Moreover, the model is still differentiable and easier to train compared to the hard attention.

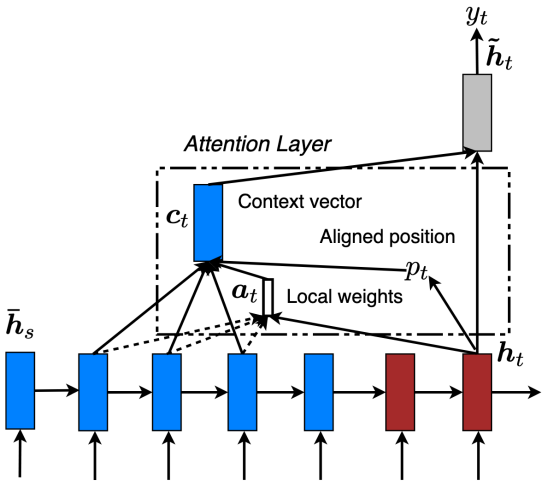


Figure 4.7: Local attention [24]

4.4 Transformer architecture

Thanks to the addition of the attention mechanism, the encoder-decoder with attention surpassed the original encoder-decoder architecture and was considered the state-of-the-art for sequence learning. Despite the success, the usage of attention together with recurrent networks still preserves the problems that inherently come with sequential nature. In [37] authors proposed the Transformer architecture, which gets rid of recurrence and relies only on attention mechanism. Moreover, the model also allows parallelization. Transformers created a breakout in the NLP field and became state-of-the-art for many NLP tasks.

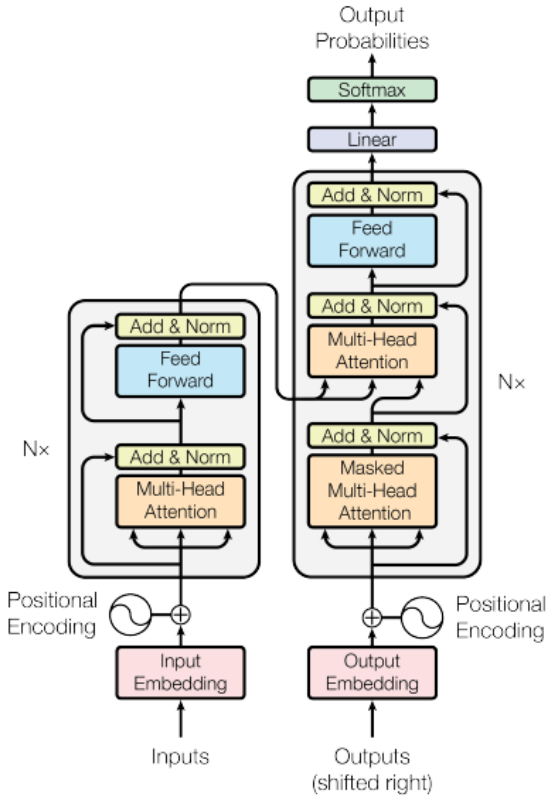


Figure 4.8: Transformer architecture [37]

From a high point perspective, the transformer model still follows encoder-decoder architecture. However, this time there are no recurrent networks. The encoding component is a stack of identical encoders. Each encoder layer consists of two sub-layers. The first is a multi-headed self-attention layer, and the second one is a fully connected feed-forward network. Moreover, residual connection followed by layer normalization is applied to each of the sub-layers.

Similar to the encoding component, a stack of identical decoder layers form the decoding one. The decoder differs from the encoder by an additional sub-layer between the multi-headed attention layer and the feed-forward layer. Moreover, the self-attention sub-layer is modified so that the predictions for position i can attend only to positions less than i.

Overall, the authors brought many new ideas that improved state of the art for sequence learning. They replaced RNNs with self-attention. Furthermore, they introduced scaled dot-product attention and multi-headed attention, and parameter-free positional encoding.

Scaled Dot-Product Attention

The authors introduced Scaled Dot-Product Attention. The function maps a query and a set of key-value pairs to an output. Despite the different notations, the main principles of attention are similar to what was described in [4; 24]. The output is still computed as a weighted sum of hidden states, in this case, values. The weights depend on the compatibility of the query with the corresponding key.

Specifically, the input consists of queries and keys of dimension d_k, and values of dimension d_v. The scores are computed as dot products of the query with all keys and normalized dividing by $\sqrt{d_k}$. Afterward, the softmax function is applied so that we get the final weight distribution. The output of the attention layer is a weighted sum of weights with values. However, the great thing is that all of these operations can be written in matrix form, making the computation more effective.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

Multi-Headed Attention

Another novel idea introduced in the paper is Multi-Headed Attention. The idea is to apply attention function h times on linearly projected queries, keys, and values instead of only once. The idea is analogical to the application of multiple smaller convolutional filters to the image in CNNs. By doing so, the model can learn multiple input representations where each can focus on a different aspect. These representations are concatenated and multiplied by a final weight matrix which results in the final output representation.

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)W^O$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

$$W_i^Q, W_i^K \in \mathbb{R}^{d_{model} \times d_k}, W_i^V \in \mathbb{R}^{d_{model} \times d_v} and W_i^O \in \mathbb{R}^{hd_v \times d_{model}}$$

Positional Encoding

The last important idea that we will talk about is positional encoding. Thanks to that, the authors were able to get rid of recurrence but still make use of the information about the order of the sequence. They add positional encodings to the input embeddings at the bottom of the encoder and decoder components. In the paper, the authors propose the following positional encodings.

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$

The intuition behind those functions is that *sin* and *cos* are periodical functions, which makes the encoding of particular positions independent of the length of the sequence. If we would fix all the parameters except of *pos* we can see that the positional embedding for the certain position is independent on the sequence length. However, we can also observe that because of the periodicity, some positions would be encoded similarly. This problem is solved by the i parameter. We can see that each of the embedding dimensions is encoded differently for the positions that would otherwise be encoded similarly in the case of fixed parameters.

4.5 BERT

BERT, which stands for Bidirectional Encoder Representations from Transformers, is a language representation model introduced by Google in 2018 [13]. The authors solved the problem of the traditional language models being unidirectional, despite the language understanding being bidirectional. The model created a big stir since it obtained state-of-the-art results on eleven NLP tasks at that time. Another reason was that the pre-trained models and the libraries were made publicly available.

The authors train the model in two steps. First, they pre-train the model on unlabeled data from multiple tasks. Afterward, the model is initialized with pre-trained parameters and fine-tuned using labeled data for the specific task. It should be pointed out that the pre-trained parameters are the same for each task. That goes hand in hand with the fact that BERT architecture is unified across different tasks.

The model architecture itself is just a stack of bidirectional Transformer encoders based on the implementation described Section 4.4.

To tackle multiple NLP tasks with one model, the authors needed to come with an input representation that is able to represent both a single sentence or a pair of sentences in one token sequence. An input sentence which is an arbitrary span of contiguous text, is encoded by WordPiece embeddings introduced in [39]. They use 30000 token vocabulary. A special classification token ([CLS]) is at the beginning of every input sequence. The purpose of the [CLS] token is to embed the information about the whole input sequence in the final hidden state. In case the sequence is a sequence pair, the sentences are separated by a special separation token ([SEP]). Moreover, a learned embedding indicating whether the token belongs to either sentence A or B is added to each token. The final representation of a token is given by a sum of the token, segment, and position embedding.

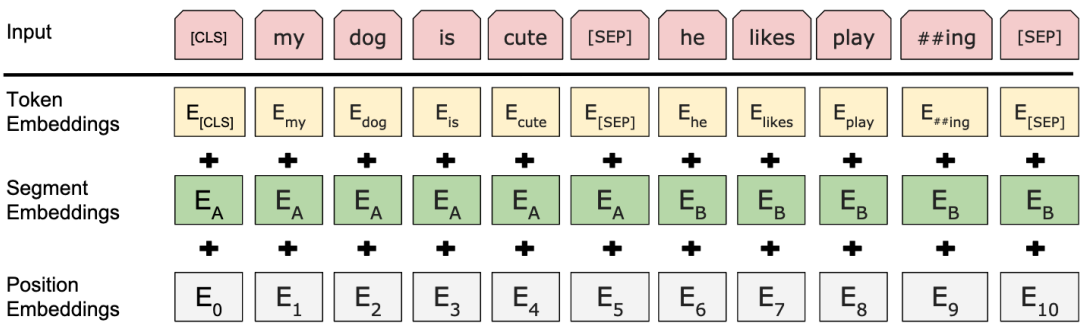


Figure 4.9: Bert input representation [13]

4.5.1 Pre-training

Instead of using traditional left-to-right or right-to-left language models, authors pre-trained BERT using two unsupervised tasks. The first one, which we will be more interested in, is masked language modeling. The second unsupervised task is the next sentence prediction. The pre-training is done on unlabeled data from BooksCorpus and English Wikipedia.

Masked Language Model (MLM)
In the masked language modeling task, a random percentage of the input tokens is masked. Later on, the goal is to predict the masked tokens. During the training, 15%

of the input token positions is selected. Out of these positions, 80% of the positions are replaced by the [MASK] token, 10% is replaced by the random token and the last 10% remains unchanged. Why did the authors choose to replace using this scheme?

The reason not to replace only with a [MASK] token is mitigating that the [MASK] should not appear during fine-tuning unless we are fine-tuning the language model. Moreover, the model would likely produce meaningful token representations only for masked words but not for the non-masked ones. Now, if we would replace only by [MASK] token and random word token, then the model would never believe that the observed word is correct. On the other hand, if we would replace only by [MASK] token or kept observed token instead, then the model would not have to rely on context.

Next Sentence Prediction (NSP)
In the next sentence prediction task, the authors try to capture the relationship between two sentences. The data for the task itself are generated from a monolingual corpus. The procedure is the following, they choose two sentences A and B as a training example. In half of the cases, sentence B is the sentence that follows sentence A in a training corpus. In the other half of the cases, it is a random sentence from the corpus. The authors show that despite the task simplicity, the pre-training on NSP is beneficial to both Question Answering and Natural Language Inference tasks which are based on understanding the relationship between two sentences.

4.5.2 Fine-tuning

Considering the fine-tuning part, there is nothing special. The procedure is pretty straightforward and requires only plugging in the task-specific inputs and outputs into BERT and fine-tuning the parameters of the network. Leveraging transfer learning by using the BERT pre-trained models and fine-tuning them was made very easy by authors. Moreover, since fine-tuning requires only the addition of an untrained feed-forward layer on top of the pre-trained BERT, it is considered inexpensive compared to the pre-training.

Chapter 5

Problem Statement

5.1 Problem Formulation

A conventional formulation of an inflectional word-form prediction task is to predict the target word-form given the base-form and morpho-syntactic tags.

The example input in Czech when considering the traditional approach could look something like this:

"Praha <Noun><F><Sg><Locative>".

Where string *"Praha"* is the Czech base form of the word Prague, and the accompanying morpho-syntactic tags describe part-of-speech, gender, number, and case.

While the traditional approaches have been proven to work well and have been helpful in tasks like machine translation or text correction, most of the inflection prediction models were highly dependent on the handcrafted input features or rules Chapter 2. However, these handcrafted input features or rules are often quite challenging to create and can look unintuitive for non-expert people outside of the target field. Under these circumstances, the natural question that arises is: Would the inflection generation work without relying on handcrafted input features or rules?

With the previously mentioned question in mind, we aim to tackle inflection prediction in a novel way as a context-aware prediction of inflectional word-forms. As the thesis topic suggests, we aim to derive the target word-form from the context of the sentence without any additional features or rules.

The example input in Czech should be:

"Byl jsem v #Praha".

Which in translation means:

"I was in #Prague".

Here, we again want to inflect the Czech base form *"Praha"*, but this time instead of the morpho-syntactic features, we have access to the context of the sentence.

5.2 Use cases

As mentioned earlier in Chapter 2, the traditional use cases of inflection prediction are mainly in machine translation when the target language is morphologically rich. Nevertheless, we intend to utilize the inflection prediction in a different way.

In our case, the objective is to utilize inflection prediction in conversational applications. Specifically, the Czech conversational applications since in weakly inflected English, the usage of the system would be less common, mainly because the English words do not have such a high number of different word-forms.

In conversational applications, there are many cases when the system extracted entities from the previous turns of conversation with a user by applying named-entity recognition (NER) [25]. The extracted entity mentions are names of essential elements in a text, like names of people, places, brands, and organizations.

With regard to the user side of interaction with the conversational app, it is undoubtedly positive to have a system that listens and reacts to the things that the user said. To put it another way, talking about recognized entity mentions makes the conversational system more engaging. Moreover, the conversational flow sounds more natural than if we would talk about predefined random things in the conversational system, which might be irrelevant for the user.

If we consider the side of the conversational system, we concluded that it is beneficial to use previously recognized entities. Let us imagine that we have recognized an entity Matrix that refers to the movie. Moreover, let us assume that we have prepared multiple dialogues about the topic of movies. Given the recognized movie entity, the desired use case is to use the correctly inflected entity in the various movie dialogues where the context differs.

On a similar note, another case where inflection prediction could be beneficial in conversational applications is having a predefined sentence where only one word varies. For example, let us take an example sentence:

"[slovo] den."

Which in translation means:

"Have a [word] day."

In this example, the variable [word] could be any positive adjective like good, nice, great, etc. As a consequence, while in English, the adjectives are not inflective except for degrees of comparison, in Czech, adjectives can take many forms even when being in one degree. This is precisely the place where the context-aware inflectional prediction model could come in handy.

Outside of conversational applications, another use case could be text correction. This use case is somewhat secondary and is inspired by my own experience. It is not unusual that foreigners say or write sentences where they do not inflect some of the words and say or write base forms instead of the correctly inflected word-forms. Thus, the inflection correction model could serve as an alternative to the existing language models.

5.3 Motivation

The motivation behind selecting a context-aware approach to the inflection prediction problem is the rise of the Transformer architecture in the field of NLP Section 4.4. Thanks to Transformers, encoding of context and long-term dependencies has advanced considerably compared to the previous age of recurrent networks.

Moreover, leveraging transfer learning methods together with large-scale Transformer language models that helped researchers achieve state-of-the-art results in multiple NLP tasks was made effortless and accessible thanks to the public libraries and a large number of publicly available pre-trained models [38].

Despite the proven power of Transformers, relying solely on the sentence context should make the prediction task itself more complex. Considering this, we still think that sentences should generally contain enough information to derive the correct word-form. But of course, there might be cases when sentence context is insufficient or ambiguous, especially if the input sentence is short.

Considering the insufficient or ambiguous instances in our conversational application use case, one could oppose that the conversational app designers can simply input the target word-forms for each context sentence. However, the aim is to make the inputs reusable for multiple context sentences.

For example, it is not hard to see that positive adjectives can be used in multiple context sentences. Similarly, various verbs representing sports activities can be present in context sentences about sports. By that, we remove as much unnecessary human work for designers. In the end, requiring inputting target word-forms would be even worse than requiring inputting morpho-syntactic features, which we already wanted to avoid.

Not considering the context-aware part of the selected research topic, a personal reason for choosing the general topic of inflection is seeing people struggle with Czech inflection. The general belief is that foreigners have difficulty learning the Czech language. However, we argue that it is not only the case for foreigners. We believe that there are also native speakers struggling with inflection in the Czech language, at least during primary and secondary school education. Moreover, concerning the general level of the Czech language in recent years, it seems to decline since the younger generation tends to include anglicisms in daily conversations.

Chapter 6

Methods

6.1 Lemmatized Language Model Approach

In our approach, we used BERT architecture described in Section 4.5 that achieved state-of-the-art results on various NLP tasks [13]. In particular, we were inspired by the masked language model, where the model is trained to predict a word corresponding to the [MASK] token. For example, the output of a pre-trained BERT base model for an English input

"I live in [MASK]".

is token *"Brooklyn"*.

As we can see in this example, BERT learned to predict the name of a city that fits into the context. However, there is no straightforward way to influence the model's exact choice of the city that it would predict. We aim to influence the model decision by inputting lemma as a base form instead of masking the word. We will refer to the model as lemmatized language model (LLM).

The model itself is a BERT Czech language model trained from scratch. However, instead of the masked input format like

"I live in [MASK]".

the model accepts input like

"I live in #Prague".

where *"#Prague"* is the lemma.

Given the lemma, we assume that the model will have an easier time learning how to predict the correct word-form than the MLM since it has additional information about how the target word should look. Therefore, the only thing left for a model is to predict the correct word-form of a given lemma based on the learned representation of sentence context.

6.1.1 Dataset Creation

For a dataset, we have chosen the Czech part of the OpenSubtitles dataset [22]. OpenSubtlitles dataset is a collection of parallel corpora. The source is similarly named database of movie and TV subtitles. It includes 2.6 billion sentences across more than 60 languages. The Czech corpus that we chose contains 135.9 million sentences. We split the dataset into training, validation, and test splits with 110, 15, and 10 million sentences, respectively. The sample of the OpenSubtitles dataset can be seen in Table 6.1.

Additionally to the OpenSubtitles dataset, we cooperated with linguists that created a small testing dataset for human evaluation. The dataset tries to cover all types of inflection ranging from nouns, adjectives, pronouns, numerals to verbs. The intended use

Table 6.1: Sample of OpenSubtitles data

Myslíš , že jen tak kecám .
Jestli budeš dál psát ty historky, tak lidé v ty nesmysly uvěří a zpanikaří .
Doufám , že na příští akci nebudete chybět .
Je vážně milý a přátelský .
Nazdárek , Astrid .
Doufal jsem , že jsme se poučili z našich chyb , ale zdá se , že někteří z nás se nepoučili .
Až ti nadhodím , sleduj míček .
Kolik různých jazyků existuje ?
Musíš to vidět bez toho všeho .
Klienti si můžou vybrat , co chtějí ... a kohokoliv chtějí .

is to get a better insight into what kind of mistakes the model makes and whether it can handle all types of inflection.

Since the datasets themselves do not contain lemmas, we need to lemmatize the data first. For that purpose, we used MorphoDiTa [35] a morphological dictionary and tagger. It is an open-source tool for morphological analysis of natural language texts. It can be used for morphological analysis, morphological generation, tagging, and tokenization.

We choose to lemmatize the whole datasets at once and then pick the lemmas instead of lemmatizing only the inputs that are used for training since the total size of datasets is quite large, and lemmatization is time-consuming. The lemmatized sample is shown in The sample of the dataset can be seen in Table 6.2.

Table 6.2: Lemmatized sample of OpenSubtitles data

myslit , že jen tak kecat .
jestli být dál psát ten historka , tak člověk v ten nesmysl uvěřit a zpanikařit .
doufat , že na příští akce nebýt chybět .
být vážně milý a přátelský . .
nazdárek , Astrid .
doufat být , že být se poučit z můj chyba , ale zdát se , že některý z já se nepoučit .
až ty nadhodit , sledovat míček .
kolik různý jazyk existovat ?
muset ten vidět bez ten všechen .
klient se moci vybrat , co chtít . . . a kdokoli chtít .

However, when we were designing the system, we found out that classical lemmatization could cause problems. Specifically, we were thinking of degrees of comparisons that relate to adjectives and adverbs. If we would use standard lemmatization, then the base form of *"dobrý"("good")*,*"lepší"("better")*, *"nejlepší"("best")* would in all cases be positive degree *"dobrý"("good")*. This example illustrates that the standard lemmatization would lose the information about the degree of comparison.

Therefore, if we would had Czech input

"Dnes je #dobrý den."

meaning

"Today is #good day."

the model would have no additional information about the degree of comparison of the target word-form.

We try to tackle the issue by returning custom *"lemma"* that represents a target adjective or adverb. While there is only one grammatical variation of an adjective or an adverb in the particular degree of comparison in English, it is not the case in the Czech language.

Moreover, a similar issue with context ambiguity holds for verbs and their negations. *"To be, or not to be."* That could be the question not only for Hamlet but also for our prediction model in the case of the verb inflection prediction. With this in mind, we also try to customize the verb lemmatization and return the lemma prepended with a prefix representing negation so that the model gains additional information.

6.1.2 Implementation Details

In this part, we will briefly describe the implementation details of the first proposed model. As we are working with Transformer architecture [37] namely with BERT [13], we decided to use widely-used open-source library called HuggingFace Transformers [38].

Firstly, we needed to train our own tokenizer since we are training a language model from scratch. The reason behind not using a pre-trained model is that we could not find any model that trained on the Czech language outside of the multilingual BERT model, which seemed too large. Because of this, we trained a byte-level BPE tokenizer on the whole OpenSubtitles Czech dataset.

Afterward, we needed to make some adjustments since our task is not the same as the original masked language task. In the original BERT pre-trained task, 15% of tokens are either masked or replaced by a random token or left as they were. In our task, we also chose 15% of tokens. However, we replaced them with lemmas so that the model can learn sentence representation with additional information about the lemma.

Another tweak that had to be made was so-called whole-word masking. Originally, it could happen that during the masking, only a part of a multi-token word would be masked, leaving the rest of the word pieces visible. Seeing a part of a word made the prediction of a mask token easier. In our case, it would make no sense if it would happen that the target affix token would remain visible for prediction. Therefore, whole-word masking was forced.

Since we were training a language model from scratch, a large dataset was needed for the model to learn. However, we could not use standard data loaders with a sizeable dataset since the data would not fit into the memory. For that reason, we stored both lemmatized sentences and inflected sentences in files and wrote a custom data loader that lazily reads the data.

Other than that, we used the standard HuggingFace Transformers Trainer class that is responsible for the training loop. We used the default loss for language models, which is negative log-likelihood. The model was trained on two V100 GPUs with a total memory of 32 GB on the CIIRC computational cluster. Thanks to the size of the memory, we were able to use a batch size of 512. However, the training still took about 4 days because of the training data size.

6.1.3 Automatic Evaluation

For the automatic evaluation of our task, we use the standard precision, recall, f1 score, and accuracy metrics. What interests us is whether the model learned when to inflect and when not to and, of course, whether it inflects correctly. In our task, we interpret TP as the number of correctly predicted word-forms when we were supposed to inflect. On the other hand, FP is the number of inflected cases that should not have been inflected. Furthermore, TN is the number of cases where the predicted word-form was the same as

	Original		
	High	Medium	Low
de	65.72 (NBL)	60.26 (UZH)	59.15 (UZH)
en	71.90 (CPH)	68.08 (UZH)	68.08 (UZH)
es	51.05 (NBL)	42.50 (CPH)	32.68 (UZH)
fi	34.82 (NBL)	27.06 (UZH)	24.40 (UZH)
fr	61.51 (CPH)	45.62 (CPH)	29.53 (CPH)
ru	56.73 (BME-HAS)	54.02 (UZH)	28.11 (UZH)
sv	55.96 (CPH)	47.87 (UZH)	32.77 (UZH)

Figure 6.1: Results of Sigmorphon 2018 Task 2 [9]

the input lemma, and finally, FN is the number of cases where we predicted lemma, but the target form should have been different.

- $Precision = \frac{TP}{TP+FP}$
- $Recall = \frac{TP}{TP+FN}$
- $F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision+Recall}$
- $Accuracy = \frac{TP+TN}{TP+FP+TN+FN}$

We evaluated the LLM model on the test split of the OpenSubtitles dataset that consists of 10 million sentences. Furthermore, we tested the baseline approach's performance that copies the input lemma to the output.

Table 6.3: LLM Automatic Evaluation

Model	**Precision**	**Recall**	**F1**	**Accuracy**
LLM	65.36	39.95	49.59	56.54
BASE	-	-	-	46.49

If we take a look at the Table 6.3, the resulting metrics do not look the best. While the model outperforms the lemma copying baseline, one would hope that it would do so in a more sizeable fashion. The main problem seems to be a low recall which signifies that the model often predicts lemma when it should inflect. To put it another way, it seems like the model rather struggles with learning which words to inflect and which not, than with correctly inflecting when it already decides to inflect.

Another insight that we could get is comparing with the results closest task that we could have found, the Sigmorphon 2018 inflection in context [9]. In Sigmorphon context inflection tackle similar task however in different languages. Besides, they evaluate their systems only with respect to the inflection of nouns, verbs, and adjectives only.

In Figure 6.1, we can observe that the best result in Russian, which is representative of the Slavic language, also reaches the accuracy of 56% in a high data setting with 100000 tokens. Again, it should be noted that the data and the evaluation are both different. Therefore, this comparison cannot be taken as seriously.

6.1.4 Human Evaluation

We also manually evaluated the LLM on the evaluation dataset created by linguists. Out of the 100 sentences, there were 23 mistakes if we would consider the exact labels. The 77% accuracy is not bad, but it should be noted that the dataset is relatively small and was made mainly for the insight into how the model works.

With that mind, if we would also consider the other labels possible for the similar context, then the number of mistakes would decrease to 16 mistakes since 6 of the cases can be considered debatable as can be seen in Table 6.6.

Firstly, we will look at the correctly evaluated sentences. The positive fact is that the model is capable of predicting correct target word-forms for all inflective parts of speech. The examples are shown in Table 6.4.

Table 6.4: Examples of correctly predicted sentences

Input sentence	**Target form**	**Part of speech**
Byl jsem v #Praha	Praze	noun
Co je v #dům není pro mě	domě	noun
Po dvoře honila #strakatý krávu	strakatou	adjective
Dali jsme si #španělský ptáčka	španělského	adjective
Je #já zima	mi	pronoun
#já se to nějak nezdá	Mně	pronoun
Skončila jsem na #třetí místě	třetím	numeral
Tento rok vyrazila už na #druhý dovolenou	druhou	numeral
Včera mi #volat moje kamarádka Radka	volala	verb
To #být moje boty	jsou	verb

If we take a closer look, we can see that the model seems to handle quite difficult inflections. In the first example where "Praha" inflects to "Praze", we see that model learned the change of phoneme h to z. Moreover, while the inflection of adjectives and numerals seems pretty standard, the inflection of pronouns seems rather challenging. This is especially true if we check how a pronoun "já" can be inflected into multiple word-forms with no signs of regularity. Nevertheless, the LLM still manages to tackle even this challenge. The last part of the speech left to discuss is verbs. The irregular verb "být" is inflected correctly, and also, the verb "volat" in the other example is inflected in the correct tense based on the context sentence.

Although LLM somehow learned various inflections, we are mainly interested in the model's mistakes. All incorrect examples are shown in Table 6.5.

The first thing that caught our attention is that many nouns are inflected incorrectly, even though we previously showed that LLM could inflect nouns. In the end, inflecting lemma #princezna to princezně does not seem that difficult since it is only a change of one letter. On the other hand, inflecting lemma #dům to domě seems more challenging, yet it was done correctly. So, where exactly is the problem?

Instead of finding the answer when looking at other noun inflections, we found it when checking the third example where adjective #horký should have been inflected to horkého. The first clue is that the target affix #ého was not present in the possibilities at all. Another clue was that a target form has more syllables than a lemma. However, it is rather an intuitive clue since the tokenizer does not work based on the number of syllables.

One could guess the problem lies in the fact that the target form has different tokens than the inputted lemma. Since the LLM model predicts which tokens most probably fit

Table 6.5: Examples of incorrectly predicted sentences

Input sentence	**Predicted form**	**Target form**
Babička mi vyprávěla pohádku o pyšné #princezna	princezny	princezně
V ruce držela šálek #horký čaje	horké	horkého
Kouřil #jedna za druhou	jedna	jednu
Bez #svůj mobilu se neobejdu.	svýho	svého
Premiér vyzval k #podpora ostatní země EU	podpory	podpoře
Maminka upekla ovocný koláč s #drobenka	drobenky	drobenkou
Tištěné noviny ztrácí na #význam	význam	významu
jeden z #my	my	nás
Zatluč to #kladivo	kladivo	kladivem
Panenka #vsítit dloubákem rozhodující gól a stadion propukl v jásot	vsítila	vsítil
Přišel jsem, viděl jsem, #zvítězit jsem	zvítězit	zvítězil
#Honzův plány vůbec nemá cenu se zabývat	Honzovy	Honzovými
Tatínek si koupil #kožený boty	kožený	kožené
dokonce jsem získala i pár nových #funkce	funkce	funkcí
Už se těším, jak ji #překvapit dárkem, který jsem jí koupil	překvapit	překvapím

Table 6.6: Debatable sentences for Lemmatized Language Model

Input sentence	**Predicted form**	**Target form**
Včera jsme s #kamarád vyrazili na houby	kamarády	kamarádem
Vyrazili jsme za #oni na návštěvu	ním	nimi
V řece hromadně #uhynout ryby	uhynou	uhynuly
Výběr #žárovka může mít vliv na zdraví	žárovek	žárovky
Někdy bych ale radši #jít hrát tenis	šel	šla
mně osobně ananas teda nijak #nevadit	nevadil	nevadí
nejsem vůbec moc #šikovný	šikovný	šikovná

the context in place of lemmatized tokens, there is no way for this model to predict additional tokens outside of the lemmatized tokens. Similarly, the model will also struggle in the opposite case when the lemma would have more tokens than the target form, which should be very rare but possible.

There are other problems not as severe as the one just mentioned but still to be noted. One of the problems directly follows from the nature of the model relying only on the sentence context. As we can see in the table, there are multiple cases where there could be various target word-forms just based on the sentence context. However, it seems that the model tends to prefer masculine gender instead of a feminine. The examples of this phenomena are #jít to šel instead of šla and similarly in the case of lemma #šikovný. Another problem seems to be that model learns some ungrammatical word-form inflections.

6.2 Lemmatized Sequence to Sequence Approach

Despite LLM having a solid performance, the major flaw tied to the different number of tokens of lemma and target word-form needs to be solved. A natural solution seems to be a sequence-to-sequence model. However, training a sequence-to-sequence model requires longer training compared to just training a language model using BERT.

Since we needed 2 V100 GPUs and 32 GB of memory just for training LLM from scratch, we tried to avoid training the sequence-to-sequence model from scratch, as it would likely take a longer time to train, and it was not simple to get the computational resources. Instead, we tried to use pre-trained models to save time and leverage transfer learning.

The decision of using pre-trained checkpoints for sequence-to-sequence tasks is also based on the article that describes the advantages of initializing encoder-decoder models with pre-trained checkpoints [31]. In their work, they introduced Transformer-based sequence-to-sequence models compatible with publicly available pre-trained BERT [13], GPT-2 [30] and RoBERTa [23] checkpoints. Based on their experiments, a pre-trained encoder is an essential component for the successful encoder-decoder model. Moreover, their models resulted in new state-of-the-art results on Machine Translation, Text Summarization, Sentence Splitting, and Sentence Fusion.

The model itself is a bert2bert model [31] and we will refer to it as lemmatized sequence-to-sequence model (LS2S). It uses pre-trained BERT for both the encoder and the decoder component. Since the target language is Czech, we used a pre-trained BERT multilingual cased checkpoint which is also trained on the Czech language. Thanks to that, we avoid computationally expensive training of the Czech language model. As a consequence, we can focus on the inflection generation itself.

Similarly to the Lemmatized Language Model approach, we want the model to encode the information about the input lemma. However, this time we do it differently. The example input representation looks like:

"I [SOL] be [EOL] John.",

where *"[SOL]"* and *"[EOL]"*, which stand for the start and end of the lemma, respectively.

The aim is to give the model a way to encode lemmas of the various token length. We hope that *"[SOL]"* and *"[EOL]"* tokens for lemmas will work on the same principle like *"[CLS]"* token for sentences in BERT. We suppose that the model could possibly learn the representation of context on the left of the lemma in the output corresponding to *"[SOL]"* token, and the right context representation in the output corresponding to *"[EOL]"* token. Moreover, the model still should have a full token representation of the lemma present in between *"[SOL]"* and *"[EOL]"* tokens.

6.2.1 Dataset Creation

Another problem that we found out during the hand evaluation of the first approach was possibly the OpenSubtitles dataset. We observed that the model learned ungrammatical word-forms in some cases, which might have been because of the language used in movies and TV series. Moreover, training a sequence-to-sequence model on the same amount of data would be practically impossible for us because of the computational resources. Therefore, we tried to find a more suitable Czech dataset.

In the end, we decided to use Czech Grammar Agreement Dataset for Evaluation of Language Models by [5]. The AGREE dataset contains 10 million Czech sentences with marked verbs in the past tense. The original purpose of the dataset is to predict the correct form of the verb in past tense based on the sentence context. As we can see, the task is somehow related to our context-aware word-form prediction. The difference is

that we want to predict word-forms for all inflectable parts of speech while also learning which words not to inflect, whereas the authors of AGREE dataset were focused only on verbs in the past tense. Despite this difference, we think that having a dataset designated for a related task could be helpful for the training of our model.

Similarly to the OpenSubtitles dataset, we need to lemmatize all sentences first. Moreover, we need to remove the marks from the marked verbs to create standard sentences. Afterward, we randomly choose a word to be selected as a base-form. Furthermore, we prepend and append the special *"[SOL]"* and *"[EOL]"* tokens to the selected word. After this, the preprocessing of the chosen base-form is finished, and the remainder of the sentence serves as context for prediction again.

6.2.2 Implementation Details

Following with the implementation details of lemmatized sequence-to-sequence model, we again worked with the HuggingFace Transformers library [38]. The library contains EncoderDecoder class that is suitable for initializing a working with sequence-to-sequence models. It can be used to initialize the encoder component with any pre-trained autoencoding model. Similarly, any pre-trained autoregressive model can initialize the decoder.

We initialized both components of the encoder-decoder model with pre-trained BERT multilingual cased checkpoints [13]. Therefore, compared to the LLM approach, there was no need to train a tokenizer from scratch since we could use the pre-trained one from BERT multilingual model. The input sentence is encoded to the sequence of maximum length of 64 tokens. The encoded sequence serves as an input to the decoder which output is limited to the maximum length of 8 tokens. Like this, we should be able to cover most of the input sentences and output word-forms with respect to the number of tokens while keeping the memory requirements as low as possible for more efficient training.

We again had to use a custom data loader that loads the inputs and labels lazily from preprocessed input files. The rest of the training loop is left again to the Trainer class. During the training, we again used the negative log-likelihood as a loss function. Furthermore, we applied early stopping with the patience of 3.

This time we trained the model on one V100 GPU with a memory of 16 GB. The computational resources allowed us to use a batch of 128 sentences. With these settings, the training took approximately two days.

6.2.3 Automatic Evaluation

Similarly to the Lemmatized Language Approach case, we also performed an automatic evaluation of Lemmatized Sequence to Sequence model. However, while we used similar metrics, we used the evaluation split of the AGREE dataset consisting of 996 sentences [5].

Table 6.7: LLM Automatic Evaluation

Model	Precision	Recall	F1	Accuracy
LS2S	97.71	93.93	95.78	95.48
BASE	-	-	-	45.38

Contrary to the automatic results of the previous model, the resulting metrics of the Lemmatized Sequence-to-Sequence Model on the AGREE dataset showed in Table 6.7 look much better.

However, the only thing that we can take from these results is that the model learned well on AGREE dataset specifically. There is not much else to deduce since the evaluation split of the AGREE dataset differs from the evaluation split of the OpenSubtitles dataset by four orders of magnitude. Moreover, the size of the evaluation split of the OpenSubtitles dataset is similar to the whole training size of AGREE dataset. Thus, we hope to gain more insight into the model in Human Evaluation.

6.2.4 Human Evaluation

In human evaluation on the dataset constructed by linguists, the LS2S model performed worse than its counterpart. Considering the strict labels, the model made 40 mistakes and therefore reached 60% accuracy.

Thinking about the positives first, the LS2S model managed to learn some correct word-forms, which the previous approach could not handle because of the different token number of the input and target form. For example, the model returned *"horkého"* given the input *"V ruce držela šálek #horký čaje"*. This is a positive fact as the primary motivation for the sequential approach was to manage these kinds of problems.

On the other hand, the model made significantly more mistakes. Even if we would subtract the 11 debatable cases, which can be seen in Table 6.8, the model still obviously struggles with correct inflection more than the LLM. The mistakes are shown in Table 6.9

Considering the controversial cases, most of them are verbs in a different tense than the target verb inflections. However, that is because the AGREE dataset contains verbs in the past tense for the sake of the subject-predicate agreement task.

However, moving back to the undisputable mistakes, the new model shows several cases that predict different words than the lemma suggests. For example, in some cases, it is the following word in the context, but on the other hand, in some cases, it seems like a random word.

Furthermore, the model seems to struggle much more on all the parts of speech except for verbs, especially with the complex inflection of Czech pronouns. We again attribute this to the fact that initially, the AGREE dataset was built for verb inflection. With that in mind, it seems like either the randomness during the base-form picking was not sufficient, or the dataset simply does not contain enough examples for various types of inflections.

Table 6.8: Debatable sentences for Lemmatized Sequence-to-Sequence Model

Input sentence	**Predicted form**	**Target form**
docela mě to #bavit	bavilo	baví
moje maminka nerada #vařit	vařila	vaří
i když taková diskuze také někdy #trvat poměrně dlouho	trvala	trvá
jiní lidé ho naopak #milovat	milovali	milují
všechno teď #běžet mnohem rychleji	běželo	šla
na druhou stranu #jíst trochu méně	jedl	jím
zvlášť v létě vypiji vždy hodně #aperol	aperolů	aperolu
Já už tu na tebe #čekat hodinu	čekal	čekám
Natrhala jsem kvítí na #louka	loukách	louce
Včera jsme s #kamarád vyrazili na houby	kamarády	kamarádem
Už se těším, jak ji #překvapit dárkem, který jsem jí koupil	překvapí	překvapím

Table 6.9: Incorrect sentences for Lemmatized Sequence-to-Sequence Model

Input sentence	**Predicted form**	**Target form**
Premiér vyzval k #podpora ostatní země EU	podporám	podpoře
Tento rok vyrazila už na #druhý dovolenou	druhé	druhou
Stala se obětí #domácí násilí	domácích	domácího
Kvůli #vážný nemoci musí držet přísnou dietu	vážnému	vážné
#Granada je krásné město	Granadě	Granada
Zítra půjdeme do #Alhambra	Alhambra	Alhambry
Jeden z #my	my	já
Dali jsme si #španělský ptáčka	španělský	španělského
Je #já zima	zima	mi
To #být moje boty	je	jsou
Zatluč to #kladivo	kladivo	kladivem
Nakrájej nožem #ten bramboru	bral	tu
#ten květiny potřebujou zalít	to	ty
#já se to nějak nezdá	mě	mně
Neviděl jsem #on už mnoho let	jich	ho
Za koho #já máš?	máš	mě
Zakopl jsem v předsíni o #tátův boty	tátově	tátovy
#Honzův plány vůbec nemá cenu se zabývat	Honzovou	Honzovy
Tatínek si koupil #kožený boty	kožený	kožené
Nekoukej na #ona takhle	ní	ni
Stala se #padesátý držitelkou ceny	padesátých	padesátou
#nikdo se nic nestalo	nikdo	nikomu
Tenhle rok #být ale pořád zima	jsme	je
Mnoho lidí nesnáší ananas na #pizza	pizzu	pizze
Ale takové diskuze zaberou vždy hodně #čas	časů	času
Podle mě jsem vytvořila opravdu #báječný věty	báječný	báječné
Proto studuju na #univerzita	univerzitách	univerzitě
Někdy bych ale radši #jít hrát tenis	jít	šla
Když #můj maminka nerada vaří	můj	moje

Chapter 7

Conclusion

The aim of our thesis was to create a context-aware model for the prediction of inflectional word-forms. This objective was motivated by the vision of a user-friendly inflectional system without any need for hand-crafted features. Moreover, the context-aware approach was supported by the recent success of the Transformer and its excellent accessibility, as we tried to rely solely on the context encoding for the prediction.

To better understand the task of inflectional word-form prediction, we first researched current state-of-the-art methods. Afterward, we studied the linguistic concepts needed to carry out the linguistically oriented research task successfully. Furthermore, we analyzed fundamental technical ideas related to sequential learning. Notably, we got acquainted with the architecture of Transformers and their use in the field of NLP.

Based on previous research, we have proposed two models that address context-aware prediction of inflectional word-forms. Furthermore, we also prepared lemmatized Czech datasets for training and model evaluation using MorphoDiTa.

The first model we designed was a Lemmatized Language Model. Here, we trained a Czech language model from scratch. We took inspiration for the Lemmatized Language Mode from the BERT Masked Language Model. However, instead of predicting the value of the masked inputs using the context of surrounding words, we tried to predict the value of the lemmatized tokens. The model accepts inputs like: *"I #be John."*, and the goal is to inflect base-form *"#be"* to the target form *"am"*.

The Lematized Language Model was trained and evaluated on Czech lemmatized OpenSubtitles data which consists of 135.9 million sentences. We measured the automatic evaluation on the evaluation split formed of 10 million sentences. While the baseline copying model reached the accuracy of 46.49%, the resulting accuracy of LLM was 56.54%, which suggested room for improvement for the LLM model despite outperforming the baseline. Solely based on the metrics, we could conclude that the model often predicts lemma when it should inflect. However, deeper insight into the model was provided by human evaluation.

In human evaluation, we performed analysis on the small dataset constructed by linguists. We found out that the model inflected surprisingly well and achieved 77% accuracy on these data. On the other hand, we found a major issue that the model could not predict the cases when the target form has different tokens than the inputted lemma. Furthermore, we found more minor problems like the ambiguity of the context and bias of the model towards specific grammatical categories.

As a follow-up to the first model and the found issues, we suggested a second approach. Since the main problem was a different token length of base-forms and targets, we decided that the possible solution could be a lemmatized sequence-to-sequence model. The model itself is a bert2bert model where a pre-trained BERT checkpoint initialized both encoder and decoder. Specifically, we used a pre-trained BERT multilingual

model trained on 104 languages, including Czech, instead of training from scratch like in the previous case. The model also differs in the input formatting, which we can see in the example input : *"I [SOL] be [EOL] John."*, where *"[SOL]"* and *"[EOL]"* are special tokens denoting the start and the end of lemma.

The training and automatic evaluation of the Lemmatized Sequence-to-Sequence model was performed on lemmatized Czech Agreement dataset, which was originally a dataset for the related task of grammatical alignment.

The results of the automatic evaluation showed a surprisingly high accuracy of 95.48%. However, we could only conclude LS2S model learned well on the AGREE data. Other than that, we could not make any conclusions as the size of the evaluation split of AGREE dataset differs from the evaluation split of the OpenSubtitles dataset by four orders of magnitude.

Our assumption that the model learned well only on AGREE data was confirmed during the human evaluation, where the LS2S performed worse than its counterpart. While the model was capable of solving the cases when the target form has different tokens than the inputted lemma, it exhibited multiple other mistakes. Notably, the model showed several cases when it predicted different words than the lemma suggests. Furthermore, the model seemed to struggle more with all the parts of speech except for verbs.

Overall, there was a notable difference between automatic evaluation and human evaluation. There were several factors that influenced that. One of them was the small size of the human evaluation dataset compared to the size of automatic evaluation datasets. Another was the fact that in human evaluation, we focused specifically on certain parts of speech like nouns, pronouns, and verbs, while in the original training setting, the model was supposed to learn whether to inflect before learning how to inflect.

As future work, we believe that changing the random masking for choosing the base-form would improve the performance of the systems. For example, we could examine POS tags of the words to create a balanced dataset, or we could focus specifically on certain types of inflections.

Bibliography

[1] R. Adam et al. *Morfologie*. Univerzita Karlova v Praze, Nakladatelství Karolinum, 2015.

[2] R. Aharoni and Y. Goldberg. Morphological inflection generation with hard monotonic attention. *arXiv preprint arXiv:1611.01487*, 2016.

[3] M. Aronoff and K. Fudeman. *What is morphology?*, volume 8. John Wiley & Sons, 2011.

[4] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[5] V. Baisa. Czech grammar agreement dataset for evaluation of language models. In *RASLAN*, pages 63–67, 2016.

[6] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[7] G. G. Chowdhury. Natural language processing. *Annual review of information science and technology*, 37(1):51–89, 2003.

[8] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[9] R. Cotterell, C. Kirov, J. Sylak-Glassman, G. Walther, E. Vylomova, A. D. McCarthy, K. Kann, S. J. Mielke, G. Nicolai, M. Silfverberg, et al. The conll–sigmorphon 2018 shared task: Universal morphological reinflection. *arXiv preprint arXiv:1810.07125*, 2018.

[10] F. Deloche. Gru unit, Apr 2018. [Online; accessed 24-April-2018].

[11] F. Deloche. Lstm unit, Apr 2018. [Online; accessed 24-April-2018].

[12] F. Deloche. Recurrent neural network, Apr 2018. [Online; accessed 24-April-2018].

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

[14] G. Durrett and J. DeNero. Supervised learning of complete morphological paradigms. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1185–1195, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

[15] M. Faruqui, Y. Tsvetkov, G. Neubig, and C. Dyer. Morphological inflection generation using character sequence to sequence learning. *arXiv preprint arXiv:1512.06110*, 2015.

[16] A. Fraser, M. Weller, A. Cahill, and F. Cap. Modeling inflection and word-formation in smt. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 664–674, 2012.

[17] S. Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

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

[19] M. Hulden, M. Forsberg, and M. Ahlberg. Semi-supervised learning of morphological paradigms and lexicons. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 569–578, 2014.

[20] P. Koehn. *Statistical machine translation*. Cambridge University Press, 2009.

[21] J. Lafferty, A. McCallum, and F. C. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[22] P. Lison and J. Tiedemann. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. 2016.

[23] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[24] M.-T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[25] D. Nadeau and S. Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[26] G. Nicolai, C. Cherry, and G. Kondrak. Inflection generation as discriminative string transduction. In *Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 922–931, 2015.

[27] J. Nouza, J. Zdansky, P. Cerva, and J. Silovsky. Challenges in speech processing of slavic languages (case studies in speech recognition of czech and slovak). In *Development of Multimodal Interfaces: Active Listening and Synchrony*, pages 225–241. Springer, 2010.

[28] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. PMLR, 2013.

[29] M. Pravdová and I. Svobodová. *Akademická příručka českého jazyka*. Academia, 2014.

[30] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[31] S. Rothe, S. Narayan, and A. Severyn. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*, 8:264–280, 2020.

[32] H. Schmid, A. Fitschen, and U. Heid. SMOR: A German computational morphology covering derivation, composition and inflection. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal, May 2004. European Language Resources Association (ELRA).

[33] H. Schütze, C. D. Manning, and P. Raghavan. *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge, 2008.

[34] A. Stolcke. *Bayesian learning of probabilistic language models*. PhD thesis, University of California, Berkeley, 1994.

[35] J. Straková, M. Straka, and J. Hajic. Open-source tools for morphology, lemmatization, pos tagging and named entity recognition. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 13–18, 2014.

[36] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215*, 2014.

[37] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[38] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

[39] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[40] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR, 2015.

[41] M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.