Studies in Language Structure using Deep Learning

Adam Ek

Department of Philosophy, Linguistics and Theory of Science

UNIVERSITY OF
GOTHENBURG

I hereby declare that the work presented in this thesis is my own work and that the research reported here has been conducted by myself unless indicated otherwise in the preface of each chapter

[...] whatever happens.

Fyodor Dostoevsky
Notes from the Underground

Acknowledgemets

First and foremost I would like to express my gratitude and thanks to my supervisors Jean-Phillipe Bernardy and Stergios Chatzikiriakidis, for helpful guidance, discussions, and extra-linguistic entertainment. This thesis would also be improbable without the discussions over food and coffee with my fellow Ph.D. students and colleagues at Gothenburg University.

Post-scribbling one should not forget to express gratitude to the author, myself. Thank you Adam, for our unending curiosity and the work put into this thesis.

A special thanks go out to the Climbing team, for the adventures both on and above ground, with a good mixture of moods.

To all others whom it may concern, those who have been there both directly and indirectly - Thank you.

Table of Contents

List of Figures

List of Tables

Part I

Preliminaries

”When you think about it, thinking about thinking is the hardest sort of thinking there is. Which makes you think.”

Philomena Cunk

Chapter 1

Introduction

In recent years, the use of computers for analysing language have become more and more prevalent. This rise of the machines for processing language have been facilitated by advances in statistical methods. But even with the field moving at a rapid pace several fundamental issues remain. In this thesis we are interested in the ability of computers to construct meaning representations of language using grammatical information.

When a human encounters a piece of text the meaning can be understood based on the linguistic competence and previous experiences. Computers on the other hand need some set of instructions to analyse language. For this purpose rule-based, statistical and neural network methods has been developed as generic procedures for teaching computers to perform some language-based task.

As an example of a language task we can consider a basic type of analysis: *who did what to whom?*. That is, given a sentence we want to identify *who* did something, *what* they did, and to *whom* they did it. For this analysis the grammar of the language help us answer the question by considering the structure of the sentence. [1] For example, consider sentence (1) below:

[1] Grammar and structure are used interchangably throughout this thesis.

(1) the cat chases a dog

The grammar of a language specifies how different linguistic properties are expressed, such as the *who*, *what*, and *whom*. Using our knowledge of English grammar then we see that *the cat* (who, the subject) is *chasing* (what, the verb) *a dog* (whom, the object). This is because in English syntax subjects are placed to the left of the verb, and objects to the right as shown in Figure 1.1.

the	cat	chases	a	dog
-	SUBJECT	VERB	-	OBJECT

Figure 1.1: Subject-Verb-Object structure in English.

We can also note that it is a specific cat that is chasing some dog, which is expressed by a definite (*the*) or indefinite (*a*) article. These relations are expressed by the relations between words, but words themselves also have structure. A notion that becomes important when considering the question of *who did what to whom* is the PREDICATE-ARGUMENT STRUCTURE. The predicate-argument structure is how a predicate (typically a verb) takes arguments, this can both be expressed syntactically and semantically. In the example above (Figure 1.1) the syntactic predicate-argument structure can be formalized as the word *chases* takes two arguments (transitive); the SUBJECT and the OBJECT. Other verbs do not have the same arguments; for example *give* requires three arguments (ditransitive), the SUBJECT *gives* the OBJECT an INDIRECT OBJECT. There are also verbs like *ran* that only take one argument (intransitive), the SUBJECT.

Consider the verb *chases*: it is indicating that something is happening currently. So going back to (1), we see that the cat currently chases a dog. But many more fine-grained details can be expressed by modifying the structure of *chases*. For example, if the *chases* happened in the past, this is expressed by changing the word form to *chased*.

Changing a word's internal structure to indicate linguistic properties is a common process. Many languages in the world use this instead of the order of the words in relation to other words for indicating what is the subject and object. One such language is Russian which uses a CASE-system to indicate this. In Russian the NOMINATIVE case is used to indicate the subject, while the ACCUSATIVE case is used to indicate the object. For example, if a cat is chasing a dog this can be expressed as:

(2) Кошка преследует собаку
cat.NOM chase dog.ACC
'the cat chases a dog'

In (2), the subject is expressed by the internal structure of the word rather than by its positioning in relation to the verb. This means that it is possible to rearrange their positioning without altering the meaning:

(3) Собаку преследует кошка
dog.ACC chase cat.NOM
'the cat chases a dog'

From this small comparison between Russian and English, we can observe that the structure of the languages is different. Most languages use different structures to indicate linguistic processes. However, many languages use similar structures for expressing linguistic properties. For example, both Swedish and English use word order to indicate the subject and object. We can then see that grammar is a set of rules (or conventions) for expressing information such that it is successfully transmitted between two or more people.

But in analyzing language, we are often interested in more than its structure. We want to find an interpretation of its *meaning*. Because simply knowing that *cat* is the subject and *dog* is the object, without any information about what these two words refer to, does

not help us ascertain what the sentence means. To find the meaning we need to consider the lexical meaning of each word, then consider how they relate to each other. Thus, to interpret the meaning of a sentence both how words relate to each other, how the words' internal structure looks, and what the words themselves mean have to be considered.

However, for computers, this is a lot to take into account, especially when a computer only performs the actions specified by its program. So to analyze a sentence, a computer would need a full set of instructions both on how the language is structured and some notion of the words meaning. But specifying this can be a difficult and tedious task. For some narrow problems it is sufficient to find a smaller set of rules that can perform some tasks, but scaling up presets issues. To circumvent this issue, statistical methods have been used. The advantage of statistical methods for processing natural language is that what a system learns is dependent on the data, rather than rules. So statistical systems can be adapted to whatever data is available, they can learn what statistical regularities occur, for example, that the subject is somewhere to the left of the verb. These learned statistical regularities can then be used to analyze new sentences that the system has not seen. In particular, neural networks have been developed and used for learning how to analyze (and produce) language in recent years. Neural networks build vector representations of words and learn to produce informative vectors such that they are useful for the task at hand.

In this thesis, we explore how a computer can learn the structure of grammar, using both rule-based and neural methods. Additionally, we also investigate how grammatical information can help computers understand the meaning of language. This leads us to pose two general research questions, the first one concerns how to discover grammatical structure, or more specifically, how we can construct computational representations of grammatical structure:

RQ1 How to obtain representations of grammatical structure?

The second research question concerns how to use the representations of grammatical structure to obtain semantic representations:

RQ2 How to predict semantic phenomena based on representations of grammatical structure?

This thesis is organized into two parts. In Part I, an introduction to how neural networks learn to analyze language using statistical methods is given in Chapter 2. Chapter 3 focuses on introducing the relevant linguistic background for our investigations. Chapter 4 expands the motivation behind the research questions given the neural and linguistic preliminaries and also presents a summary of the produced research along with how these answer the research questions. Part II presents the papers published towards addressing the research questions.

"We don't have time to be ourselves. We only have time to be happy."

Albert Camus

Chapter 2

Neural Networks

In recent years neural networks have become the predominant way to process language with computers. One of the main strengths of neural networks over other statistical and rule-based methods is their ability to effectively model and learn from large amounts of data. This allows models to find answers[1] to natural language tasks, such as providing semantic and grammatical information about words and sentences.

To successfully perform a task, neural network models represent words as vectors and sentences as matrices of word vectors. The vectors are transformed in such a way that the objective of a task can be predicted. The unreasonable effectiveness of neural networks (Sejnowski, 2020) comes from the ability to search for effective parameters θ that transform the input to solve a task. The parameters used by a model are found using gradient descent-based methods. Thus, given some input x and parameters θ, an output $\hat{y}$ is produced:

$$f(x, \theta) = \hat{y} \tag{2.1}$$

The remainder of this chapter is dedicated to exploring how rea-

[1] In different tasks there are many acceptable answers. When "answer" or "solution" is used in this thesis we merely refer to one possible answer.

sonable outputs are obtained, how different neural architectures are constructed for different tasks, and how parameters can be learned from data.

2.1 Language modeling

One task that neural networks excel at is language modeling (Bengio et al., 2000). In language modeling the objective is to predict missing words[2] of a sentence. For example, consider the following sentence with a missing word:

(4) *the cat sat on the [MASK]*

The language modeling task predicts which word fits instead of the [MASK] token. Formally, the language modeling objective for predicting the [MASK] token at position n in the sentence can be formulated as:

$$P(w_n|w_{<n}) \tag{2.2}$$

Of course, many different words fit as a continuation, but what is considered correct is what appears in a *reference* sentence. A reference sentence is a naturally occurring sentence from a corpus.

To predict missing words in a sentence, a model builds contextual representations of the words that occur before the mask token. The representations are obtained by reading the sentence from right-to-left (*uni-directional*), or by reading the sentence in both the right-to-left and left-to-right directions (*bi-directional*). Typically, but not necessarily, language modeling is based on co-occurrence statistics. That is, how often does a word occur in the context of another word. However, other approaches exist for modeling language. In particular, approaches that combine statistical methods with rules, as given

[2] [MASK] will be used to indicate an unknown word that should be predicted.

by a context-free grammar (Meteer, Rohlicek, 1993; Jurafsky et al., 1995).

In Example (4) the missing word is the last word of the sentence. But the language modeling objective can also be extended to predicting a randomly missing word (*masked language modeling*) in a sentence. For example:

(5) *the [MASK] sat on the mat and purred*

To predict the missing word in masked language modeling there may be context both to the left and right of the missing word. Thus, the objective for predicting the [MASK] token at position n in the sentence can be formulated as:

$$P(w_n|w_{<n}, w_{>n}) \tag{2.3}$$

Thus, in masked language modeling, we are given the complete sentence where the missing word needs to be predicted. Whereas in standard language modeling, only context to the left is given. These two formulations of language modeling give a model different contexts to represent. But to produce a reasonable word in place of the missing word a model needs to use the representation of the context to predict not only the word that fit semantically but also various grammatical features[3] and sentence structures such as punctuation. When mentioning language modeling in the remained of this thesis we refer to masked language modeling unless otherwise specified.

Language modeling has been used to construct models (Peters et al., 2018a; Devlin et al., 2019; Liu et al., 2019b) trained on large amounts of data. These models are trained *end-to-end* meaning that given an input the output is produced with no intermediate processing steps, such as explicitly extracting the important features. Instead, models are expected to learn what the important features are

[3]If the lemma *run* should be produced, which of the various word forms: *runs*, *ran*, and so on, fit into the context.

from observing the data. The purpose of these models is not necessarily to perform well in language modeling but on other tasks, such as answering questions or generating utterances in a dialogue. [4] Training large models with the language modeling objective to obtain representations is called *pre-training*. The model is then applied to another task; and if the representations are trained further on the task, it is called *fine-tuning*. If the model is applied to another task, and is not further trained, it is called *zero-shot learning*. The intuition is roughly: are the representations obtained from the pretraining sufficient to complete another task without informing the model of the specifics of the task.

Language modeling can be seen as a general language understanding task. Many aspects of language are needed to perform language modeling, such as knowledge of sentence structure and semantic plausibility. For instance, if a model is passed the masked sentence: *A word is a collection of [MASK]* and predicts that [MASK] should be replaced with *cats*, this would indicate that the model considers *cats* as a plausible continuation to the sentence, which of course is nonsense. However, if the model predicts *cats* it did something right; it correctly predicts that a collection of something indeed is more than one thing. These two factors, awareness of grammar and semantic plausibility, allow the representations obtained from language models to be applied to other language tasks successfully. When the representations obtained from a language model are used for another task, it is an example of *transfer learning* (Pan, Yang, 2010; Howard, Ruder, 2018; Peters et al., 2019). Transfer learning is the ability to adapt representations learned in one language from one task to another task, to a new writing style, or to a new language.

[4] To evaluate these models the GLUE (Wang et al., 2018) and SuperGLUE (Wang et al., 2019a) benchmarks are often used.

2.2 Tokenization and word embeddings

The goal of processing language is to build an understanding of its meaning. As such, we must define and represent the different units of language (Webster, Kit, 1992). This often means identifying words and sentences and assigning a meaningful representation to them, which parameters of a model can manipulate. Identifying words in a text is called tokenization. The words we find are then represented as d-dimensional vectors, also known as *embeddings* (Rumelhart et al., 1986; Mikolov et al., 2013a,b), and sentences are represented as matrices of size (k, d), where k is the number of words in the sentence and d the size of the word embeddings. However, it is problematic to identify what a *word* is across languages. The problem with words is that there seem to be no good set of guidelines for identifying them (Haspelmath, 2017). In this section, we briefly explore this problem regarding tokenization.

An English sentence can be tokenized by identifying words as a sequence of characters surrounded by white space. However, it is problematic to assign a word embedding to the text fragment *isn't* surrounded by white space. The problem is that it is a contraction of the two words *is* and *not*. So if an embedding is assigned to *isn't* this embedding has to represent two words. Therefore, it is good to separate *isn't* into *is* and *not* and assign an embedding to each. This is because otherwise the embedding for *isn't* will encode two words, and then the information for the embedding of *isn't* will not share information with the embeddings for *is* and *not*. Another example is proper names such as *New York City*. All the words in the proper name carry meaning individually; however, the words in combination have a specific meaning that would get lost if they are assigned an embedding each. Thus, *New York City* should not be split at white spaces and instead an embedding should be created for the sequence. In the word *New York City* we do not obtain the meaning from the composition of the parts but rather the parts as

one unit. This problem becomes more acute in other languages that, for example, do not use white-space, like Mandarin (Huang et al., 2007) or Thai (Haruechaiyasak, Kongthon, 2013). Another aspect of language to keep in mind is how the meaning of words is augmented. For example, in agglutinative languages (Pan et al., 2020; Li et al., 2020) different parts of a word have distinct grammatical meanings. A natural question is whether words should be represented as-is or by splitting texts into smaller units such as sub-word or character embeddings (Bojanowski et al., 2017; Martin et al., 2020).

A general problem for representing words is that the vocabulary of a language is theoretically infinite, while a computer's memory is finite. As such, having a separate representation for each word form (for example *run*, *runs*, *running*, ...) becomes infeasible if we want to capture as much as possible from the vocabulary of a language. A recent approach to tackling this problem is considering *sub-word* tokens (Sennrich et al., 2016; Kudo, 2018; Bojanowski et al., 2017; Church, 2020) instead of words. A sub-word is a substring of a word; for example, the word *moreover* can be separated into the sub-word *more* and *over*. To create the word *moreover* we would add the sub-words *more* and *over* together. Using sub-words allows us to represent a larger vocabulary in the computer memory since we can compose the sub-words to form larger words. Another benefit of this method is that the sub-words *more* and *over* will obtain the information from the cases when they do not occur in the word *moreover*.

But words can be broken down into smaller units, the characters. By assigning a representation to each character the relations between them can be modeled accurately, but at the cost of additional compute (Edman et al., 2022). Since character embeddings represent words by considering all the characters, it has been used for fine-grained word predictions such as word similarity (Chen et al., 2015) and named entity recognition (Santos dos, Guimarães, 2015). Additionally, it has also been proposed as a technique for dealing with

languages that have very large vocabularies (Yu, Vu, 2017; Özates et al., 2018).

In neural networks the embeddings of words, sub-words, or characters are obtained from a matrix, typically called an embedding layer, of size (n, d) where n is the number of items that have a representation, and d is the dimensionality of the vectors. The embeddings are typically initialized from a standard normal distribution $\mathcal{N}(0, 1)$ (Kocmi, Bojar, 2017), which is then trained on some task.

2.2.1 Sentence embeddings

Sentences are composed of words in a linear order, and neural networks initially represent sentences in the same manner. That is, a sentence is a matrix of size (n, d), where n is the number of words in the sentence, and d the dimensionality of the embeddings (Blacoe, Lapata, 2012; Mitchell, Lapata, 2010; Wieting et al., 2016). For some tasks, we want to predict whether a sentence is positive or negative (sentiment analysis), thus a sentence composed of n words or sub-words with a dimensionality of d should be compressed to a single vector of size d. For this, we need to *pool* the information contained in the word embeddings. One can also avoid pooling the information from the word representations by using the final or summary representation produced by some models. However, one of the main motivations for pooling the information is compositionality. That is, by reiterating Partee (1995):

> *The meaning of a whole is a function of the meaning of the parts*

By the same process, we can consider meaning in neural networks to be a function of the word embeddings of the sentence. Where the function is the structure of the neural network along with its parameters. By considering the sentence as word embeddings in a matrix, we can compose and combine them with various methods

that allow us to have more control over how the meaning representation for the sentence is constructed.

Several pooling techniques have been developed, with popular ones being the max, sum, or mean pooling (Reimers, Gurevych, 2019; Ek, Bernardy, 2020a; Ács et al., 2021). These techniques, the max, mean, or sum are computed over the columns to compress a matrix of size (n, d) to size d (Conneau et al., 2017). A sentence X, composed of three words with d dimensions, is shown in Equation (2.4).

$$X = \begin{bmatrix} x_0^0 \dots x_d^0 \\ x_0^1 \dots x_d^1 \\ x_0^2 \dots x_d^2 \end{bmatrix} \tag{2.4}$$

A new vector X' (the sentence embedding) is computed that represents the words in a sentence by passing each column in X through a function f, which is a function that takes the input values and reduces them to one value. This is so that sentences that have different lengths are compressed to the same size, regardless of the sentence length. The size of the sentence representation will be that of the dimension of the word embeddings it contains.

$$x = [f(x_0^0, \dots, x_0^2), \dots, f(x_d^0, \dots, x_d^2)] \tag{2.5}$$

When using pooling, a neural network learns to select information from a particular dimension from all words, such that the selected information is most predictive for the task. More advanced sentence representation techniques such as Arora et al. (2017); Wang et al. (2017) have also been developed.

2.3 Linear and non-linear transformations

The purpose of neural networks is to predict something, such as which word completes a sentence. To do so, we must extract the information we want from the word embeddings. Crucial to this is the

so-called linear transformation.[5] A linear transformation takes as input a vector v that is multiplied with a parameter θ and parameter b (the *bias*[6]) is added to the result. The goal of this transformation is to manipulate the input such that it is easier to predict the expected output.

A linear transformation can be written as $f(v; \theta, b) = v^\top \theta + b$, where v is the input. We can think of the input (a word vector) as occupying a region, or a set of points, in a vector space, where the values in v define this region. The parameter θ then transforms this region which v occupies by contracting or expanding it. Finally, the bias parameter b is used in a linear layer to improve the generalization capabilities of the layer, for a more in-depth exposition of the bias the interested reader is referred to Bishop, Nasrabadi (2006, Chapter 3.1).

Linear transformations are used to manipulate the input to become more informative for a given task. Another use of a linear transformation is to make a prediction, for example, a label associated with the input. If we want to predict a label for some representation, we want the output to be of size l, where l is the number of possible labels. A design choice in linear transformation predictions is that they contain a parameter for each label l which we predict. So, if our input is of size 2 and there are four possible outputs, the operations performed[7] are shown in Equation (2.6), where x is the input and the w's are the values of the parameter θ.

$$\begin{bmatrix} w_0^0 & w_1^0 \\ w_0^1 & w_1^1 \\ w_0^2 & w_1^2 \\ w_0^3 & w_1^3 \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} x_0 \cdot w_0^0 \\ x_0 \cdot w_0^1 \\ x_0 \cdot w_0^2 \\ x_0 \cdot w_0^3 \end{bmatrix} + \begin{bmatrix} x_1 \cdot w_1^0 \\ x_1 \cdot w_1^1 \\ x_1 \cdot w_1^2 \\ x_1 \cdot w_1^3 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} \tag{2.6}$$

[5] Linear transformation and linear layer will be used interchangeably.

[6] When adding the bias, we have an *affine* transformation.

[7] The bias parameter have been removed from the computation to simplify it.

The representation of the labels is each row in W. As such, we can extract and inspect the parameters associated with each label and compare their structures and similarities. When the scores for the possible labels have been computed, the softmax (Equation (2.7)) function is applied. The label that is assigned the highest score is selected as the predicted label.

Activation functions

In the previous section linear transformations were introduced, but not all relationships are linear. To model relationships that are not linear we can augment linear transformations with activation functions. By adding an activation function to a linear transformation we obtain a non-linear transformation. There are several activation functions with different strengths and weaknesses for this purpose. We briefly present some of the more common activation functions below in Table 2.1.

Activation	Function
Sigmoid	$\frac{1}{1+e^{-x}}$
Tanh	$\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
ReLU (Nair, Hinton, 2010)	$\mathsf{max}(0, x)$
Leaky ReLU (Maas et al., 2013)	$\mathsf{max}(0, x) + 0.01w \times \mathsf{min}(0, x)$

Table 2.1: Activation functions.

The output of a linear layer is a vector containing some values. However the purpose is to predict some labels, these values, the *logits*, are unbounded and can take on any value. For learning purposes, we want to normalize the logits so they can be interpreted as probabilities when predicting a label. We obtain a probabilistic interpretation of the logits by using the softmax function Equation (2.7).

$$\text{Softmax}(x) = \frac{e^{x_i}}{\sum_K^{j=1} e^{x_j}} \tag{2.7}$$

The softmax function takes a vector containing unbounded values as input and re-scales them into the range $[0, 1]$. The sum of the values now sums to 1, allowing us to interpret each value as a probability.

2.4 Learning

For now, we can consider a neural network simply as a function f, that takes as input some data x and produces an output $\hat{y}$. We can formalize learning a task as finding a neural network $f(x, \theta) = \hat{y}$ such that $\hat{y} = y$ where y are what label a human would assign to the input x given the parameters θ. Learning a task in its essence is finding a suitable θ while finding f is the modeling process, i.e., finding suitable transformations f which produce an effective θ.

If we have annotated data, like in the example above, and attempt to learn the mapping from x to y, we call it supervised learning. However, we may also want to learn without having annotated data as annotating the data can be expensive to produce and includes annotation biases, which in addition to producing poorly annotated data where models learn the wrong thing, can result in social biases such as racism and sexism. For an overview of this problem, the interested reader is referred to (Blodgett et al., 2020).

Models learn their parameters based on the data used to train them. From this it follows that for a model to learn efficiently, there must be a sufficient amount of data for the model to learn from. But the data available does not always allow for this. Training a model in these cases is called *low-resource learning* (Magueresse et al., 2020; Singh, 2008). For models to learn in these cases, heuristics and data augmentation techniques can be used. We discuss several of these

techniques in Chapter 3 when describing approaches to natural language processing.

The task of a neural network is to predict an $\hat{y}$ from the data presented to the model. Where the $\hat{y}$ may be words, labels, or something else. Let us consider a task that has three labels (y_1, y_2, y_3), the model assigns a score to each of these labels. To obtain these scores, the neural network passes the input x through a set of parameters $\theta_0, \ldots, \theta_n$. The goal of passing the data through the parameters $\theta_0, \ldots, \theta_n$ is that the output should be as close as possible to what a human annotator assigned to the example.

For the neural network to produce output labels that resemble the human-assigned labels, the parameters $\theta_0, \ldots, \theta_n$ have to transform the input data in an informative manner. To accomplish this, neural systems make use of three core components:

- **Backpropagation**: The backpropagation algorithm (Rumelhart et al., 1986) estimates how the values in the parameters should change to produce a better output the next time it gets input.
- **Loss function**: The loss function estimate how good or bad the final prediction is.
- **Optimizer**: The optimizer applies the updates to the parameters that the backpropagation estimated based on the loss function.

2.4.1 Backpropagation

In neural networks, learning from data means manipulating the parameters in the model such that they transform the input so that the output gets closer and closer to our desired output. An outline of how information flows in a neural network is given in Figure 2.1.

This neural network is composed of two parameters, θ_0 and θ_1 ordered sequentially. First, the input x, let us imagine it is a tokenized sentence, is passed to an embedding layer (θ_0) where word embeddings are extracted. Next, the word embeddings are transformed with θ_1 followed by a softmax activation function to produce an output $\hat{y}$.

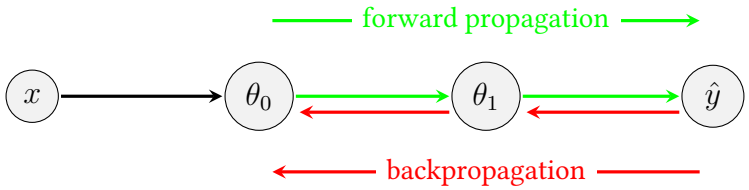


Figure 2.1: The flow of back (red arrows) and forward (green arrows) propagation in a simple neural network with two parameters, θ_0 and θ_1.

We compute the transformations that result in specific outputs during the forward propagation. To estimate how well the neural network performed in the task, we use a *loss function*. When calculating the loss, it becomes inefficient to do it one example at a time. Instead of computing one example at a time, several examples are collected into a *batch* for which the loss is calculated. With a loss for a batch of examples, we can train a neural network by computing the partial derivatives for each parameter given the value of the loss function. When we have obtained this information, we can use an *optimizer* to update the values in our parameters to produce a better output next forward propagation.

2.4.2 Loss functions

A loss function is a way of quantifying or estimating how well the model is performing on a task. To estimate how much a prediction resembles the annotated labels, we want to estimate how wrong the

prediction is, then minimize the model's errors. We often want to predict some discrete categorical variable for which cross-entropy loss can be used. For example, we might want to predict a relationship between two sentences (*classification*) or the part-of-speech tag associated with a word (*sequence labeling*). As a minimal example, let us consider part-of-speech tagging with a subset of the parts-of-speech labels (NOUN, VERB, and DETERMINER). For example, if we want to tag the sentence

(6) The cat chases squirrels

with our set of labels, the neural network produces a score for each label that is converted to a probability with the softmax function for each word. We then obtain an output like Table 2.2.

Word	NOUN	VERB	DETERMINER
The	0.08	0.02	0.9
cat	0.7	0.25	0.05
chases	0.2	0.75	0.05
squirrels	0.9	0.02	0.08

Table 2.2: An example of the softmax output for part-of-speech tagging.

The models' output should then be compared to the labels in the dataset. To do this we create a probability distribution with all probability mass on the correct part-of-speech tag, e.g., cat $= [1.0, 0.0, 0.0]$. To compare the output with the gold, we use the *cross-entropy* equation, defined as:

$$\mathcal{L}_{\text{Cross-Entropy}} = -\sum_i p(y_i) \log p(\hat{y}_i|x_i) \tag{2.8}$$

Where i is the index of a label in the example, $p(y_i)$ is the probability assigned by annotators of the dataset, and $p(\hat{y}_i|x_i)$ is the probability estimated by the model. First, the negative sum of each pair of

word-gold probability distributions is computed and used to measure how wrong the model is, then the objective is to minimize this value. This makes it possible to compute a distance measurement between the probabilities an annotator has assigned and the probabilities a model has estimated. For the interested reader, a more comprehensive description of the cross-entropy can be found in Goodfellow et al. (2016, Chapter 3.13). Alternatively, when performing a regression task, predicting a single number, the Mean Square Error (MSE) can be used, defined as follows:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.9}$$

The MSE loss calculates how far from the reference the model is using the squared Euclidean distance. If $\hat{y}_i$ is larger than y_i, subtracting one from the other will result in a negative term. However, remember that our objective is to *minimize* the loss term. If we try to minimize this term (which can be negative), our goal will be to maximize the value of $\hat{y}_i$, which is not what we want.

2.4.3 Optimization

For optimization, the goal is to update the parameters of a neural network such that we minimize the loss function. This requires some method of estimating how the parameters θ should change given the input and loss.

Stochastic Gradient Descent We can use backpropagation to estimate the importance of different parameters in the final output. However, we also need an algorithm to update the parameters' values. One of the earliest methods for doing this is the Stochastic Gradient Descent (SGD) (Kiefer, Wolfowitz, 1952) method. SGD goes over a dataset of examples $(x^0, y^0), \ldots, (x^n, y^n)$ and perform an up-

date of the parameters based on the loss obtained from each example. The update performed for each example is the following, where θ is the parameters of the model, η is the learning rate, $\mathcal{L}$ is the loss function used, and $\nabla_\theta \mathcal{L}(x^n, y^n, \theta)$ the gradient obtained form the loss function over the example (x^n, y^n). The SGD algorithm is shown in Algorithm 1.

Algorithm 1: Stochastic Gradient Descent

for $1 \dots N \in D$ **do**
  $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(x^n, y^n, \theta)$

The learning rate η defines how large of a step we should take in each parameter update. That is, we do not want to take too small updates because it will take a long time to learn a task, but we also do not want to make too large updates. The goal of updating the parameters is for the loss function to converge towards local or global minima.

However, individual updates for each example (x, y) in the dataset are costly when the dataset contains many examples. Therefore, the loss is calculated over a mini-batch of examples to mitigate this. To perform updates on mini-batches the dataset D is divided into M mini-batches of examples. These examples can be sampled randomly from the dataset or according to some schedule, for example as suggested by Bengio et al. (2009). To perform an update, we take a mini-batch M' from the dataset and calculate the updated parameter θ according to Algorithm 2 (Goodfellow et al., 2016, Chapter 8).

Mini-batch SGD thus calculates a loss for each example in a mini-batch, then computes the mean of the losses. The parameters θ are then updated based on this mean loss.

Algorithm 2: Mini-batch Stochastic Gradient Descent

for $1 \dots M \in D$ **do**
  $\theta \leftarrow \theta - \eta \frac{1}{|M'|} \nabla_\theta \sum_{m \in M'} \mathcal{L}(x^m, y^m, \theta)$

Adam The Adam optimizer (Kingma, Ba, 2015) is a variant of the SGD algorithm that combines two advances in optimization: AdaGrad (Duchi et al., 2011) and RMSProp (Tieleman et al., 2012). The Adam update is defined as follows (Ruder, 2016):

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t + \epsilon}} m_t \tag{2.10}$$

The Adam optimizer runs per-parameter learning rates, so each neural network parameter has a specific learning rate. This learning rate is then scaled based on the previous update steps. The advantage of using per-parameter learning rates is that the learning rate for each parameter can be adapted to how good the output is given the importance of that parameter. There is no global learning rate that has to be applied to all the parameters. This becomes important when neural networks contain up to billions of parameters. Scaling the learning rate for each parameter based on previous updates have the advantage of being more flexible in local minima. The local minima can be regarded as a point in the loss landscape where the loss is lower than before. However, some updates will still result in a lower loss. For the model to get to an even lower loss, it has to take additional steps which move the loss out of the local minima. This is illustrated in Figure 2.2.

Two parameters are used to scale the learning rate based on previous updates: m_t and v_t, where t is the current step. To estimate the first moment, m_t uses the mean of previous gradients, and the second moment is estimated by v_t, the uncentered variance of the previous gradients.

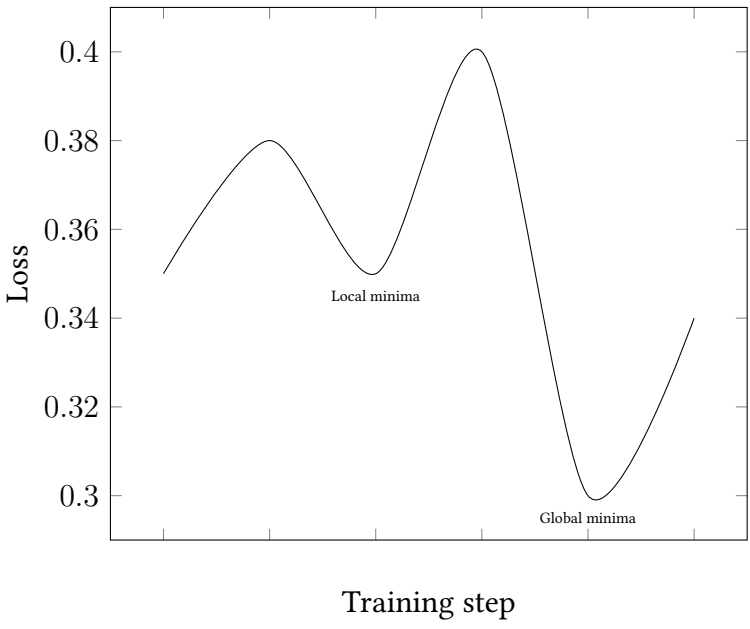


Figure 2.2: Local and global minima.

2.5 Attention

A strength of neural networks is their ability to select salient information from embeddings. So that given word or sentence embeddings, we want to identify and keep the important information related to the task and discard the otherwise available information. A powerful technique for this is using the attention mechanisms (Bahdanau et al., 2015). The intuition behind the attention mechanism is that we can use an embedding (k) to estimate the importance of some other embeddings ($S = (s_0, s_1, \ldots, s_n)$). For this, some function f is used, that produces an output a_n that indicates the importance of each pair (k, s_n). The process is illustrated in Figure 2.3.

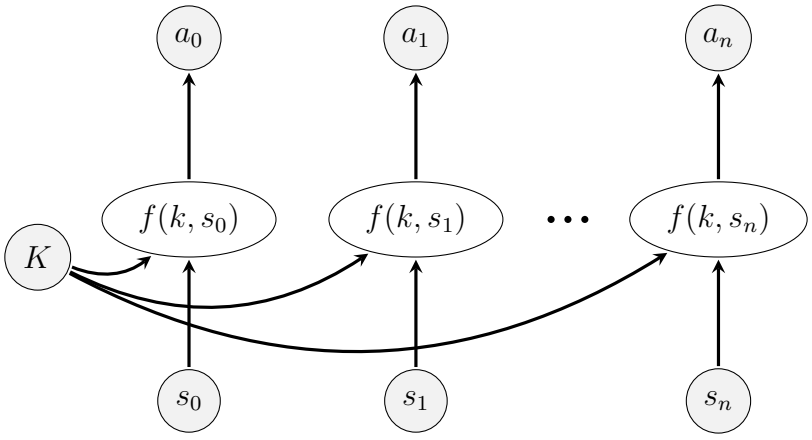


Figure 2.3: Generic structure for computing attention scores.

These attention scores $a_0, \ldots, a_n$ are not bounded and can potentially take on any value. However, we are interested in how the different inputs relate to the key k in log proportion to each other. To compute this, we can apply the softmax function over the attention sequence:

$$A = \mathsf{softmax}(a_0, \ldots, a_n) \tag{2.11}$$

Because the softmax function scale values in A, their magnitudes are now relative to each other in addition to being relative to k. If our goal is to predict something for the whole sequence, the weighted sum or mean can be used. Where we take the sum or mean of all the embeddings $s_0, \ldots, s_n$, and each embedding is multiplied by its corresponding attention value. However, if we do not want a representation for the whole sequence but a sequence of weighted embeddings, we multiply each s by its corresponding attention value.

When working with attention modules, we consider a key representation k and a sequence $s_0, \ldots, s_n$. For this, we want to compute a value for every element in the sentence, such that the salient elements given k are assigned higher values. We describe some commonly used attention modules below:

Dot-product In dot-product (also known as general attention) (Luong et al., 2015), the attention score between two representations k and s is calculated by considering the dot-product between the two representations.

$$f(k, s) = k^\top s \tag{2.12}$$

Cosine In cosine-based attention (Graves et al., 2014) the similarity between the k and s is estimated through their cosine similarity.

$$f(k, s) = \mathsf{cosine}(k, s) \tag{2.13}$$

Additive/Concatenative In additive attention (Bahdanau et al., 2015) (also known as concatenative attention) there are three learnable parameters, v, θ_a, and θ_b.

$$f(k, s) = v^\top \mathsf{tanh}(\theta_a k + \theta_b s) \tag{2.14}$$

2.6 Neural Networks for Sequences

Processing natural language often involves a whole sequence of words. Previously we have seen how to transform and learn things about a single word embedding, but processing language typically involves considering a sequence of words whose meanings depend on each other.

When processing a sequence of words, two main neural architectures are used, namely Recurrent Neural Networks (RNN, (Elman, 1990)) and Transformers (Vaswani et al., 2017). In this section, we describe the inner workings of RNNs and the transformer model in the next section.

2.6.1 Recurrent architectures

The recurrent neural network architecture is composed of an RNN cell R, which takes a word embedding as input, selects some information from the input to keep in memory, and generates an output h. A generic schema of an RNN is shown in Figure 2.4, where R is the RNN cell.

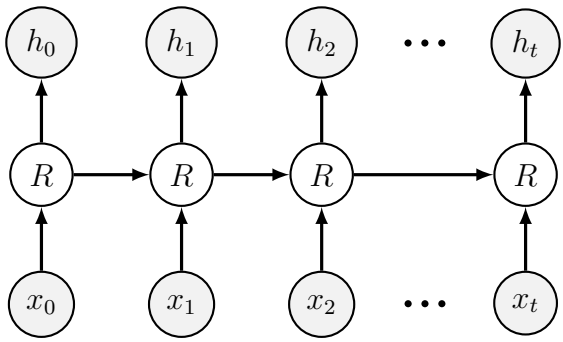


Figure 2.4: Schematic design of the RNN architecture.

The RNN works sequentially: it takes one input and processes

it, then takes the following input and processes it. But crucially, the input x_n is contextualized by the previous inputs $x_0, \ldots, x_{n-1}$ in the RNN. As such, when the hidden state is computed for x_n, this computation considers the previous inputs seen. The simplest RNNs (Elman, 1990) contain three parameters, W, U, and b, and for an input x_n the hidden state is calculated as follows, where σ is the sigmoid activation function and t the current time step:

$$h_t = \sigma(Wx_t + Uh_{t-1} + b) \tag{2.15}$$

However, a problem simple RNNs have is *long-range dependencies* (Bengio et al., 1993). As the name suggests, the problem concerns how models can keep information from the input at time-step t in memory such that it is available at a later timestep. This can be difficult because there is a tendency to overwrite the information in memory as more inputs are passed to the model. A related issue that the simple RNN had to deal with was that of *vanishing* (the gradients become very close to 0 and only small updates are performed) and *exploding* gradients (gradients become very big and large updates are performed) (Pascanu et al., 2013). To tackle these issues two architectures in particular have been suggested, GRU (Gated-Recurrent Unit) (Chung et al., 2014) and LSTM (Long-Short Term Memory) (Hochreiter, Schmidhuber, 1997). Both of these architectures are based on the principle of a gating mechanism. The gating mechanism alleviates the issue of vanishing gradient by allowing the model to select information to retain, thus allowing the model to ignore some of the input signals and not consider them when computing the gradients. The problem of exploding gradients can be tackled by imposing a constraint on how large gradients can be (*gradient clipping*) or by using L1 or L2 weight regularization (Salimans, Kingma, 2016).

In particular, we will outline the LSTM model here. To keep track of the progress in the LSTM, there is the hidden state h_t and the cell state c_t. In addition to these, there are four components

responsible, where each of the components has three parameters W, U, and b associated with it. To compute a new hidden state for the input, first, there is the forget gate f which selects the information to forget. Then there is the input gate i, which selects information to keep. Next, we want to update the cell $\tilde{c}$ for the current time step to keep track of the new information. Finally, we have an output gate o, which selects information to include in the output. To update our cell state, we multiply what should be forgotten with the previous cell state $f_t \odot c_{t-1}$, and to this, we add the information we want to keep by multiplying the current cell state $i_t \odot \tilde{c}$. Finally, the output of the LSTM is computed by multiplying the output state o_t with the cell state $\mathsf{tanh}(c_t)$. In all, the operations performed by the LSTM cell are summarized in Equation (2.16).

$$
\begin{aligned}
f_t &= \mathsf{tanh}(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \mathsf{tanh}(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \mathsf{tanh}(W_o x_t + U_o h_{t-1} + b_o) \\
\tilde{c}_t &= \mathsf{tanh}(W_c x_t + U_c h_{t-1} + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
h_t &= o_t \odot \mathsf{tanh}(c_t)
\end{aligned}
\tag{2.16}
$$

The LSTM led to a surge of developments in a variety of NLP topics, such as learning formal languages (Gers, Schmidhuber, 2001; Bernardy, 2018), Language modeling (Sundermeyer et al., 2015; Verwimp et al., 2017), Natural Language Inference (Bowman et al., 2015; Chen et al., 2017), and others (Goldberg, 2017, Chapter 16).

The RNN in Figure 2.4 shows only that we read from left to right, that is, a unidirectional RNN. However, we can also combine left-to-right with a right-to-left reading by inverting the order of the input elements and computing this new sequence with the RNN. Then, we have all the word representations from the left-to-right direction $\overrightarrow{h}_0, \ldots, \overrightarrow{h}_t$ and from the right-to-left direction $\overleftarrow{h}_0, \ldots, \overleftarrow{h}_t$.

To combine them, we can concatenate the forward and backward pass of each word representation:

$$\begin{bmatrix} \overrightarrow{h}_0, \overleftarrow{h}_0 \\ \vdots \\ \overrightarrow{h}_t, \overleftarrow{h}_t \end{bmatrix}$$

Note that the output is twice as large - we are concatenating the left-to-right and right-to-left word representations. This allows a model which reads a sentence in a linear order to consider both the left and right context.

2.6.2 Transformer architectures

In recent years the transformer model (Vaswani et al., 2017) has emerged as the dominant model in natural language processing (Wolf et al., 2020). The transformer is based on the self-attention architecture presented in the next section. The transformer is often trained with the MLM objective presented in Section 2.1. The original transformer model used as a language model (Devlin et al., 2019) was also trained with an additional objective, next sentence prediction. This objective is to predict whether another sentence 7b follows the masked sentence 7a:

(7) a. The bowl is [MASK] and of unknown origin.
b. It contained spices.

The benefit of the following sentence prediction objective is that it allows a model to account for additional context beyond the sentence level, making the representations consider what surrounding context is relevant for the sentence. More recent models such as (Liu et al., 2019b) exclude this objective.

The transformer uses some form of sub-words, for example, Byte-Pair Encoding (Sennrich et al., 2016), WordPiece (Wu et al.,

2016), or SentencePiece (Kudo, Richardson, 2018), to split words into sub-words and map them to vectors. Additionally, attention generally does not encode positions, i.e., where in a sentence a word occurs. To obtain positional information, the sub-word vectors are concatenated with so-called positional encodings that help the model determine what token occurs before or after another (Ke et al., 2021).

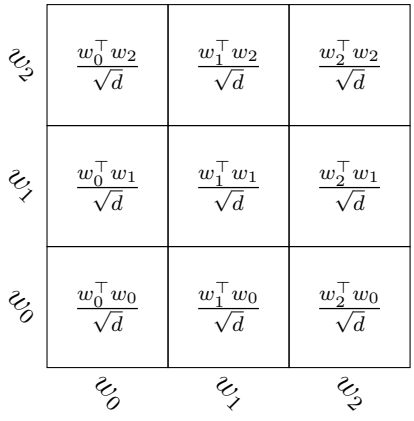


Figure 2.5: The self-attention computation $\frac{QK^\top}{\sqrt{d}}$ given the three tokens w_0, w_1 and w_2.

Self-attention Heads Self-attention is the basis of the transformer model, defined as:

$$\mathsf{SelfAttention}(Q, K, V) = \mathsf{softmax}(\frac{QK^\top}{\sqrt{d}})V \tag{2.17}$$

Where Q (query), K (key), and V (value) are parameterized transformations of the input. With these three representations, we first compute the dot-product of all possible combinations and divide by the square root of their dimension ($\sqrt{d}$). The $QK^\top$ part of the self-attention formula can be visualized as Figure 2.5. Combined with

softmax along the rows in Figure 2.5, i.e. the cases when w_n is the key, gives us a matrix of attention scores between word combinations. We then use these scores to scale the V representation, such that V_0 is scaled by the scores obtained from combining the input s_0 with the other words in the sentence. Thus, we take a scaled version of the dot product and use the attention scores to compute a weighted sum of V.

Transformer Structure Self-attention modules are organized in a stacked fashion using different layers. Each layer in the transformer is composed of m self-attention heads, such that each layer learns m different self-attended variations of the input, which are then combined and passed to the next layer. Before passing the output to the next layer, we compute layer normalization (Ba et al., 2016) on the concatenated representations from each attention head. A schematic representation of the transformer architecture is shown in Figure 2.6.

At each layer, the transformer learns different representations of the input. Research has shown that at different layers and attention heads, different linguistic patterns appear to be processed (Kovaleva et al., 2019; Geva et al., 2021). For example, lower layers in the model typically learn information about morphology Section 3.3 and syntax Section 3.4. In contrast, the latter layers of the model learn semantic and pragmatic aspects (Clark et al., 2019; Tenney et al., 2019). In general, we can note that certain parameters appear to be better, or worse, at encoding different parts of an input. Thus, isolating which parameter is responsible for what in the output become difficult.

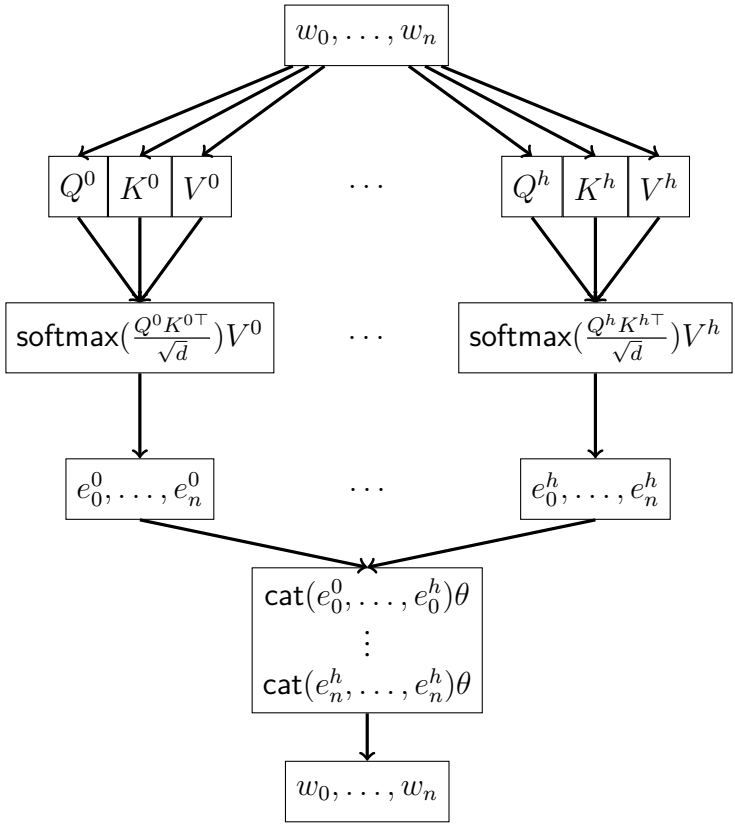


Figure 2.6: One layer in the transformer with h self-attention heads. Here θ is a linear transformation that performs the dimension reduction $(|n|, h \cdot d) \rightarrow (|n|, d)$ that produces the representations $w_0, \ldots, w_n$. The final layer normalization step has been omitted.

One popular technique for this is freezing layers of the transformer (Michel et al., 2019; Lee et al., 2019) to asses how the model perform while only training parts of the model on a downstream task. Besides probing a model by manipulating its parameters, Alajrami, Aletras (2022) explores how different pre-training objectives influence a model's capacity to learn linguistic properties. They per-

form their experiments by considering masking strategies which can be categorized as linguistically motivated or not. Their findings suggest that the largest factor in how well models learn linguistic properties lies in the data used to train and the architecture of the model, rather than the pre-training objective.

Multilingual Transformer Models As the transformer models gain-ed popularity and showed an impressive performance for a variety of tasks, a line of research began exploring the possibility of constructing a multi-lingual transformer model (Xue et al., 2021; Conneau et al., 2020; Liu et al., 2020b). That is, a transformer-based language model that can process many different languages simultaneously. Generally, these models are trained in the same manner as monolingual models but with data from many languages.

This allows us, for example, to fine-tune a multilingual model on an English dataset and evaluate it on a Greek test set. Briefly, because all languages share a single vector space when this space is adapted to English data when fine-tuning, an effect of this is that the word representations in another language are also modified. As a result, these models generally show strong performance and an impressive ability to model many languages. A question that naturally arises with multilingual transformer models is how does it work, and in what scenarios can these models transfer information successfully between languages; this has been investigated by (Pires et al., 2019) among others. Additionally, multilingual models have been shown to encode some language-specific properties in their representations (Rama et al., 2020).

2.7 Encoders and Decoders

As of yet, we have seen models that encode words in a sentence as representations, which are then used to predict a label.

The neural architectures we have seen are able to generate a fixed set of predictions, and the number of outputs the models should produce is known. But in the cases where this information is not known a different type of architecture has to be employed. For this, encoder-decoder (also known as sequence-to-sequence) models (Sutskever et al., 2014) can be used. Encoder-decoder models are particularly useful for tasks that require models to produce new words, such as machine translation, where a sentence in one language is to be translated into another language, or image captioning, where the model is to generate a description for an image. In these cases, there is no strict correspondence between the words which should be generated and the input size. In Example (8) when translating from English (in this case, the source sequence) to Swedish (the target sequence), we can note that the determiner (*the*) is encoded in the word *katten* in Swedish. This means that there is not a 1-1 correspondence between the words in the source and target sentence.

(8) Katten springer
cat.DET run.PRS
The cat runs

To translate, a model needs to generate n representations, where n is the number of words in the target sentence. To do this sentences are encoded with the start- and end-of-sequence special tokens surrounding the sentence. The purpose of these special tokens is to inform the model where the sentence starts and ends. The end token also functions as a command to the model to stop generating representations.

The decoder is another model with independent parameters, that produce a sequence of new representations. Put simply, the final state of the encoded sequence is passed through the decoders' parameters that produce a new representation. This new representation is then decoded into a token and passed as the next input to the decoder. The decoder will continue to produce new representations

until one is decoded as the end-of-sequence token. Figure 2.7 shows how the encoder-decoder model generates a translation using an RNN model.

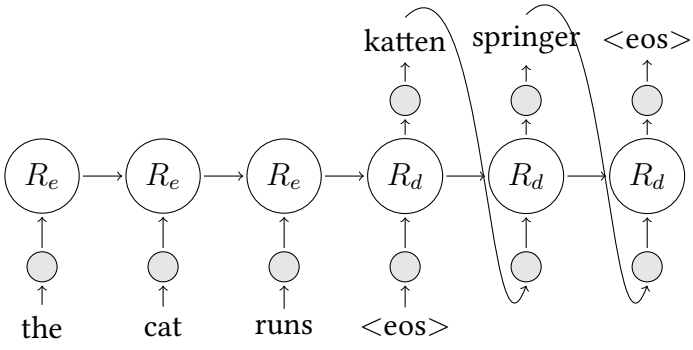


Figure 2.7: A RNN based Encoder (R_e) and Decoder (R_d) architecture for translating *The cat runs* to Swedish.

The objective of an decoder is to produce the most likely target sequence $y_1, \ldots, y'_t$ given the source sequence $x_1, \ldots, x_t$ as shown in Equation (2.18).

$$p(y_1, \ldots, y_T | x_1, \ldots, x_T) = \prod_{t=1}^{T} p(y_t | x_1, \ldots x_t, y_1, \ldots, y_{t-1}) \quad (2.18)$$

To produce the most probable sequence, a decoding algorithm is employed on the generated representations. For this many different algorithms can be used but commonly greedy- or beam-search (Dept., 1977) is used. Greedy search selects the most likely token at each time-step. To make the decoding more varied and take the previously generated tokens into account beam search is commonly

used. Beam search operates by finding several possible decoding sequences and then selecting the most probable one.

These types of models produce a sequence of new representations, rather than predicting a label for the input representations. However, the models can be effectively used for tasks that traditionally require the model to predict labels. Examples of this include T5 (Raffel et al., 2020) and its multilingual version mT5 (Xue et al., 2021). The advantage of this approach is that a model may be trained on many tasks simultaneously, using the same learning objective, training procedure, and decoding strategy (Raffel et al., 2020). This allows a model to transfer information learned in one task to another.

> "We are not interested in the fact that the brain has the consistency of cold porridge."
>
> Alan Turing

Chapter 3

Linguistic Structures

In language, the lexical meaning of a word is the word meaning in the absence of contextual information. Typically, what this meaning is can be found in a dictionary. But the dictionary meaning (the *lemma*) can also be modified by contextual and non-linguistic information. For example, consider the word WARM, it may both refer to temperature (*It is warm outside*) and to clothing that makes you feel warm (*This fleece is warm*) (Lee, 2021). However, structures in language can further specify the semantic or syntactic properties of the word and relate it to other words in a sentence. The meaning expressed by a word in a sentence is an interplay between the semantics of the word, word, and sentence structures. This interplay between meaning and structure allows language to express a wide variety of concepts and situations. For example, while the noun *cat* denotes the concept of a CAT, this tells us nothing about the real-world situation that elicited the occurrence of *cat* in the language. Without any specification describing the concept's details, we can only make a semantic interpretation based on the lexical meaning.

Languages contain many different structures that help shape the meaning expressed; this section is dedicated to exploring them. Thro-ughout the presented papers only a limited number of these

structures are explored. Following the themes explored in the published papers, this section is dedicated to grammatical structures such as morphology and syntax, and how they relate to meaning.

3.1 Structure and Meaning

A core property of language is that words carry lexical meaning, the concept or function a word signifies. However, language is more than a collection of lexical meanings; it is lexical meanings augmente-d to express more fine-grained properties and aspects, organized in a particular way in the context of other words to relate their lexical meanings. However, to talk about how meaning and structure relate, we need to consider compositionality and the syntax-semantics homomorphism. This homomorphism postulates that there is a mapping of the syntactic structure to the semantic meaning (Partee, 2014). In particular, this notion was formalized by Montague (Montague, 1970), stating that we can analyze the structure of words in a sentence to derive its semantic meaning. That is, the meaning of a sentence can be viewed as a function that composes the elements of its syntactic structure.

In linguistics, when talking about the structure of a language, the morphological and syntactic patterns are often referred to. On an approximate level, we can draw a line between morphology and syntax by saying that morphology deals with structures within a single word and that syntax deals with how a collection of words are structured. But looking closer at the separation between morphology and syntax, this line becomes less clear. For example, how words express fine-grained meaning can influence the word form and how it should be placed in a sentence, for example, by *agreement*. An example of agreement is shown later in Figure 3.2. This is an example of when morphology influences how syntactic structures are realized. Instead of considering morphology and

syntax as two separate areas of study, we can consider them jointly as morpho-syntax. We further discuss the interactions between morphology and syntax in Section 3.3.

The question of how these different morphological, syntactic, and morpho-syntactic structures give rise to complex meaning is still an open question. One way of exploring this is through entailment problems. Entailment problems can generally be formulated as follows: one sentence (the premise, denoted as **P**), does this give sufficient evidence to claim that another sentence (the hypothesis, denoted as **H**) is True. For example:

(9) **P** John sees Mary playing
H Mary is playing

Here we can see that *John* is doing some action (*sees*), and the thing John is seeing is *Mary playing*. The premise, in this case, gives sufficient evidence that *Mary is playing* because that is what John is seeing [1]. We can say that the premise is sufficient to justify the hypothesis because of word order. If the order of *John* and *Mary* were swapped, it would not justify the premise, as the SUBJECT of an action is given by its position relative to the verb, i.e., to the left of the verb.

Languages such as English mainly rely on syntactic structures to evoke meaning representations, and its morphological system is relatively weak. This is in sharp contrast to agglutinative languages such as Turkish or Finnish, which rely mainly on the morphological structure to evoke meaning representations. However, in any language, words carry meaning, and the word-internal structure further modifies that meaning. For example, the word-internal structure can help us disambiguate when something happens as shown in Example (10).

[1] We assume here that John is not hallucinating.

(10) **P** The cat chased after the mouse fast
H The cat is running

In this example, we can note that in the verb *chased*, the inflection indicates that it happened in the past (past tense, PST). While in the hypothesis, we have a sentence about something that is going on at this moment (progressive tense, PROG), which forces us to evaluate the entailment problem as False given that premise says that something happened before, while the hypothesis says that something is happening now.

While syntactic ambiguity in English is common, morphological ambiguity can be difficult to find, as the morphological system is minimal at best. However, one type of ambiguity surrounds adjectives of the form *un-X-able* (Vikner, Vikner, 2008):

(11) **P** The chest is unlockable
H Someone can open the chest

Where the adjective ”unlockable” can be interpreted in two ways, depending on how the morphemes are attached to each other:

- that which cannot be locked
- that which can be unlocked

We have to use the second interpretation to evaluate the entailment problem as True. However, this interpretation depends on which part of the word the prefix *un-* is attached to. Thus, how morphemes are attached changes the semantic meaning of the word. The difference between the two interpretations is shown in Figure 3.1.

We can also consider cases where syntactic structures, such as word order, gives rise to ambiguous meaning of a sentence. We can consider an entailment problem that is determined by how the English syntactic structure is interpreted, such as Example (12).

(a) The X cannot be locked (b) The X can be unlocked

Figure 3.1: Interpretation of *un-X-able* adjectives.

(12) **P** John saw the girl with a smile
H John had a smile

To determine whether this entailment problem should be evaluated as True or False, the syntactic structure needs to be considered to parse this problem as True or False. To make the entailment True, the parse of the first sentence needs to attach *with a smile* to *John*, as the hypothesis says that *John* is the one who was smiling. If, on the other hand, "with a smile" is attached to the girl, then *John* is not the one who is smiling, and we have to evaluate the problem as False. So, to make a meaningful judgment of Example (12), a human reader of these two sentences needs to make a decision whether *with a smile* is modifying *John* or *the girl* which adds another layer of complexity.

We can also consider cases where different syntactic structures, gives rise to the same meaning representations. Consider the two sentences in Example (13).

(13) a. The man painted the house
b. The house was painted by the man

The meaning of the two sentences is the same, but the word order is different. So we cannot merely make the judgment that because

two sentences have different word orders, they also have different meanings. Crucially, the semantic interpretation that the syntactic structure evokes needs to be taken into account.

3.2 Grammatical Structures

Besides lexical meaning, words in sentences have indicators that help identify the category of words (*part-of-speech*), specify the meaning of the inflection of a word (*grammatical features*), and how words modify each other (*grammatical relations*).

Part of Speech Part of Speech (POS), also known as lexical categories, broadly categorize the role of a word in a sentence. But in addition to describing how a word can be used in a sentence, it also informs us what grammatical features can be added to the word. POS tags can be divided into two categories, open and closed sets. This means that closed POS tags, such as PRONOUNS (e.g. *he*, *she*) and CONJUNCTIONS (e.g. *and*, *or*), have a fixed vocabulary size. These classes do not generally allow for new words to be added. This, of course, is contentious as both Swedish and English (Hord, 2016) have recently been enhanced with various new pronouns that mitigate gender connotations. On the other hand, open word classes allow for new words (also called *neologisms*) to be added. The POS tags that are typically considered to be open are VERBS (actions), NOUNS (things), ADJECTIVES (descriptions of things), and ADVERBS (descriptions of actions). Thus when encountering a new unknown word, we can generally say that it belongs to one of the open word classes.

The task of assigning POS tags has been one of the first tasks to be explored in NLP; one reason for this is the usefulness of shallow syntactic analyses, which can either be used in other systems (Marcheggiani et al., 2017; Ren et al., 2017; Cheng et al., 2020) or used

as a preprocessing and analysis tool for text (Östling, Wirén, 2013; Saphra, Lopez, 2018; Oktavianti, Ardianti, 2019). Systems have become good at predicting POS tags, reaching up to 97% accuracy on English data (Manning, 2011). Common approaches to POS tagging using neural networks are to pass a sentence through a bidirectional LSTM network and then run the outputs through a linear transformation to obtain POS labels (Wang et al., 2015; Yasunaga et al., 2018).

In recent times, the transformer model has become more popular, especially for extending POS tagging capabilities beyond English (Kondratyuk, Straka, 2019). Other languages with many available resources are also performing well, but for low-resource languages, the accuracy of POS taggers still lags behind larger languages such as English. For low-resource languages, several techniques have been developed to overcome this. For example, by infusing lexical knowledge into models (Plank, Klerke, 2019), or by including character-level information in the training objective (Kann et al., 2018). These are examples where additional information can be obtained from the same language. Another method is to use untagged data to project labels from one language to another (Fang, Cohn, 2016) or incorporate data from other languages using transfer learning techniques (Vries de et al., 2022).

Grammatical Features In the previous section, we introduced how words can be categorized, both in terms of how a word can be used in a sentence and simultaneously distinguishing between what things a word signifies. The notion of POS encodes how words work in language, that is, how they relate to sentence structure. To facilitate communication about the particular details of a word, speakers can attach properties to words, namely *grammatical features*. For example, verbs refer to actions, and when communicating that an action was undertaken, an essential piece of information is when it took place, that is, the TENSE. Additionally, a verb may take several other grammatical features, such as MOOD (how the verb is ex-

pressed, for example, whether it is a command, IMPERATIVE, or that the verb is possible, POTENTIAL) and EVIDENTIALITY (what evidence exist from the speaker point of view of the event) among others. For example, the Finnish verb ”palai” in (14) has a number of grammatical features associated with it.

(14) Tietokone palai
computer.NOM.SING burn.IND.SING.3PRS.PST.FIN.ACT
täysin
down
en: The computer burned completely down

This expresses the concept of BURN, and also indicate that the burning happened previously (PST), that it is a completed action (FIN), it is in the third person perspective (3PRS), that it happened (IND), that the subject of the sentence is the agent of the verb (ACT) and agreement with the NUMBER of the subject (SING).

The features a particular word can take varies between languages. In many languages, the tense of a verb is expressed with inflection (that is, by adding a morpheme representing when an action took place). However, for isolating languages that do not employ morphology to a large extent, for example, Mandarin Chinese, other devices than inflection are used. In some cases, the context can help disambiguate the tense of a verb. In other cases, lexical items denoting time (such as *yesterday* and *now*) are used.

Grammatical features can be of a syntactic (morpho-syntactic) or a semantic (morpho-semantic) nature (Kibort, Corbett, 2010, Chapter 4). Syntactic features are those which are influenced by syntax, for example, NUMBER in English and Finnish are involved in agreement. When assigning NUMBER to a word the value of this feature then influences the NUMBER feature in other words in the sentence. An example in English of this is shown in Figure 3.2, where the NUMBER of the subject and verb have to be the same. Another example

of this we saw previously in Example (14).

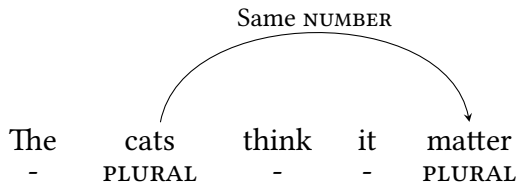


Figure 3.2: Subject-verb number agreement.

In this case, because the subject has the NUMBER feature of plural, the verb also has to have it. If the subject would be singular instead *cat*, then the verb must take the singular form *matters*. However, some features are not influenced by syntax, for example, TENSE. When TENSE is assigned to a word to indicate when it took place, there are no syntactic constraints that prevent a certain TENSE assignment.

In computational linguistics, discovering the grammatical features of a word have many applications. For example, in machine translation, it can help with the translation of low-resource and morphologically rich language with many inflections (Ataman et al., 2020), or in Named Entity Recognition (Güngör et al., 2019). Additionally, grammatical features are essential in typological studies, where they often help distinguish languages from each other (Ponti et al., 2019). Approaches to tagging grammatical features are similar to those of POS tagging. In both tasks, the goal is to assign a tag (*part-of-speech*) or a set of tags (*grammatical features*). In grammatical feature tagging[2] systems generally rely on character embeddings (Matteson et al., 2018) or sub-word embeddings (Kondratyuk, Straka, 2019) to identify inflectional features. Then the embeddings are passed through a model to obtain hidden states from which the grammatical features can be predicted.

[2]Sometimes referred to as morphological tagging or morphological analysis.

Grammatical Relations Grammatical relations concern how words are related to each other. This is in contrast to grammatical features, which primarily augment the meaning of a single word. For example, verbs describe an action, and actions typically have someone doing something. Thus, a relation exists between the doer (the subject) and the action (the verb). Additionally, the action performed may be performed to something (the direct object) and perhaps with something (the indirect object). For example, in English, the subject, direct object, and indirect object are given by the word order, so we can annotate a sentence as follows:

(15) Adam gives Bill a megaphone
SUBJECT Dir-OBJECT Ind-OBJECT

However, in other languages this information is given by case markings on words, for example in Russian as we saw in the introduction:

(16) Кошка преследует собаку
SUBJECT Dir-OBJECT
'the cat chases a dog'

Depending on the language, there are different ways of identifying subjects, direct and indirect objects. There are many grammatical relations that all describe how words are related to each other in a sentence. This is further discussed in Section 3.4.

3.3 Morphology

As we have seen, the notion of a word is integral to the study of structures in language. Many strategies we have looked at use words to build structures that give rise to semantic meaning. Besides being used to build structure, words themselves have internal structures.

The study of their structure is commonly called morphology. However, the notion of words and their structure is problematic. We saw a consequence of this when describing how neural networks tokenize text. Trying to define word boundaries (even in English which has very limited morphology) using whitespace leads to non-intuitive "words" which in turn yields non-intuitive representations in computers. To some extent, the separation between morphology and syntax across the languages in the world has no basis (Haspelmath, 2017). Instead, a common domain for the study of words and sentences has been proposed, morpho-syntax. While there may not be a clear boundary between morphology and syntax on a larger scale, we can speak about words given a specific language. In our exploration of morphology, we consider words not as a universal linguistic unit but rather as a language-specific phenomenon.

Words as a language-specific phenomenon can be viewed in two different ways, as a *lexeme* (the abstract meaning of a word) and as a *lemma* (the dictionary form of the word). The lemma of a lexeme expresses the main source of semantic meaning, which can then be further modified by other morphological processes to generate the different *word forms*, which can express more fine-grained information as we saw in Section 3.2. The structure of a word can be broken down into two units, the free morpheme, and the bound morphemes. A free morpheme is a unit that can occur independently of other morphemes, while bound morphemes can not.

As an example of how free and bound morphemes work together to create the word forms of a lexeme, we can consider a partial morphological paradigm (Blevins, 2001) of the verb *walk*, shown in Table 3.1. In Table 3.1, *walk* is a free morpheme, while *-s* is a bound morpheme expressing that only one person is involved in the activity of walking. To express the PLURAL feature instead, that several people are involved in the activity, the null morpheme (∅) is used.

Inflectional Morphology The meaning of a lexeme can be modified to express more fine-grained information. For example, by adding the bound morpheme, *-ing* to the free morpheme, *write*. This process augments the meaning of *write* with the PROGRESSIVE grammatical feature. We illustrate this example in Figure 3.3, where the root is a free morpheme, and the affix is a bound morpheme.

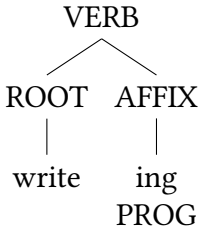


Figure 3.3: The free morpheme *write* is inflected using to bound morpheme *-ing* to express the PROGRESSIVE grammatical feature.

Building a word form is called inflection, or inflectional morphology. The purpose of inflectional morphology is to augment a word so that in addition to expressing the lexical meaning, it also expresses some grammatical meaning, in this case, that the activity of writing is ongoing (PROGRESSIVE, abbreviated as PROG).

Morphological inflection is the task of producing the resulting word form (the *target*) given a word (the *source*) and a set of grammatical features. For example, the Finnish lemma *palaa* which is a verb, should be transformed with the grammatical features

Free morpheme	Bound morpheme	Feature
walk	-s	SINGULAR
walk	∅	PLURAL

Table 3.1: Possible word forms of the lexeme "walk" with respect to the grammatical feature of TENSE.

IND.SING.3PRS .PST.FIN.ACT to produce its inflected form *palai*. Approaches to this task generally use a sequence-to-sequence model that generates a new word, character by character, given the lemma of a word and a set of grammatical features as input (Faruqui et al., 2016; Aharoni, Goldberg, 2017; Anastasopoulos, Neubig, 2019). The model can either proceed in a standard fashion and output characters, or it can produce edit-operations (Makarov et al., 2017) that predict actions to take given the lemma and grammatical features. This approach has the advantage of modeling the copy mechanism (Figure 3.4) explicitly in its training objective, which is a core problem facing all systems which attempt to generate morphological inflection.

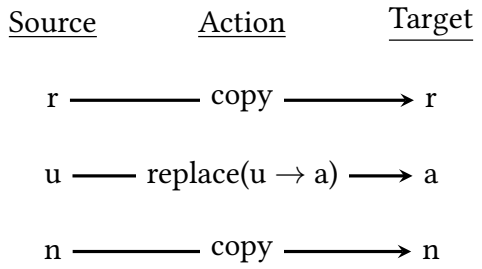


Figure 3.4: Inflecting the word *run* to the past tense (PST) using the copy mechanism.

Systems for morphological inflection generally perform well, achie-ving high accuracy for many of the languages it has been tested on. However, the number of languages these systems have been tested on is only a fraction of the languages in the world. Thus, truly how applicable these systems are to unseen languages remains to be shown. As with POS tagging, the performance on low-resource languages is lower than for high-resource ones. Concerning learning morphological inflection for low-resource languages, several techniques have been developed that mainly focus on artificially increasing the amount of training data, for instance by gener-

ating data hallucinations (Anastasopoulos, Neubig, 2019) or generating random new examples based on corpus statistics (Bergmanis et al., 2017).

3.4 Dependency Grammar

Dependency grammar is a way of formalizing syntax in natural language (Tesnière, 1959). The theory posits that the central notion in syntax is the relationship between two lexical items. This is opposed to constituency grammar which considers groups of lexical items to be the central notion. To represent relations between words, directed trees are used. The relationship between two lexical items is shown by an arc connecting them. The item from which the arc originates is the head of the relationship, and the target of the arc is the dependent. In plain language, the dependent modifies the head in some way. How one item modifies another item is indicated by attaching a label to the arc connecting them. For example, in the dependency fragment in Figure 3.5, the word "cats" is modified by "green", which is indicated by the syntactic relation of nominal modifier NMOD.

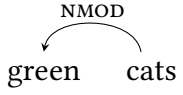


Figure 3.5: Dependency fragment of "green cats".

Heads, Dependents and Grammatical relations Grammatical relations are formalized as head-dependent relations in a dependency grammar. This means that one lexical item (the dependent) modifies another lexical item (the head). For example, this is shown in Figure 3.6.

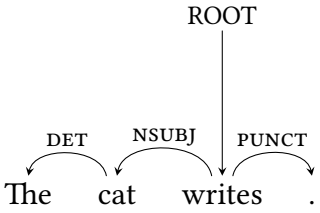


Figure 3.6: Dependency tree of the sentence *The cat writes.*.

The word *the* is modifying *cat* while *cat* and . are modifying *writes*. From this, we see how the lexical items modify each other. Dependency structures are fundamentally about connecting two words. But in addition to realizing that two words are connected, we also want to know how they depend on each other. In other words, what is the relationship between, for example, *cat* and *writes* in Figure 3.6. In the tree above, *cat* is the SUBJECT of the verb *writes*.

A special and important relation is the ROOT relation. The ROOT identifies the sentences' main, if any, verb and acts as the top node in the dependency tree. Verbs typically have subjects, objects, and indirect objects attached to them. The subject is coded as NSUBJ, the object as OBJ, and the indirect object and other auxiliary arguments as OBL. In English, determiners indicate the grammatical feature of definiteness of a noun and are encoded as the DET relation. To facilitate research into dependency structures the Universal Dependencies (UD) (Nivre et al., 2016) dataset is commonly used.

Enhanced Dependency Grammar For Natural Language Understanding tasks, the basic structure and relations proposed in Nivre et al. (2016) (Universal Dependencies) can be underwhelming, as the basic structure mainly focuses on strict syntactic relations. But for the understanding of language, we are more concerned with discovering relations between content words which are often omitted. (Schuster, Manning, 2016) propose an extension to the schema of

(Nivre et al., 2016) by including more semantically relevant relations. In particular, these extensions focus on adding NSUBJ and OBJ relations to sentences with conjunctions. An example of this can be found in Figure 3.7.

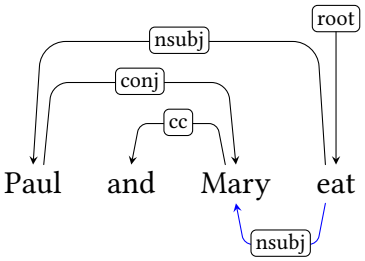


Figure 3.7: Enhanced reperentations of the sentence *Paul and Mary eat*. The additional arc introduced by the enhanced universal dependencies schema is shown in blue.

In this case, the conjunction indicates that both *Paul* and *Mary* are eating, so they are both the SUBJECT of the verb *eat*. The enhanced universal dependencies schema makes this explicit. Similarly, in cases where two verbs share the same SUBJECT and/or OBJECT, this is not shown in the universal dependency schema but is made explicit in the enhanced universal dependencies, as shown in Figure 3.8.

More types of arcs are introduced, that follow the theme of making the semantically relevant relations more prominent. The interested reader is referred to (Schuster, Manning, 2016) for further information.

3.4.1 Parsing Dependency Structures

There are two general approaches to dependency parsing: graph and transition parsing. Both types of parsers provide a syntactic analysis of a sentence, however, they use different methods to do so.

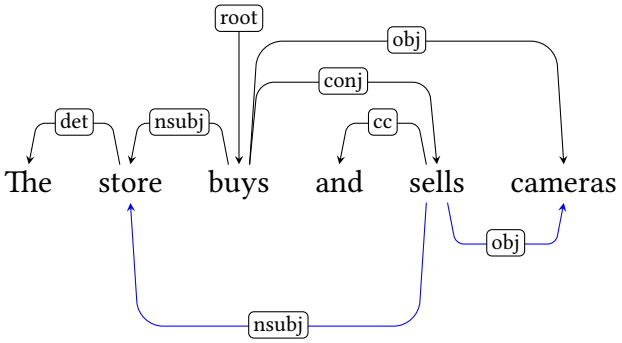
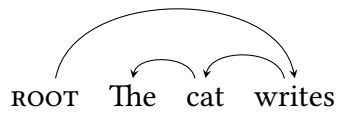


Figure 3.8: Enhanced reperentations of the sentence *The store buys and sells cameras*. The additional arcs introduced by the enhanced universal dependencies schema are shown in blue.

Graph parsing In neural graph-based dependency parsing, a neural network outputs a graph matrix of size $(n + 1, n + 1)$, where n is the sentence length, the additional row, and the column is the entry of the ROOT token, and each cell indicates the predicted score that the word is its head. An example of this is shown in Figure 3.9. However, because the output of the model is a matrix, there is nothing that ensures that the resulting parse tree is connected, which is a requirement of a dependency tree, i.e. that each node is reachable from the ROOT node. To address this issue, the Chu-Liu-Edmonds algorithm (Chu, 1965; Edmonds, others, 1967) is used when decoding the matrix into a tree. The loss is then calculated between the predicted tree and the annotated tree from the dataset. Arguably, using neural network for graph dependency parsing have been spearheaded by two parsers in particular, that of (Kiperwasser, Goldberg, 2016) which introduced a method for parsing text using a bidirectional LSTM network. This work was further built upon in (Dozat, Manning, 2017) where deep affine attention was introduced which improved the performance further. Since, the ideas introduced by (Dozat, Manning, 2017) were used by (Kondratyuk, Straka, 2019)

	ROOT	the	cat	writes
ROOT	0.8	1.3	2.5	4.3
the	1.8	1.3	4.5	2.1
cat	3.5	2.3	3.5	7.1
writes	5.5	1.9	3.2	0.3

(a) Matrix representation of a dependency tree.



(b) Dependency tree of Figure 3.9a.

Figure 3.9: Dependency tree for the sentence *the cat writes* predicted by a (fictional) graph-based dependency parser.

where the bidirectional LSTM network was replaced by the popular Transformer architecture, in particular the BERT variant.

Transition parsing In transition parsing [3], instead of outputting a tree represented as a matrix the system outputs a series of actions which corresponds to building the dependency tree from the input words. A dependency tree can be constructed by using a *stack* that reads and adds the input words, which are contained in the *buffer*, to a list. Words are then removed from the stack using two different actions, the `left-arc` action, which connects the stack's top-most word to the stack's second top-most word and pops the second top-most word. Then there is the `right-arc` action, which does the

[3] There are many variants of transition parsers, here we give a summary of the arc-standard transition parser (Nivre et al., 2006).

same operation in reverse: the second top-most word is connected to the top-most word, and the top-most word is popped from the stack. Additionally, there is one more action, `shift`. The shift action reads another symbol from the input sentence and adds it to the stack. The loss is then computed between the predicted actions and actions from a gold standard. An example of how these actions are used to parse a sentence is given in Table 3.2.

Step	Action	Stack	Buffer	Result
1	`shift`	ROOT	the cat writes	
2	`shift`	ROOT, the	cat writes	
4	`shift`	ROOT, the, cat	writes	
5	`l-arc`	ROOT, cat	writes	the ← cat
6	`shift`	ROOT, cat, writes		
7	`l-arc`	ROOT, writes		cat ← writes
8	`r-arc`	ROOT		ROOT → writes

Table 3.2: Transition parsing actions given the sentence *the cat writes nothing*.

There are many variants of the transition parser (Attardi, 2006; Nivre, Fernández-González, 2014), mainly these variants introduce new actions that can be performed.

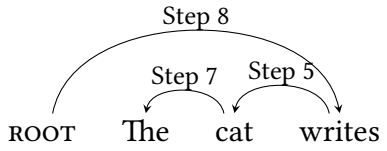


Figure 3.10: Dependency tree for the sentence *the cat writes* predicted by a (fictional) transition-based dependency parser.

Other approaches In addition to these systems, there have been several other proposed methods, among others using a sequence-to-

sequence (Strzyz et al., 2019b,a) model, and a stack-pointer network (Vinyals et al., 2015; Ma et al., 2018; Fernández-González, Gómez-Rodríguez, 2019) inspired from learning graph structures. Common to many dependency parsers is that POS tags are used in the input to help disambiguate which words are connected. There are two general approaches to using POS tags; either the POS tags are taken from the data annotations, or the model predicts them. Another direction that has gained traction is simply parsing dependency trees without any POS tags (Lhoneux de et al., 2017). In the approach where POS tags are selected from the data, we can only parse texts with POS tags annotated, limiting the system to sentences that have POS information available.

Probing models for dependency structures Comparing the performance of graph and transition parsers reveals that when using bidirectional LSTM networks there are distinct differences between the approaches (Kulmizev et al., 2019). In particular, transition-based parsers tend to have lower accuracy than graph-based parsers on long-range dependencies and dependencies that occur close to the ROOT. But, recent transformer appears to have changed these differences, bringing the performance of the two approaches closer to each other with respect to the types of errors they tend to make.

The internal structure of the transformer model has been used by Htut et al. (2019); Raganato, Tiedemann (2018) to create dependency trees (See Section 3.4) by running, for example, the Maximum Spanning Tree (Chu, 1965; Edmonds, others, 1967) algorithm over the attention matrix. The results from the experiments can be used to assess whether the attention scores of different attention heads in the transformer correspond to linguistically motivated word-word combinations. Similarly, Hewitt, Manning (2019) explores how the L_2 distance between token representations allows a model to reconstruct dependency trees. This line of work naturally connects with the work of (Michel et al., 2019; Lee et al., 2019), since it also

explores how different parts and representations of a model result in interpretable outputs. (Søgaard et al., 2018) show that additional work needs to be put on how to process punctuation.

"I stick my finger into existence and it smells of nothing."

Sören Kierkegaard

Chapter 4

Research questions

In NLP there is a recent growing trend of producing systems end-to-end; systems are trained to take a text as input and are expected to produce an output. These end-to-end systems show an impressive performance on a variety of NLP tasks, and new varieties are continuously being produced. This approach has produced systems that appear to perform language tasks on a near human-level performance. However, when looking more closely at what systems are doing on a per-example basis it becomes clear that while impressive these systems also fail on many simple examples. This gives rise to our research questions, namely; given that end-to-end systems perform well on a global metric but seems to fail on certain types of examples, what are the factors which are responsible for this behavior.

Our main contributions to NLP research in this thesis are an exploration of how these systems can be used effectively to encode the structure of the input, and how the structure of the input result in meaningful representations. To reiterate, we posed the following research questions in the introduction:

RQ1 How to obtain representations of grammatical structure?

RQ2 How to predict semantic phenomena based on representations of grammatical structure?

The first research question concerns how representations of grammatical structure can be obtained. That is, the input to end-to-end trained models is a piece of text that has some implicit grammatical structure, and humans use these cues to build a meaning representation of a text. So, these structures seem to be relevant for processing language. The first research question targets whether this grammatical structure can be predicted from the representations obtained by passing input through a model. In particular, we explore whether and how grammatical properties can be encoded in the models' representations.

The second research question concerns how representations of grammatical structure can be used to analyze and obtain meaning representations. That is, technically models are able to minimize the loss of an objective and show an impressive performance if we only consider global metrics, which indicate that they are successful at the tasks presented. But, looking more closely at the performance of these models it appears as if models resort to simple patterns of reasoning to analyze meaning. The patterns of reasoning used by these models can produce proper results, but it is by no means guaranteed. In particular, it has been shown for a number of tasks that models exploit statistical patterns that appear in the data (Poliak et al., 2018; Gururangan et al., 2018), patterns that correspond to biases present in the data. Exploiting statistical patterns can reveal insights about how some language phenomenon is used and constructed, but one has to be careful so that the patterns exploited are truly representative of the phenomena. That is, models should learn from patterns that reveal insights about language, and not from those patterns which are spurious. To circumvent this behaviour we are interested in exploring how grammatical pattern help improve the semantic capabilities of these models.

4.1 Summary of Papers

In this section, the published papers are summarised. Each paper is summarised as follows: First, a general introduction to the problem and the approach taken is given. Then how the paper answers the research questions is presented. Some ideas for future research are then given. Finally, the current authors' contributions to the papers are declared.

The conclusions we come to with respect to the research papers are presented on a per-paper basis. Because as it stands, this thesis opened more questions than it answered. This is to be expected given the complexity and rapid advancements recently in the field.

Composing Byte-Pair Encodings for Morphological Sequence Classification

In this paper, we tackle the task of morphological sequence tagging, or simply morphological tagging. Because modern transformer models use Byte-Pair encodings a "word" can be composed of two different vectors. Previous work has used the first Byte-Pair token to predict morphological features and has ignored the remaining ones. In this paper, we build a transformer model that combines all Byte-Pairs a word consists of and then predicts the morphological features. As combining embeddings is not trivial we consider three different methods and contrast the results with simply using the first Byte-Pair in a word.

Contributions: The first research question is addressed as follows:

- We show that the way of constructing word representations from sub-word representations proposed by Devlin et al. (2019) produces sub-par word representations with respect to predicting morphological features.

- We find that simple methods that take the information contained in each sub-word into account are more effective than relying on the transformers' ability to pool the predictive information into a representation.
- By using the sub-word representations obtained from our methods, it is possible to produce representations that better encode grammatical features, by more carefully considering how to combine sub-word representations.

Future directions A natural extension of our work is to evaluate what method of constructing word embeddings is appropriate given the task, and the linguistic information that should be used. That is, we show that for a specific task, our three methods perform better than the default method for constructing word representations. However, how dependent is this on the task, and are there certain characteristics of the task, both in terms of machine learning objective and the linguistic information we are after, which favors certain methods.

Statement of contribution In this project I came up with the idea of composing Byte-Pair encodings, implemented the model, and performed all experiments. The analysis was done jointly with Jean-Phillipe Bernardy.

Can the Transformer Learn Nested Recursion with Symbol Masking?

In this paper, we consider the popular transformer model and how well it can learn nested recursive structures which commonly occur in language (Hauser et al., 2002; Dehaene et al., 2015). As we are specifically interested in the transformers' ability to learn a certain type of structure, we train and test the model on the

Dyck language which constructs strings using opening and closing pairs of characters. A benefit of the Dyck language is that we can easily control the levels of nested recursion and investigate at what point it becomes difficult to learn. In addition to this, we also investigate how the structure of the transformer in terms of layers and attention heads influences the performance.

Contributions The first research question is addressed by the following findings

- A minimal version of the transformer model is able to produce representations from which nested structures in a formal language can relatively confidently be predicted.
- We find that to achieve this the model resort to a simple reasoning strategy that does not allow for generalization to more complex instances of the same phenomena.
- Thus, to obtain informative representations given a semantic phenomenon attention needs to be put on how the training is set up, which ensures that the model obtains strategies that can generalize.

Future directions Based on our conclusions with respect to the research question a line of work that could be pursued is a more detailed analysis of how tasks are set up and how this influences the learning strategies models obtain. This could shed more light on how models can be improved by biasing models to learn or find, appropriate strategies for a task. This can be done by constructing test sets that require generalizable methods to solve.

Statement of contribution In this project I implemented the transformer model from scratch and performed the experiments. The

idea for the experiments and the analysis were done jointly with Jean-Phillipe Bernardy and Vlad Maraev.

Can Predicate-Argument relationships be extracted from UD trees?

In this paper, we explore how predicate-argument structures can be extracted from a QA-SRL (Question-Answering Semantic Role Labeling) dataset. To extract predicate-argument structures we produce enhanced UD graphs for each sentence in the dataset. We then consider another rule-based system for extracting the predicate-argument structures. To ascertain the effectiveness of enhanced UD we contrast the results by extracting predicate-argument structures from UD trees. Additionally, we explore the difference between two UD parsers.

Contributions The second research question is addressed by the following findings

- In principle, we find that the upper bound of a rule-based model exceeds 98% given that the correct rules are found.
- We find that enhancing UD parsers with semantic information performs better than not enhancing them for the task of extracting semantic predicate-argument structures.
- We find that the inventory of phenomena used to enhance the parsers with semantic information lacks some components which are crucial for extracting additional semantic predicate-argument structures.

Future directions This project led to several interesting findings regarding what phenomena to represent in a syntax-semantic schema. One future direction that can be pursued is to what extent

these different phenomena are relevant for a certain task. That is, given that we want to train a model to do some task, can we a priori identify what part of a syntax-semantics schema is relevant to accomplish the task. This could be accomplished by discovering some form of weight of the different phenomena.

Statement of contribution In this paper, I came up with the idea for the investigation. I did the initial parsing of the dataset into UD trees. The rule-based system for extracting predicate-argument structures was jointly developed with Jean-Phillipe Bernardy, and the analysis of the rule-based system and dataset was done jointly with Jean-Phillipe Bernardy and Stergios Chatzikiriakidis.

Language Modelling with Syntactic and Semantic representations for Acceptability Predictions

This project explores unsupervised prediction of acceptability judgments using syntactic and semantic representations. The idea is that if this structure is explicitly modeled it could help systems better predict the acceptability. We consider both grammatical relations and tree structures, as well as semantic labels. In particular, we investigate how adding these components individually and combined affects the performance of our model.

Contributions The second research question is addressed by the following findings

- We note that neither syntactic nor semantic information improves the correlation between model and human acceptability judgments.
- We find that enhancing LSTM language models with syntactic information can provide useful information, while the semantic information does not, in terms of model perplexity.

- Thus, we find that in terms of perplexity, the model's performance has improved, but not in terms of its performance on the task.

Future directions Given that there is a vast amount of structured information available, what are the plausible strategies for including this in a model for a given task. That is, while the information may not directly improve the global metric of the task, it can still provide useful information. How can this resource be leveraged in such a way that the undesirable effects are minimized while the desired effects are maximized?

Statement of contribution In this project the idea was developed jointly by me, Jean-Phillipe Bernardy, and Shalom Lappin. I did the implementation of the models and performed the experiments. The analysis was done jointly with Jean-Phillipe Bernardy and Shalom Lappin.

How does Punctuation affect Neural Models in Natural Language Inference

In this paper, we explore a very common phenomenon in written language, namely the usage of punctuation symbols. While being common, the research about the effect it has on predictions given by neural models has not been explored much. In particular, we consider the influence of punctuation in the task of natural language inference. We analyze this in two ways, we add missing final stops to sentences that do not already have them, and we remove all punctuation as most of it does not have a direct influence on the sentences meaning. Furthermore, we also develop a small dataset to investigate fine-grained shifts in meaning that punctuation may introduce.

Contributions The second research question is addressed by the following findings

- We find that different models produce sub-par representations of punctuation for predicting types of inference. Similar results are obtained in (Søgaard et al., 2018), but for the task of dependency parsing.
- RNN representations are sensitive to irrelevant punctuation when producing meaning representations, while BERT is not, showing that BERT produces more informative representations in this aspect.
- We note that neither model appears to produce representations that properly take into account semantically relevant punctuation symbols.

Future directions A question that arises from this project is why systems produce such strange behavior to the common phenomena of punctuation. Because, it is used in basically every text to indicate various structures, but models do not seem to pick up on this. This is rather strange given how good of a representation these models generally are able to create for common phenomena, so a productive future direction would be to investigate how, and why, punctuation is so hard for models to represent.

Statement of contribution In this project me and Stergios Chatzikiriakidis jointly came up with the idea, I implemented the model and ran the experiments, and the analysis was done jointly with Jean-Phillipe Bernardy and Stergios Chatzikiriakidis.

Papers not included in this thesis

During my Ph.D., I was also a part of other papers that were not included in this thesis. The reason for not including these papers

is that my contribution to these papers was minor, they were written before my Ph.D. started but published during or they were not related to my main interests/research topics.

- Vector Norms as an Approximation of Syntactic Complexity (Ek, Ilinykh, 2023)
- Fine-grained Entailment: Resources for Greek NLI and Precise Entailment (Amanaki et al., 2022)
- We went to look for meaning and all we got were these lousy representations: aspects of meaning representation for computational semantics (Dobnik et al., 2022)
- UniMorph 4.0: Universal Morphology (Batsuren et al., 2022)
- Training Strategies for Neural Multilingual Morphological Inflection (Ek, Bernardy, 2021)
- SIGMORPHON 2021 Shared Task on Morphological Reinflection: Generalization Across Languages (Pimentel et al., 2021)
- How much of enhanced UD is contained in UD? (Ek, Bernardy, 2020b)
- Annotation Guideline No. 7: Guidelines for annotation of narrative structure (Wirén et al., 2020)
- Synthetic propaganda embeddings to train a linear projection (Ek, Ghanimifard, 2019)
- Distinguishing narration and speech in prose fiction dialogues (Ek, Wirén, 2019)

Part II

Papers

Chapter 5

Composing Byte-Pair Encodings for Morphological Sequence Classification

Abstract

Byte-pair encodings is a method for splitting a word into sub-word tokens, a language model then assigns contextual representations separately to each of these tokens. In this paper, we evaluate four different methods of composing such sub-word representations into word representations. We evaluate the methods on morphological sequence classification, the task of predicting grammatical features of a word. Our experiments reveal that using an RNN to compute

Published in the article: Adam Ek and Jean-Philippe Bernardy. 2020. Composing Byte-Pair Encodings for Morphological Sequence Classification. In Proceedings of the Fourth Workshop on Universal Dependencies (UDW 2020), pages 76–86, Barcelona, Spain (Online). Association for Computational Linguistics.

word representations is consistently more effective than the other methods tested across a sample of eight languages with different typology and varying numbers of byte-pair tokens per word.

5.1 Introduction

After its introduction, the Transformer model (Vaswani et al., 2017) has emerged as the dominant architecture for statistical language models, displacing recurrent neural networks, in particular, the LSTM and its variants. The Transformer owes its success to several factors, including the availability of pretrained models, which effectively yield rich contextual word embeddings. Such embeddings can be used as is (for so-called *feature extraction*), or the pre-trained models can be finetuned to specific tasks.

At the same time as Transformer models became popular, the tokenization of natural language texts have shifted away from methods explicitly oriented towards words or morphemes. Rather, statistical approaches are favoured: strings of characters are split into units which are not necessarily meaningful linguistically, but rather have statistically balanced frequencies. For example, the word "scientifically" may be composed of the tokens: "scient", "ifical", "ly" — here the central token does not correspond to a morpheme. That is, rather than identifying complete words or morphemes, one aims to find relatively large sub-word units occurring significantly often, while maximizing the coverage of the corpus (the presence of the "out of vocabulary" token is minimized). Approaches for composing words from sub-word units have focused on combining character n-grams (Bojanowski et al., 2017), while other approaches have looked at splitting words into *roots* and *morphemes* (El Kholy, Habash, 2012; Chaudhary et al., 2018; Xu, Liu, 2017), and then combining them.

In this paper, we consider Byte-Pair Encodings (BPE) (Sennrich et al., 2016). BPE has been popularized by its usage in translation

and the BERT Transformer model (Devlin et al., 2019). The BPE algorithm does not specifically look for either character n-grams or morphs, but rather it aims at splitting a corpus $\mathcal{C}$ into N tokens, where N is user defined. Even though BPE is not grounded in morphosyntactic theory, the characteristics of the sub-word units generated by BPE will be directly influenced by morphosyntactic patterns in a language. In particular, it is reasonable to expect that the statistical characteristics of BPE to be different between languages with different typologies. One issue with this tokenization scheme is that models based on BPE provide vector representations for the BPE tokens (which we call *token embeddings* from now on), while one is typically interested in representations for the semantically meaningful units in the original texts, *words*. In sum, one wants to combine *token embeddings* into *word embeddings*.

Our main goal is to explore how to best combine token embeddings in the context of sequence classification on words, that is, the task of assigning a label to every word in a sentence. Coming back to our example, we must combine the token embeddings assigned to the BPE tokens "scient", "ifical" and "ly" to form a word representation of "scientifically" (as a vector) which we can then assign a label to.

To our knowledge, this is a little-studied problem. For the original BERT model Devlin et al. (2019) simply state that for named entity recognition the first sub-word token is used as the word representation. For morphological sequence classification Kondratyuk, Straka (2019); Kondratyuk (2019) report that only small differences in performance were found between averaging, taking the maximum value or first sub-word token. In this paper we explore the problem in further detail and identify the effect that different methods have on the final performance of a model. Additionally, with the increased interest in multilingual NLP it becomes important to explore how different computational methods perform cross-linguistically. That is, because languages are different morphosyn-

tactically, one can expect various computational methods not to be uniformly effective.

5.2 Task

To investigate composition methods for token embeddings we focus on the task of morphological sequence classification. The task is to assign a tag to a word that represent its grammatical features, such as gender, number and so on. In addition to the word-form, the system can use information from context words as cues. While the grammatical features primarily are given by the word-form, useful information is also found in the context.

Thus, we have to identify k different tags for a word, each with C_i possible classes, making the task a multi-class classification problem. We simplify the classification problem by combining the different tags into a composite tag with up to $\prod_i^k C_i$ classes (instead of making k separate predictions). This task is suitable for our goal as the output space is large, ranging from 100 to 1000 possible tags for a word, depending on the grammatical features present in the language[1], and is directly linked to the affixes in the word-form. A system must efficiently encode information about the structure of the target words as well as the context words to be able to predict the correct grammatical features.

5.3 Data

For both training and testing data, we use the Universal Dependencies dataset (Nivre et al., 2018) annotated with the UniMorph schema (McCarthy et al., 2018). We are mainly interested in how the accu-

[1]For practical reasons, we only consider tag combinations observed in the dataset

Language	Typology	$\frac{\text{BPE}}{\text{word}}$	Tags	Train	Validation	Test
Basque-BDT	Agglutinative	1.79	919	97336	12206	11901
Finnish-TDT	Agglutinative	1.98	591	161791	19876	20541
Turkish-IMST	Agglutinative	1.73	1056	46417	5708	5734
Estonian-EDT	Agglutinative	1.86	512	346986	43434	43825
Spanish-AnCora	Fusional	1.25	177	439925	55196	54449
Arabic-PADT	Fusional	1.39	300	225494	28089	28801
Czech-CAC	Fusional	1.77	990	395043	50087	49253
Polish-LFG	Fusional	1.75	634	104730	13161	13076

Table 5.1: Treebank statistics showing the language typology, average number of BPE tokens per word, the number of (composite) morphological tags and the size of the datasets in terms of words.

racy is influenced by different composition methods, but also consider the type of morphology a language uses as a factor in this task. With this in mind, we consider both languages that use agglutinative morphology where each morpheme is mapped to one and only one grammatical feature, and languages that use fusional morphology where a morpheme can be mapped to one or more grammatical features. The fusional languages that we consider are Arabic, Czech, Polish and Spanish, and the agglutinative languages that we consider are Finnish, Basque, Turkish, and Estonian. We show the size, the average number of BPE tokens per word, and the number of morphological tags for each treebank in Table 5.1.

The fusional languages were chosen such that two of them (Czech and Polish) have a higher BPE per word ratio than the other two (Arabic and Spanish). We make this choice because one factor that impacts the accuracy obtained by a composition method may be the BPE per word ratio. By having both fusional and agglutinative languages with similar BPE per word ratio we can take this variable into account properly in our analysis.

5.4 Method

In this section we present the model used for morphological sequence classification, the methods that we use to compose token embeddings, and how the model is trained. [2]

5.4.1 Model

Our model is composed of three components, each of them detailed below. First, the input sequence of BPE tokens is fed to a Transformer model, which yields a contextual vector representation for each BPE token. The contextual information here is the surrounding BPE tokens in the sentence. Then, the token embeddings are combined using a composition module, which we vary for the purpose of evaluating each variant. This component yields one embedding per original word. Then we pass the word embeddings through a bidirectional LSTM, which is followed by two dense layers with GELU (Hendrycks, Gimpel, 2016) activation. These dense layers act on each word embedding separately (but share parameters across words). An outline of the model is presented in Figure 5.1, where f represents the different methods we use to combine token embeddings.

5.4.1.1 Underlying Transformer Model

To extract a embeddings for each BPE token, we use the XLM-RoBERTa (Conneau et al., 2020) model[3]. XLM-R is a masked language model based on the Transformer, specifically RoBERTa (Liu et al., 2019b), and trained on data from 100 different languages, using

[2] Our code is available at: `https://github.com/adamlek/ud-morphological-tagging`

[3] We use the huggingface implementation `https://huggingface.co/transformers/model_doc/xlmroberta.html`

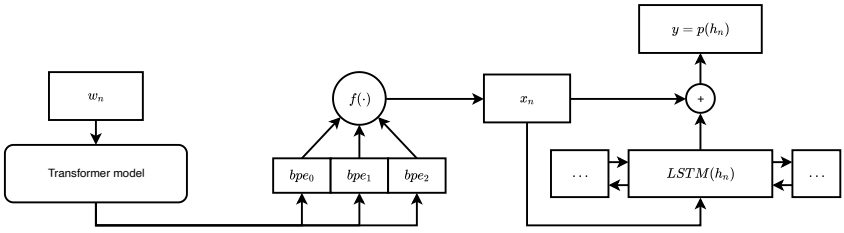


Figure 5.1: Model outline for one input. A word w_n is tokenized into k BPE tokens. The Transformer model produces one embedding per token per layer. We then calculate a weighted sum over the layers to obtain one representation per token. The resulting token embeddings are then passed to a composition function f that combines the k different token embeddings into a word embedding. The word embedding is then passed to an LSTM followed by a dense prediction layer.

a shared vocabulary of 250000 BPE tokens. All the languages that we test are included in the XLM-R model. In this experiment we use the XLM-R_{base} model with 250M parameters. It has 12 encoder layers, 12 attention heads and use 768 dimensions for its hidden size.

5.4.1.2 Feature extraction

The XLM-R model uses 12 layers to compute a vector representation for a BPE token. It has been shown in previous research (Kondratyuk, Straka, 2019; Raganato, Tiedemann, 2018; Liu et al., 2019a) that the different layers of the Transformer model encode different types of information.

To take advantage of this variety, we compute token embeddings as a weighted sum of the layer representation (Kondratyuk, Straka, 2019), using a weight vector w, of size l, where l is the number of layers in the Transformer model. The weight vector w is initialized from a normal distribution of mean 0 and standard deviation 1. If

r_{ji} is the layer representation at layer j and token position i, we calculate the weighted sum as follows:

$$x_i = \sum_{j=1}^{l} \mathsf{softmax}(w)_j r_{ji} \tag{5.1}$$

Consequently, in end-to-end training, the optimiser will find a weight for extracting information from each layer ($\mathsf{softmax}(w)_j$) which maximizes performance.

5.4.1.3 Composition of BPE token embeddings

The weighted sum yields a token embedding for each BPE token. We proceed to combine them into words as they appear in the data. The model that we use to combine token embeddings is as follows. For each sentence we extract n token embeddings x^0 to x^{n-1} from XLM-R_{base}, and then align them to words. We then pass all token embeddings in a word to a function f which combines the tokens into a word embedding.

We consider four methods for composing token embeddings: taking the first token embedding, summation, averaging, and using an RNN. Taking the first token embedding, summation and averaging have been used in previous work (Sachan et al., 2021; Kondratyuk, 2019; Devlin et al., 2019), but using an RNN has not been explored before to our knowledge.

First: The first method is the standard one used by Devlin et al. (2019), which is to use the first token embedding in a word.

Sum: For the Sum method, we use an element-wise sum. That is, we calculate the vector sum of the token embeddings. Assuming that we have T token embeddings in a word X (the word is a matrix

of size $(T, 768)$), for dimension i we calculate the word embedding by summing the token embeddings:

$$f(X)_i = \sum_{j=1}^{T} x_i^j \tag{5.2}$$

Mean: In the mean method we calculate the sum as above and divide by the number of BPE tokens in the word. Thus, for word X we calculate the word embedding by averaging over the sum:

$$f(X)_i = \frac{1}{T} \sum_{j=1}^{T} x_i^j \tag{5.3}$$

RNN: For this method we employ a bidirectional LSTM to compose the token embeddings. For each word, we pass the sequence of token embeddings through an LSTM and use the final output as the word representation.

5.4.1.4 Word-level features and classification

The above methods of composing BPE tokens produce one contextual embedding per word. We then pass the word embeddings through an LSTM to take into account the word contexts. While the BPE token embeddings are already contextual, they are conditioned on the BPE token context, not word context. We pass the hidden states for each word to a residual connection with the pre-LSTM representation. We then pass this to two dense layers with GELU activation followed by a dense layer that computes class-scores for each word. We then use a softmax layer to assign probabilities and compute the loss accordingly.

Commonly, systems analyzing morphology use character embeddings as an additional source of information. We opted not to

include character embeddings because this would obfuscate the effect of the composition method and may mask some of the effects of the different methods.

5.4.1.5 Label smoothing

Given that many of the languages have a large number of morphological tags, we want to prevent the model from growing overconfident for certain classes. To address this issue we introduce label smoothing Szegedy et al. (2016), that is, instead of the incorrect classes having 0% probability and the correct class 100% probability we let each of the incorrect classes have a small probability.

Let α be our smoothing value, in our model we follow (Kondratyuk, Straka, 2019) and use $\alpha = 0.03$, and C the number of classes, then given a one-hot encoded target vector t of size C, we calculate the smoothed probabilities as:

$$t_{smooth} = (1 - \alpha)t + \frac{\alpha}{C} \tag{5.4}$$

In words, we remove α from the correct class then distribute α uniformly among all classes.

5.4.2 Training

In our experiments we consider two possible training regimes. In the first regime we finetune the XLM-R model's parameters, in the second we only extract weights for BPE tokens, that is, we use the model as a feature extractor. In all cases, we use end-to-end training.

When finetuning the model we freeze the XLM-R parameters for the first epoch, effectively not finetuning at first. When training the model we use a cosine annealing learning rate (Loshchilov, Hutter, 2017) with restarts every epoch, that is, the learning rate starts high then incrementally decreases to 1×10^{-12} over N steps, where N is the number of batches in an epoch. We use the Adam optimizer

Parameter	Value
Epochs	15
Batch size	4 / 32
Word LSTM size	768
Linear transform size	1536
Optimizer	Adam
Learning rate	0.001
Learning rate$_{xlmr}$	1×10^{-6}
Weight decay	0.05
Label smoothing	0.03

Table 5.2: Hyperparameters used for training the model. Slashed indicates the value of a parameter when we finetune or extract features.

with standard parameters, with a learning rate of 0.001 for layer importance parameter (w in Section 5.4.1.2), the parameters of the Word-LSTM, of the classification layer, and of the BPE-combination module (when an RNN is used). For the Transformer parameters, we use a lower learning rate of 1×10^{-6}. We summarize the hyperparameters used in Table 5.2.

As an additional regularization in addition to weight decay and adaptive learning rate, we use dropout throughout the model. Generally, we apply dropout before some feature is computed. Initial experiments revealed that a high dropout yielded the best results. We summarize the dropout used as: We replace 20 percent of the BPE tokens with `<UNK>`. Then, we compute a weighted sum of the layer representations, to regularize this operation we apply dropout on layer representations with a probability of 0.1, that is we set all representations in the layer to 0. We then combine the token embeddings into word embeddings and apply a dropout of 0.4%, and pass these into the Word-LSTM. Before the contextualized representation

		Finetuning				Feature extraction			
Treebank	Baseline	First	Sum	Mean	RNN	First	Sum	Mean	RNN
Basque-BDT	.676	.857	.884	.877	**.901**	.759	.789	.780	**.834**
Finnish-TDT	.751	.961	.958	.960	**.965**	.853	.856	.847	**.899**
Turkish-IMST	.620	.848	.859	.855	**.884**	.742	.741	.735	**.775**
Estonian-EDT	.740	.956	.955	.955	**.961**	.855	.856	.853	**.901**
Spanish-AnCora	.842	.977	.977	.977	**.979**	.951	.954	.952	**.962**
Arabic-PADT	.770	.946	.946	.947	**.951**	.920	.923	.920	**.936**
Czech-CAC	.771	.968	.968	.968	**.975**	.863	.887	.881	**.924**
Polish-LFG	.657	.956	.953	.953	**.959**	.828	.844	.840	**.878**
Average	.728	.933	.937	.936	**.946**	.846	.856	.851	**.888**

Table 5.3: Accuracy for morphological tagging. We show scores both for finetuning the XLM-R model and extracting features.

is passed to the classification layer, we apply a dropout of 0.4%.

5.5 Results

Even though our aim is to compare the relative performance of various BPE-combination methods rather than to improve on the state of the art in absolute terms, we compare our results against the baseline reported by McCarthy et al. (2019). This comparison serves the purpose of checking that our system is generally sound. In particular, the actual state of the art, as reported by McCarthy et al. (2019); Kondratyuk (2019), uses treebank concatenation or other methods to incorporate information from all treebanks available in a language, which means that results are not reported on a strict per-treebank basis and thus our numbers are not directly comparable. We report the accuracy of prediction morphological tags for each of our composition methods, and for our two training regimes in Table 5.3.

Our system performs better than the baseline. As a general trend we see that the RNN method tends to perform better than all other tested methods. This trend is consistent across both language families (agglutinative and fusional) and training regimes showing that, while the advantage of the RNN is small, it occurs consistently. In

	Finetuning				Feature extraction			
Treebank	First	Sum	Mean	RNN	First	Sum	Mean	RNN
Basque-BDT	.739	.802	.790	**.835**	.657	.715	.703	**.774**
Finnish-TDT	.940	.946	.946	**.952**	.780	.805	.794	**.861**
Turkish-IMST	.730	.780	.778	**.818**	.653	.683	.664	**.711**
Estonian-EDT	.938	.939	.939	**.949**	.779	.805	.803	**.868**
Spanish-AnCora	.956	.961	.959	**.964**	.922	.937	.930	**.947**
Arabic-PADT	.889	.896	.898	**.907**	.902	.909	.906	**.923**
Czech-CAC	.940	.947	.947	**.959**	.786	.849	.840	**.900**
Polish-LFG	.917	.920	.918	**.927**	.696	.761	.752	**.812**
Average	.881	.899	.897	**.913**	.772	.808	.799	**.849**

Table 5.4: Accuracy for morphological tagging on all words that are composed of two or more BPE tokens.

general we find that finetuning yields higher accuracy than plain feature extraction, on average the difference is about 5.8 percentage points. This difference is to be expected when finetuning has 250M more parameters tuned to the task than the feature extraction.

Focusing on the finetuning regime only, we see the largest benefits of the RNN method for Basque with an increased performance of 3.25 points, and 2.7 points for Turkish over using mean or averaging. The First method for Basque and Turkish performs worse with a decrease of 4.4 percentage points for Basque and 3.6 points for Turkish compared to the RNN method. In the bare features extraction regime, we see a larger benefit for the RNN, of 3.7 percentage points (Turkish) and 4.95 points (Basque). Again, this is not unexpected: When finetuning the error rate is smaller, and therefore there is a smaller margin for a subsequent phase to yield and improvement.

Table 5.3 reports average accuracy for every word, including those which are only composed of a single BPE token. To highlight the strengths and weaknesses of each composition method, we also compute the accuracy for longer words only (composed of two or more BPE tokens). The results can be seen in Table 5.4. We see the

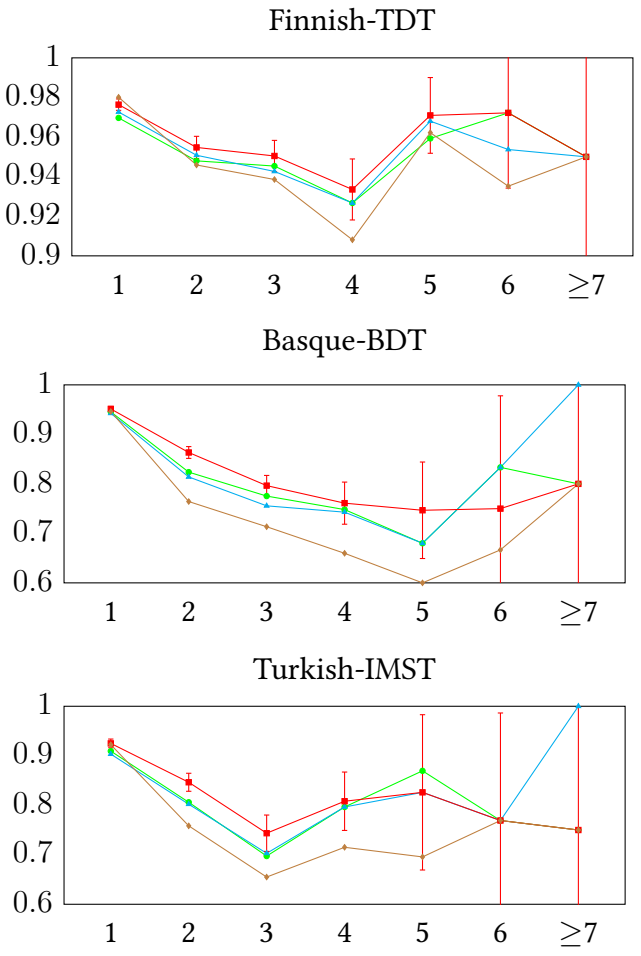
same trend for accuracy on words that are composed of two or more BPE tokens, as in the overall accuracy, where the RNN outperforms all other methods. We can also see that the average increase in accuracy when using an RNN is larger. This holds both when finetuning or extracting bare features. Given that the number of BPE tokens per word varies in the different languages, we also look at the accuracy of the different methods given the number of BPE tokens. We show per-language performance with the different methods in Figure 5.2.

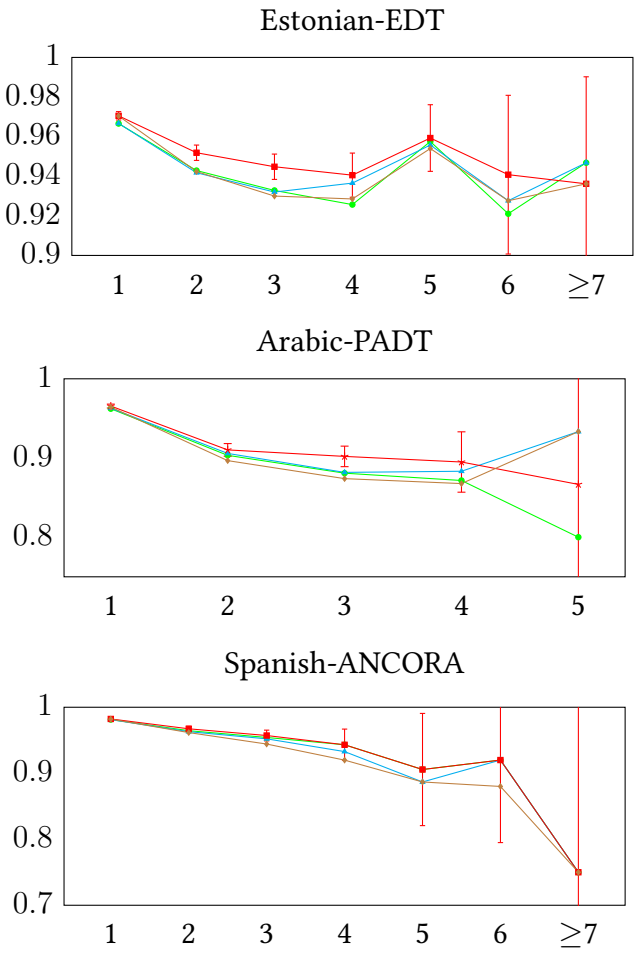
5.6 Discussion

For predicting morphological features, the RNN method is more effective than the other proposed methods (summing, averaging or taking the first BPE token). This holds regardless of training regime (finetuning versus feature extraction) and across languages with different BPE per word ratios.

As we see it, the advantage of the RNN over commutative methods (Sum, Mean) and taking the first BPE token is that it can take the order of elements into account. In broad terms, information about the order of elements in morphology allows a system to determine what is a stem, prefix, or suffix. Thus allowing a model to collect more predictive information from token embeddings.

We can suspect that the average BPE per word ratio in a language affects the performance of the composition method used. To further control this variable, in Figure 5.3 we plot the average number of BPE tokens per word in each language (x-axis), and compare this average against the gain in accuracy yielded by using the RNN method over summation (y-axis). For finetuning we see that in general the average number of BPE tokens does not matter that much. The two cases where it does matter is for Turkish and Basque, where we see a substantial improvement of about 3 percentage points. We note however that these are also the languages with the lowest


Finnish-TDT
1
0.98
0.96
0.94
0.92
0.9
1 2 3 4 5 6 ≥7
Basque-BDT
1
0.9
0.8
0.7
0.6
1 2 3 4 5 6 ≥7
Turkish-IMST
1
0.9
0.8
0.7
0.6
1 2 3 4 5 6 ≥7


Estonian-EDT
1
0.98
0.96
0.94
0.92
0.9
1 2 3 4 5 6 ≥7
Arabic-PADT
1
0.9
0.8
1 2 3 4 5
Spanish-ANCORA
1
0.9
0.8
0.7
1 2 3 4 5 6 ≥7

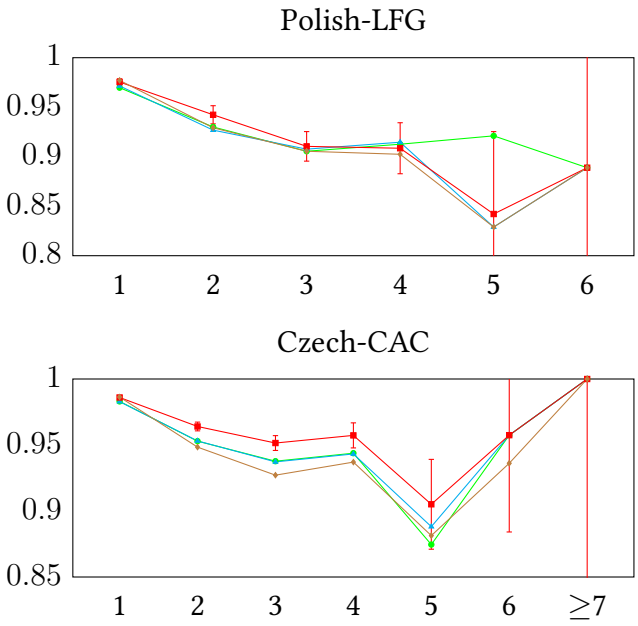


Figure 5.2: Per-language accuracy on tokens with different numbers of BPE components, for the finetuning training regime. The last data point on the x-axis refers to all tokens composed of seven or more BPE tokens. We indicate the method by encoding First as brown, summation as green, averaging as blue and RNN as red. The accuracy is given on the y-axis. We show the Agresti-Coull approximation of a 95%-confidence interval for the RNN method (Agresti, Coull, 1998). We do not show the intervals for other methods to avoid excessive clutter.

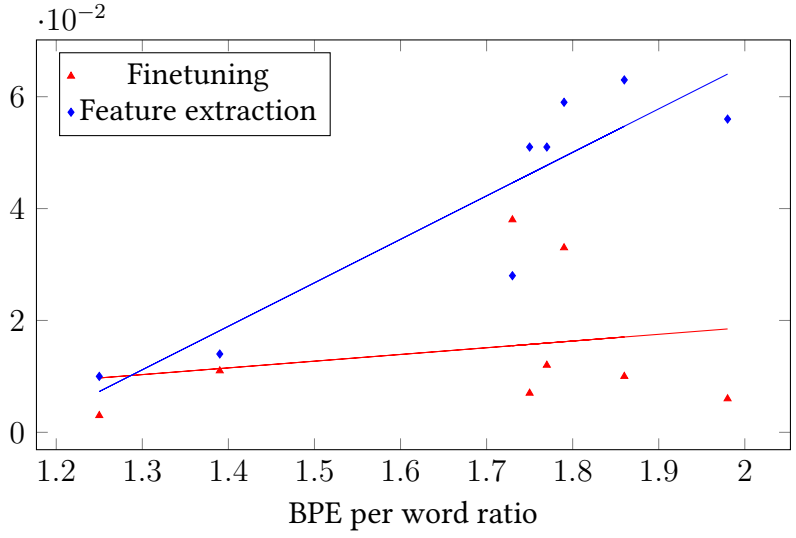


Figure 5.3: The difference in accuracy between summation and RNN plotted against average number of BPE tokens per word in all languages, with a linear regression line.

amount of training data. For the other languages the improvements lie in the range .6 to 1.2 percentage points. This indicates that when finetuning, the model can provide information that allows commutative methods to properly compose BPE tokens. However, looking at bare feature extraction we see that there is a larger gap between the low BPE-ratio and the high BPE-ratio languages.

Our sample of languages contain both fusional and agglutinative languages, and the typology does not appear to have an effect in our experiments. We see about the same trends for the fusional languages with a high BPE per word ratio as the agglutinative languages.

5.6.1 First method

The idea behind the First method is that the Transformer is sufficiently powerful to pool the relevant information into the first BPE token embedding. However, our experiments reveal that it is less efficient than any other method we tested for morphological sequence classification across languages. We see in Table 5.3 that the method is, on average, $.4$ and 1 percentage points lower than the next lowest scoring method for finetuning and feature extraction respectively. This effect is further enhanced when we consider the accuracy of words composed of more than two BPE tokens in Table 5.4, where the difference is 1.6 and 2.7 points, compared against the next lowest scoring method, for finetuning and feature extraction respectively. When we compare the performance against the RNN this difference only increases, showing a gain of 3.2 percentage points and 7.7 points for finetuning and feature extraction respectively.

While the First method may be effective, primarily because of the expressivity of the Transformer architecture, the method forces the model to push the predictive information of several BPE token embeddings into the first one. This puts an additional burden on the Transformer model, and we believe that this is the reason for the performance degradation which we observe. Besides, putting this burden on the model is not necessary: pooling information from several BPE embeddings can be done effectively using additional layers.

5.6.2 Sum and Mean

When we consider the commutative methods of combining token embeddings, summation or averaging, we see no clear advantage for either of them over the other one, when doing finetuning. However, when extracting features only we see hints that summation is more effective than averaging. For feature extraction, summation is $.5$ percentage points better than averaging, and words composed of

two or more BPE tokens exhibit an advantage of .9 point for summation.

This discrepancy suggests that by averaging, we are removing some predictive information from the pretrained BPE token embeddings, that is, by reducing the values in the token embeddings uniformly across a sequence of token embeddings we lose useful information. We believe that some token embeddings contain more predictive information than others, and by summing them we retain all the information. But when we finetune, the difference between summing and averaging almost disappears: the model appears to learn how to distribute the information uniformly across the token embeddings that compose a word and is thus able to retain the information better. Interestingly, the model learns to distribute the information across multiple BPE token embeddings more efficiently than pushing the information into the first token. This is shown by the large difference in accuracy between finetuning and feature extraction for the First and averaging method.

5.6.3 Parameterization of First, Sum and Mean

One question that arises when looking at Figure 5.2, specifically considering the performance on words composed of only one BPE token is the following: can the superiority of the RNNs be attributed to its ability to take context into account, or simply to containing more parameters and extra layers? We would expect that for the words with only one BPE token, the performance of the model would be the same for all methods. For practical reasons, we push all word embeddings through an RNN, effectively doing a non-linear transformation with tanh activations on the words composed of only one BPE token. Typically, the difference in accuracy between various methods for one-BPE-token words is small (barely visible in Figure 5.2). But for example in Finnish, we see a larger difference. Although in general if we perform better on longer words consisting of BPE to-

kens that also appear as words in the data, we could also expect the performance to be better for words of BPE length one, because we will have more accurate representations of the contextual words.

		Finetuning				Feature extraction			
Treebank	Baseline	First	Sum	Mean	RNN	First	Sum	Mean	RNN
Basque-BDT	.676	.864	.894	.890	**.901**	.772	.793	.794	**.834**
Finnish-TDT	.751	.958	.959	.961	**.965**	.857	.856	.855	**.899**
Turkish-IMST	.620	.850	.875	.867	**.884**	.742	.722	.729	**.775**
Estonian-EDT	.740	.956	.958	.958	**.961**	.865	.856	.853	**.901**
Spanish-AnCora	.842	.978	.977	.978	**.979**	.953	.954	.952	**.962**
Arabic-PADT	.770	.949	.945	.947	**.951**	.925	.923	.920	**.936**
Czech-CAC	.771	.969	.972	.972	**.975**	.873	.887	.881	**.924**
Polish-LFG	.657	.957	.953	.955	**.959**	.832	.844	.840	**.878**
Average	.728	.935	.942	.941	**.946**	.852	.854	.853	**.888**

Table 5.5: The accuracy of morphological tagging when we parameterize the First, Sum and Mean method with a non-linear transformation layer.

We test this hypothesis by parameterizing the First, Sum, and Mean method. Essentially, we need to increase the capabilities of these methods. This is done by passing all BPE token embeddings through a non-linear transformation with ReLU activation before we compute the Sum, Mean, or select the first BPE-token. Our experiment, whose results are shown in Table 5.5, shows that while adding parameters to the First, Sum, and Mean method generally improve their performance slightly, ranging between a change of -0.2 and $+0.6$ percentage points, but their performance never exceeds that of the RNN method.

5.7 Conclusions and Future Work

In conclusion, our results indicate that using an RNN to compose word representations from token representations, obtained from a large Transformer model, is more efficient than two commutative

methods, summing and averaging, and also more effective than letting a Transformer model automatically pool the predictive word-level information into the first BPE token embedding. We show this for the task of morphological sequence classification, in eight different languages with varying morphology and word-lengths in term of BPE tokens, as well as for two training regimes, finetuning and feature extraction.

In future work, we want to continue experimenting with the different BPE token embedding composition methods, specifically looking at more complex syntactic and semantic tasks, such as dependency and/or constituency parsing, semantic role labeling, named entity recognition, and natural language inference. We also wish to run our experiments on the hundreds of available UD treebanks to improve the robustness of our results.

Chapter 6

Can the Transformer Learn Nested Recursion with Symbol Masking?

Abstract

We investigate if, given a simple symbol masking strategy, self-attention models are capable of learning nested structures and generalise over their depth. We do so in the simplest setting possible, namely languages consisting of nested parentheses of several kinds. We use encoder-only models, which we train to predict randomly masked symbols, in a BERT-like fashion. We find that the accuracy is well above random baseline, with accuracy consistently above 50% both when increasing nesting depth and distances between training

Published in the article: Bernardy, Jean-Philippe, Adam Ek, and Vladislav Maraev. "Can the Transformer Learn Nested Recursion with Symbol Masking?." Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021.

and testing. However, we find that the predictions made correspond to a simple parenthesis counting strategy, rather than a push-down automaton. This suggests that self-attention models are not suitable for tasks which require generalisation to more complex instances of recursive structures than those found in the training set.

6.1 Introduction

Self-attention models (Vaswani et al., 2017) enjoy broad use in NLP tasks. The best attention-based models can tackle several tasks using a unified sentence encoding (and perhaps decoding) module Raffel et al. (2020), with applications ranging from classification to inference and generation. They provide state of the art results for all such tasks, displacing the already very successful recurrent neural networks, in particular the LSTM and its variants. The availability of large pretrained models (Devlin et al., 2019) is another strong point in their favour.

However, the generalisation capabilities of self-attention models are still not well understood, and the present work is part of an ongoing effort to understand their capabilities. We study in particular their ability to learn context-free languages, which are characterised by the nested structures. For this purpose, we control the inputs to the model to the maximum, while focusing on the defining characteristic of context-free languages, namely matching opening and closing brackets. This corresponds to learning *generalised Dyck languages* (see table 6.2). In particular, we investigate the following questions:

1. Can self-attention generalise to matching open/close parenthesis at longer distances?

2. Can self-attention generalise to matching open/close parenthesis at deeper nesting levels distances?

There is a already a small body of work dealing with this question (see sec. 6.5), but our contribution is specific in the following two respects: i) We use the popular BERT-like training regime (predict a percentage of randomly masked tokens), ii) We concentrate on generalising to (much) deeper nesting.

Beyond theoretical considerations, matching brackets have applications in the NLP-style treatment of constructed languages (in particular) programming languages, for example translating between programs and their natural language descriptions.

6.2 Data Sets

We define the language $\mathcal{D}_n$ as the set of strings generated by the following context-free rules: $E ::= ; E ::= EE; E ::= oEc$, where (o, c) stands for a pair of matching parenthesis pairs. The index n stands for the number of possible pairs. In all of our tests, we will use $n = 5$ (corresponding for example to the pairs `()` , `[]`, `{}`, `<>` and «»), and thus we drop the subscript from now on.

We are interested in various characteristics of the strings of $\mathcal{D}$. First, we consider the distance between a closing parenthesis and the corresponding opening parenthesis. Given a string s of length $2N$ (N is the number of matching pairs), we will call (s) an array of length $2N$ such that if s_i is a closing parenthesis, $(s)_i$ is the distance between s_i and the closing parenthesis. If s_i is an opening parenthesis, $(s)_i$ is 0. For example, if $s =$"`{()<[](«»)>}`", $(s) = [0, 0, 1, 0, 0, 1, 0, 0, 1, 3, 9, 11]$. The second characteristic that we consider is the amount of nesting between closing and opening parentheses. We call this characteristic η(s), and likewise we define it for each closing parenthesis, and let it be zero for opening parentheses. For example, if $s =$"`{()<[](«»)>}`", $(s) = [0, 0, 1, 0, 0, 1, 0, 0, 1, 2, 3, 4]$.

To generate a string with N matching pairs, we perform a ran-

dom walk between opposite corners of a square grid of width and height N, such that one is not allowed to cross the diagonal. When not restricted by the boundary, a step can be taken either along the x or y axis with equal probability. A step along the x axis corresponds to open a parenthesis, and one along the y axis corresponds to closing one. The kind of parenthesis pair is chosen randomly and uniformly. We call the distribution of input strings sampled by this procedure D. In all our experiments we set $N = 10$ (which is enough to illlustrate our points) and we thus omit the superscript in what follows.

We also want control the maximum distance between opening and closing parentheses (so that we never train on too long distances). We do so by discarding elements s of D such that $(s)_i > d$ for some i, and call the resulting distribution $D[MaxDist = d]$.

Often we want to control the maximum depth that our model is trained or tested on. For this purpose, we generate strings s which exhibit at least one index i such that $(s)_i = d$, but no index j such that $(s)_j > d$. These paths can be generated by constraining the path on the grid to touch a diagonal at distance d to the origin diagonal, and we call the corresponding distribution $D[MaxDepth = d]$.

6.3 Model and masking strategy

We implement a variation of the transformer model as introduced by (Vaswani et al., 2017). In the model each input symbol is associated with a vector embedding of size K. A sequence of opening and closing brackets is represented by a matrix of size (N, K).

Following Devlin et al. (2019), our model then applies a series of multi-head self-attention layers organised in a hierarchical structure, such that the second layer operates on the representations generated in the first layer, and so on. We use a BERT-like, non autoregressive architecture: each layer attends to every position in the

input, including itself. Then a softmax classifier is employed to predict the symbol at the current position. Hence, we use a masking strategy to train and test the model (otherwise it could simply use the current symbol for prediction).

For training, we follow the masking strategy presented by Devlin et al. (2019). We mask 15% of the closing parenthesis tokens at random, where in 80% of the cases we replace the token with a mask token, in 10% of the cases with a random token, and in the remaining 10% of the cases we replace it with the same token.

For testing, after sampling a string s, we pick a random position i such that s_i is a closing parenthesis. Then we mask all subsequent symbols, and let the model predict s_i. There is a single possible closing parenthesis type for s_i, corresponding to the opening parenthesis found earlier in the string. The prediction is considered successful if the model predicts the right type of closing parenthesis.

6.4 Experiments & Results

Our experiments consists in training the language model for a limited version of the Dyck family (for example by limiting nesting depth () or maximum distance ()), and testing what the performance is in a more general case. Thus, because there are five types of parenthesis pairs in all our experiments, the random baseline is $\frac{1}{5} = 20\%$.

6.4.1 Generalisation to Longer Distances

In the first experiment we investigate whether the model is capable of predicting closing parenthesis at long distance from the corresponding opening parentheses, whereas it has only seen short-distances in the training data. More precisely, we train the model on strings from $D[MaxDist = 9]$ and test it on $D[MaxDist = 19]$.

We present an overview of the results in table 6.1. Our experi-

Table 6.1: Mean accuracy and standard deviation over 10 runs on generalisation to longer distances for each model configuration.

Layers	Heads	Accuracy
4	4	0.814(± 0.013)
8	2	0.643(± 0.005)
2	8	0.844(± 0.008)

ments show that the (2 layers, 8 heads) model generalises the best. Using fewer heads appears to be more detrimental to the model's accuracy than the number of layers. This is true even though the (8,2) model has many more parameters than the (2,8) model (see appendix).

The aggregated numbers however hide much of the reality of the generalisation capabilities as a function of distance. Therefore we further break down the accuracy by distance to the corresponding opening parenthesis in figure 6.1. The (8,2) model fails to learn parenthesis matching at short distances, but its accuracy is better for longer distances. In contrast the (4,4) and (2,8) models do well for adjacent parentheses, but their accuracy drops quickly until reaching a minimum at distance 13, dipping below 50% accuracy —however still above chance. Perhaps surprisingly, all models do very well at very long distances. These very long distances correspond to matching parentheses at the beginning of the input with parentheses at the end (that is, when we mask the fewest number of input symbols).

6.4.2 Generalisation to Deeper Nesting

In the second experiment we test whether the model can generalise to deeper nesting depths. That is, we train the model on $D[MaxDepth = 3]$ and test it on $D[MaxDepth = 9]$

We present an overview of the results in table 6.2. Looking at the results we see a similar pattern in terms of aggregated accuracy

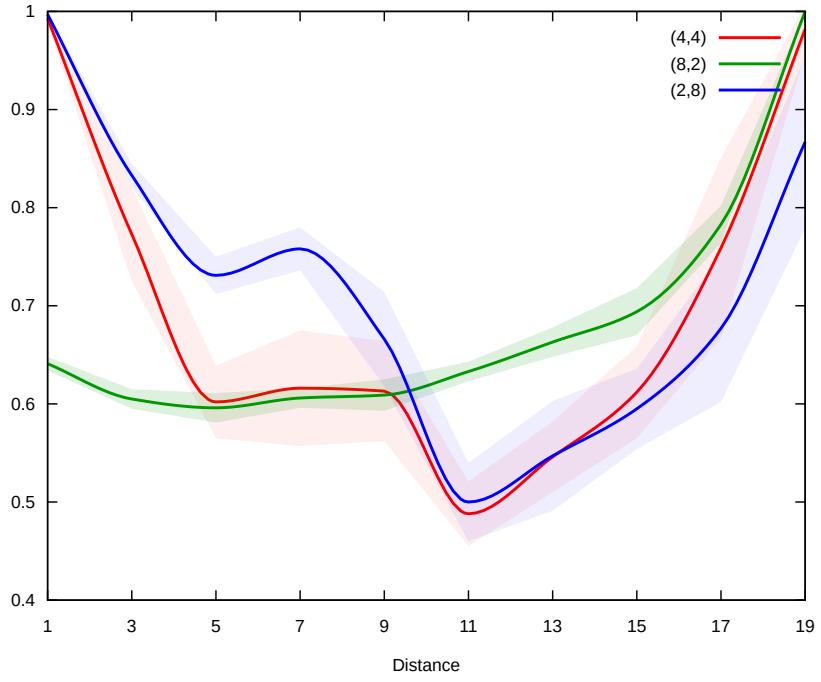


Figure 6.1: Mean model accuracy for closing parenthesis depending on a distance to corresponding opening parenthesis, over 10 runs. Shaded areas correspond to standard deviation.

Table 6.2: Mean accuracy and standard deviation over 10 runs on generalisation to deeper nesting for each model configuration.

Layers	Heads	Accuracy
4	4	0.654(± 0.012)
8	2	0.518(± 0.005)
2	8	0.672(± 0.008)

as in the previous experiment: the (2,8) setup performs the best, followed by (4,4) and finally (8,2). Breaking down accuracy by nesting depth (figure 6.2) reveals that the difference resides chiefly in the (8,2) model failing to predict shallow nesting.

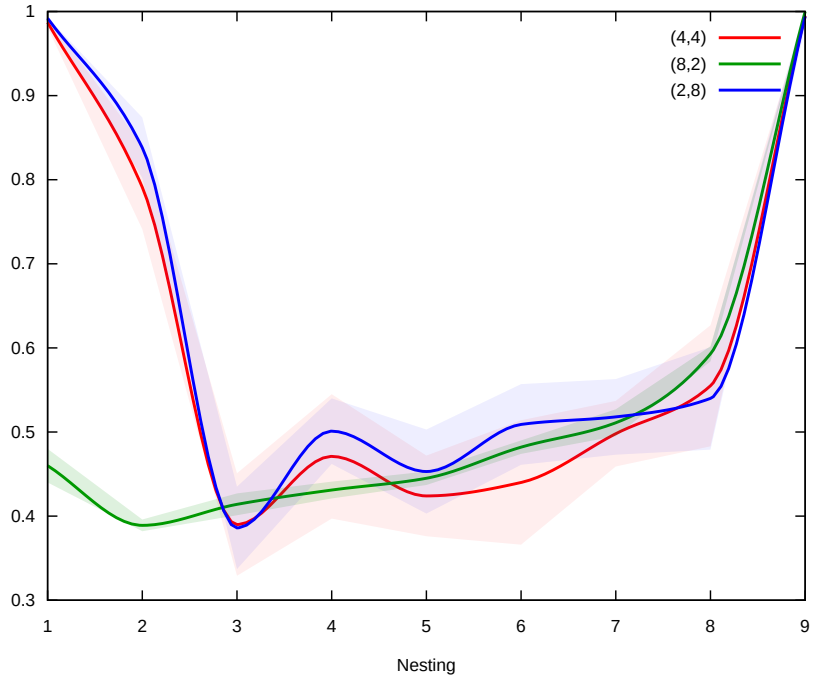


Figure 6.2: Mean model accuracy for closing parenthesis depending on a distance to corresponding opening parenthesis over 10 runs. Shaded areas correspond to standard deviation.

6.4.3 Analysis of attention heads

We have analysed attention heads by manual inspection of softmax score for attention heads for each layer, on several sequence from our training set (see Appendix for the corresponding heat maps).

Looking at the behaviour of the attention heads we note that the first layer in the (2,8) and (4,4) models focuses its attention on the previous symbol. Then, in the final layer of the (2,8) model the attention of the start of the sequence focuses on the end, and vice-versa.

In the (4,4) model, the second layer appears to often focus on the non-masked symbols while in the third layer the attention is distributed more evenly between masked and non-masked symbols. A notable feature of the third layer is that a lot of self-attention occurs on the masked symbols. In the final layer, the attention of all symbols is put almost exclusively on the masked symbols.

The (8,2) model is the only model which does not have a clear layer that looks at the preceding token. It appears that in the (8,2) model, the earlier layers focus their attention on the beginning of the sequence, then it moves towards the latter part of the sequence. The heat maps also show that the (8,2) model focuses heavily on certain symbols, which are the least frequent symbols used in the sequence, for later layers. In earlier layers the model appears to focus on the frequent symbols. This analysis is compatible with the (8,2) model using a symbol counting method.

In summary, the (4,4) model appears to first look at the previous symbol in the sequence. There are two steps of searching where first the model ignores the masked symbols and distributes the attention over the other symbols. In the second step, the model again focuses all around the sequence, but the masked symbols receive a lot of attention. For the (2,8) model, the behaviour is more straightforward. First it looks at the previous symbol, then all around the sequence. To the best of our knowledge, the (8,2) model is counting symbols by distributing its attention on frequent and less frequent symbols.

6.5 Related work

Studying the ability of language models to learn Dyck languages is emerging as a standard way to test the ability to generalise to deeper nesting levels. Before self-attention, this test was applied to RNNs. Bernardy (2018) proposed non-standard stack-based RNN models, which can approach perfect accuracy for generalised Dyck-language, although the accuracy of standard RNNs was higher than random but far from perfect. Hewitt et al. (2020) presented a theoretical proof that RNNs are able to learn Dyck languages with maximum nesting depth m using $O(m)$ memory. Sennhauser, Berwick (2018) present contrasting evidence, concluding that LSTMs can learn very limited range of rules.

A number of studies have considered self-attention models, especially in the past year. Ebrahimi et al. (2020) investigated self-attention models using Dyck languages, and claimed that self-attention models with a starting symbol are able to generalise to longer sequences and deeper structures without learning recursion, as competitive LSTM models do. In contrast to us, they studied models trained autoregressively only. Bhattamishra et al. (2020) studies how autoregressive Transformer architecture learns a subset of formal languages, including Dyck language and its generalisations. In contrast to our study, they examine Shuffle-Dyck languages, which allows constructions like "`([)]`" and provide theoretical and experimental evidence that the Transformer is capable of learning such a language. On the other hand, Hahn (2020) points at the limitation of using self-attention models. He indicates that in theory the LSTM should perform better than the autoregressive Transformer, because the transformer cannot emulate a stack, general finite-state automata, or use recursion.

6.6 Conclusion and future work

Our experiments show that, with a random masking strategy, the transformer is able to discover a way to make good predictions when generalising to longer distances and deeper nesting. However, this strategy is not using the history of opening and closing parentheses in a way a push-down automaton would.

Indeed, the analysis reveals that the best accuracy is obtained when few symbols have been masked. This can be explained by the model having learned a counting strategy. When a single symbol is masked, predicting the kind of missing parenthesis can be done by subtracting the number of closing parentheses by the number of opening parentheses for each type, and predict the type which exhibits a discrepancy. For short distances our (2,8) and (4,4) models were able to learn to remember preceding symbols and act accordingly. We suspect that for intermediate levels of nesting and distance, the models act according to a mixture of the above two strategies.

In consequence, we recommend not to use a BERT-like masking strategy for applications where generalising to longer distances or deeper nesting is critical. Rather, auto-regressive models should be used, such as auto-regressive attention or RNNs.

Appendix A: Experimental setup and reproducibility information

Our implementation is based on pytorch, with a custom re-implementation of the transformer architecture, exactly following (Vaswani et al., 2017). The runtime is under one day for the whole set of experiments using a Titan X (Pascal) GPU.

The hyperparameters we use are listed in table 6.3.

Table 6.3: Hyperparameters used and the number of data examples used.

Parameter	Value
Optimiser	Adam
Learning rate	0.0001
Epochs	10
Batch size	512
Training examples	102400
Validation examples	20480

In our experiments we consider three different transformer architectures, corresponding to different values for the number of multi-head self-attention layers, and the size of the heads. Specifically, we considers the setups presented in section 6.4

Table 6.4: Model configurations and the number of parameters in each configuration

Layers	Heads	Parameters
8	2	897 292
4	4	1 191 820
2	8	1 781 452

In each case, we have used 64-dimensional embeddings throughout the models.

Appendix B: Attention heat-maps

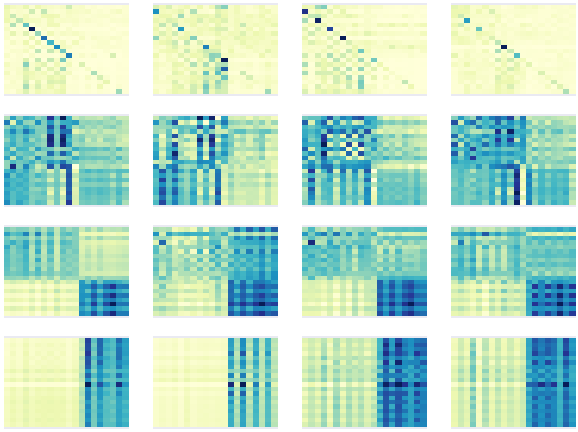


Figure 6.3: Attention heatsmaps for the model with 4 heads and 4 layers on the input +-+<[+[([()])]-]>-.

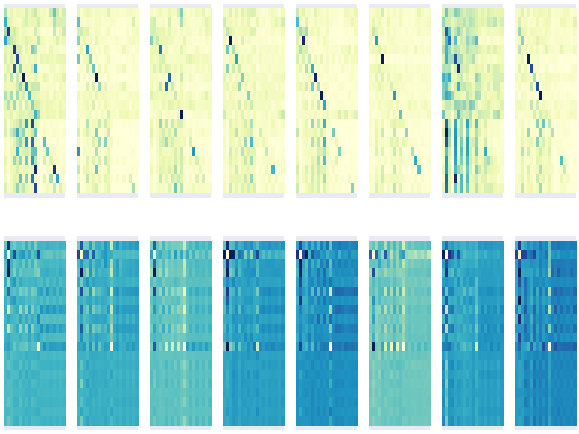


Figure 6.4: Attention heatsmaps for the model with 2 heads and 8 layers on the input +-+<[+[([()])]-]>-.

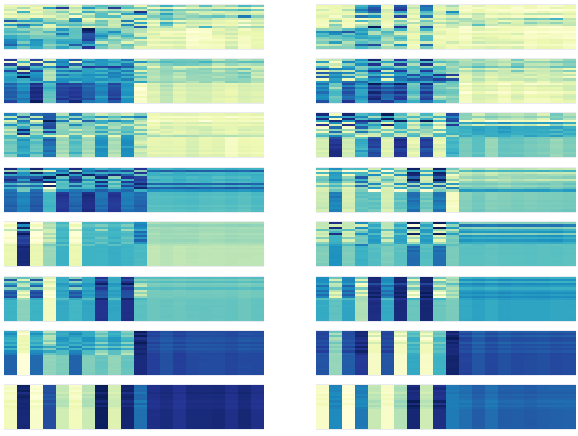


Figure 6.5: Attention heatsmaps for the model with 8 heads and 2 layers on the input +-+<[+[([()])]-]>-.

Chapter 7

Can Predicate-Argument relationships be extracted from UD trees?

Abstract

In this paper we investigate the possibility of extracting predicate-argument relations from UD trees (and enhanced UD graphs). Concretely, we apply UD parsers on an English question answering/semantic role labeling data set FitzGerald et al. (2018) and check if the annotations reflect the relations in the resulting parse trees, using a small number of rules to extract this information. We find that 79.1% of the argument-predicate pairs can be found in this way, on

Published in the article: Adam Ek, Jean-Philippe Bernardy, and Stergios Chatzikyriakidis. 2021. Can predicate-argument relationships be extracted from UD trees?. In Proceedings of the Joint 15th Linguistic Annotation Workshop (LAW) and 3rd Designing Meaning Representations (DMR) Workshop, pages 46–55, Punta Cana, Dominican Republic. Association for Computational Linguistics.

the basis of Udify Kondratyuk, Straka (2019). Error analysis reveals that half of the error cases are attributable to shortcomings in the dataset. The remaining errors are mostly due to predicate-argument relations not being extractible algorithmically from the UD trees (requiring semantic reasoning to be resolved). The parser itself is only responsible for a small portion of errors. Our analysis suggests a number of improvements to the UD annotation schema: we propose to enhance the schema in four ways, in order to capture argument-predicate relations. Additionally, we propose improvements regarding data collection for question answering/semantic-role labeling data.

7.1 Introduction

Universal Dependencies (UD), can be seen as a compromise, a balancing act between six principles, referred to as Manning's law Nivre et al. (2016):

1. UD needs to be satisfactory for analysis of individual languages
2. UD needs to be good for linguistic typology
3. UD must be suitable for rapid, consistent annotation
4. UD must be suitable for computer parsing with high accuracy
5. UD must be easily comprehended and used by a non-linguist
6. UD must provide good support for downstream language understanding tasks

Support for natural language understanding downstream tasks in the UD schema has been shown in a number of studies including event extraction, negation scope detection and opinion analysis

Fares et al. (2018), information extraction Angeli et al. (2015), image retrieval Schuster et al. (2015), question-answering Reddy et al. (2017), and Natural Language Inference Mishra et al. (2020), among many others.

However, certain syntactic dependencies relevant to semantics are not included in the original formulation of UD. For example, a word may be the subject of two conjoined verbs, but in UD the subject is only connected to one of the verbs. To discover that the word is the subject of two verbs it has to be inferred from the conjunction. However, this creates unnecessary burdens for models using the UD schema. The *enhanced UD schema* (EUD) Schuster, Manning (2016) includes such edges, with the aim to make semantics more explicit. Recently there has been a surge of interest and development of EUD, spurred on by its applicability on semantic downstream tasks such as information extraction Tiktinsky et al. (2020); Sun et al. (2020). Research into EUD has also be facilitated recently by two shared tasks on EUD parsing Bouma et al. (2020, 2021), which has resulted in a mix of machine learning and rule-based approaches for producing EUD graphs. We come back to an evaluation of the EUD schema in Section 7.5.1.

The support provided by UD w.r.t. downstream NLU tasks raises the question of how much "semantics" UD actually contains, or better put, how much semantic reasoning can one perform by using just the information provided by UD. This is also related to the question of whether UD dependencies should be seen as semantic, syntactic, or maybe something between the two. To some extent all three possibilities have been considered. One way to approach this question is to check the amount of semantic knowledge that UD exhibits, explicitly or implicitly, in relation to specific semantic tasks or features. Silveira (2016) argues that the way to see UD is as a representation "for" semantics, not "of" semantics. Under this view, UD can be seen as a kind of scaffolding where some proper semantic backbone will be built upon. Again, however, this begs the question of the na-

ture of the scaffolding. Silveira (2016) claims that UD has implicit semantic role information and also shows that their enhanced version, which, as they argue, mirror semantic relations more closely, perform better than normal UD in an event extraction task involving a model that extracts dependency features from different parses. Previous research has shown the opposite to be the case, i.e. UD performing better than the enhanced version in this task Miwa et al. (2010a,b); Buyko, Hahn (2010), even though these pieces of work are not directly tested on enhanced UD, but on previous related efforts to expand basic UD (Silveira, 2016). UD has been also criticized by researchers working in Theoretical Linguistics (Osborne, Gerdes, 2019). According to them, UD fails to observe Manning's first desideratum because "UD annotation choices are not satisfactory on linguistic analysis grounds because they result from a mixture of semantic and syntactic criteria". Lastly, one could argue that approaches that attempt to combine UD with an explicit logical semantics interface implicitly assume that UD is syntactic and/or missing crucial semantic information.

In this paper, we propose a way to test the semantic capabilities of UD *parsers* for English by using their output to infer answers in a Question-Answering task. More precisely, what we want to investigate is the question of whether predicate-argument relations are correctly captured by UD parsers. We believe that this is an important question to be posed, because, if this is the case and there is enough ground/scaffolding, then a more fine-grained semantic representation may be build on top of UD (for example, some correspondence between UD syntactic trees and logical semantics). A related question is to what extent enhanced dependencies are better, if at all, in precisely encoding predicate-argument relations.

7.2 Dataset

We perform experiments on the question-answering/semantic-role-labeling dataset of (FitzGerald et al., 2018), which is based on the work of (He et al., 2015), simply referred to as "QA-SRL" below. The rationale is that, in the QA-SRL dataset, question-answers pairs are directly concerning predicate-argument structures. Each question has a passage which it refers to. For example, the dataset might contain the passage "UN published a report" together with the question "What did something publish?". The answers are provided by annotators selecting a contiguous span of text in the passage which answers the question, in this case the object "a report".

The dataset contains passages from 3 domains in English: Wikipedia, Wikinews and science, with questions and answers generated by crowdsourcing. For each verbal predicate in the passage, questions about one of the arguments are constructed by the annotators using question templates. In total the dataset contain 265156 valid questions over 76397 passages. The QA-SRL dataset also contains an automatically generated dataset. However, we have not included this part and only consider the crowdsourced part.

7.3 Task and Method

The most obvious way to test whether UD parsers can correctly identify the semantic arguments of verbs would be to map the form of a QA-SRL question to an UD role, then retrieve the subtree of the argument from the UD tree and check if it matches the human annotations.

Unfortunately it is not easy to map the argument types of the QA-SRL dataset to UD roles. One difficulty is the mismatch of passive and active voice between questions and answers. Another problem is that the non-subject UD roles (obj/obl/advcl/etc) are in n-to-n

correspondence with the QA-SRL argument types (locations, time, etc). Converting these relationship to a functional mapping would require the use of some statistical model to extract these features from the sentence. Using a statistical model would make unclear whether it is UD that captures argument-predicate relationships, or the model. Thus, to keep the method simple we resort to checking if the UD trees obtained from a parser contains the annotated QA-SRL argument. To avoid the question of *which* semantic role should be extracted, we check if *any* of the children of the verb matches the answer. We make two further amendments to the task: 1. we enhance UD trees with EUD arcs and 2. we check for arguments in the parent position.

The second amendment helps with cases when the sentence has the form of a copula or when the verb plays the role of adjectival phrase. For example, given the passage "Paleontologists are interested in fossils" and the question "Who is interested in something?", then one should be able to recover "Paleontologists" as an argument. However, in the UD tree, "Paleontologists" is the parent of "interested". Likewise, given "The observed animals were tortoises." and the question "What was observed?" should point to "animals"; which is the parent of "observed" in the UD tree.

The first amendment is to use the EUD schema rather than plain UD. While the state-of-the-art UD parsers do not provide this information, it is possible to automatically add most EUD edges using a number of rules Silveira (2016); Ek, Bernardy (2020b). Thus our pipeline consists in first running a plain UD parser, we test both the Stanza parser (Qi et al., 2020) and the Udify parser (Kondratyuk, Straka, 2019), and then we apply the following enhancements to the UD trees, using the system developed in (Ek, Bernardy, 2020b):

1. Propagation of incoming dependencies to conjuncts;

2. Propagation of outgoing dependencies from conjuncts;

3. Propagation of subject relations for direct control and raising constructions;

4. Addition of co-reference arcs in relative clause constructions

To recapitulate, after adding enhanced edges for each question in the test set, we proceed to:

1. Find the verb index relevant to the question. Generally this information is given by the QA-SRL data. In rare cases some adjustments need to be made, for example if the parser counted words differently than the dataset we adjust the verb index accordingly;

2. Collect all possible arguments according to the EUD graph;

3. Extract the constituent for each argument by following the child edges;

4. Normalize the text of each constituent by removing punctuation, leading prepositions, and determiners. Indeed, the annotations are inconsistent regarding whether prepositions and determiners should be part of the argument or not;

5. If any of the gold answers match any of the arguments retrieved, we consider the argument retrieval a success

7.4 Results and Analysis

In this section we present the results obtained from extracting predicate-argument relations, and provide an analysis of the errors observed.

7.4.1 Baseline

As a side experiment, we have attempted to find if the argument can be found *anywhere* as a constituent in the UD parse tree.

Model	Upper bound
Udify	98.9%
Stanza	98.6%

Table 7.1: Dependency parsers upper bound performance.

Table 7.1 shows that in 98.9 and 98.6 of the cases, it is possible to extract the semantic arguments from the syntactic structure by finding an appropriate root of the tree. Thus, the above numbers place a theoretical upper bound on the method, as the accuracy that we could achieve if arguments were always correctly attached to their predicate. This means that the above numbers provide a sanity check for the approach: in 98.9% of the cases, the gold correspond to something which Udify has identified *somewhere* in the sentence.

7.4.2 Extracting predicate-argument relations

In Table 7.2 we report the accuracy for both parsers, with and without the applying the enhancements described in Section 7.3. The

Parser	Plain UD	EUD
Udify	0.683	0.791
Stanza	0.722	0.744

Table 7.2: Accuracy of UD trees with and without enhancements using the Udify and Stanza parsers.

results show a clear superiority for Udify, which is more than 4 percentage points above Stanza in both configurations. Taking into ac-

count enhancement edges gives a large benefit to Udify parser, and a small benefit to Stanza.

To get a better sense of where the errors are coming from, we have performed manual analysis as follows. Focusing on the best performing configuration (Udify with enhanced dependencies), we picked 100 test cases at random, and, by manual inspection, we determined if the error is imputable to either the parser, the dataset or the method. Our classification criteria are as follows:

Parser If the used UD parser produced a wrong parse tree.

Dataset If either the passage or the question is incorrect, either syntactically or semantically; or if the annotations do not contain the answer according to the question and passage.

Method If both the dataset and the parse tree are correct, but the argument is not related to the verb in the UD tree.

We found the following results: out of 100 cases, 49 errors were attributable to the dataset, 13 to the parser and 38 to the method. In terms of percentage points of lost accuracy, this means that 10.2 points are attributable to the dataset, 2.7 points to the parser and 7.9 points to the method. We further analyze error cases below.

7.4.3 Shortcomings of the data set

We found 49 errors imputable to shortcomings in the QA-SRL dataset in our sample. In 20 cases out of those, we found that the annotators chose an answer which is a semantic superset of the answer found in the passage. This situation is illustrated in Section 7.4.3.

(17) An error due to a superset relation between the gold and the retrieved answer

Passage: Placards on the courtyard wall explain it served as headquarters for Field marshal Kollowrat-Krakowsky battling Napoleonic forces in the 1796 Siege of Kehl

Question: Where was someone battling?

Gold: 'Siege of Kehl'

Retrieved: in the 1796 Siege of Kehl

In this example, the correct answer is only "Kehl", as the "siege of" indicates something which happened at "Kehl". Thus, the gold provided by the annotators include the actual gold answer, but provide additional information.

Another issue that arises in the dataset (7 cases in our sample) is incorrect or incomprehensible questions. This is frequently caused by considering a word which is a noun or an adjective in the passage as verb (or part of a verb, e.g. a past participle in a passive verbal form) about which to ask questions. This concerns either homophonous forms or forms that can be formed by using a base form which is a homophone to the word in the passage. An example is shown below:

(18) An error due to changing the POS of a word in the passage

Passage: In 1977 a swamp created by heavy rains was found to contain 8 toxic materials, including 11 suspected cancer-causing chemicals

Question: When was something being swamped?

Gold: 'in 1977'

In this example the noun 'swamp' is turned to a past participle, part of the passive past continuous verbal form "was being swamped".

In the following example the incomprehensibility is caused by plain ungrammaticality:

(19) An error due to an ungrammatical question

> **Passage**: A Texas man was rescued earlier this week after being adrift at sea for 31 hours, according to media reports on Monday
>
> **Question**: Who was something according to?
>
> **Gold**: 'media reports'

Lastly, in 9 cases the actual answer is just not in the provided passage. Despite this problem, annotators did provide a gold answer. The following is such an example:

(20) An example where the answer is not in the passage

> **Passage**: What this entails is a more complex relationship to technology than either techno-optimists or techno-pessimists tend to allow.
>
> **Question**: What isn't being allowed?
>
> **Gold**: 'complex relationship to technology', 'a more complex relationship to technology', more complex relationship to technology'

Here the passage tells us that "techno-optimists" allow do not allow *simple (or less complex) relationships to technology.* However neither the word "less" or "simple" or equivalent are found in the passage. Thus, the gold simply cannot be annotated as a span in the passage, even though annotators did attempt to do so.

Another notable issue is the incorrect identification of a verb occurrence which occurs more than once in the passage (the question is about one occurrence and the answer about another), accounting for two cases in our sample. In another two cases, the syntax of the passage was plainly incorrect, and thus the parser could not recover any useful UD tree.

7.4.4 Shortcomings of the parser

In most cases, parsing errors are attributable to difficulty in handling punctuation (in particular quotes)- and attachment errors.

In 5 out of the 13 parse error cases in our sample, Udify interpreted quotation marks as sentence final markers and terminated the parsing, as in the sentence: *After summarizing his career , Matisse refers to the possibilities the cut-out technique offers , insisting " [...] "* where the parser stops after the first quotation mark.

Another common error (6 cases out of 13) is incorrect attachment. That is, a subtree of the dependency tree is attached to the wrong head, as in: *Churchill was a prolific writer, often under the pen name " Winston S. Churchill " , which he **used** [...]* where "used" is attached to "writer" rather than "name". Of course, in this case, a correct attachment demands a fine understanding of the sentence, so one might wonder if this it reasonable to expect such precision from the parser. Indeed, this is precisely what we intend to estimate by our experiment.

7.4.5 Shortcomings of the method

Seen as a way to test parsers, our method relies on the assumption that predicate-argument relationships are either directly encoded in the UD syntax, or can be directly inferred from it. Thus, conversely, the predicate-argument relationship can serve as a proxy for testing UD parser. Even though the assumption generally holds (not withstanding parsing errors), it sometimes fails. In the rest of the section we analyze the cases when this happens.

Insufficient propagation of arguments The first class of issues is related to the propagation of argument to all the predicates where they apply. This sort of situation accounts roughly for one third of the errors attributable to shortcomings of the method. While EUD

mandates subject control propagation, there are other kinds of argument propagation which can apply.

The first main case occurs when purpose clauses are present. Consider the following passage and question: "Public officials in Texas have urged citizens to receive a flu shot. Who receives something?" Here the answer can be retrieved from a relation between *citizens* and *receive*, but the relationship is not direct: it is mediated by a purpose clause, and this mediation is not identified explicitly in the UD representation.

The second main case involves topicalization of prepositional phrase. The following example illustrates. "In the summer, the glacier melts rapidly, producing a thick deposit of sediment. When is something produced?" In this case the temporal clause is not syntactically attached to *producing*. Rather, it is topicalized and thus attached to the top level node.

Semantic or pragmatic reasoning is necessary In the second class of issues, some sort of semantic and/or pragmatic reasoning is necessary to understand the relationship between arguments and their predicates. The following passage illustrates the problem: "New South Wales premier Mike Baird said people should leave work early and arrive home before dark, as storms were predicted to intensify. Why did someone say something?" Here the cause is not syntactically related to the verb "say". Furthermore, locating the cause cannot be a matter of traversing the syntax tree, using *any* method. Instead, proper identification of the answer relies on the lexical semantics of the passage. We attribute roughly one fourth of the shortcomings of the methods to this class. We stress however that the lines are blurred between various classes of errors. Even though the classification is done according to the best of our judgement it is not easy to make the difference between this case and the previous one.

Anaphora resolution Another cause of errors is the lack of anapho-ra resolution layer in the processing pipeline. For example, the search for syntactic arguments may find the pronoun "it", but the annotators could have resolved the anaphora to a noun phrase (say "the power plant"). This class of errors causes only a tenth of the method shortcomings. This low number may come as a surprise. Its relatively low weight can be explained by two factors: the first one is that annotators are allowed to point to pronouns when identifying arguments. In this case anaphora resolution plays no role. Additionally, each passage is only one sentence long. Therefore, the possibilities for anaphora resolution are limited.

Shortcomings of the parent heuristic When the answer is one of the children, we consider the whole subtree as a candidate answer. When the answer should be looked up in the parent node, we cannot do the same thing: the parent node would contain the whole phrase, which is wrong. For example, when trying to answer "Who observed?" given "The observed animals were tortoises", the parent is "animals", which is the root of the sentence. The heuristic that we apply is to subtract the subtree which contains the verb to obtain the candidate answer. Often, this works well, but in this example we obtain nonsense. This problem accounts roughly for 15 percent of method errors.

Other issues The above list covers roughly 80 percent of errors. The remaining issues include various idiosyncratic interpretations of passages and questions (parataxis, non-deterministic selection of non-specific relative clauses, etc.). Some of them seem as if they could be handled by special rules to identify arguments, but we have preferred not to implement such rules in order to keep the results more directly linked to the syntactic trees which we analyze.

7.5 Suggested improvements to annotation schemes

In this section we leverage our understanding of EUD and QA-SRL, and provide advice to creators of datasets featuring either annotation schema.

7.5.1 EUD

While the main UD format prescribes dependency *trees*, UD also specifies an enhanced format which allows for additional semantically relevant edges to be added (thus obtaining a graph). As Candito et al. (2017) among others note, different tasks seem to require different semantic representations. Thus, our suggestions to the EUD schema focus on how to extract arguments indicated by some question.

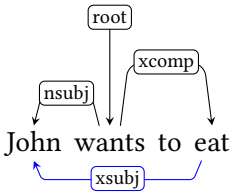
Our analysis shows that EUD is able to model the predicates and arguments in QA-SRL to a high degree (when probed with our fairly straightforward rule-based system) providing an appreciable increase in accuracy compared to plain UD, see Table 7.2. Yet, as far as we understand, the EUD annotation standard is lacking in clarity when it comes to how much semantic relations should be reflected in the structure. The standard reference appears to be the UD website[1], where all enhancements seem to be deducible algorithmically from the plain UD tree. However, as seen in Section 7.4.5, certain predicate-argument relationships are not present in the dependency structure, even after applying the algorithmic enhancements.

We believe that a variant of the EUD scheme with full reflection of predicate-argument structure would be beneficial for many downstream tasks. In the light of our experiment, we propose a number

[1] `https://universaldependencies.org/u/overview/enhanced-syntax.html`

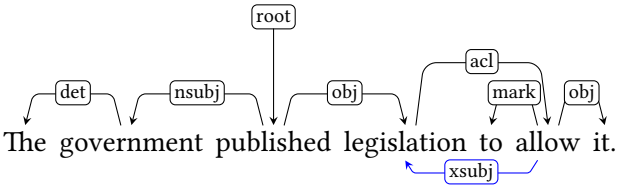
of following arcs to be added, as we list below.

EUD mandates the propagation of subjects through control verbs. As an illustration, consider the sentence "John wants to eat.". The UD tree contains the arcs in black, and EUD mandates to add the blue arc:



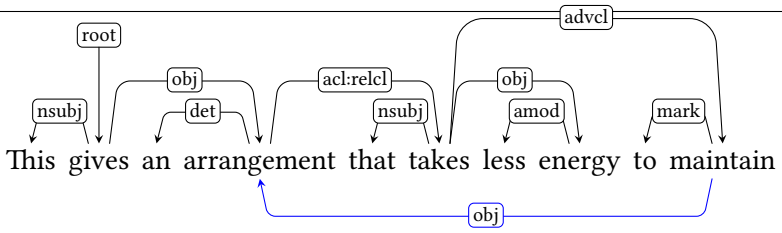
However, we have found that the predicate, the argument and the control verb are not arranged in fixed syntactic patterns, which makes adding the relevant arcs difficult. The main source of difficulties appear to be that the relationship between the argument and predicate can be mediated by a purpose clause.

To illustrate the complexity of the problem, we show two typical examples.



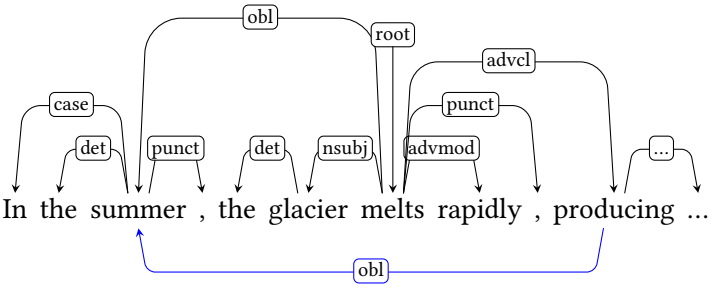
Above, the (semantic) subject of "allow" is "government", which is syntactically a grandfather node of allow. ("Legislation" is another candidate, but it also cannot be identified using a simple syntactic pattern.)

In the example below, we face two difficulties. First, "take" is not a control verb. Second, even though the desired argument of "maintain" (which is "arrangement") can be identified as an argument of "take", this can only be done *via* a relative clause. Third, the roles do not match (a subject becomes an object).



In sum, we purport that, in general, the semantic subject (or object) of a predicate can be found anywhere in the sentence.

Another shortcoming observed is regarding topicalization. Topicalization occurs when a phrase in a sentence is moved to the front of the sentence, to make the phrase more prominent. In the case of prepositional phrases, often indicating semantic roles pertaining to the location, time, or manner in which something happens, is typically expressed with the role obl. However, two verbs may be associated with a prepositional phrase indicating time. Thus, the obl argument should be propagated similarly to how the subject and object roles are propagated in control-like verb construction. An example from the dataset, with out proposed enhancement in blue:



This addition allows for a straightforward interpretation of “when” things happen, by associating both “melts” and “producing” (which is a consequence of “melts”) with the phrase “in the summer”. This allows us to more easily extract the answer to the question: “when was something produced?”.

Finally, anaphoric relationships should be noted as well. This is a well-studied topic which we won't comment further upon, however, we refer readers to the Universal Anaphora project Poesio et al. (1999).

It should be noted that, contrary to the algorithmic transformations of UD trees, some of the above arcs cannot be deduced without a certain amount of semantic understanding of the sentence (in the sense that substituting lexemes by others with the same POS would change the structure). However, this kind of effect is already present when deciding the attachment of constituents, and therefore already affects plain UD.

7.5.2 QA-SRL

We have discovered several possible improvements regarding the QA-SRL data collection. One prevalent source of ambiguity regards the selection of a general or specific phrase, as in Example (7.4.3). A way to remedy this ambiguity in future versions of the QA-SRL datasets is to give annotators more specific instructions for cases like these. A solution that seems to be viable is to instruct annotators to give the most specific answer as this is found in the text, which correctly answers the question. In plain words, this is the longest possible substring that correctly answers the question. In the case of Example (7.4.3), that would be the substring "in the 1796 Siege of Kehl". Note that the relations subset and superset have a more restricted meaning here, as they are bound by the specific syntax found in the passage. As such, the gold and the retrieved answer stand in a subset relation, if the former is a superstring (thus, more specific) of the latter and, vice versa, in a superset relation, if the former is a substring of the latter. An instruction to select the longer string would also lift the ambiguity inherent to selection of non-specific relative clauses. To illustrate, consider the passage-question pair "Matisse's wife Amélie , who suspected that he was having an affair, ended

their 41-year marriage. Who ended something?" For this example the annotators marked 'Amélie' and 'Matisse's wife Amélie' as possible answers, but 'Matisse's wife Amélie , who suspected that he was having an affair' is the longest acceptable string.

To prevent incomprehensible questions (like Section 7.4.3), additional validation tests should be run to safeguard against the formation of ungrammatical questions. One way to do this is to validate at least part of the questions in the dataset using a syntactic acceptability task. This helps identify the ungrammatical questions and replace them with grammatical ones. We observed that annotators tend to make attempts at such meaningless questions as well as questions which do not have an answer in the passage. This is presumably caused by annotators "trying their best", but results in bogus answers. One idea to filter those would be to turn proposed answers into inference problems, as suggested by Demszky et al. (2018). If the constructed problem is not an entailment, then the answer should be rejected. For instance, Example 7.4.3 would be turned into the following problem:

(21) NLI pair for Example (Section 7.4.3

> **Premise**: What this entails is a more complex relationship to technology than either techno-optimists or techno-pessimists tend to allow.
>
> **Hypothesis**: Complex relationship to technology isn't being allowed.

Even though the double-negation complicates reasoning, in this case, one can reasonably expect that the absence of entailment could be detected. This could be done by another round of annotations, perhaps helped by a statistical model which would select doubtful cases.

7.6 Related Work

In addition to our suggestions, there has been several other proposals to extend syntactic dependency trees to more explicitly cover semantic phenomena, including the work of Silveira (2016), already discussed in the introduction. Additionally, Candito et al. (2017) notably propose additions to the EUD schema mainly focusing on extracting the arguments of non-finite verbs and dealing with syntactic alterations in a French treebank. The Universal Decompositional Semantics project (White et al., 2016; Zhang et al., 2017) is another attempt at extending the UD framework to cover semantic phenomena. They develop the Semantic proto-role labeling protocol (SPR1 and SPR2), to find proto-semantic roles by decomposing semantic roles such as "Agent" into more fine-grained properties. Working more generally on dependency trees, Stanovsky et al. (2016) develop a framework to enhance dependency trees such that semantic propositions are more easily recoverable which includes a similar propagation of subjects and objects as in EUD. However they do not appear to take any special note of purpose clauses or topicalization.

7.7 Conclusion and Future Work

We have found that a state-of-the-art UD parser such as Udify only fails to produce a semantically correct UD trees in rare cases. If we exclude difficulties in handling quotes, only 8 cases out of 100 errors are imputable to the parser.

However, in a lot of cases the semantic relationship cannot possibly be present in the UD format, due to its tree structure. To express this, enhancing the structure with additional arcs is needed. Some of those arcs can be found by algorithmic means (as listed in Section 7.3), boosting the accuracy by a several points, see Table 7.2. One could expect that the EUD schema would mandate the addition

of all semantically relevant arcs, but this is not the case. We have advocated for an update to the EUD standard which fills this gap, as discussed in Section 7.5.1.

While the goals of the QA-SRL appear to align perfectly with ours, and the annotation for QA-SRL was both effective and relatively cheap, we notice some shortcomings in the annotations (Section 7.4.3). Sometimes annotators get something wrong because of a tricky phenomena or they are presented with a badly formulated question about the passage. We have proposed a number of strategies to improve data collection for future similar datasets (Section 7.5.2). Another point to consider is that it is much cheaper to annotate QA-SRL than full EUD parse trees. Therefore QA-SRL could be a proxy for training EUD parsers on predicate-argument structures, together with for example multi-task learning. That is, in addition to training a system to predicting arcs, the system would be optimized on selecting the spans of text corresponding to the arguments of predicates.

Chapter 8

Language Modelling with Syntactic and Semantic representations for Acceptability Predictions

Abstract

In this paper, we investigate the effect of enhancing lexical embeddings in LSTM language models (LM) with syntactic and semantic representations. We evaluate the language models using perplexity, and we evaluate the performance of the models on the task of predicting human sentence acceptability judgments. We train LSTM

Published in the article: Adam Ek, Jean-Philippe Bernardy, and Shalom Lappin. 2019. Language Modeling with Syntactic and Semantic Representation for Sentence Acceptability Predictions. In Proceedings of the 22nd Nordic Conference on Computational Linguistics, pages 76–85, Turku, Finland. Linköping University Electronic Press.

language models on sentences automatically annotated with universal syntactic dependency roles (Nivre et al., 2017), dependency tree depth features, and universal semantic tags (Abzianidze, Bos, 2017) to predict sentence acceptability judgments. Our experiments indicate that syntactic depth and tags lower the perplexity compared to a plain LSTM language model, while semantic tags increase the perplexity. Our experiments also show that neither syntactic nor semantic tags improve the performance of LSTM language models on the task of predicting sentence acceptability judgments.

8.1 Introduction

Lau et al. (2014) show that human acceptability judgments are graded rather than binary. It is not entirely obvious what determines sentence acceptability for speakers and listeners. However, syntactic structure and semantic content are clearly central to acceptability judgments. In fact, as Lau et al. (2015, 2017) show, it is possible to use a language model, augmented with a scoring function, to predict acceptability. Standard RNN language models perform fairly well on the sentence acceptability prediction task.

By experimenting with different sorts of enrichments of the training data, one can explore their effect on both the perplexity and the predictive accuracy of the LM. For example, Bernardy et al. (2018) report that including contextual information in training and testing improves the performance of an LSTM LM on the acceptability task, when contextual information is contributed by preceding and following sentences in a document.

Here we report several experiments on the possible contribution of symbolic representations of semantic and syntactic features to the accuracy of LSTM LMs in predicting human sentence acceptability judgments. [1]

[1]Our training and test sets, and the code for generating our

For semantic tags, we use the Universal Semantic Tagging scheme, which provides language independent and fine-grained semantic categories for individual words (Abzianidze et al., 2017). We take our syntactic roles from the Universal Dependency Grammar scheme (Nivre et al., 2016). This allows us to assign to each word in a sentence a semantic and a syntactic role, respectively.

Our working hypothesis is that for a language model the syntactic and semantic annotations will highlight semantic and syntactic patterns observed in the data. Therefore sentences that exhibit these patterns should be more acceptable than sentences which diverge from them. One would expect that if we get lower perplexity for one of the tagging scheme LMs, then its performance would improve on the acceptability prediction task. Clearly, better performance on this task indicates that tagging supplies useful information for predicting acceptability.

8.2 Experimental Setup

First, we train a set of language models, some of them on tag annotated corpora, and some on plain text. While we are interested in the effect of the tags on model perplexity, our main concern is to measure the influence of the tags on an LSTM LM's predictive power in the sentence acceptability task.

We implement four variants of LSTM language models. The first model is a plain LSTM that predicts the next word based on the previous sequence of words. The second, third and fourth models predict next the word w_i conditioned on the previous sequence of words and tags, for which we write $P_M(w_i)$. For a model M that use syntactic or semantic information:

$$P_M(w_i) = P(w_i|(w_{i-1}, t_{i-1}), ..., (w_{i-n}, t_{i-n})) \tag{8.1}$$

LSTM LM models are available at `https://github.com/GU-CLASP/predicting-acceptability`.

We stress that the current tag (t_i) is not given when the model predicts the current word (w_i). Following a previous similar experiment (Bernardy et al., 2018), all language models use a unidirectional LSTM of size 600. We apply a drop-out of 0.4 after the LSTM layer. The models are trained on a vocabulary of 100,000 words. We randomly initialise word embeddings of size 300 dimensions, and tag embeddings of size 30 dimensions. Each model is trained for 10 epochs.

Following the literature on acceptability (Lau et al., 2015, 2017; Bernardy et al., 2018), we predict a judgment by applying a variant of the scoring function SLOR (Pauls, Klein, 2012) to a model's predictions.

8.2.1 SLOR

To estimate sentence acceptability, we use a length-normalized *syntactic log-odds ratio* (hereafter simply referred to as SLOR). We use SLOR rather than any other measurements since it was shown to have the best results in a previous study (Lau et al., 2015). It is calculated by taking the logarithm of the ratio to the probability of the sentence s predicted by a model M (P_M) with the probability predicted by the unigram model (P_U), divided by the length of the sequence $|s|$.

$$SLOR_M(s) = \frac{\log(P_M(s)) - \log(P_U(s))}{|s|} \tag{8.2}$$

where $P_M(s) = \prod_{i=1}^{|s|} P_M(w_i)$ and $P_U(s) = \prod_{i=1}^{|s|}(P_U(w_i))$. This formula takes into account the effect of both word frequency and sentence length on the acceptability score that it assigns to the sentence. SLOR has been found to be a robustly effective scoring function for the acceptability prediction task (Lau et al., 2015).

8.2.2 Model evaluation

We evaluate the model by calculating the Weighted Pearson correlation coefficient between the SLOR score assigned by the model and the judgments assigned by the annotators. Even though we show only the mean judgment in Figure 8.3, each data point comes also with a variance (there is heteroscedasticity). Thus we have chosen to weight the data points with the inverse of the variance when computing the Pearson correlation, as is standard when computing least square regression on heteroscedastic data.

We report the weighted correlation point wise between all models, and between each model and the human judgments. Additionally, we perform three experiments where we shuffle the syntactic and semantic representations in the test sentences. This is done to evaluate if the tags provide useful information for the task.

8.2.3 Language Model Training Data

For training the LMs we selected the English part of the CoNLL 2017 dataset (Nivre et al., 2017). The input sentences were taken from a subset of this corpus. We used only 1/10 of the total CoNNL 2017 Wikipedia corpus, randomly selected. We took out all sentences whose dependency root is not a verb, thus eliminating titles and other non-sentences. We also removed all sentences longer than 30 words. After filtering, the training data contained 87M tokens and 5.3M sentences.

8.3 Semantic Tags

We train a LSTM model for predicting semantic tags. We use this model to tag both the training set extracted from the CoNLL 2017 corpus, and the crowdsource annotated test set (described in Section 6).

The Universal semantic tagging scheme provides fine-grained semantic tags for tokens. It includes 80 different semantic labels. The semantic tags are similar to Part-of-Speech (POS) tags, but they are intended to generalise and to semantically disambiguate POS tags. For many purposes, POS tags do not provide enough information for semantic processing, and this is where semantic tags come into play. A significant element of POS disambiguation consists in assigning proper nouns to semantic classes (named entities). In this way, the scheme also provides a form of named entity recognition. The scheme is designed to be language independent. Annotations currently exist for English, German, Dutch and Italian, but we only use the English labels in our model.

The corpus of semantically tagged sentences that we use comes from the Parallel Meaning Bank (PMB) (Abzianidze et al., 2017). It contains 1.4M tagged tokens divided into 68,177 sentences[2]. The datas-et is extracted from a variety of sources: Tatoeba, News Commentary, Recognizing Textual Entailment (RTE), Sherlock Holmes stories, and the Bible. The sentences are split into gold and silver annotations, where the gold have been manually annotated, and the silver has been annotated by a parser with manual corrections. The silver annotations are mostly correct, but may contain some errors.

Example 22 below is a semantically tagged sentence, taken from the PMB corpus. It includes two pronouns 'he' and 'his'.

(22) He took his book .
PRO EPS HAS CON NIL

Both of these instantiate the same POS, but their semantic classes are distinct. The first is a simple third person pronoun, while the second is a possessive pronoun. Semantic tags are able to handle this distinction, by assigning PRO (pronoun) to the third person pronoun, and HAS (possessive) to the possessive pronoun.

[2]Available for download at https://pmb.let.rug.nl/releases/sem-0.1.0.zip

8.3.1 Semantic Tagging Model

To assign semantic tags to the CoNNL 2017 training corpus and our training set we use a bidirectional LSTM of size 256, with a standard configuration. The model is trained with a batch size of 512 sentences. The word embeddings are of size 256 and are randomly initialized. The model is implemented with `keras` Chollet, others (2015). We stress that this model is separate from the language models used to predict sentence acceptability.

The semantic tagging model is trained for a maximum of 1024 epochs, with early stopping if the validation loss does not improve after 32 epochs. For each epoch, we feed the model 64 batches of 512 randomly selected sentences. For each epoch, the model observes 32,768 sentences (e.g. roughly half of the corpus). To select the best model we left out 1024 gold annotated sentences, randomly selected, and we used them for validation.

Performance The model was validated on 1.5% of the sentences with gold annotations. The remaining data were used for training. This split was chosen because the primary goal of this model is a downstream task, namely tagging data for language modeling. We wish to maximise the number of sentences in the training data. The model finished after 33 epochs, with a final validation loss of 0.317 and a validation accuracy of 91.1%. The performance of our model is similar to that of (Bjerva et al., 2016).

8.4 Syntactic Tags

To introduce syntactic information to our model in an explicit way, we provide it with Universal Dependency Grammar (UD) roles. The UD annotation scheme seeks to develop a unified syntactic annotation system that is language independent (Nivre et al., 2016). UD implements a syntactic annotation through labelled directed graphs,

where each edge represents a *dependency relation*. In total, UD contains 40 different dependency relations (or tags). For example, the sentence *'There is no known cure'* (taken from the CoNLL2017 Wikipedia corpus) is annotated as the dependency graph shown in Figure 1.

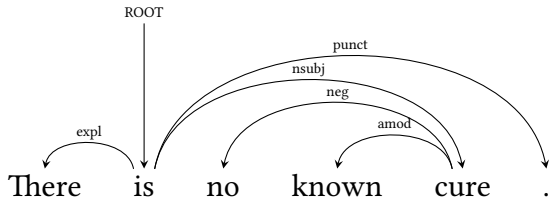


Figure 8.1: Dependency Graph

The model gives the label of the dependency originating from each word, which we call the *syntactic role* of the word. This label is provided as an additional feature for each word *in the input* to our language model. The model does not attempt to predict these roles. For the above sentence, the information given to our syntactic tag trained models would be:

(23) There is no known cure
`expl root neg amod nsubj`

We use the Stanford Dependency Parser to generate syntactic tags for the training and test sets (Chen, Manning, 2014).

8.5 Syntactic Depth

In addition to using syntactic and semantic tags, we also experiment with syntactic depth. To assign a depth to word n, we compute the number of common ancestors in the tree between word n and word $n + 1$. The last word is arbitrarily assigned depth 0. This method

was proposed by Gómez-Rodríguez, Vilares (2018) for constituent trees, but the method works just as well for dependency trees. An example tree is shown below:

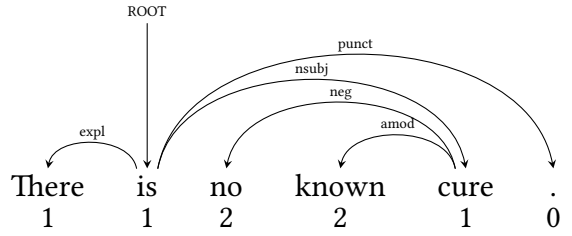


Figure 8.2: Linearized dependency graph

8.6 Test Set

The test set for evaluating our LMs comes from the work of Lau et al. (2015, 2017). 600 sentences were extracted from the BNC corpus (BNC Consortium, 2007) and filtered for length ($8 < |s| < 25$). After this filtering 500 sentences remained and were put through a round-trip machine translation process, from English to Norwegian, Spanish, Chinese or Japanese, and then back to English. In total, the test set contains 2500 sentences: 500 original sentences and 500 from each language used for round-trip translation (i.e. Norwegian, Spanish, Chinese and Japanese). The purpose of using round-trip MT is to introduce a wide variety of infelicities into some of the sentence in our test set. This insures variation in acceptability judgements across the examples of the set.

We used Amazon Mechanical Turk (AMT) crowdsourcing to obtain acceptability judgments. The annotators were asked to rate the sentences based on their naturalness (as opposed to the theoretically committed notion of *well-formedness*) on a scale of 1 to 4. On av-

erage, each sentence had 14 annotators after filtering (for a more detailed description see (Lau et al., 2017)).

The results are shown in Table 8.1. The original sentences, and the sentences that were round-trip translated through Norwegian and Spanish have a higher mean rating than the sentences translated through Japanese and Chinese. The standard deviation is slightly higher for all the sentences which underwent round-trip translation, which is to be expected.

Sentences	Mean	st-dev
en	3.51	0.46
en-no-en	3.13	0.70
en-es-en	3.12	0.69
en-zh-en	2.42	0.72
en-ja-en	2.14	0.74

Table 8.1: Mean judgments and standard deviation for the test set.

	Human	LSTM	+Syn	+Syn*	+Sem	+Sem*	+Depth	+Depth*
Human	1.00							
LSTM	0.58	1.00						
+Syn	0.55	0.96	1.00					
+Syn*	0.39	0.76	0.75	1.00				
+Sem	0.54	0.81	0.78	0.61	1.00			
+Sem*	0.52	0.81	0.78	0.63	0.96	1.00		
+Depth	0.56	0.97	0.97	0.74	0.79	0.79	1.00	
+Depth*	0.46	0.87	0.85	0.73	0.72	0.72	0.86	1.00

Table 8.2: Weighted Pearson correlation between prediction from different models on the SMOG1 dataset. * indicates that the tags have been shuffled.

8.7 Results

Below we denote the plain LSTM LM by LSTM, the LM with syntactic tags as +Syn, the LM with semantic tags as +Sem and the LM with syntactic tree depth as +Depth. We denote the models with shuffled tags by using the star (*) as a modifier.

8.7.1 Language Model Perplexity

We report in Table 8.3 the training loss for the plain-LSTM language model, and for the LSTM language models enhanced with syntactic and semantic tags.

Model	Loss	Accuracy
LSTM	5.04	0.24
+Syn	4.79	0.26
+Sem	5.23	0.21
+Depth	4.88	0.27

Table 8.3: Training loss and accuracy for the language modeling task.

At the end of the training, the language model conditioned on syntactic tags shows the lowest loss. By definition loss is the logarithm of the perplexity. The semantic tag LM exhibits the highest degree of loss. It seems that the syntactic tags reduce LM perplexity, while the semantic tags increase it.

8.7.2 Acceptability Predictions

The matrix in Table 8.2 gives the results for the sentence acceptability prediction task. Each entry r_j^i indicates the weighted Pearson correlation r between $SLOR_i$ and $SLOR_j$. Scatter plots showing

the correlation between human and model predictions are given in Figure 8.3.

The plain LSTM performs close to the level that Bernardy et al. (2018) report for the same type of LM, trained and tested on English Wikipedia data. This indicates the robustness of this model for the sentence acceptability prediction task, given that, unlike the LSTM of Bernardy et al. (2018), it is trained on Wikipedia text, but tested on a BNC test set. Therefore, it sustains a relatively high level of performance on an out of domain test set.

We also tested a model that combined depth markers and syntactic tags, which is, in effect, a full implicit labelled dependency tree model. Interestingly, the correlation (0.54) was lower than the ones achieved by the syntactic tag and depth LSTM LMs individually. None of the enhanced language models increases correlation with human judgments compared to the plain LSTM. Neither does the additional information significantly reduce correlation.

Shuffling the tags causes a drop of 0.16 in correlation for syntactic tags, and a drop of 0.1 for tree depth. Shuffling the semantic tags also lowers the correlation, but only by a small amount (-0.02).

8.8 Discussion

8.8.1 Semantic Tags

As can be observed in Table 8.3, the semantic tags show the highest loss during training. This indicates that semantic tags increase the perplexity of the model, and do not help to predict the next word in a sentence. Despite this, +SEM correlates fairly well with human judgments ($r = 0.54$).

The results obtained with shuffled semantic tags (+Sem*) are revealing. They yield a correlation factor nearly as high as the non-shuffled tags ($r = 0.53$). This suggests that the semantic tags *do*

not provide any useful information for the prediction task. This hypothesis is further confirmed by the high correlation between the non-shuffled and the shuffled semantic tag LMs ($r = 0.96$).

The question of why semantic tags do not reduce perplexity, or why randomly assigned semantic tags are almost as good as non-shuffled tags at predicting acceptability requires further study. One possibility is that the tagging model does not perform as well on the ConLL 2017 Wikipedia subset, or the BNC test set, as it does on the PMB corpus. It may be the case that since the domains are somewhat different, the model is not able to accurately predict tags for our training and test sets. Similarly, we do not know the accuracy of the Stanford Dependency Parser on the BNC test set.

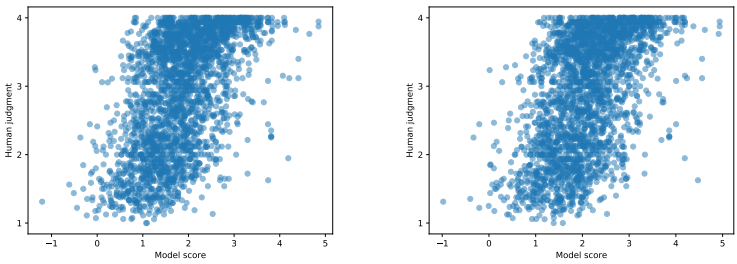
8.8.2 Syntactic Tags

Providing syntactic tags improves the language model, but not the correlation of its predictions with mean human acceptability judgments. However, shuffling the syntactic tags does lower the correlation significantly. This indicates that syntactic tags significantly influence the predictions of the language model.

8.8.3 Tree Depth

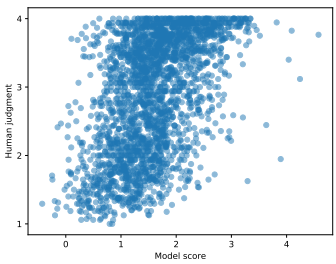
The depth marker enriched LSTM performs best of all the feature enhanced models. Shuffling the markers significantly degrades the accuracy, and it achieves a reduction in perplexity. However, it still performs below the simple LSTM on the acceptability prediction task

It may be the case that the plain LSTM already acquires a significant amount of latent syntactic information, and adding explicit syntactic role labeling does not augment this information in a way that is accessible to LSTM learning. This conclusion is supported by the work of Bernardy, Lappin (2017) on syntactic agreement. They ob-

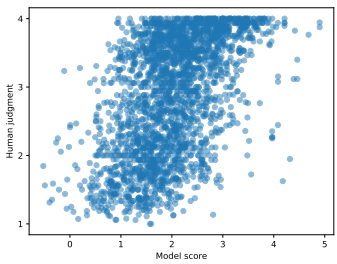


(a) Human - LSTM

(b) Human - Syn



(c) Human - Sem



(d) Human - Depth

Figure 8.3: Scatter plots showing the weighted Pearson correlation between human acceptability judgments (y-axis) and model predictions (x-axis).

serve that replacing a significant portion of the lexicon of an LSTM with POS tags degrades its capacity to predict agreement.

In general, our results do not show that syntactic and semantic information plays no role in the performance of any LM for the acceptability prediction task. It seems clear that the simple LSTM model learns both semantic and syntactic relations among words and phrases, but represents these in a distributed way through the encoding of lexical embeddings in vectors. In fact, there is a body

of work which shows that such LSTMs recognise complex long-distance syntactic relations (Lakretz et al., 2019; Bernardy, Lappin, 2017; Gulordava et al., 2018; Linzen et al., 2016).

8.8.4 Error analysis

We analyse the models in two ways. First, we explore how they score sentences in the test set as categorised by the round-trip translation language that the sentences went through. Second, we look at two example sentences for which no model did particularly well.

8.8.4.1 Model performance on test sentences

To analyse the scores assigned by the model in comparison to the human judgments we first need to normalise the scores. We do this by dividing the score assigned to each sentence by the maximum score assigned. Thus, the relative score of a sentence indicates how close it is to the highest acceptability judgment.

The mean relative score of the human judgments and model scores are presented in Table 8.4. We observe that the models generally appear to assign a lower relative score than humans. But all models also appear to follow the general trend of human judgments and assign a lower score to the Chinese and Japanese round-trip translated sentences compared to the Spanish, Norwegian and original sentences. However, looking at the numbers the difference in magnitude for Chinese and Japanese sentences is rather large. The Chinese and Japanese sentences have a lower relative score of 0.27 and 0.35 respectively. But for models, this difference is only ≈ 0.07 and ≈ 0.12 respectively. This indicates that while the models are able to see some acceptability differences between the subclasses of test sentences, the models do not penalize these sentences as much as humans.

Model	En	No	Es	Zh	Ja
Human	0.88	0.78	0.78	0.61	0.53
LSTM	0.41	0.40	0.40	0.34	0.29
+Syn	0.46	0.44	0.45	0.39	0.35
+Sem	0.39	0.36	0.37	0.30	0.28
+Depth	0.45	0.43	0.44	0.38	0.34

Table 8.4: Comparison of the average relative score assigned by the models and humans for the different sentences in the test set.

We also note that the models consistently assign much lower relative scores than the human annotators do to most of the sentences. This, biases their scores in favour of the Chinese and Japanese target sentences, since these are typically 'worse' than their original English sources, or the Norwegian and Spanish targets, according to the human judges (see Table 8.1).

As an additional analysis of the models we compare the worst scoring sentences between the models. This was done by splitting the predictions into two sets: (a) model scores above the average[3] and (b) model scores below the average. We sort these sets by their difference to the humans and select the top 20 sentences for each model. Table 8.5 shows the intersection of sentence sets for the different models.

Model	LSTM	+Syn	+Sem	+Depth
LSTM	40			
+Syn	30	40		
+Sem	19	15	40	
+Depth	30	28	17	40

Table 8.5: Shared erroneous sentences between the models.

[3]We compare scores by dividing each score by it's maximum value, as described previously.

We observe that the syntactic tag and depth models share many sentences with each other, and with the plain LSTM, but not as many with the semantic model. This shows that the difficult sentences for the semantic model are different than those for the syntactic and plain models.

8.8.4.2 Model and human performance

We use the relative scores from the previous section to select sentences for examination. We look at two types of cases, one in which the model predicts a higher score than the human judgments, and the other where the model predicts a lower score than human judgments. For both cases we select a sentence at random.

We begin by considering an example to which the model assigns a higher score than humans do. The sentence went through Chinese:

(24) '1.5% Hispanic or Latino of any race population.'

The sentence lacks a verb, and the modifier-noun construction 'race population' is lexically strange. It is interesting to note that our syntactic models (+Syn and +Depth) both assign a high score to this sentence, while the semantic and plain LM assign a lower score (which is closer to the human judgment). We would think that the model using syntactic tags would pick up on the missing verb, and so penalize the sentence. The scores for the sentence (24) are shown in Table 8.6:

Model	Relative	Absolute
Human	0.40	1.62
LSTM	0.77	3.74
+Syn	0.90	4.47
+Sem	0.71	3.29
+Depth	0.85	4.17

Table 8.6: Human judgments and model scores for sentence (24).

For (1), the LM enhanced with semantic tags gave the sentence the lowest score. The syntactic and depth model gave the sentence a high score (0.90 and 0.85 respectively). This indicates that while still assigning the sentence a relatively high score, the semantic and plain LM rate the sentence closer to humans than the syntactical LM.

In the second case (25), the sentence is one of the original English sentences:

(25) 'ACS makes a special "FAT" heavy duty BMX freewheel in 14T and 16T with 3/16 "teeth compatible only with 3/16" chains.'

The human annotators gave it an appropriately high score, but the models did not, as indicated in table Table 8.7.

Model	Relative	Absolute
Human	0.80	3.23
LSTM	-0.007	-0.03
+Syn	0.002	0.01
+Sem	0.26	1.20
+Depth	0.02	-0.01

Table 8.7: Human judgments and model scores for sentence (25).

Again, we can see that the LM enhanced with semantic tags performed the best (i.e. assigned the sentence the highest score). The sentence has a few features which might make it difficult for the standard LM and syntactically enhanced language models. The sentence contains a high number of quotations, acronyms (e.g. ACS) and specialized terms (e.g. 3/16). The dependency tags do not treat these words in any special way. Because the words are rare they are not likely candidates. The semantic tags will treat these words in a different manner, since it contains tags for named entities and quantities.

8.8.5 Pre-Trained Language Models

Recently several large pre-trained language models using transformation architecture, like BERT (Devlin et al., 2019), or bidirectional LSTM with attention, such as ELMo (Peters et al., 2018a), have achieved state of the art results across a variety of NLP tasks. We opted not to experiment with any of these pre-trained language models for our task. The LSTM architecture of our LMs is far simpler, which facilitates testing the contribution of explicit feature representation to correlation in the acceptability prediction task, and perplexity for the language modeling task.

8.9 Related Work

There has been a considerable amount of work showing that encoding tree representations in deep neural networks, particularly LSTMs, improves their performance on semantic relatedness tasks. So, for example, Tai et al. (2015) show that Tree-LSTMs outperform simple LSTMs on SemEval 2014 Task 1, and sentiment classification. Similarly, Gupta, Zhang (2018) argue that by adding progressive attention to a Tree-LSTM it is possible to improve its performance on several semantic relatedness tasks.

Williams et al. (2018a) describes a number of experiments with latent tree learning RNNs. These models learn tree structures implicitly, rather than through training on a parse annotated corpus. They construct their own parses. Williams et al. (2018a) state that they outperform Tree-LSTM and other DNN models on semantic relatedness applications, and the Stanford Natural Language Inference task. Interestingly, the parse trees that they construct are not consistent across sentences, and they do not resemble the structures posited in formal syntactic or semantic theories. This result is consistent with our finding that LSTMs learn syntactic and semantic patterns in a way that is quite distinct from the classifications posited in classical

grammatical and semantic systems of representation.

Finally, Warstadt, Bowman (2019) discuss the performance of several pre-trained transformer models on classifying sentences in their Corpus of Linguistic Acceptability (CoLA) as acceptable or not. These models exhibit levels of accuracy that vary widely relative to the types of syntactic and morphological pattern that appears in CoLA.

It is important to recognise that CoLA is a very different sort of test set from the one that we use in our experiments. It is drawn from linguists examples intended to illustrate particular sorts of syntactic construction. It is annotated for binary classification according to linguists' judgments. By contrast, our BNC test set consists of naturally occurring text, where a wide range of infelicities are introduced into many of the sentences through round trip machine translation. It is annotated through AMT crowd sourcing with gradient acceptability judgments. Given these significant differences in design and annotation between the two test sets, applying our models to CoLA would have taken us beyond the scope of the sentence acceptability task, as specified in Lau et al. (2015, 2017); Bernardy et al. (2018),

Moreover, our experiments are not focused on identifying the best performing model as such. Instead, we are interested in ascertaining whether enriching the training and test data with explicit syntactic and semantic classifier representations contributes to LSTM learning for the sentence acceptability prediction task.

8.10 Conclusions

We present experiments that explore the effect of enhancing language models with syntactic and semantic tags, and dependency tree depth markers, for the task of predicting human sentence acceptability judgments. The experiments show that neither syntactic nor semantic tags, nor tree depth indicators improve the correlation

between an LSTM LM and human judgments. Our experiments also show that syntactic tags provide information that is useful for language modeling, while semantic tags do not. However, further experiments are needed to verify our results for semantic tags. The model that we used for tagging, rather than the information in the tags themselves, may be responsible for the observed result.

Surprisingly our initial hypothesis that lower training perplexity produces better acceptability prediction has been overturned. We have not observed any correlation between the perplexity of an LM and its accuracy in acceptability prediction. The SLOR scoring function may disrupt this connection.

Our tentative conclusion from these experiments is that simple LSTMs already learn syntactic and semantic properties of sentences through lexical embeddings only, which they represent in a distributional manner. Introducing explicit semantic and syntactic role classifiers does not improve their capacity to predict the acceptability of sentences, although such information may be useful in boosting the performance of deep neural networks on other tasks.

In future work, we plan to test other sources of information for the language models. One possibility is to use constituency, rather than dependency, tree depth. We will also plan to experiment with different combinations of tags for the language models, for example semantic and syntactic roles.

Chapter 9

How does Punctuation affect Neural Models in Natural Language Inference

Abstract

Natural Language Inference models have reached almost human-level performance but their generalisation capabilities have not been yet fully characterized. In particular, sensitivity to small changes in the data is a current area of investigation. In this paper, we focus on the effect of punctuation on such models. Our findings can be broadly summarized as follows: (1) irrelevant changes in punctuation are correctly ignored by the recent transformer models (BERT)

Published in the article: Adam Ek, Jean-Philippe Bernardy, and Stergios Chatzikyriakidis. 2020. How does Punctuation Affect Neural Models in Natural Language Inference. In Proceedings of the Probability and Meaning Conference (PaM 2020), pages 109–116, Gothenburg. Association for Computational Linguistics.

while older RNN-based models were sensitive to them. (2) All models, both transformers and RNN-based models, are incapable of taking into account small relevant changes in the punctuation.

9.1 Introduction

In recent years models for Natural Language Inference (NLI) have reached almost human-level performance. These models frame inference as a classification problem, whose input is a premise/hypothesis pair. It has been noted that small changes in the pair, can flip the prediction (Glockner et al., 2018). In this paper, we explore the effect of punctuation[1] in neural models in natural language inference.

Small changes in a premise/hypothesis pair are of two kinds. First, the change can be of an irrelevant kind. For example, we can expect that removing a sentence-final stop should not change the meaning of a (final) sentence. Second, a textually small change could flip the relationship between hypothesis and premise. For example, adding a negation word is a small textual change that has a lot of semantic content. But it is not only words that can have a large impact on the meaning of a sentence. Commas, for example, may indicate which words belong together and which do not in a list. Ideally, an NLI model should be insensitive to changes of the first kind, but still, properly recognize changes of the second kind.

In this paper, we test both hypotheses for the case of punctuation. Namely:

- (H1) Deep-learning based classifiers are sensitive to irrelevant punctuation.
- (H2) Deep-learning classifiers take relevant punctuation into account correctly.

[1] The set of punctuation symbols we consider are: `'!"#$%&()*+,-./:;<=>?@[]\^_`{}| '`

This work is part of the larger question concerning the ability of NLI models to generalize. There are a number of papers that report several problems of generalizability: Glockner et al. (2018) have shown that several NLI models break considerably easily when, instead of tested on the original SNLI (Bowman et al., 2015) test set, they are tested on a test set which is constructed by taking premises from the training set and creating several hypotheses from them by changing at most one word within the premise. Talman, Chatzikyriakidis (2019) show that NLI models break down when one trains in one dataset, but then test on the test set of a similar dataset (e.g. training on MNLI (Williams et al., 2018b) and testing on SNLI). Wang et al. (2019b) report problems in generalizability when the two pairs are swapped. The idea is that one should expect the same accuracy for contradiction and neutral when the pairs are swapped (neutral remains neutral, and contradiction remains a contradiction[2]), and a lower accuracy for entailment (given that entailment turns neutral when the pairs are swapped).

9.2 Datasets and experiments

Our experiments are performed on the Multi-Genre Natural Language Inference (MNLI) corpus (Williams et al., 2018b) (and variants thereof, as described below). MNLI consists of 433k human-written sentence pairs labeled with entailment, contradiction and neutral. MNLI contains sentence pairs from ten distinct genres[3] of both written and spoken English. Only five genres are included in the training set. The development and test sets have been divided into matched and mismatched, where the former includes only sentences from the

[2]Even though one can imagine exotic, non-symmetric definitions of "neutral" and "contradiction", we are not aware of any syste or dataset using such a definition.

[3]face to face conversations, telephone ones, letters, oxford university press publications, etc.

same genres as the training data and the latter include sentences from the remaining genres not present in the training data.

We consider three variants of MNLI:

($orig$) This variant is the original MNLI with no changes whatsoever.

(p) To obtain this variant we make punctuation consistent throughout examples by adding full stops at the end of each sentence.

($\neg p$) To obtain this variant we remove all non-alphanumeric characters from each sentence. This also remove special characters that are sometimes not classified as punctuation, such as the dollar sign. However, such characters occur so seldom that they have little influence on the results, either way (see Table 9.1).

Appending a sentence-final stop is in general reasonable, especially for the non-dialogue examples. For the dialogue part of the MNLI dataset, this is unnatural as final stops typically are not expressed in dialogue.

To convey an idea of the amount of data that our transformation impact, we show the raw and relative count[4] of punctuation symbols in Table 9.1. In total, relative to word-tokens, punctuation symbols account for about 11.5% of the tokens.

9.2.1 Experiments

We perform two sets of experiments. In the first set, designed to test (H1), we train NLI models for either of the three ($orig$, p, $\neg p$) variants and test on either the p or $\neg p$ variants. Additionally, we train on $orig$ and test on $orig$, as a baseline result.

[4]Relative to the number of total tokens in the MNLI dataset

Symbol	Count	%	Symbol	Count	%
,	672354	3.544	]	1872	0.010
.	632460	3.334	&	1032	0.005
’	426014	2.246	%	1014	0.005
-	188124	0.992	_	666	0.003
)	66498	0.351	*	186	0.001
(	66210	0.349	@	162	0.001
?	41530	0.219	=	150	0.001
”	27246	0.144	#	114	0.001
;	18182	0.096	+	66	0.0003
!	11384	0.060	‘	24	0.0001
$	8724	0.046	~	12	6.32e-05
:	6162	0.033	\	12	6.32e-05
/	5746	0.030	{	12	6.32e-05
[	1920	0.010			

Table 9.1: Count of punctuation symbols used in the training examples of MNLI.

In the second set, we designed a dataset to test (H2), that is, whether NLI models are able to detect semantically relevant punctuation. This experiment is performed the same way as the first set, but we replace the MNLI test data with our own dataset. The dataset we constructed for this contain a number of problems whose correct label depends on the presence or absence of punctuation. Here are some representative examples (& separates the premise from the hypothesis, label follows in parentheses):

(26) I thank, my mother, Anna, Smith and John & I thank four people (E)

(27) I thank, my mother Anna, Smith and John & I thank two people (C)

(28) The notion of good, god, is incomprehensible & Good is incomprehensible (E)

(29) The notion of good, god, is incomprehensible & Good god is incomprehensible (C)

The first two examples are cases where the commas are used to denote the conjunction of more than one conjunct. Removing the comma between "my mother" and "Anna" in (27) has a significant effect on counting: what is taken to be two entities in (26), are one in (27). In (28) and (29), we get a different label depending on whether the hypothesis refers to the property "good" (E) or the adjectival modification "good god" (C). The test set consists of 18 examples which can be seen in Table 9.4.

9.3 Models

The experiments are performed using three models:

BiLSTM The simplest model is a bidirectional LSTM that encodes the premise and hypothesis, then applies max pooling. The model then concatenates the premise and hypothesis in the standard fashion (Conneau et al., 2017; Talman et al., 2019): $[p; h; p-h; p*h]$ where p is the premise representation and h the hypothesis representation. A three-layer perceptron with leaky ReLU activation between the layers then assigns a class to the example.

HBMP The second model is described by Talman et al. (2019). The model is a three-layer bidirectional LSTM, wherein between the layers a representation is extracted through max pooling. The final representation for each sentence is the concatenation of all intermediate representations $[h_0; h_1, h_2]$. The same representation as with the BiLSTM, $[p; h; p - h; p * h]$ where p and h respectively is the

concatenation of all intermediate representations, is then passed to a three-layer perceptron with leaky ReLU activation and dropout.

BERT Our third model is a transformer model, BERT (Devlin et al., 2019). We use the BERT base model from the transformer library (Wolf et al., 2020). To train BERT we use a three layer perceptron with Leaky ReLU activations on top of the BERT model and fine-tune. The BERT model process the premise and hypothesis is parallel and there is no need to explicitly combine them as with the previous models. For the classification of a sentence pair, we use the CLS token generated by BERT that contain information about both sentences.

9.4 Experimental setup

For each architecture (BERT, HBMP, and BiLSTM) we perform experiments by training four models, two trained and validated on the dataset with punctuation and two models trained and validated on the dataset without punctuation. To asses the effect of our data augmentation we test the model on the other dataset, i.e. a model trained and validated without punctuation is tested on the dataset with punctuation. We measure the performance in terms of accuracy.

For HBMP and the BiLSTM models we use the default hyperparameters reported by Talman et al. (2019) with GloVe (Pennington et al., 2014) word embeddings[5]. The BERT model is fine-tuned with the default model hyperparameters. We use the Adam optimizer with a learning rate of 0.00002 and a batch size of 24.

[5]Trained on 840 billion tokens.

9.5 Results

9.5.1 First experiment set

The results from the first experiment are shown below in Table 9.2. The experiment shows the accuracy for the models trained on the MNLI variations with and without punctuation and their accuracy on all variations.

Model	Test	MA	MM
BiLSTM$_{orig}$		.724	.723
BiLSTM$_{p}$	p	.723	.724
BiLSTM$_{p}$	$\neg p$	.428	.414
BiLSTM$_{\neg p}$	$\neg p$	.714	.727
BiLSTM$_{\neg p}$	p	.424	.430
HBMP$_{orig}$		.729	.733
HBMP$_{p}$	p	.728	.729
HBMP$_{p}$	$\neg p$	.430	.408
HBMP$_{\neg p}$	$\neg p$	.729	.732
HBMP$_{\neg p}$	p	.436	.427
BERT$_{orig}$		.833	.839
BERT$_{p}$	p	.835	.837
BERT$_{p}$	$\neg p$	.816	.822
BERT$_{\neg p}$	$\neg p$	.819	.820
BERT$_{\neg p}$	p	.830	.833

Table 9.2: The effect on punctuation on all three models in terms of accuracy of the MNLI dataset. MA indicate the matched and MM the mismatched test split. $original$ is trained on the unaugmented data, p models trained with punctuation and $\neg p$ models trained without punctuation

The results indicate that when the RNN-based models are tested on the same dataset as it is trained on, the results are similar to

that of the original model. However, when we test on the opposite dataset the performance drops drastically (about 30 percentage points). We see that the drop in accuracy is about the same for both the matched and mismatched test set. In contrast to the RNN-based models, the transformer model only shows a slight drop in accuracy when presented with test data different from its training data.

9.5.2 Second experiment set

Full results from the second experiment can be found in Table 9.4, a subset of the examples can be found in Table 9.3. The experiment shows the predictions by the HBMP and BERT models trained with and without punctuation on our hand-crafted dataset.

9.5.3 Experiment one analysis

The experiment shows that the BLSTM and HBMP models trained with punctuation drops significantly in accuracy when tested on data without punctuation. This indicates that when removing punctuation the model changes its prediction incorrectly. Most of the removed punctuation does not change the meaning, rather some information irrelevant the the relationship between the two sentences (such as sentence-final stop).

Inspecting the output of the HBMP model we can see that in many cases, removing a sentence-final stop flips the models' prediction. In example (30) and (31), both the model trained on punctuation and the one without fail to predict that the final stop does not add any meaning.

In examples (32) and (33)[6], the sentence-final stop has been removed, as well as a comma. In such a case, the comma does not add any meaning but acts as a separator of clauses. The removal or addition of this comma flips the prediction of the models. This shows

[6]For clarity, the premise is indicated by P and the hypothesis by H.

that irrelevant changes both involving commas and sentence-final stops can flip the model's prediction without any semantic motivation.

n	Premise	Hypothesis	Gold	Pred	Model
0	I thank, my mother, Anna, Smith and John	I thank four people	E	N	HBMP$_{\neg p}$
1	I thank, my mother, Anna Smith and John	I thank three people	E	N	HBMP$_{\neg p}$
8	I hear John says 'come here'	I hear John speaking	E	E	HBMP$_{\neg p}$
9	I hear 'John says come here'	I hear John speaking	C	N	HBMP$_{\neg p}$
14	No, god is good	God is good	E	E	HBMP$_{\neg p}$
15	No god is good	There is no good god	E	E	HBMP$_{\neg p}$
16	No, god is good	There is no good god	C	E	HBMP$_{\neg p}$
17	No god is good	God is good	C	C	HBMP$_{\neg p}$
0	I thank, my mother, Anna, Smith and John	I thank four people	E	E	HBMP$_{p}$
1	I thank, my mother, Anna Smith and John	I thank three people	E	E	HBMP$_{p}$
8	I hear John says 'come here'	I hear John speaking	E	E	HBMP$_{p}$
9	I hear 'John says come here'	I hear John speaking	C	E	HBMP$_{p}$
14	No, god is good	God is good	E	E	HBMP$_{p}$
15	No god is good	There is no good god	E	E	HBMP$_{p}$
16	No, god is good	There is no good god	C	E	HBMP$_{p}$
17	No god is good	God is good	C	C	HBMP$_{p}$
0	I thank, my mother, Anna, Smith and John	I thank four people	E	E	BERT$_{\neg p}$
1	I thank, my mother, Anna Smith and John	I thank three people	E	C	BERT$_{\neg p}$
8	I hear John says 'come here'	I hear John speaking	E	C	BERT$_{\neg p}$
9	I hear 'John says come here'	I hear John speaking	C	E	BERT$_{\neg p}$
14	No, god is good	God is good	E	E	BERT$_{\neg p}$
15	No god is good	There is no good god	E	E	BERT$_{\neg p}$
16	No, god is good	There is no good god	C	E	BERT$_{\neg p}$
17	No god is good	God is good	C	E	BERT$_{\neg p}$
0	I thank, my mother, Anna, Smith and John	I thank four people	E	E	BERT$_{p}$
1	I thank, my mother, Anna Smith and John	I thank three people	E	C	BERT$_{p}$
8	I hear John says 'come here'	I hear John speaking	E	C	BERT$_{p}$
9	I hear 'John says come here'	I hear John speaking	C	E	BERT$_{p}$
14	No, god is good	God is good	E	E	BERT$_{p}$
15	No god is good	There is no good god	E	E	BERT$_{p}$
16	No, god is good	There is no good god	C	E	BERT$_{p}$
17	No god is good	God is good	C	E	BERT$_{p}$

Table 9.3: Results on a subset of the examples in our constructed dataset. E is entailment, N is neutral and C is contradiction. The model column indicate which HBMP model configuration was used (trained with punctuation p, or without $\neg p$).

(30) not yourself . & only you . (C)
HBMP$_{p}$ = C
HBMP$_{\neg p}$ = E
BERT$_{p}$ = N
BERT$_{\neg p}$ = N

(31) not yourself & only you (C)
HBMP$_{p}$ = E

$\text{HBMP}_{\neg p}$ = C
BERT_{p} = N
$\text{BERT}_{\neg p}$ = N

(32) P = so they set about clearing the land for agriculture , setting fire to massive tracts of forest .
H = as a result , the land was devastated by erosion . (N)

HBMP_{p} = N
$\text{HBMP}_{\neg p}$ = C
BERT_{p} = N
$\text{BERT}_{\neg p}$ = N

(33) P = so they set about clearing the land for agriculture setting fire to massive tracts of forest
H = as a result the land was devastated by erosion (N)
HBMP_{p} = C
$\text{HBMP}_{\neg p}$ = N
BERT_{p} = E
$\text{BERT}_{\neg p}$ = C

BERT assigns the neutral class regardless of punctuation in examples (30) to (32), indicating that the choice of punctuation in training and test does not impact its decision. For example (8) there is no punctuation in the premise and hypothesis, but the different BERT models assign two different classes, *entailment* by the model trained on punctuation and *contradiction* by the model trained without punctuation.

A possible explanation for why the accuracy of BERT does not behave similarly to that of the LSTM based models is that the pre-training of BERT allows the model to better ignore variations in the input. However, the HBMP model also uses pre-trained information in the form of GLoVE vectors, yet we do not see HBMP handling

the discrepancy between the training and the test well. Albeit the pre-training of GLoVE and BERT are different, in the essence they are the same. Both model the meaning of words based on their surroundings in the neural architecture. Thus, the relevant difference between the models relevant to the absence or presence of punctuation is whether the model use self-attention or an LSTM to create representations of sentences. From this, we pose a tentative hypothesis that self-attention more easily learn to ignore irrelevant input tokens for a task than the LSTM. However, to confirm this we need to perform more expensive experiments.

9.5.4 Experiment two analysis

None of the models perform very well for this dataset. The HBMP$_p$ model has an accuracy of 61.1% while the HBMP$_{\neg p}$ has an accuracy of 44.4%. The BERT$_p$ model has an accuracy of 44.4% while the BERT$_{\neg p}$ has an accuracy of 38.8%.

For example, both models are tricked by comma removal in (27). An interesting case involves cases where the comma is removed from “No, god” turning it into a negative quantifier “no god”. The models are tricked when asked to infer “There is no good god” from “No, god is good” (they predict E instead of C). Another example where the models are tricked by comma removal is when listing items. In the example ”I thank, my mother, Anna Smith and John” there are three entities being thanked. The comma placement indicates that ”Anna Smith” is one person, and not two. Only HBMP$_p$ successfully predicts that ”I thank three people” is an entailment for this example. The quotation examples are also challenging. Both systems are tricked when they are asked to judge whether “I hear John speaking” follows: a) from “I hear John says ‘come here’ ”, and b) “I hear ‘John says come here’ ”. Both models correctly predict a) but fail on b). However, they give a different wrong label, (N) for HBMP$_{\neg p}$ and (E) for HBMP$_p$.

9.6 Conclusions

The conclusions of this paper can be summarized as follows: Only BERT is robust to irrelevant changes in punctuation (H1 is validated for BERT). The other models see a significant drop in performance when for any mismatch of the presence of punctuation between training and testing sets. However, the presence or absence of the full stop at the end of a sentence has little effect. This statement rests on the observation that punctuation is generally semantically insignificant in MNLI. This fact has not been tested using a model but rather relies on manual inspection of the data.

We have evidence that no model is capable of taking into account cases where punctuation is meaningful. At this stage of our research, this evidence does not rely on a large body of data. This result is not surprising because of the above observation (namely, there is not enough meaningful punctuation in the training set). Yet, we use pre-trained embeddings (BERT) which have been trained on very large dataset, and it could not be ruled out *a priori* that such embeddings did not contain information related to the meaning of punctuation.

As a general remark, it seems to us useful, if not necessary, to extend the present datasets for NLI to include examples where punctuation is actually meaningful. In general, this is part of a discussion of extending current datasets to include cases of inference where more fined-grained phenomena are taken into consideration Chatzikyriakidis et al. (2017); Bernardy, Chatzikyriakidis (2019, 2020). This also connects with the generalization capabilities of NLI models that were briefly brought up in the introduction. However, the goal should not only be to create many diverse datasets that can get very fine-grained for numerous syntactic phenomena. What we further need are models that will have the ability to generalize well to new data after they have been trained on datasets that represent a much more diverse and rich picture of NLI, and are not prone to similar problems as these have been reported in the literature (Glockner

et al., 2018; Talman, Chatzikyriakidis, 2019; Wang et al., 2019b; Poliak et al., 2018).

9.7 Future work

In future work, we plan to continue pursuing the question of model generalizability by investigating how neural models for natural language inference can be adapted to take into account fine-grained semantic phenomena. More specifically, how can models be adapted to learn what constitutes a meaningful part of a sentence, in terms of semantics, and what is not meaningful. We can notice that the phenomena of punctuation is primarily ”syntactic sugar”, by constructing a sentence in a certain way syntactically (by inserting or removing punctuation). To exploit this we plan to incorporate syntactic representations of sentences.

Appendix

n	Premise	Hypothesis	Gold	Pred	Model
0	I thank, my mother, Anna, Smith and John	I thank four people	E	N	HBMP$\neg p$
1	I thank, my mother, Anna Smith and John	I thank three people	E	N	HBMP$\neg p$
2	I thank, my mother Anna, Smith and John	I thank two people	C	E	HBMP$\neg p$
3	I thank, my mother Anna Smith and John	I thank three people	C	E	HBMP$\neg p$
4	I thank, my mother Anna, Smith and John	I thank more than two people	E	N	HBMP$\neg p$
5	I thank my mother Anna, Smith and John	My mother is called Anna Smith	N	N	HBMP$\neg p$
6	I thank my mother, Anna Smith and John	My mother is called Anna Smith	N	N	HBMP$\neg p$
7	I thank my mother Anna Smith and John	My mother is called Anna Smith	E	E	HBMP$\neg p$
8	I hear John says 'come here'	I hear John speaking	E	E	HBMP$\neg p$
9	I hear 'John says come here'	I hear John speaking	C	N	HBMP$\neg p$
10	The notion of good, god, is incomprehensible	Good is incomprehensible	E	N	HBMP$\neg p$
11	The notion of good god is incomprehensible	Good god is incomprehensible	E	E	HBMP$\neg p$
12	The notion of good, god, is incomprehensible	Good god is incomprehensible	C	E	HBMP$\neg p$
13	The notion of good god is incomprehensible	Good is incomprehensible	N	C	HBMP$\neg p$
14	No, god is good	God is good	E	E	HBMP$\neg p$
15	No god is good	There is no good god	E	E	HBMP$\neg p$
16	No, god is good	There is no good god	C	E	HBMP$\neg p$
17	No god is good	God is good	C	C	HBMP$\neg p$
0	I thank, my mother, Anna, Smith and John	I thank four people	E	E	HBMPp
1	I thank, my mother, Anna Smith and John	I thank three people	E	E	HBMPp
2	I thank, my mother Anna, Smith and John	I thank two people	C	E	HBMPp
3	I thank, my mother Anna Smith and John	I thank three people	C	E	HBMPp
4	I thank, my mother Anna, Smith and John	I thank more than two people	E	N	HBMPp
5	I thank my mother Anna, Smith and John	My mother is called Anna Smith	N	N	HBMPp
6	I thank my mother, Anna Smith and John	My mother is called Anna Smith	N	N	HBMPp
7	I thank my mother Anna Smith and John	My mother is called Anna Smith	E	E	HBMPp
8	I hear John says 'come here'	I hear John speaking	E	E	HBMPp
9	I hear 'John says come here'	I hear John speaking	C	E	HBMPp
10	The notion of good, god, is incomprehensible	Good is incomprehensible	E	E	HBMPp
11	The notion of good god is incomprehensible	Good god is incomprehensible	E	E	HBMPp
12	The notion of good, god, is incomprehensible	Good god is incomprehensible	C	E	HBMPp
13	The notion of good god is incomprehensible	Good is incomprehensible	N	C	HBMPp
14	No, god is good	God is good	E	E	HBMPp
15	No god is good	There is no good god	E	E	HBMPp
16	No, god is good	There is no good god	C	E	HBMPp
17	No god is good	God is good	C	C	HBMPp
0	I thank, my mother, Anna, Smith and John	I thank four people	E	E	BERT$\neg p$
1	I thank, my mother, Anna Smith and John	I thank three people	E	C	BERT$\neg p$
2	I thank, my mother Anna, Smith and John	I thank two people	C	E	BERT$\neg p$
3	I thank, my mother Anna Smith and John	I thank three people	C	E	BERT$\neg p$
4	I thank, my mother Anna, Smith and John	I thank more than two people	E	E	BERT$\neg p$
5	I thank my mother Anna, Smith and John	My mother is called Anna Smith	N	E	BERT$\neg p$
6	I thank my mother, Anna Smith and John	My mother is called Anna Smith	N	E	BERT$\neg p$
7	I thank my mother Anna Smith and John	My mother is called Anna Smith	E	E	BERT$\neg p$
8	I hear John says 'come here'	I hear John speaking	E	C	BERT$\neg p$
9	I hear 'John says come here'	I hear John speaking	C	E	BERT$\neg p$
10	The notion of good, god, is incomprehensible	Good is incomprehensible	E	E	BERT$\neg p$
11	The notion of good god is incomprehensible	Good god is incomprehensible	E	E	BERT$\neg p$
12	The notion of good, god, is incomprehensible	Good god is incomprehensible	C	E	BERT$\neg p$
13	The notion of good god is incomprehensible	Good is incomprehensible	N	E	BERT$\neg p$
14	No, god is good	God is good	E	E	BERT$\neg p$
15	No god is good	There is no good god	E	E	BERT$\neg p$
16	No, god is good	There is no good god	C	E	BERT$\neg p$
17	No god is good	God is good	C	E	BERT$\neg p$

0	I thank, my mother, Anna, Smith and John	I thank four people	E	E	BERTp
1	I thank, my mother, Anna Smith and John	I thank three people	E	C	BERTp
2	I thank, my mother Anna, Smith and John	I thank two people	C	E	BERTp
3	I thank, my mother Anna Smith and John	I thank three people	C	E	BERTp
4	I thank, my mother Anna, Smith and John	I thank more than two people	E	E	BERTp
5	I thank my mother Anna, Smith and John	My mother is called Anna Smith	N	E	BERTp
6	I thank my mother, Anna Smith and John	My mother is called Anna Smith	N	E	BERTp
7	I thank my mother Anna Smith and John	My mother is called Anna Smith	E	E	BERTp
8	I hear John says 'come here'	I hear John speaking	E	C	BERTp
9	I hear 'John says come here'	I hear John speaking	C	E	BERTp
10	The notion of good, god, is incomprehensible	Good is incomprehensible	E	E	BERTp
11	The notion of good god is incomprehensible	Good god is incomprehensible	E	E	BERTp
12	The notion of good, god, is incomprehensible	Good god is incomprehensible	C	E	BERTp
13	The notion of good god is incomprehensible	Good is incomprehensible	N	E	BERTp
14	No, god is good	God is good	E	E	BERTp
15	No god is good	There is no good god	E	E	BERTp
16	No, god is good	There is no good god	C	E	BERTp
17	No god is good	God is good	C	E	BERTp

Table 9.4: Constructed dataset. E is entailment, N is neutral and C is contradiction. The Model column indicate which model was used (trained with punctuation p, or without $\neg p$).

Bibliography

Abzianidze Lasha, Bjerva Johannes, Evang Kilian, Haagsma Hessel, Van Noord Rik, Ludmann Pierre, Nguyen Duc-Duy, Bos Johan. The parallel meaning bank: Towards a multilingual corpus of translations annotated with compositional meaning representations // arXiv preprint arXiv:1702.03964. 2017.

Abzianidze Lasha, Bos Johan. Towards universal semantic tagging // arXiv preprint arXiv:1709.10381. 2017.

Ács Judit, Kádár Ákos, Kornai András. Subword Pooling Makes a Difference // Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021. 2021. 2284–2295.

Agresti Alan, Coull Brent A. Approximate is better than "exact" for interval estimation of binomial proportions // The American Statistician. 1998. 52, 2. 119–126.

Aharoni Roee, Goldberg Yoav. Morphological Inflection Generation with Hard Monotonic Attention // Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers. 2017. 2004–2015.

Alajrami Ahmed, Aletras Nikolaos. How does the pre-training objective affect what large language models learn about linguistic

properties? // Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022. 2022. 131–147.

Amanaki Eirini, Bernardy Jean-Philippe, Chatzikyriakidis Stergios, Cooper Robin, Dobnik Simon, Karimi Aram, Ek Adam, Giannikouri Eirini Chrysovalantou, Katsouli Vasiliki, Kolokousis Ilias, Mamatzaki Eirini Chrysovalantou, Papadakis Dimitrios, Petrova Olga, Psaltaki Erofili, Soupiona Charikleia, Skoulataki Effrosyni, Stefanidou Christina. Fine-grained Entailment: Resources for Greek NLI and Precise Entailment // Proceedings of the Workshop on Dataset Creation for Lower-Resourced Languages within the 13th Language Resources and Evaluation Conference. Marseille, France: European Language Resources Association, VI 2022. 44–52.

Anastasopoulos Antonios, Neubig Graham. Pushing the Limits of Low-Resource Morphological Inflection // Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019. 2019. 984–996.

Angeli Gabor, Johnson Premkumar Melvin Jose, Manning Christopher D. Leveraging Linguistic Structure For Open Domain Information Extraction // Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). Beijing, China: Association for Computational Linguistics, VII 2015. 344–354.

Arora Sanjeev, Liang Yingyu, Ma Tengyu. A Simple but Tough-to-Beat Baseline for Sentence Embeddings // 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. 2017.

Ataman Duygu, Aziz Wilker, Birch Alexandra. A Latent Morphology Model for Open-Vocabulary Neural Machine Translation // 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. 2020.

Attardi Giuseppe. Experiments with a Multilanguage Non-Projective Dependency Parser // Proceedings of the Tenth Conference on Computational Natural Language Learning, CoNLL 2006, New York City, USA, June 8-9, 2006. 2006. 166–170.

BNC Consortium . The British National Corpus, version 3 (BNC XML Edition). 2007 // Distributed by Oxford University Computing Services on behalf of the BNC Consortium. 2007.

Ba Lei Jimmy, Kiros Jamie Ryan, Hinton Geoffrey E. Layer Normalization // CoRR. 2016. abs/1607.06450.

Bahdanau Dzmitry, Cho Kyunghyun, Bengio Yoshua. Neural Machine Translation by Jointly Learning to Align and Translate // 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. 2015.

Batsuren Khuyagbaatar, Goldman Omer, Khalifa Salam, Habash Nizar, Kieras Witold, Bella Gábor, Leonard Brian, Nicolai Garrett, Gorman Kyle, Ate Yustinus Ghanggo, Ryskina Maria, Mielke Sabrina J., Budianskaya Elena, El-Khaissi Charbel, Pimentel Tiago, Gasser Michael, Lane William, Raj Mohit, Coler Matt, Samame Jaime Rafael Montoya, Camaiteri Delio Siticonatzi, Rojas Esaú Zumaeta, Francis Didier López, Oncevay Arturo, Bautista Juan López, Villegas Gema Celeste Silva, Hennigen Lucas Torroba, Ek Adam, Guriel David, Dirix Peter, Bernardy Jean-Philippe, Scherbakov Andrey, Bayyr-ool Aziyana, Anastasopoulos Antonios, Zariquiey Roberto, Sheifer Karina, Ganieva Sofya, Cruz Hilaria,

Karahóga Ritván, Markantonatou Stella, Pavlidis George, Plugaryov Matvey, Klyachko Elena, Salehi Ali, Angulo Candy, Baxi Jatayu, Krizhanovsky Andrew, Krizhanovskaya Natalia, Salesky Elizabeth, Vania Clara, Ivanova Sardana, White Jennifer, Maudslay Rowan Hall, Valvoda Josef, Zmigrod Ran, Czarnowska Paula, Nikkarinen Irene, Salchak Aelita, Bhatt Brijesh, Straughn Christopher, Liu Zoey, Washington Jonathan North, Pinter Yuval, Ataman Duygu, Wolinski Marcin, Suhardijanto Totok, Yablonskaya Anna, Stoehr Niklas, Dolatian Hossep, Nuriah Zahroh, Ratan Shyam, Tyers Francis M., Ponti Edoardo M., Aiton Grant, Arora Aryaman, Hatcher Richard J., Kumar Ritesh, Young Jeremiah, Rodionova Daria, Yemelina Anastasia, Andrushko Taras, Marchenko Igor, Mashkovtseva Polina, Serova Alexandra, Prud'hommeaux Emily, Nepomniashchaya Maria, Giunchiglia Fausto, Chodroff Eleanor, Hulden Mans, Silfverberg Miikka, McCarthy Arya D., Yarowsky David, Cotterell Ryan, Tsarfaty Reut, Vylomova Ekaterina. UniMorph 4.0: Universal Morphology // CoRR. 2022. abs/2205.03608.

Belinkov Yonatan, Glass James. Analysis Methods in Neural Language Processing: A Survey // Transactions of the Association for Computational Linguistics. 2019. 7. 49–72.

Bengio Yoshua, Ducharme Réjean, Vincent Pascal. A Neural Probabilistic Language Model // Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA. 2000. 932–938.

Bengio Yoshua, Frasconi Paolo, Simard Patrice Y. The problem of learning long-term dependencies in recurrent networks // Proceedings of International Conference on Neural Networks (ICNN'88), San Francisco, CA, USA, March 28 - April 1, 1993. 1993. 1183–1188.

Bengio Yoshua, Louradour Jérôme, Collobert Ronan, Weston Jason. Curriculum learning // Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal,

Quebec, Canada, June 14-18, 2009. 382. 2009. 41–48. (ACM International Conference Proceeding Series).

Bergmanis Toms, Kann Katharina, Schütze Hinrich, Goldwater Sharon. Training Data Augmentation for Low-Resource Morphological Inflection // Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection, Vancouver, BC, Canada, August 3-4, 2017. 2017. 31–39.

Bernardy Jean-Philippe. Can recurrent neural networks learn nested recursion? // Linguistic Issues in Language Technology. 2018. 16.

Bernardy Jean-Philippe, Chatzikyriakidis Stergios. What kind of Natural Language Inference are NLP systems learning: Is this enough? // ICAART (2). 2019. 919–931.

Bernardy Jean-Philippe, Chatzikyriakidis Stergios. Improving the precision of natural textual entailment problem datasets // Proceedings of the 12th Language Resources and Evaluation Conference. 2020. 6835–6840.

Bernardy Jean-Philippe, Lappin Shalom. Using Deep Neural Networks to Learn Syntactic Agreement // Linguistic Issues In Language Technology. 2017. 15, 2. 15.

Bernardy Jean-Philippe, Lappin Shalom, Lau Jay Han. The influence of context on sentence acceptability judgements // Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). 2018. 456–461.

Bhattamishra Satwik, Ahuja Kabir, Goyal Navin. On the Ability and Limitations of Transformers to Recognize Formal Languages // Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). Online: Association for Computational Linguistics, XI 2020. 7096–7116.

Bishop Christopher M, Nasrabadi Nasser M. Pattern recognition and machine learning. 4, 4. 2006.

Bjerva Johannes, Plank Barbara, Bos Johan. Semantic tagging with deep residual networks // arXiv preprint arXiv:1609.07053. 2016.

Blacoe William, Lapata Mirella. A Comparison of Vector-based Representations for Semantic Composition // Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea. 2012. 546–556.

Blevins James P. Introduction: Morphological paradigms // Transactions of the Philological Society. 2001. 99, 2. 207–210.

Blodgett Su Lin, Barocas Solon, III Hal Daumé, Wallach Hanna M. Language (Technology) is Power: A Critical Survey of "Bias" in NLP // Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020. 2020. 5454–5476.

Bojanowski Piotr, Grave Edouard, Joulin Armand, Mikolov Tomás. Enriching Word Vectors with Subword Information // Trans. Assoc. Comput. Linguistics. 2017. 5. 135–146.

Bouma Gosse, Seddah Djamé, Zeman Daniel. Overview of the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies // Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies. Seattle, US, July 2020.

Bouma Gosse, Seddah Djamé, Zeman Daniel. From raw text to enhanced universal dependencies: The parsing shared task at iwpt 2021 // Proceedings of the 17th International Conference on Parsing Technologies (IWPT 2021). 2021. 146–157.

Bowman Samuel R., Angeli Gabor, Potts Christopher, Manning Christopher D. A large annotated corpus for learning natural language inference // Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015. 2015. 632–642.

Bugliarello Emanuele, Cotterell Ryan, Okazaki Naoaki, Elliott Desmond. Multimodal Pretraining Unmasked: A Meta-Analysis and a Unified Framework of Vision-and-Language BERTs // Transactions of the Association for Computational Linguistics. 2021. 9. 978–994.

Buyko Ekaterina, Hahn Udo. Evaluating the impact of alternative dependency graph encodings on solving event extraction tasks // Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing. 2010. 982–992.

Candito Marie, Guillaume Bruno, Perrier Guy, Seddah Djamé. Enhanced UD Dependencies with Neutralized Diathesis Alternation // Proceedings of the Fourth International Conference on Dependency Linguistics (Depling 2017). Pisa,Italy: Linköping University Electronic Press, IX 2017. 42–53.

Chatzikyriakidis Stergios, Cooper Robin, Dobnik Simon, Larsson Staffan. An overview of Natural Language Inference Data Collection: The way forward? // Proceedings of the Computing Natural Language Inference Workshop. 2017.

Chaudhary Aditi, Zhou Chunting, Levin Lori, Neubig Graham, Mortensen David R, Carbonell Jaime G. Adapting Word Embeddings to New Languages with Morphological and Phonological Subword Representations // Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. 2018. 3285–3295.

Chen Danqi, Manning Christopher. A fast and accurate dependency parser using neural networks // Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014. 740–750.

Chen Qian, Zhu Xiaodan, Ling Zhen-Hua, Wei Si, Jiang Hui, Inkpen Diana. Enhanced LSTM for Natural Language Inference // Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers. 2017. 1657–1668.

Chen Xinxiong, Xu Lei, Liu Zhiyuan, Sun Maosong, Luan Huan-Bo. Joint Learning of Character and Word Embeddings // Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015. 2015. 1236–1242.

Cheng Jianpeng, Dong Li, Lapata Mirella. Long Short-Term Memory-Networks for Machine Reading // Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. Austin, Texas: Association for Computational Linguistics, XI 2016. 551–561.

Cheng Kefei, Yue Yanan, Song Zhiwen. Sentiment Classification Based on Part-of-Speech and Self-Attention Mechanism // IEEE Access. 2020. 8. 16387–16396.

Chollet François, others . Keras. 2015.

Chu Yoeng-Jin. On the shortest arborescence of a directed graph // Scientia Sinica. 1965. 14. 1396–1400.

Chung Junyoung, Gülçehre Çaglar, Cho KyungHyun, Bengio Yoshua. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling // CoRR. 2014. abs/1412.3555.

Church Kenneth Ward. Emerging trends: Subwords, seriously? // Nat. Lang. Eng. 2020. 26, 3. 375–382.

Clark Kevin, Khandelwal Urvashi, Levy Omer, Manning Christopher D. What Does BERT Look at? An Analysis of BERT's Attention // Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@ACL 2019, Florence, Italy, August 1, 2019. 2019. 276–286.

Conneau Alexis, Khandelwal Kartikay, Goyal Naman, Chaudhary Vishrav, Wenzek Guillaume, Guzmán Francisco, Grave Edouard, Ott Myle, Zettlemoyer Luke, Stoyanov Veselin. Unsupervised Cross-lingual Representation Learning at Scale // Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020. 2020. 8440–8451.

Conneau Alexis, Kiela Douwe, Schwenk Holger, Barrault Loïc, Bordes Antoine. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data // Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017. 2017. 670–680.

Dehaene Stanislas, Meyniel Florent, Wacongne Catherine, Wang Liping, Pallier Christophe. The neural representation of sequences: from transition probabilities to algebraic patterns and linguistic trees // Neuron. 2015. 88, 1. 2–19.

Demszky Dorottya, Guu Kelvin, Liang Percy. Transforming Question Answering Datasets Into Natural Language Inference Datasets // CoRR. 2018. abs/1809.02922.

Speech understanding systems: summary of results of the five-year research effort at Carnegie-Mellon University. // . 4 1977.

Devlin Jacob, Chang Ming-Wei, Lee Kenton, Toutanova Kristina. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding // Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers). 2019. 4171–4186.

Dobnik Simon, Cooper Robin, Ek Adam, Noble Bill, Larsson Staffan, Ilinykh Nikolai, Maraev Vladislav, Somashekarappa Vidya. In Search of Meaning and Its Representations for Computational Linguistics // Proceedings of the 2022 CLASP Conference on (Dis)embodiment. Gothenburg, Sweden: Association for Computational Linguistics, IX 2022. 30–44.

Deep Biaffine Attention for Neural Dependency Parsing. // . 2017.

Duchi John C., Hazan Elad, Singer Yoram. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization // J. Mach. Learn. Res. 2011. 12. 2121–2159.

Ebrahimi Javid, Gelda Dhruv, Zhang Wei. How Can Self-Attention Networks Recognize Dyck-n Languages? // Findings of the Association for Computational Linguistics: EMNLP 2020. Online: Association for Computational Linguistics, XI 2020. 4301–4306.

Edman Lukas, Toral Antonio, Noord Gertjan van. Subword-Delimited Downsampling for Better Character-Level Translation // CoRR. 2022. abs/2212.01304.

Edmonds Jack, others . Optimum branchings // Journal of Research of the national Bureau of Standards B. 1967. 71, 4. 233–240.

Ek Adam, Bernardy Jean-Philippe. Composing Byte-Pair Encodings for Morphological Sequence Classification // Proceedings of the

Fourth Workshop on Universal Dependencies (UDW 2020). 2020a. 76–86.

Ek Adam, Bernardy Jean-Philippe. How much of enhanced UD is contained in UD? // Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies. 2020b. 221–226.

Ek Adam, Bernardy Jean-Philippe. Training Strategies for Neural Multilingual Morphological Inflection // Proceedings of the 18th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology. Online: Association for Computational Linguistics, VIII 2021. 260–267.

Ek Adam, Ghanimifard Mehdi. Synthetic Propaganda Embeddings To Train A Linear Projection // Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda. Hong Kong, China: Association for Computational Linguistics, XI 2019. 155–161.

Ek Adam, Ilinykh Nikolai. Vector Norms as an Approximation of Syntactic Complexity // Proceedings of the Second Workshop on Resources and Representations for Under-Resourced Languages and Domains (RESOURCEFUL-2023). Tórshavn, the Faroe Islands: Association for Computational Linguistics, V 2023. 121–131.

Ek Adam, Wirén Mats. Distinguishing narration and speech in prose fiction dialogues // DHN. 2019.

El Kholy Ahmed, Habash Nizar. Orthographic and morphological processing for English–Arabic statistical machine translation // Machine Translation. 2012. 26, 1-2. 25–45.

Elman Jeffrey L. Finding Structure in Time // Cogn. Sci. 1990. 14, 2. 179–211.

Ettinger Allyson. What BERT Is Not: Lessons from a New Suite of Psycholinguistic Diagnostics for Language Models // Transactions of the Association for Computational Linguistics. 2020. 8. 34–48.

Fang Meng, Cohn Trevor. Learning when to trust distant supervision: An application to low-resource POS tagging using cross-lingual projection // Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016. 2016. 178–186.

Fares Murhaf, Oepen Stephan, Øvrelid Lilja, Björne Jari, Johansson Richard. The 2018 shared task on extrinsic parser evaluation: on the downstream utility of English Universal Dependency Parsers // Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. 2018. 22–33.

Faruqui Manaal, Tsvetkov Yulia, Neubig Graham, Dyer Chris. Morphological Inflection Generation Using Character Sequence to Sequence Learning // NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016. 2016. 634–643.

Fernández-González Daniel, Gómez-Rodríguez Carlos. Left-to-Right Dependency Parsing with Pointer Networks // Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers). 2019. 710–716.

FitzGerald Nicholas, Michael Julian, He Luheng, Zettlemoyer Luke. Large-Scale QA-SRL Parsing // Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Vol-

ume 1: Long Papers). Melbourne, Australia: Association for Computational Linguistics, VII 2018. 2051–2060.

Freedman David, Pisani Robert, Purves Roger. Statistics (international student edition) // Pisani, R. Purves, 4th edn. WW Norton & Company, New York. 2007.

Futrell Richard, Mahowald Kyle, Gibson Edward. Large-scale evidence of dependency length minimization in 37 languages // Proc Natl Acad Sci U S A. VIII 2015. 112, 33. 10336–10341.

Gers Felix A., Schmidhuber Jürgen. LSTM recurrent networks learn simple context-free and context-sensitive languages // IEEE Trans. Neural Networks. 2001. 12, 6. 1333–1340.

Geva Mor, Schuster Roei, Berant Jonathan, Levy Omer. Transformer Feed-Forward Layers Are Key-Value Memories // Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021. 2021. 5484–5495.

Glockner Max, Shwartz Vered, Goldberg Yoav. Breaking NLI Systems with Sentences that Require Simple Lexical Inferences // Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers. 2018. 650–655.

Goldberg Yoav. Neural network methods for natural language processing // Synthesis lectures on human language technologies. 2017. 10, 1. 1–309.

Gómez-Rodríguez Carlos, Vilares David. Constituent Parsing as Sequence Labeling // arXiv:1810.08994 [cs]. X 2018. arXiv: 1810.08994.

Goodfellow Ian J., Bengio Yoshua, Courville Aaron C. Deep Learning. 2016. (Adaptive computation and machine learning).

Graves Alex, Wayne Greg, Danihelka Ivo. Neural Turing Machines // CoRR. 2014. abs/1410.5401.

Gulordava Kristina, Bojanowski Piotr, Grave Edouard, Linzen Tal, Baroni Marco. Colorless Green Recurrent Networks Dream Hierarchically // Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). New Orleans, Louisiana: Association for Computational Linguistics, VI 2018. 1195–1205.

Güngör Onur, Güngör Tunga, Üsküdarli Suzan. The effect of morphology in named entity recognition with sequence tagging // Nat. Lang. Eng. 2019. 25, 1. 147–169.

Gupta Amulya, Zhang Zhu. To Attend or not to Attend: A Case Study on Syntactic Structures for Semantic Relatedness // Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Melbourne, Australia: Association for Computational Linguistics, VII 2018. 2116–2125.

Gururangan Suchin, Swayamdipta Swabha, Levy Omer, Schwartz Roy, Bowman Samuel R., Smith Noah A. Annotation Artifacts in Natural Language Inference Data // Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers). 2018. 107–112.

Hacohen Guy, Weinshall Daphna. On The Power of Curriculum Learning in Training Deep Networks // Proceedings of the 36th

International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA. 97. 2019. 2535–2544. (Proceedings of Machine Learning Research).

Hahn Michael. Theoretical Limitations of Self-Attention in Neural Sequence Models // Transactions of the Association for Computational Linguistics. 2020. 8. 156–171.

Haruechaiyasak Choochart, Kongthon Alisa. LexToPlus: A thai lexeme tokenization and normalization tool // Proceedings of the 4th Workshop on South and Southeast Asian Natural Language Processing. 2013. 9–16.

Haspelmath Martin. The indeterminacy of word segmentation and the nature of morphology and syntax // Folia linguistica. 2017. 51, s1000. 31–80.

Hauser Marc D, Chomsky Noam, Fitch W Tecumseh. The faculty of language: what is it, who has it, and how did it evolve? // science. 2002. 298, 5598. 1569–1579.

He Luheng, Lewis Mike, Zettlemoyer Luke. Question-Answer Driven Semantic Role Labeling: Using Natural Language to Annotate Natural Language // Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. Lisbon, Portugal: Association for Computational Linguistics, IX 2015. 643–653.

Hendrycks Dan, Gimpel Kevin. Gaussian error linear units (gelus) // arXiv preprint arXiv:1606.08415. 2016.

Hewitt John, Hahn Michael, Ganguli Surya, Liang Percy, Manning Christopher D. RNNs can generate bounded hierarchical languages with optimal memory // Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). Online: Association for Computational Linguistics, XI 2020. 1978–2010.

Hewitt John, Manning Christopher D. A Structural Probe for Finding Syntax in Word Representations // Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers). 2019. 4129–4138.

Hochreiter Sepp, Schmidhuber Jürgen. Long Short-Term Memory // Neural Comput. 1997. 9, 8. 1735–1780.

Hord Levi CR. Bucking the linguistic binary: Gender neutral language in English, Swedish, French, and German // Western Papers in Linguistics. 2016. 3, 1.

Howard Jeremy, Ruder Sebastian. Universal Language Model Fine-tuning for Text Classification // Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers. 2018. 328–339.

Htut Phu Mon, Phang Jason, Bordia Shikha, Bowman Samuel R. Do Attention Heads in BERT Track Syntactic Dependencies? // CoRR. 2019. abs/1911.12246.

Huang Chu-Ren, Simon Petr, Hsieh Shu-Kai, Prévot Laurent. Rethinking Chinese Word Segmentation: Tokenization, Character Classification, or Wordbreak Identification // ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, June 23-30, 2007, Prague, Czech Republic. 2007.

Ilinykh Nikolai, Dobnik Simon. What Does a Language-And-Vision Transformer See: The Impact of Semantic Information on Visual Representations // Frontiers in Artificial Intelligence. 2021. 4.

Jing Yingqi, Liu Haitao. Mean Hierarchical Distance Augmenting Mean Dependency Distance // Proceedings of the Third International Conference on Dependency Linguistics (Depling 2015). Uppsala, Sweden: Uppsala University, Uppsala, Sweden, VIII 2015. 161–170.

Jurafsky Daniel, Wooters Chuck, Segal Jonathan, Stolcke Andreas, Fosler Eric, Tajchman Gary N., Morgan Nelson. Using a stochastic context-free grammar as a language model for speech recognition // 1995 International Conference on Acoustics, Speech, and Signal Processing, ICASSP '95, Detroit, Michigan, USA, May 08-12, 1995. 1995. 189–192.

Kann Katharina, Bjerva Johannes, Augenstein Isabelle, Plank Barbara, Søgaard Anders. Character-level Supervision for Low-resource POS Tagging // Proceedings of the Workshop on Deep Learning Approaches for Low-Resource NLP, DeepLo@ACL 2018, Melbourne, Australia, July 19, 2018. 2018. 1–11.

Katharopoulos Angelos, Fleuret François. Not All Samples Are Created Equal: Deep Learning with Importance Sampling // Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018. 80. 2018. 2530–2539. (Proceedings of Machine Learning Research).

Ke Guolin, He Di, Liu Tie-Yan. Rethinking Positional Encoding in Language Pre-training // 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. 2021.

Kibort Anna, Corbett Greville G. Features: Perspectives on a Key Notion in Linguistics. 08 2010.

Kiefer Jack, Wolfowitz Jacob. Stochastic estimation of the maximum of a regression function // The Annals of Mathematical Statistics. 1952. 462–466.

Kingma Diederik P., Ba Jimmy. Adam: A Method for Stochastic Optimization // 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. 2015.

Kiperwasser Eliyahu, Goldberg Yoav. Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations // Trans. Assoc. Comput. Linguistics. 2016. 4. 313–327.

Kobayashi Goro, Kuribayashi Tatsuki, Yokoi Sho, Inui Kentaro. Attention is Not Only a Weight: Analyzing Transformers with Vector Norms // Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020. 2020. 7057–7075.

Kocmi Tom, Bojar Ondrej. An Exploration of Word Embedding Initialization in Deep-Learning Tasks // Proceedings of the 14th International Conference on Natural Language Processing, ICON 2017, Kolkata, India, December 18-21, 2017. 2017. 56–64.

Kondratyuk Dan. Cross-Lingual Lemmatization and Morphology Tagging with Two-Stage Multilingual BERT Fine-Tuning // Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology. 2019. 12–18.

Kondratyuk Daniel, Straka Milan. 75 Languages, 1 Model: Parsing Universal Dependencies Universally // Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019. 2019. 2779–2795.

Kovaleva Olga, Romanov Alexey, Rogers Anna, Rumshisky Anna. Revealing the Dark Secrets of BERT // Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019. 2019. 4364–4373.

Kudo Taku. Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates // Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers. 2018. 66–75.

Kudo Taku, Richardson John. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing // Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018: System Demonstrations, Brussels, Belgium, October 31 - November 4, 2018. 2018. 66–71.

Kulmizev Artur, Lhoneux Miryam de, Gontrum Johannes, Fano Elena, Nivre Joakim. Deep Contextualized Word Embeddings in Transition-Based and Graph-Based Dependency Parsing - A Tale of Two Parsers Revisited // Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019. 2019. 2755–2768.

Lakretz Yair, Kruszewski German, Desbordes Theo, Hupkes Dieuwke, Dehaene Stanislas, Baroni Marco. The emergence of number and syntax units in LSTM language models // arXiv preprint arXiv:1903.07435. 2019.

Lau Jey Han, Clark Alexander, Lappin Shalom. Measuring gradience in speakers' grammaticality judgements // Proceedings of the Annual Meeting of the Cognitive Science Society. 2014.

Lau Jey Han, Clark Alexander, Lappin Shalom. Unsupervised Prediction of Acceptability Judgements // Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). Beijing, China: Association for Computational Linguistics, 2015. 1618–1628.

Lau Jey Han, Clark Alexander, Lappin Shalom. Grammaticality, Acceptability, and Probability: A Probabilistic View of Linguistic Knowledge // Cognitive Science. VII 2017. 41, 5. 1202–1241.

Lee Jaejun, Tang Raphael, Lin Jimmy. What Would Elsa Do? Freezing Layers During Transformer Fine-Tuning // CoRR. 2019. abs/1911.03090.

Lee Yun Joon Jason. Polysemous Words in English Movies, Learning Obstacles or Gifted Talent? // Journal of English Teaching through Movies and Media. 2021. 22, 4. 14–26.

Lhoneux Miryam de, Shao Yan, Basirat Ali, Kiperwasser Eliyahu, Stymne Sara, Goldberg Yoav, Nivre Joakim. From Raw Text to Universal Dependencies - Look, No Tags! // Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Vancouver, Canada, August 3-4, 2017. 2017. 207–217.

Li Zhe, Li Xiuhong, Sheng Jiabao, Slamu Wushour. AgglutiFiT: Efficient Low-Resource Agglutinative Language Model Fine-Tuning // IEEE Access. 2020. 8. 148489–148499.

Lin Zhouhan, Feng Minwei, Santos Cícero Nogueira dos, Yu Mo, Xiang Bing, Zhou Bowen, Bengio Yoshua. A Structured Self-Attentive

Sentence Embedding // 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. 2017.

Linzen Tal, Dupoux Emmanuel, Goldberg Yoav. Assessing the ability of LSTMs to learn syntax-sensitive dependencies // Transactions of the Association for Computational Linguistics. 2016. 4. 521–535.

Liu Haitao. Dependency distance as a metric of language comprehension difficulty // Journal of Cognitive Science. 2008. 9, 2. 159–191.

Liu Nelson F, Gardner Matt, Belinkov Yonatan, Peters Matthew E, Smith Noah A. Linguistic Knowledge and Transferability of Contextual Representations // Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). 2019a. 1073–1094.

Liu Xuebo, Lai Houtim, Wong Derek F., Chao Lidia S. Norm-Based Curriculum Learning for Neural Machine Translation // Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020. 2020a. 427–436.

Liu Yinhan, Gu Jiatao, Goyal Naman, Li Xian, Edunov Sergey, Ghazvininejad Marjan, Lewis Mike, Zettlemoyer Luke. Multilingual Denoising Pre-training for Neural Machine Translation // Trans. Assoc. Comput. Linguistics. 2020b. 8. 726–742.

Liu Yinhan, Ott Myle, Goyal Naman, Du Jingfei, Joshi Mandar, Chen Danqi, Levy Omer, Lewis Mike, Zettlemoyer Luke, Stoyanov Veselin. RoBERTa: A Robustly Optimized BERT Pretraining Approach // CoRR. 2019b. abs/1907.11692.

Loshchilov Ilya, Hutter Frank. SGDR: Stochastic Gradient Descent with Warm Restarts // 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. 2017.

Lu Yu, Zhang Jiajun. Norm-based Noisy Corpora Filtering and Refurbishing in Neural Machine Translation // Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing. 2022. 5414–5425.

Luong Thang, Pham Hieu, Manning Christopher D. Effective Approaches to Attention-based Neural Machine Translation // Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015. 2015. 1412–1421.

Ma Xuezhe, Hu Zecong, Liu Jingzhou, Peng Nanyun, Neubig Graham, Hovy Eduard H. Stack-Pointer Networks for Dependency Parsing // Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers. 2018. 1403–1414.

Maas Andrew L, Hannun Awni Y, Ng Andrew Y, others . Rectifier nonlinearities improve neural network acoustic models // Proc. icml. 30, 1. 2013. 3.

Magueresse Alexandre, Carles Vincent, Heetderks Evan. Low-resource Languages: A Review of Past Work and Future Challenges // CoRR. 2020. abs/2006.07264.

Makarov Peter, Ruzsics Tatiana, Clematide Simon. Align and Copy: UZH at SIGMORPHON 2017 Shared Task for Morphological Reinflection // Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection, Vancouver, BC, Canada, August 3-4, 2017. 2017. 49–57.

Manning Christopher D. Part-of-Speech Tagging from 97% to 100%: Is It Time for Some Linguistics? // Computational Linguistics and Intelligent Text Processing - 12th International Conference, CICLing 2011, Tokyo, Japan, February 20-26, 2011. Proceedings, Part I. 6608. 2011. 171–189. (Lecture Notes in Computer Science).

Marcheggiani Diego, Frolov Anton, Titov Ivan. A Simple and Accurate Syntax-Agnostic Neural Model for Dependency-based Semantic Role Labeling // Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017). Vancouver, Canada: Association for Computational Linguistics, VIII 2017. 411–420.

Marneffe Marie-Catherine de, Manning Christopher D., Nivre Joakim, Zeman Daniel. Universal Dependencies // Computational Linguistics. 07 2021. 47, 2. 255–308.

Martin Louis, Müller Benjamin, Suárez Pedro Javier Ortiz, Dupont Yoann, Romary Laurent, Clergerie Éric de la, Seddah Djamé, Sagot Benoît. CamemBERT: a Tasty French Language Model // Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020. 2020. 7203–7219.

Matteson Andrew, Lee Chanhee, Kim Young-Bum, Lim Heuiseok. Rich Character-Level Information for Korean Morphological Analysis and Part-of-Speech Tagging // Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018. 2018. 2482–2492.

McCarthy Arya D, Silfverberg Miikka, Cotterell Ryan, Hulden Mans, Yarowsky David. Marrying Universal Dependencies and Universal Morphology // Proceedings of the Second Workshop on Universal Dependencies (UDW 2018). 2018. 91–101.

McCarthy Arya D, Vylomova Ekaterina, Wu Shijie, Malaviya Chaitanya, Wolf-Sonkin Lawrence, Nicolai Garrett, Kirov Christo, Silfverberg Miikka, Mielke Sebastian J, Heinz Jeffrey, others . The SIGMORPHON 2019 Shared Task: Morphological Analysis in Context and Cross-Lingual Transfer for Inflection // Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology. 2019. 229–244.

Meteer Marie, Rohlicek Jan Robin. Statistical language modeling combining N-gram and context-free grammars // IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '93, Minneapolis, Minnesota, USA, April 27-30, 1993. 1993. 37–40.

Michel Paul, Levy Omer, Neubig Graham. Are Sixteen Heads Really Better than One? // Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada. 2019. 14014–14024.

Mikolov Tomás, Chen Kai, Corrado Greg, Dean Jeffrey. Efficient Estimation of Word Representations in Vector Space // 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings. 2013a.

Mikolov Tomás, Sutskever Ilya, Chen Kai, Corrado Gregory S., Dean Jeffrey. Distributed Representations of Words and Phrases and their Compositionality // Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States. 2013b. 3111–3119.

Mishra Anshuman, Patel Dhruvesh, Vijayakumar Aparna, Li Xiang, Kapanipathi Pavan, Talamadupula Kartik. Reading Comprehen-

sion as Natural Language Inference:A Semantic Analysis // Proceedings of the Ninth Joint Conference on Lexical and Computational Semantics. Barcelona, Spain (Online): Association for Computational Linguistics, XII 2020. 12–19.

Mitchell Jeff, Lapata Mirella. Composition in Distributional Models of Semantics // Cogn. Sci. 2010. 34, 8. 1388–1429.

Miwa Makoto, Pyysalo Sampo, Hara Tadayoshi, Tsujii Jun'ichi. Evaluating Dependency Representations for Event Extraction // Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010). Beijing, China: Coling 2010 Organizing Committee, VIII 2010a. 779–787.

Miwa Makoto, Pyysalo Sampo, Hara Tadayoshi, Tsujii Jun'ichi. A comparative study of syntactic parsers for event extraction // Proceedings of the 2010 Workshop on Biomedical Natural Language Processing. 2010b. 37–45.

Montague Richard. Universal grammar // Theoria. 1970. 36, 3. 373–398.

Nair Vinod, Hinton Geoffrey E. Rectified Linear Units Improve Restricted Boltzmann Machines // Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel. 2010. 807–814.

Nivre Joakim, Abrams Mitchell, Agić Željko, Ahrenberg Lars, Antonsen Lene, Aplonova Katya, Aranzabe Maria Jesus, Arutie Gashaw, Asahara Masayuki, Ateyah Luma, Attia Mohammed, Atutxa Aitziber, Augustinus Liesbeth, Badmaeva Elena, Ballesteros Miguel, Banerjee Esha, Bank Sebastian, Barbu Mititelu Verginica, Basmov Victoria, Bauer John, Bellato Sandra, Bengoetxea Kepa, Berzak Yevgeni, Bhat Irshad Ahmad, Bhat Riyaz Ahmad, Biagetti Erica, Bick Eckhard, Blokland Rogier, Bobicev Victoria, Börstell Carl,

Bosco Cristina, Bouma Gosse, Bowman Sam, Boyd Adriane, Burchardt Aljoscha, Candito Marie, Caron Bernard, Caron Gauthier, Cebiroğlu Eryiğit Gülşen, Cecchini Flavio Massimiliano, Celano Giuseppe G. A., Čéplö Slavomír, Cetin Savas, Chalub Fabricio, Choi Jinho, Cho Yongseok, Chun Jayeol, Cinková Silvie, Collomb Aurélie, Çöltekin Çağrı, Connor Miriam, Courtin Marine, Davidson Elizabeth, Marneffe Marie-Catherine de, Paiva Valeria de, Ilarraza Arantza Diaz de, Dickerson Carly, Dirix Peter, Dobrovoljc Kaja, Dozat Timothy, Droganova Kira, Dwivedi Puneet, Eli Marhaba, Elkahky Ali, Ephrem Binyam, Erjavec Tomaž, Etienne Aline, Farkas Richárd, Fernandez Alcalde Hector, Foster Jennifer, Freitas Cláudia, Gajdošová Katarína, Galbraith Daniel, Garcia Marcos, Gärdenfors Moa, Garza Sebastian, Gerdes Kim, Ginter Filip, Goenaga Iakes, Gojenola Koldo, Gökırmak Memduh, Goldberg Yoav, Gómez Guinovart Xavier, Gonzáles Saavedra Berta, Grioni Matias, Grūzītis Normunds, Guillaume Bruno, Guillot-Barbance Céline, Habash Nizar, Hajič Jan, Hajič jr. Jan, Hà Mỹ Linh, Han Na-Rae, Harris Kim, Haug Dag, Hladká Barbora, Hlaváčová Jaroslava, Hociung Florinel, Hohle Petter, Hwang Jena, Ion Radu, Irimia Elena, Ishola Ọlájídé, Jelínek Tomáš, Johannsen Anders, Jørgensen Fredrik, Kaşıkara Hüner, Kahane Sylvain, Kanayama Hiroshi, Kanerva Jenna, Katz Boris, Kayadelen Tolga, Kenney Jessica, Kettnerová Václava, Kirchner Jesse, Kopacewicz Kamil, Kotsyba Natalia, Krek Simon, Kwak Sookyoung, Laippala Veronika, Lambertino Lorenzo, Lam Lucia, Lando Tatiana, Larasati Septina Dian, Lavrentiev Alexei, Lee John, Lê Hồng Phương, Lenci Alessandro, Lertpradit Saran, Leung Herman, Li Cheuk Ying, Li Josie, Li Keying, Lim KyungTae, Ljubešić Nikola, Loginova Olga, Lyashevskaya Olga, Lynn Teresa, Macketanz Vivien, Makazhanov Aibek, Mandl Michael, Manning Christopher, Manurung Ruli, Mărănduc Cătălina, Mareček David, Marheinecke Katrin, Martínez Alonso Héctor, Martins André, Mašek Jan, Matsumoto Yuji, McDonald Ryan, Mendonça Gustavo, Miekka Niko, Misirpashayeva Margarita, Missilä Anna,

Mititelu Cătălin, Miyao Yusuke, Montemagni Simonetta, More Amir, Moreno Romero Laura, Mori Keiko Sophie, Mori Shinsuke, Mortensen Bjartur, Moskalevskyi Bohdan, Muischnek Kadri, Murawaki Yugo, Müürisep Kaili, Nainwani Pinkey, Navarro Horñiacek Juan Ignacio, Nedoluzhko Anna, Nešpore-Bērzkalne Gunta, Nguyễn Thị Lương, Nguyễn Thị Minh Huyền, Nikolaev Vitaly, Nitisaroj Rattima, Nurmi Hanna, Ojala Stina, Olúòkun Adédayọ̀, Omura Mai, Osenova Petya, Östling Robert, Øvrelid Lilja, Partanen Niko, Pascual Elena, Passarotti Marco, Patejuk Agnieszka, Paulino-Passos Guilherme, Peng Siyao, Perez Cenel-Augusto, Perrier Guy, Petrov Slav, Piitulainen Jussi, Pitler Emily, Plank Barbara, Poibeau Thierry, Popel Martin, Pretkalniņa Lauma, Prévost Sophie, Prokopidis Prokopis, Przepiórkowski Adam, Puolakainen Tiina, Pyysalo Sampo, Rääbis Andriela, Rademaker Alexandre, Ramasamy Loganathan, Rama Taraka, Ramisch Carlos, Ravishankar Vinit, Real Livy, Reddy Siva, Rehm Georg, Rießler Michael, Rinaldi Larissa, Rituma Laura, Rocha Luisa, Romanenko Mykhailo, Rosa Rudolf, Rovati Davide, Roșca Valentin, Rudina Olga, Rueter Jack, Sadde Shoval, Sagot Benoît, Saleh Shadi, Samardžić Tanja, Samson Stephanie, Sanguinetti Manuela, Saulīte Baiba, Sawanakunanon Yanin, Schneider Nathan, Schuster Sebastian, Seddah Djamé, Seeker Wolfgang, Seraji Mojgan, Shen Mo, Shimada Atsuko, Shohibussirri Muh, Sichinava Dmitry, Silveira Natalia, Simi Maria, Simionescu Radu, Simkó Katalin, Šimková Mária, Simov Kiril, Smith Aaron, Soares-Bastos Isabela, Spadine Carolyn, Stella Antonio, Straka Milan, Strnadová Jana, Suhr Alane, Sulubacak Umut, Szántó Zsolt, Taji Dima, Takahashi Yuta, Tanaka Takaaki, Tellier Isabelle, Trosterud Trond, Trukhina Anna, Tsarfaty Reut, Tyers Francis, Uematsu Sumire, Urešová Zdeňka, Uria Larraitz, Uszkoreit Hans, Vajjala Sowmya, Niekerk Daniel van, Noord Gertjan van, Varga Viktor, Clergerie Eric Villemonte de la, Vincze Veronika, Wallin Lars, Wang Jing Xian, Washington Jonathan North, Williams Seyi, Wirén Mats, Woldemariam Tsegay, Wong Tak-sum, Yan Chunxiao, Yavrumyan

Marat M., Yu Zhuoran, Žabokrtský Zdeněk, Zeldes Amir, Zeman Daniel, Zhang Manying, Zhu Hanzhi. Universal Dependencies 2.3. 2018. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Nivre Joakim, Agić Željko, Ahrenberg Lars, al. et. Universal Dependencies 2.0. 2017. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Nivre Joakim, De Marneffe Marie-Catherine, Ginter Filip, Goldberg Yoav, Hajic Jan, Manning Christopher D, McDonald Ryan T, Petrov Slav, Pyysalo Sampo, Silveira Natalia, others . Universal Dependencies v1: A Multilingual Treebank Collection. // LREC. 2016.

Nivre Joakim, Fernández-González Daniel. Arc-Eager Parsing with the Tree Constraint // Comput. Linguistics. 2014. 40, 2. 259–267.

Nivre Joakim, Hall Johan, Nilsson Jens. MaltParser: A Data-Driven Parser-Generator for Dependency Parsing // Proceedings of the Fifth International Conference on Language Resources and Evaluation, LREC 2006, Genoa, Italy, May 22-28, 2006. 2006. 2216–2219.

Oktavianti Ikmi Nur, Ardianti Novi Retno. A corpus-based analysis of verbs in news section of The Jakarta Post: How frequency is related to text characteristics // JOALL (Journal of Applied Linguistics and Literature). 2019. 4, 2. 203–214.

Osborne Timothy, Gerdes Kim. The status of function words in dependency grammar: A critique of Universal Dependencies (UD) // Glossa (Online). 2019.

Compounding in a Swedish blog corpus. // . 2013.

Özates Saziye Betül, Özgür Arzucan, Gungor Tunga, Öztürk Balkiz. A Morphology-Based Representation Model for LSTM-Based Dependency Parsing of Agglutinative Languages // Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Brussels, Belgium, October 31 - November 1, 2018. 2018. 238–247.

Pan Sinno Jialin, Yang Qiang. A Survey on Transfer Learning // IEEE Trans. Knowl. Data Eng. 2010. 22, 10. 1345–1359.

Pan Yirong, Li Xiao, Yang Yating, Dong Rui. Morphological Word Segmentation on Agglutinative Languages for Neural Machine Translation // CoRR. 2020. abs/2001.01589.

Partee Barbara. Lexical semantics and compositionality // An invitation to cognitive science. 1995. 1. 311–360.

Partee Barbara H. A brief history of the syntax-semantics interface in Western formal linguistics // Semantics-Syntax Interface. 2014. 1, 1. 1–21.

Pascanu Razvan, Mikolov Tomás, Bengio Yoshua. On the difficulty of training recurrent neural networks // Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013. 28. 2013. 1310–1318. (JMLR Workshop and Conference Proceedings).

Pauls Adam, Klein Dan. Large-scale syntactic language modeling with treelets // Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1. 2012. 959–968.

Pennington Jeffrey, Socher Richard, Manning Christopher D. Glove: Global Vectors for Word Representation // Proceedings of the 2014

Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL. 2014. 1532–1543.

Peters Matthew E., Neumann Mark, Iyyer Mohit, Gardner Matt, Clark Christopher, Lee Kenton, Zettlemoyer Luke. Deep contextualized word representations // Proc. of NAACL. 2018a.

Peters Matthew E., Neumann Mark, Zettlemoyer Luke, Yih Wen-tau. Dissecting Contextual Word Embeddings: Architecture and Representation // Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. Brussels, Belgium: Association for Computational Linguistics, X-XI 2018b. 1499–1509.

Peters Matthew E., Ruder Sebastian, Smith Noah A. To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks // Proceedings of the 4th Workshop on Representation Learning for NLP, RepL4NLP@ACL 2019, Florence, Italy, August 2, 2019. 2019. 7–14.

Pimentel Tiago, Ryskina Maria, Mielke Sabrina J., Wu Shijie, Chodroff Eleanor, Leonard Brian, Nicolai Garrett, Ghanggo Ate Yustinus, Khalifa Salam, Habash Nizar, El-Khaissi Charbel, Goldman Omer, Gasser Michael, Lane William, Coler Matt, Oncevay Arturo, Montoya Samame Jaime Rafael, Silva Villegas Gema Celeste, Ek Adam, Bernardy Jean-Philippe, Shcherbakov Andrey, Bayyr-ool Aziyana, Sheifer Karina, Ganieva Sofya, Plugaryov Matvey, Klyachko Elena, Salehi Ali, Krizhanovsky Andrew, Krizhanovsky Natalia, Vania Clara, Ivanova Sardana, Salchak Aelita, Straughn Christopher, Liu Zoey, Washington Jonathan North, Ataman Duygu, Kieraś Witold, Woliński Marcin, Suhardijanto Totok, Stoehr Niklas, Nuriah Zahroh, Ratan Shyam, Tyers Francis M., Ponti Edoardo M., Aiton Grant, Hatcher Richard J., Prud'hommeaux Emily, Kumar Ritesh, Hulden Mans, Barta Botond, Lakatos Dorina, Szolnok Gábor, Ács Judit, Raj

Mohit, Yarowsky David, Cotterell Ryan, Ambridge Ben, Vylomova Ekaterina. SIGMORPHON 2021 Shared Task on Morphological Reinflection: Generalization Across Languages // Proceedings of the 18th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology. Online: Association for Computational Linguistics, VIII 2021. 229–259.

Pires Telmo, Schlinger Eva, Garrette Dan. How Multilingual is Multilingual BERT? // Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers. 2019. 4996–5001.

Plank Barbara, Klerke Sigrid. Lexical Resources for Low-Resource PoS Tagging in Neural Times // Proceedings of the 22nd Nordic Conference on Computational Linguistics, NoDaLiDa 2019, Turku, Finland, September 30 - October 2, 2019. 2019. 25–34.

Competence-based Curriculum Learning for Neural Machine Translation. // . 2019. 1162–1172.

Poesio Massimo, Bruneseaux Florence, Romary Laurent. The MATE meta-scheme for coreference in dialogues in multiple languages // ACL'99 Workshop Towards Standards and Tools for Discourse Tagging. 1999. 65–74.

Poliak Adam, Naradowsky Jason, Haldar Aparajita, Rudinger Rachel, Durme Benjamin Van. Hypothesis Only Baselines in Natural Language Inference // Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics, *SEM@NAACL-HLT 2018, New Orleans, Louisiana, USA, June 5-6, 2018. 2018. 180–191.

Ponti Edoardo Maria, O'Horan Helen, Berzak Yevgeni, Vulic Ivan, Reichart Roi, Poibeau Thierry, Shutova Ekaterina, Korhonen Anna.

Modeling Language Variation and Universals: A Survey on Typological Linguistics for Natural Language Processing // Comput. Linguistics. 2019. 45, 3. 559–601.

Qi Peng, Zhang Yuhao, Zhang Yuhui, Bolton Jason, Manning Christopher D. Stanza: A Python Natural Language Processing Toolkit for Many Human Languages // Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, ACL 2020, Online, July 5-10, 2020. 2020. 101–108.

Raffel Colin, Shazeer Noam, Roberts Adam, Lee Katherine, Narang Sharan, Matena Michael, Zhou Yanqi, Li Wei, Liu Peter J. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer // J. Mach. Learn. Res. 2020. 21. 140:1–140:67.

Raganato Alessandro, Tiedemann Jörg. An Analysis of Encoder Representations in Transformer-Based Machine Translation // Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2018, Brussels, Belgium, November 1, 2018. 2018. 287–297.

Rama Taraka, Beinborn Lisa, Eger Steffen. Probing Multilingual BERT for Genetic and Typological Signals // Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020. 2020. 1214–1228.

Reddy Siva, Täckström Oscar, Petrov Slav, Steedman Mark, Lapata Mirella. Universal Semantic Parsing // Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. Copenhagen, Denmark: Association for Computational Linguistics, IX 2017. 89–101.

Reimers Nils, Gurevych Iryna. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks // Proceedings of the

2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019. 2019. 3980–3990.

Ren Xiang, Wu Zeqiu, He Wenqi, Qu Meng, Voss Clare R., Ji Heng, Abdelzaher Tarek F., Han Jiawei. CoType: Joint Extraction of Typed Entities and Relations with Knowledge Bases // Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017. 2017. 1015–1024.

Ruder Sebastian. An overview of gradient descent optimization algorithms // CoRR. 2016. abs/1609.04747.

Rumelhart David E, Hinton Geoffrey E, Williams Ronald J. Learning representations by back-propagating errors // nature. 1986. 323, 6088. 533–536.

Do Syntax Trees Help Pre-trained Transformers Extract Information? // . 2021. 2647–2661.

Salimans Tim, Kingma Diederik P. Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks // Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain. 2016. 901.

Santos Cícero Nogueira dos, Guimarães Victor. Boosting Named Entity Recognition with Neural Character Embeddings // Proceedings of the Fifth Named Entity Workshop, NEWS@ACL 2015, Beijing, China, July 31, 2015. 2015. 25–33.

Saphra Naomi, Lopez Adam. Language Models Learn POS First // Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP. Brussels, Belgium: Association for Computational Linguistics, XI 2018. 328–330.

Schuster Sebastian, Krishna Ranjay, Chang Angel, Fei-Fei Li, Manning Christopher D. Generating Semantically Precise Scene Graphs from Textual Descriptions for Improved Image Retrieval // Proceedings of the Fourth Workshop on Vision and Language. Lisbon, Portugal: Association for Computational Linguistics, IX 2015. 70–80.

Schuster Sebastian, Manning Christopher D. Enhanced english universal dependencies: An improved representation for natural language understanding tasks // Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16). 2016. 2371–2378.

Sejnowski Terrence J. The unreasonable effectiveness of deep learning in artificial intelligence // Proc. Natl. Acad. Sci. USA. 2020. 117, 48. 30033–30038.

Sennhauser Luzi, Berwick Robert. Evaluating the Ability of LSTMs to Learn Context-Free Grammars // Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP. Brussels, Belgium: Association for Computational Linguistics, XI 2018. 115–124.

Sennrich Rico, Haddow Barry, Birch Alexandra. Neural Machine Translation of Rare Words with Subword Units // Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers. 2016.

Silveira Natalia G. Designing syntactic representations for NLP: An empirical investigation. 2016.

Singh Anil Kumar. Natural Language Processing for Less Privileged Languages: Where do we come from? Where are we going?

// Third International Joint Conference on Natural Language Processing, IJCNLP 2008, Hyderabad, India, January 7-12, 2008. 2008. 7–12.

Søgaard Anders, Lhoneux Miryam de, Augenstein Isabelle. Nightmare at test time: How punctuation prevents parsers from generalizing // Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2018, Brussels, Belgium, November 1, 2018. 2018. 25–29.

Stanovsky Gabriel, Ficler Jessica, Dagan Ido, Goldberg Yoav. Getting more out of syntax with props // arXiv preprint arXiv:1603.01648. 2016.

Strzyz Michalina, Vilares David, Gómez-Rodríguez Carlos. Sequence Labeling Parsing by Learning across Representations // Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers. 2019a. 5350–5357.

Strzyz Michalina, Vilares David, Gómez-Rodríguez Carlos. Viable Dependency Parsing as Sequence Labeling // Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers). 2019b. 717–723.

Sun Mingming, Hua Wenyue, Liu Zoey, Wang Xin, Zheng Kangjie, Li Ping. A Predicate-Function-Argument Annotation of Natural Language for Open-Domain Information eXpression // Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). 2020. 2140–2150.

Sundermeyer Martin, Ney Hermann, Schlüter Ralf. From Feedforward to Recurrent LSTM Neural Networks for Language Modeling //

IEEE ACM Trans. Audio Speech Lang. Process. 2015. 23, 3. 517–529.

Sutskever Ilya, Vinyals Oriol, Le Quoc V. Sequence to Sequence Learning with Neural Networks // Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada. 2014. 3104–3112.

Szegedy Christian, Vanhoucke Vincent, Ioffe Sergey, Shlens Jon, Wojna Zbigniew. Rethinking the inception architecture for computer vision // Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. 2818–2826.

Tai Kai Sheng, Socher Richard, Manning Christopher D. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks // Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). Beijing, China: Association for Computational Linguistics, VII 2015. 1556–1566.

Talman Aarne, Chatzikyriakidis Stergios. Testing the Generalization Power of Neural Network Models across NLI Benchmarks // Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@ACL 2019, Florence, Italy, August 1, 2019. 2019. 85–94.

Talman Aarne, Yli-Jyrä Anssi, Tiedemann Jörg. Sentence embeddings in NLI with iterative refinement encoders // Nat. Lang. Eng. 2019. 25, 4. 467–482.

Tenney Ian, Das Dipanjan, Pavlick Ellie. BERT Rediscovers the Classical NLP Pipeline // Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence,

Italy, July 28- August 2, 2019, Volume 1: Long Papers. 2019. 4593–4601.

Tesnière Lucien. Éléments de syntaxe structurale. 1959.

Tieleman Tijmen, Hinton Geoffrey, others . Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude // COURSERA: Neural networks for machine learning. 2012. 4, 2. 26–31.

Tiktinsky Aryeh, Goldberg Yoav, Tsarfaty Reut. pybart: Evidence-based syntactic transformations for ie // arXiv preprint arXiv:2005.01306. 2020.

Vaswani Ashish, Shazeer Noam, Parmar Niki, Uszkoreit Jakob, Jones Llion, Gomez Aidan N., Kaiser Lukasz, Polosukhin Illia. Attention is All you Need // Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA. 2017. 5998–6008.

Verwimp Lyan, Pelemans Joris, hamme Hugo Van, Wambacq Patrick. Character-Word LSTM Language Models // Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers. 2017. 417–427.

Vig Jesse, Belinkov Yonatan. Analyzing the Structure of Attention in a Transformer Language Model // Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP. Florence, Italy: Association for Computational Linguistics, VIII 2019. 63–76.

Vikner Carl, Vikner Sten. Hierarchical morphological structure and ambiguity // Merete Birkelund, Maj-Britt Mosegaard Hansen and Coco Norén, eds. 2008. 541–560.

Vinyals Oriol, Fortunato Meire, Jaitly Navdeep. Pointer Networks // Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada. 2015. 2692–2700.

Vries Wietse de, Wieling Martijn, Nissim Malvina. Make the Best of Cross-lingual Transfer: Evidence from POS Tagging with over 100 Languages // Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022. 2022. 7676–7685.

Wang Alex, Pruksachatkun Yada, Nangia Nikita, Singh Amanpreet, Michael Julian, Hill Felix, Levy Omer, Bowman Samuel R. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems // Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada. 2019a. 3261–3275.

Wang Alex, Singh Amanpreet, Michael Julian, Hill Felix, Levy Omer, Bowman Samuel R. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding // Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2018, Brussels, Belgium, November 1, 2018. 2018. 353–355.

Wang Haohan, Sun Da, Xing Eric P. What if We Simply Swap the Two Text Fragments? A Straightforward yet Effective Way to Test the Robustness of Methods to Confounding Signals in Nature Language Inference Tasks // The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial In-

telligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019. 2019b. 7136–7143.

Wang Peilu, Qian Yao, Soong Frank K., He Lei, Zhao Hai. Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Recurrent Neural Network // CoRR. 2015. abs/1510.06168.

Wang Shaonan, Zhang Jiajun, Zong Chengqing. Learning Sentence Representation with Guidance of Human Attention // Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017. 2017. 4137–4143.

Warstadt Alex, Bowman Samuel R. Grammatical Analysis of Pretrained Sentence Encoders with Acceptability Judgments // CoRR. 2019. abs/1901.03438.

Webster Jonathan J., Kit Chunyu. Tokenization As The Initial Phase In NLP // 14th International Conference on Computational Linguistics, COLING 1992, Nantes, France, August 23-28, 1992. 1992. 1106–1110.

White Aaron Steven, Reisinger Drew, Sakaguchi Keisuke, Vieira Tim, Zhang Sheng, Rudinger Rachel, Rawlins Kyle, Van Durme Benjamin. Universal decompositional semantics on universal dependencies // Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. 2016. 1713–1723.

Wieting John, Bansal Mohit, Gimpel Kevin, Livescu Karen. Towards Universal Paraphrastic Sentence Embeddings // 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings. 2016.

Wilcox Ethan, Levy Roger, Futrell Richard. Hierarchical Representation in Neural Language Models: Suppression and Recovery of

Expectations // Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP. Florence, Italy: Association for Computational Linguistics, VIII 2019. 181–190.

Williams Adina, Drozdov Andrew, Bowman Samuel R. Do latent tree learning models identify meaningful structure in sentences? // Transactions of the Association of Computational Linguistics. 2018a. 6. 253–267.

Williams Adina, Nangia Nikita, Bowman Samuel R. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference // Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers). 2018b. 1112–1122.

Wirén Mats, Ek Adam, Kasaty Anna. Annotation Guideline No. 7: Guidelines for annotation of narrative structure // Journal of Cultural Analytics. 2020. 4, 3. 11772.

Wolf Thomas, Debut Lysandre, Sanh Victor, Chaumond Julien, Delangue Clement, Moi Anthony, Cistac Pierric, Rault Tim, Louf Rémi, Funtowicz Morgan, Brew Jamie. HuggingFace's Transformers: State-of-the-art Natural Language Processing // CoRR. 2019. abs/1910.03771.

Wolf Thomas, Debut Lysandre, Sanh Victor, Chaumond Julien, Delangue Clement, Moi Anthony, Cistac Pierric, Rault Tim, Louf Rémi, Funtowicz Morgan, Davison Joe, Shleifer Sam, Platen Patrick von, Ma Clara, Jernite Yacine, Plu Julien, Xu Canwen, Scao Teven Le, Gugger Sylvain, Drame Mariama, Lhoest Quentin, Rush Alexander M. Transformers: State-of-the-Art Natural Language Processing // Proceedings of the 2020 Conference on Empirical Meth-

ods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020. 2020. 38–45.

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

Xu Yang, Liu Jiawei. Implicitly incorporating morphological information into word embedding // arXiv preprint arXiv:1701.02481. 2017.

Xue Linting, Constant Noah, Roberts Adam, Kale Mihir, Al-Rfou Rami, Siddhant Aditya, Barua Aditya, Raffel Colin. mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer // Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021. 2021. 483–498.

Yang Baosong, Wang Longyue, Wong Derek F., Chao Lidia S., Tu Zhaopeng. Assessing the Ability of Self-Attention Networks to Learn Word Order // Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. Florence, Italy: Association for Computational Linguistics, VII 2019. 3635–3644.

Yasunaga Michihiro, Kasai Jungo, Radev Dragomir R. Robust Multilingual Part-of-Speech Tagging via Adversarial Training // Pro-

ceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers). 2018. 976–986.

Yu Xiang, Vu Ngoc Thang. Character Composition Model with Convolutional Neural Networks for Dependency Parsing on Morphologically Rich Languages // Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 2: Short Papers. 2017. 672–678.

Zeman Daniel, Popel Martin, Straka Milan, Hajic Jan, Nivre Joakim, Ginter Filip, Luotolahti Juhani, Pyysalo Sampo, Petrov Slav, Potthast Martin, Tyers Francis M., Badmaeva Elena, Gokirmak Memduh, Nedoluzhko Anna, Cinková Silvie, Jr. Jan Hajic, Hlavácová Jaroslava, Kettnerová Václava, Uresová Zdenka, Kanerva Jenna, Ojala Stina, Missilä Anna, Manning Christopher D., Schuster Sebastian, Reddy Siva, Taji Dima, Habash Nizar, Leung Herman, Marneffe Marie-Catherine de, Sanguinetti Manuela, Simi Maria, Kanayama Hiroshi, Paiva Valeria de, Droganova Kira, Alonso Héctor Martínez, Çöltekin Çagri, Sulubacak Umut, Uszkoreit Hans, Macketanz Vivien, Burchardt Aljoscha, Harris Kim, Marheinecke Katrin, Rehm Georg, Kayadelen Tolga, Attia Mohammed, El-Kahky Ali, Yu Zhuoran, Pitler Emily, Lertpradit Saran, Mandl Michael, Kirchner Jesse, Alcalde Hector Fernandez, Strnadová Jana, Banerjee Esha, Manurung Ruli, Stella Antonio, Shimada Atsuko, Kwak Sookyoung, Mendonça Gustavo, Lando Tatiana, Nitisaroj Rattima, Li Josie. CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies // Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Vancouver, Canada, August 3-4, 2017. 2017. 1–19.

Zhang Sheng, Rudinger Rachel, Van Durme Benjamin. An evalua-

tion of PredPatt and open IE via stage 1 semantic role labeling // IWCS 2017—12th International Conference on Computational Semantics—Short papers. 2017.

Zhao Peilin, Zhang Tong. Stochastic Optimization with Importance Sampling for Regularized Loss Minimization // Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015. 37. 2015. 1–9. (JMLR Workshop and Conference Proceedings).