Relation Extraction with Attention-based Transfer Learning

RheinMain University of Applied Sciences
Faculty of Design Computer Science Media
Master Computer Science

Presented to obtain the Master of Science (M.Sc.)

Markus Eberts
April 17, 2019

Wiesbaden

Advisor: Prof. Dr. Adrian Ulges

Co-Advisor: Prof. Dr. Dirk Krechel

Erklärung gem. ABPO, Ziff. 6.4.3

Ich versichere, dass ich die Master-Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Wiesbaden
April 17, 2019

Markus Eberts

Erklärung zur Verwendung der Masterthesis

Hiermit erkläre ich mein Einverständnis mit den im Folgenden aufgeführten Verbreitungsformen dieser Master-Arbeit:

Verbreitungsform	**Ja**	**Nein**
Einstellung der Arbeit in die Hochschulbibliothek mit Datenträger	×	
Einstellung der Arbeit in die Hochschulbibliothek ohne Datenträger	×	
Veröffentlichung des Titels der Arbeit im Internet	×	
Veröffentlichung der Arbeit im Internet	×	

Wiesbaden
April 17, 2019

Markus Eberts

Abstract

The Generative Pre-training Transformer (GP-Transformer), a deep attention-based neural network, has recently achieved state-of-the-art results in many natural language processing tasks. The model is trained in an unsupervised manner on a language modeling objective and produces context-aware embeddings, which are then fine-tuned on the target domain. In this work, the pre-trained GP-Transformer is employed for relation extraction. Using the GP-Transformer as the core component of task-specific models, three relation extraction subtasks are addressed: “Relation Classification”, “Few-Shot Relation Classification” and “Joint Entity and Relation Extraction”: (1) In relation classification, the task is to predict the relation that is expressed in a sentence between a given entity pair. This work explores several methods to encode the target entities for the GP-Transformer and also investigates how unsupervised pre-training improves the generalization capabilities of the model. In experiments on the SemEval dataset, the model achieves competitive results with a macro-F1 score of 87.34. (2) In few-shot relation classification the objective is to generalize to new relations not encountered during training. Here the GP-Transformer is employed in a prototypical network to classify relations based on only a few examples of the target relation. The model achieves state-of-the-art results on the FewRel dataset, including an accuracy of 92.11% in the 5-way 5-shot and 72.51% in the 10-way 1-shot setting. It is shown that the GP-Transformer especially outperforms previous approaches when provided with just a single example per target relation. (3) In joint entity and relation extraction, entities must be predicted alongside the relation using a joint model. This work investigates a novel approach that is based on an exhaustive search over candidate pairs. By employing the GP-Transformer as an elaborate feature extractor, it is demonstrated that a fast exhaustive search is feasible even on a large search space and that adding negative samples is crucial for the model’s performance.

Finally, in a case study the use of the GP-Transformer for German text is explored: To learn the syntax and the semantics of German text, the model is pre-trained on a large and diverse German corpus. It is shown that the model is able to generate plausible German text and to produce context-aware embeddings. When employed for joint entity and relation extraction, the pre-trained model demonstrates remarkably generalization capabilities when trained on a small-scale German-language dataset.

Contents

Chapter 1

Introduction

Neural networks have been the most important driving factor for machine learning development in the recent years. To solve a target problem, they are typically trained in a supervised manner: By presenting the neural network a set of labeled samples, it learns to recognize the recurring patterns that characterize the problem so as to map the network's input to the desired output. Since a small set of labeled samples can hardly cover the characteristics of a complex problem, neural networks are often trained on a vast amount of data. However, the annotation of training data is an expensive and time consuming task commonly done by human annotators. Therefore such large quantities of labeled samples may not be available in the target domain and can only be obtained at great expense. To circumvent this requirement, an approach called *transfer learning* emerged: The neural network is pre-trained on data that is available in a sufficient amount and the gained knowledge is then transferred to the target domain or task.

Natural language processing (NLP) covers tasks that are required for computers to structure and understand human language, such as machine translation, sentiment analysis or coreference resolution. In NLP transfer learning has recently gained traction with the introduction of learned continuous real-valued word representations called *word embeddings*. Word embeddings are trained in an unsupervised manner solely on the basis of unstructured text and therefore require no manual annotation. Since word embeddings capture the semantics of words, they can be employed in task-specific downstream models to improve generalization capabilities. However, they come with one crucial limitation: Because word embeddings are fixed they remain the same regardless of the specific sentence context where they occur. This is especially adverse for polysemous and homonymous words, i.e. words that can have varying context-dependent meanings. In a recent development, more complex model's are pre-trained on a language modeling objective and transferred to the target domain (e.g. Radford et al. 2018, Peters et al. 2018, Devlin et al. 2018). In contrast to static word embeddings, these models are able to produce context-aware word representations and capture more sophisticated relations between words. Fine-tuning pre-trained models on the target task has lead to significant performance improvements in a large variety of NLP tasks.

A model based on such language model pre-training is investigated in this work and applied to one of the key components of natural language processing: the *extraction* and *classification* of *semantic relations* between *entities* in a particular sentence. The term entity refers to names such as those of persons, organizations, etc., but also to other noun phrases that are somehow related in a sentence. An example sentence with multiple entities and the relations that are expressed between them is illustrated in Figure 1.1.

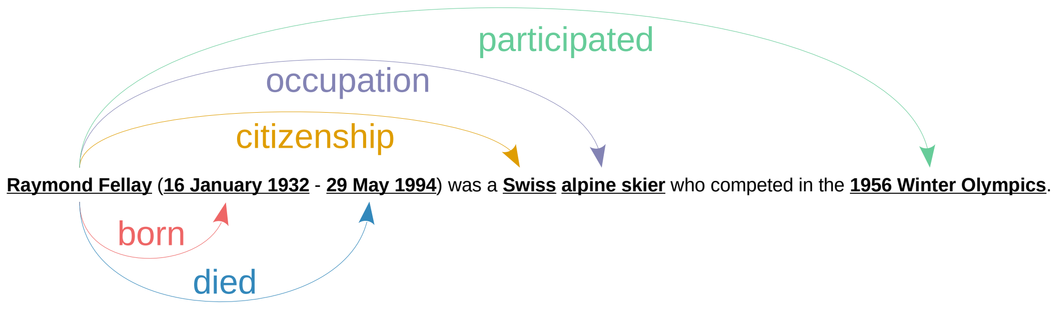

participated
occupation
citizenship
Raymond Fellay (16 January 1932 - 29 May 1994) was a Swiss alpine skier who competed in the 1956 Winter Olympics.
born
died

Figure 1.1: Relation extraction aims at the extraction of semantic relations between two entities.

Applications of relation extraction are manifold: Many companies have access to an enormous amount of unstructured text data, which includes text freely available on the world wide web and internal company data. Information that is useful for the companies objectives is scattered throughout this data and can only be collected and filtered manually with high effort. Among others, relation extraction provides a way to automatically structure text in the form of knowledge graphs (Xu, Li, et al. 2014, Byrne 2006, Toutanova et al. 2015). This information in turn plays an integral role in many information extraction tasks: From medical (Chang Wang and Fan 2014) and general (C. Wang et al. 2012) question and answering systems ("Which therapy was mentioned for a certain diagnosis?", "Was a specific damage ever been detected on a product of our company?", ...) to opinion mining (Kobayashi, Inui, and Matsumoto 2007), relation extraction is frequently required to gain valuable insights into various types of problems.

In this work, the GP-Transformer (Radford et al. 2018), a deep and attention-based model that is pre-trained on a language modeling objective, is employed for three relation extraction subtasks: "Relation Classification", "Few-shot Relation Classification" and "Joint Entity and Relation Extraction". In experiments on two public standard datasets the following questions are particularly addressed:

- The GP-Transformer was pre-trained by Radford et al. (2018) on an English-language corpus and can therefore only be utilized for English data. How well does the GP-Transformer perform when pre-trained on a German corpus and is employed for a small-scale German-language relation extraction dataset?

- In relation classification, the model is required to predict the relation between a specific entity pair in the sentence. How can the GP-Transformer be adjusted to take the target entities into account and to what extend does the unsupervised pre-training benefits generalization?
- How can the GP-Transformer be adjusted to work in a few-shot scenario, i.e. a setting where the model is required to predict a relation based on only a few samples per relation type, and how well does it perform compared to other state-of-the-art models?
- Can the GP-Transformer be used as an elaborate feature extractor in a fast exhaustive search to jointly not only extract the relation type but also localize the entities between which the relation holds?

This work is divided into five chapters: In Chapter 2 the GP-Transformer as well as its main component, the attention mechanism, is introduced. This chapter also includes the pre-training of the GP-Transformer on a diverse German dataset as well as an investigation into the model's capabilities in producing context-aware word embeddings. Chapter 3 continues with an introduction into relation extraction and introduces two relation extraction datasets. Chapter 4 (relation classification), Chapter 5 (few-shot relation classification) and Chapter 6 (joint entity and relation extraction) deal with the three relation extraction subtasks explored in this work and investigate how task-specific models that include the GP-Transformer as its core part can be employed for these tasks. Finally, Chapter 7 reasons on the applicability of the GP-Transformer for relation extraction and gives an outlook into possible future research on this topic.

Chapter 2

The Generative Pre-Training Transformer

While basic recurrent neural networks are a viable choice for NLP tasks due to their sequential structure, they struggle at processing long-term information. The problem was reduced with the introduction of gating mechanisms, such as in long short-term memory (LSTM, see Hochreiter and Schmidhuber 1997) or gated recurrent unit (GRU, see Cho et al. 2014) neural networks, which control the information flow in and out of the RNN's hidden state. Despite these improvements, however, RNNs tend to focus on the most recent inputs, in the worst case loosing important information which was obtained multiple tokens before. This behavior is especially bad for tasks such as machine translation, where models need to attend to context-specific occurrences of words in the input sentence in order to translate them.

Recently, major successes were achieved by incorporating a new approach into recurrent neural networks that enables them to focus on words over long ranges of the input sentence: The *Attention Mechanism* (Bahdanau, Cho, and Bengio 2014). Traditionally, machine translation models comprise of an encoder and decoder part. In the case of RNNs, the encoder compresses the information of the source sentence into a single vector representation. After that, the decoder uses this compressed version of the input in order to output the translated words one by one. By adding attention, the decoder does not only process the last hidden state of the encoder to output the next word, but incorporates every hidden state observed so far by computing a weighted sum over the hidden states based on their similarity with the so far translated sequence.

With the success of the attention mechanism in mind, Vaswani et al. (2017) proposed a new model for machine translation that is based entirely on attention, getting rid of recurrence altogether. This model, known as the *Transformer*, relies on a variant of attention called *self-attention* by relating words in the input sentence to other words of the same input. Self-attention is applied in both the encoder and decoder in addition to a decoder-encoder attention. Additionally, Vaswani et al. made major contributions by proposing new attention extensions, such as the scaled dot-product and multi-head attention, which will both be discussed in this chapter. They achieved a new state-of-the art in machine translation at a lower training cost compared to recurrent neural networks, which are known to require more training time compared to their sequential counterparts and are hard to parallelize beyond batches of samples.

To employ the Transformer model for tasks that do not rely on an encoder-decoder architecture, such as relation extraction, Radford et al. (2018) showed that the Transformer decoder alone, a variant first used by Liu et al. (2018) for summarizing Wikipedia sentences, achieves state-of-the-art results in many natural language processing tasks: This includes natural language inference, Q&A and semantic similarity. By pre-training the Transformer decoder in an unsupervised manner on a common language modeling objective, predicting the next word of a sentence given the previous words, they show that only little fine-tuning on the target task domain and no task specific architecture is required to outperform previous models. This model is denoted as GP-Transformer (Generative Pre-training Transformer) from now on. They key goal of this work is to explore the use of the GP-Transformer for relation extraction, a domain which was not explored in the original work by Radford et al. As a basis, transfer learning and particularly the GP-Transformer is discussed in-depth in this chapter. Besides an discussion of the GP-Transformer's architecture it is also shown that the model produces context-aware embeddings and is able to generate syntactically and semantically plausible text when pre-trained on a large German-language corpus.

2.1 Recent Transfer Learning Approaches in NLP

Humans develop a natural understanding of language through practice and increasing knowledge throughout the years. This includes the ability to differentiate the meaning of words based on their context, the awareness of word similarity (e.g. synonyms) and the recognition of semantic relations between words. Computers on the other hand have no inherent knowledge about individual words. To them words are just sequences of characters with no associated meaning. Perhaps the most crucial factor of the advances which were made in natural language processing in the recent years is the unsupervised pre-training of dense word representations (usually in form of real-valued vectors), which both capture syntactic and semantic properties of words. These word representations are used as the input in a large variety of different machine learning models to solve almost any NLP problem. More recent approaches, like the GP-Transformer which is employed in this work for relation extraction, go a step further and pre-train whole complex models in an unsupervised manner on large text corpora. These models are then *transfered* to a new domain and fine-tuned on the target task. In contrast to *fixed* word representations, they are able to learn more complex interactions between words and to distinguish the meaning of words based on the context they appear in, for example to resolve polysemes and homonyms. This section gives a brief overview of the advances in transfer learning for natural language processing in the recent years.

One-Hot Encoding

One baseline representation of words is the famous one-hot encoding, where each word of the vocabulary is mapped to a vector (with the length of the vocabulary size) where the value at the corresponding index is set to 1 (and all others to 0). Because words are treated as a distinct entity that share no overlapping properties, this leads to the aforementioned problems: With the one-hot encoding, each word is equally "distant" to all other words in the vocabulary, while in reality words can be more similar to one another because of certain properties (synonyms, relations, reflection, ...). For example, a desirable property of word encodings is that the vector representations of semantically similar words are close together in the d-dimensional encoding space, while words with no overlapping meaning lie far apart. Ultimately, this allows a relation extraction model to generalize better: It should not matter if "Raymond Fellay competed in the 1956 Winter Olympics" or "Raymond Fellay participated in the 1956 Winter Olympics", because both sentences have the same meaning and the usage of "participated" instead of "competed" does not alter the semantic relation between the two entities "Raymond Fellay" and "1956 Winter Olympics". Basically, word representations should capture prior knowledge of words, which are hard to obtain on an eventually small scale dataset that is used to train a machine learning system for a certain domain and target problem.

Fixed Word Embeddings

To counter the down sides of one-hot representations, an approach called word embeddings emerged. A popular implementation called Word2vec was proposed by Mikolov, Chen, et al. (2013). The term "embedding" describes the transfer of words into a d-dimensional feature space and usually refers to dense, real-valued feature vectors that also capture word semantics. Here the entries of the d-dimensional embedding (e.g. d=300) can be interpreted as weights, depicting the proportion of a specific property belonging to the word. Although Mikolov et al. were not the first to represent words as a distributed vector, they made the training of word embeddings feasible even on huge datasets and vocabularies (see Mikolov, Chen, et al. 2013 and Mikolov, Sutskever, et al. 2013).

There are two variants of Word2Vec, CBOW and Skip-gram. The core idea of both is that words are related if they appear in a similar context. This is especially clear for interchangeable words: Because "participated" and "competed" have roughly the same meaning they appear in similar contexts. Also terms like "cat" and "dog", albeit not synonyms, probably appear in similar contexts and are therefore related somehow. CBOW and Skip-gram employ different language modeling objectives: In the Skip-gram network architecture, the training objective is to predict the probability that a word from the vocabulary is a randomly selected nearby word. The training objective of the CBOW architecture is essentially the reverse of the Skip-Gram model: Given the surrounding words (the *context*), predict the probability of each word of the vocabulary to be the target word. A favorable property of Word2Vec and related models like GloVe (Pennington, Socher, and Manning 2014) is that they can be trained in an unsupervised manner. As a consequence they do not rely on an expensive annotation process but can be trained solely on an unlabeled corpus of text data, which is fortunately freely available in the world wide web in a more than sufficient amount.

Context-Aware Word Embeddings

Although word embeddings produced by Word2Vec or similar techniques lead to significant performance gains in many NLP tasks, they are usually fixed and therefore remain the same independent of the domain and specific sentence context. More complex interactions between words and additional context sensitive information can be utilized by transferring whole pre-trained models like the GP-Transformer to a new domain or task. This is especially useful in order to disambiguate polysemous and homonymous words, which can stand for different things in different contexts. A common example for computer scientists is the lemma *tree*, which can stand for multiple concepts (e.g. the botany tree or a data structure) depending on the surrounding context. When searching for *tree* in WordNet (Fellbaum 1998), a large lexical database, multiple senses are returned:

- “A tall perennial woody plant having a main trunk and branches forming a distinct elevated crown [...]”
- “Tree diagram (a figure that branches from a single root) [...]”
- “Sir Herbert Beerbohm Tree (English actor and theatrical producer noted for his lavish productions of Shakespeare (1853-1917))”
- “Force a person or an animal into a position from which he cannot escape”
- “tree, shoetree (stretch (a shoe) on a shoetree)”
- ...

Another example is the word *bank*, which may describe (among other things) a river or a financial bank dependent on the context. Actually the meaning or sense of a large portion of words we use in our everyday language varies dependent on the specific context. Besides being able to capture context-dependent information, pre-trained models were shown to converge faster, require fewer training samples and ultimately generalize better to new data (e.g. in Peters et al. (2018) or Radford et al. (2018)).

To learn context-aware word embeddings, whole models are nowadays trained on a language modeling objective and then employed and fine-tuned on the target task. One objective which is frequently applied in state-of-the-art transfer learning models is the *forward language modeling* objective, which computes the probability of observing a sequence of tokens $(t_1, t_2, ..., t_l)$ by calculating the probability of each token t_i with respect to the tokens $(t_1, t_2, ..., t_{i-1})$ occurring before t_i:

$$P(t_1, t_2, ..., t_l) = \prod_{i=1}^{l} P(t_i \mid t_1, t_2, ..., t_{i-1}) \tag{2.1}$$

Here l refers to the length of the so-called context, i.e. the sequence of words which is taken into account when the language model is trained on unstructured text. State-of-the-art models operate on increasingly bigger contexts, the GP-Transformer for example was pre-trained by Radford et al. (2018) on sequences of 512 tokens. With this, language models are capable of learning even more complex relations between words that potentially span over multiple sentences. Besides the GP-Transformer, which is described in detail in the upcoming sections, multiple other language models, which produce context-aware embeddings, have drawn attention in the recent years. A brief description of three other popular models, namely ELMo (Peters et al. 2018), ULMFit (Howard and Ruder 2018) and BERT (Devlin et al. 2018), is given in the following:

ELMo: ELMo (Peters et al. 2018), which stands for "Embeddings from Language Models", is a bidirectional LSTM-based language model, which is pre-trained on a large text corpus of 30 million sentences. Since the model is bidirectional, the final context-aware token embeddings are a function of the entire sentence: Two LSTMs (each with two layers) operate on the forward and backward direction of the sentence. While the former is trained to solve the forward language modeling objective of Equation 2.1, the latter essentially solves the reverse objective, i.e. by minimizing the probability of observing t_i given the sequence $(t_{i+1}, t_{i+2}, ..., t_l)$ of subsequent tokens. ELMo then concatenates the LSTMs' hidden states of the forward and backward direction for each token in both layers. To obtain the final context-aware embedding of a token, a task-specific weighting of the two layers and the initial token representations, which are formed by character-based convolutional filters to map out-of-vocabulary words, is learned. Instead of fine-tuning ELMo on the target task, Peters et al. employ state-of-the-art models for each evaluated task (e.g. coreference resolution and named entity extraction) and use the context-aware ELMo embeddings as input. By employing ELMo embeddings, the authors reported state-of-the-art results in all considered tasks, with error reductions from 6% to 20% over competitive baseline models. Moreover, they report a faster training convergence of downstream models and simultaneously require fewer training samples.

ULMFit: Howard and Ruder (2018) also pre-train a (three) layer bidirectional LSTM on the forward language modeling objective. In contrast to Peters et al. (2018), they do not employ task-specific downstream models but instead fine-tune the ULMFit (Universal Language Model Fine-tuning) model on each target task. After pre-training the model on the Wikitext-103 (Merity et al. 2016) corpus, it is fine-tuned on the target domain in two steps: (1) Fine-tune the model by language modeling on the task specific samples, essentially adapting the language model to the task domain. (2) Add an additional layer (e.g. linear softmax layer for classification tasks) and minimize the respective target task objective. In order to prevent the model to forget useful information which was obtained in the pre-training step during fine-tuning, Howard et al. propose two novel fine-tuning techniques, namely discriminative fine-tuning (Discr) and slanted triangular learning rates (STLR). While the former uses different learning rates in each LSTM layer, the latter first linearly increases the learning rate and then linearly decreases it during training. Howard et al. also gradually unfreeze each layer during fine-tuning, starting with the last. ULMFit outperformed state-of-the-art models in six different classification tasks with an error reduction of up to 24%.

BERT: BERT (Bidirectional Encoder Representations from Transformers) by Devlin et al. (2018) was recently released and is similar in architecture to the GP-Transformer. It consists of a Transformer decoder that is pre-trained on language modeling. In contrast to the GP-Transformer, BERT is trained, as the name suggests, in a bidirectional fashion by training the model simultaneously on two unsupervised language modeling objectives: (1) In the *masked LM* task, randomly selected tokens of the context are replaced with a special *Mask* token and must be predicted by the model. Since the Mask token is not used during fine-tuning and the model is required to generate embeddings for every position, selected tokens are also randomly replaced with other words or left unchanged. The authors claim, that with this adjustment the model learns context-aware embedding for every input token. (2) For the binary *next sentence prediction* task, the input context is divided into two segments. In 50% of the cases, the second segment is replaced with a random other one of the corpus. The task is to predict if the second segment actually follows the first (binary yes/no decision). Both tasks are pre-trained on a combination of the BooksCorpus (Zhu et al. 2015) and English Wikipedia. Devlin et al. (2018) report state-of-the-art results in 11 natural language processing tasks, even outperforming models such as the aforementioned ELMo, ULMFit and the unidirectional GP-Transformer.

2.2 GP-Transformer Model

The GP-Transformer, which is employed in this work for semantic relation extraction, relies on a component called *self-attention* and is pre-trained on a language modeling objective. They key components of the GP-Transformer, such as multi-head self-attention, position embeddings and byte pair input encoding, are described in this section.

2.2.1 Self-Attention

Relating words to each other is one of the fundamentals of machine learning models for natural language processing. In recurrent neural networks, words are related indirectly by incorporating them in a contextual hidden state based on previous words. Convolutional neural networks on the other hand merge information of words by applying numerous filter masks, stretching over an increasingly bigger context with each additional layer. Attention takes a more direct approach by relating words based on a relevance scoring between them and adjusts the representation of words with each attention application.

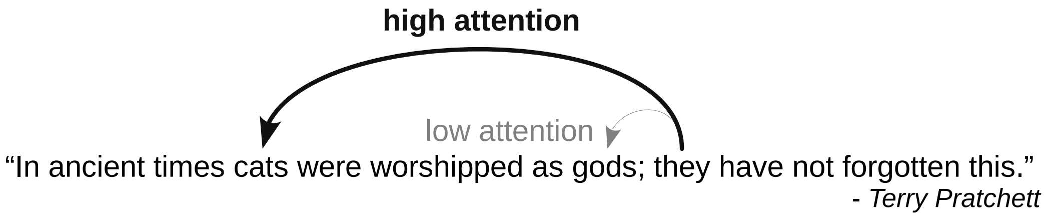
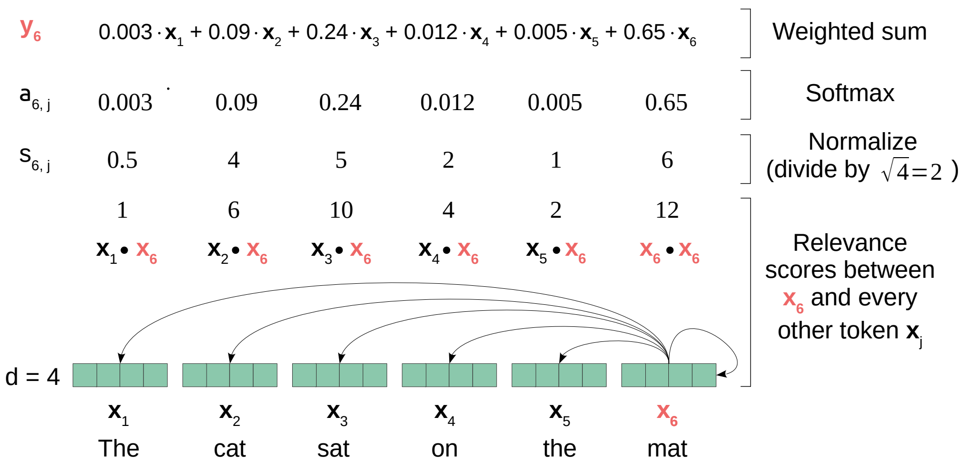

high attention
low attention
"In ancient times cats were worshipped as gods; they have not forgotten this."
- Terry Pratchett

Figure 2.1: Attention is used to focus on relevant words of the input context, here by detecting what the word "they" is referring to.

Figure 2.1 illustrates the idea behind attention: By paying attention to relevant words in the sentence, neural network's can detect important patterns of the sentence and ignore surrounding noise, here by detecting that the word "they" refers to "cats" and not "gods".

The attention function can be described as the mapping of a *query* (e.g. the word "they" in Figure 2.1) to an output by computing the weighted sum of the so-called



y6
0.003 · x1 + 0.09 · x2 + 0.24 · x3 + 0.012 · x4 + 0.005 · x5 + 0.65 · x6
Weighted sum
a6, j
0.003 0.09 0.24 0.012 0.005 0.65
Softmax
s6, j
0.5 4 5 2 1 6
Normalize (divide by √4=2)
1 6 10 4 2 12
x1 • x6 x2 • x6 x3 • x6 x4 • x6 x5 • x6 x6 • x6
Relevance scores between x6 and every other token xj
d = 4
x1 x2 x3 x4 x5 x6
The cat sat on the mat

Figure 2.2: Exemplary self-attention between vectors $\mathbf{x}_j \in \mathbb{R}^4$ (bottom). Depicted is the attention from the last word of the sentence, "mat" ($\mathbf{x}_6$), to all other words of the sentence, including "mat" itself. In a first step a relevance score between $\mathbf{x}_6$ and the embeddings of all other words of the sentence are computed by a dot product between the corresponding vectors. Next, the resulting scores are normalized by $\sqrt{d} = \sqrt{4} = 2$. The softmax function is applied to obtain weights in the range $[0, 1]$. The new representation of "mat", $\mathbf{y}_6$, which incorporates other context words, is then computed by a weighted sum over all vectors $\mathbf{x}_j$.

values based on the similarity between their corresponding so-called *keys* and the query (Vaswani et al. 2017). Generally, the query, keys and values can come from different sequences, e.g. the source and target sequence as in the decoder-encoder attention of the original machine translation Transformer. However, in the case of *self-attention*, which is employed in the GP-Transformer, attention is conducted on tokens of the same sequence (Figure 2.2). In the first layer of the GP-Transformer, the tokens are represented by their initial token embeddings $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_l$, where l denotes the length of the sequence and $\mathbf{x}_i \in \mathbb{R}^d$, with d referring to the dimensionality of the vectors. Note that the initial token embeddings are also learned during pre-training. Suppose we want to compute the attention function for an arbitrary token $\mathbf{x}_i$ of the input (e.g. the last token, as illustrated in Figure 2.2): In each attention layer, the representation of the token is changed in three steps. In the first step, the similarity between token $\mathbf{x}_i$ and all other tokens of the input is calculated based on a scoring function. This can be imagined as computing the relevance of each token with respect to the target token $\mathbf{x}_i$. The GP-Transformer model employs a simple dot-product scoring between the vectors that correspond to each token. The resulting relevance scores are further scaled by a factor of $\frac{1}{\sqrt{d}}$, resulting in:

$$s_{i,j} = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\sqrt{d}} \tag{2.2}$$

where "·" denotes the dot product between the two vectors $\mathbf{x}_i$ and $\mathbf{x}_j$. In the second step, the scores are normalized by a softmax function in order to obtain a weight of each token $\mathbf{x}_j$ with respect to token $\mathbf{x}_i$.

$$a_{i,j} = \text{softmax}(s_{i,j}) = \frac{e^{s_{i,j}}}{\sum_{j'=1}^{l} e^{s_{i,j'}}} \tag{2.3}$$

Finally, the output of attention, i.e. the new embedding of token $\mathbf{x}_i$, is obtained by taking a sum over all tokens of the input, including $\mathbf{x}_i$ itself, weighted by the corresponding weights $a_{i,j}$:

$$\mathbf{y}_i = \sum_{j=1}^{l} a_{i,j} \cdot \mathbf{x}_j \tag{2.4}$$

In practice, self-attention is carried out for all tokens at once by packing the token embeddings $\mathbf{x}_i$ into a matrix $X \in \mathbb{R}^{l \times d}$:

$$\text{Self-Attention}(X) = \text{softmax}\left(\frac{X \cdot X^T}{\sqrt{d}}\right) \cdot X \tag{2.5}$$

where the *softmax* function is applied row-wise to $\frac{X \cdot X^T}{\sqrt{d}}$. While the attention weights can theoretically be computed with arbitrary scoring functions, Vaswani et al. (2017) explain the reasoning of using scaled dot-product attention as being much faster and more space-efficient compared to other common functions like the additive attention used in Bahdanau, Cho, and Bengio (2014).

One of the key contributions of Vaswani et al. (2017) is the application of numerous attentions at once on the same input, each operating on a different representation subspace. They refer to this approach as the *multi-head attention* (Figure 2.3). For each head, there are actually three mappings: In the case of self-attention, X is first projected linearly to $Q_m = XW_m^Q$, $K_m = XW_m^K$ and $V_m = XW_m^V$ in each head m. The projected vector representation of a single token has the dimensionality $d_p = \frac{d}{z}$, where z refers to the number of attention heads. The linear projections are matrices $W_m^Q, W_m^K, W_m^V \in \mathbb{R}^{d \times d_p}$, which are learned during training. The attention in a single head m is computed for all tokens by:

$$Y_m = \text{softmax}\,(\frac{Q_m K_m^T}{\sqrt{d_p}})V_m \tag{2.6}$$

with $Y_m \in \mathbb{R}^{l \times d_p}$. The final token embeddings are then obtained by concatenating every head $(Y_1, ..., Y_z)$ and mapping the resulting vectors back to their original dimensionality d (here $\circ$ denotes vector concatenation):

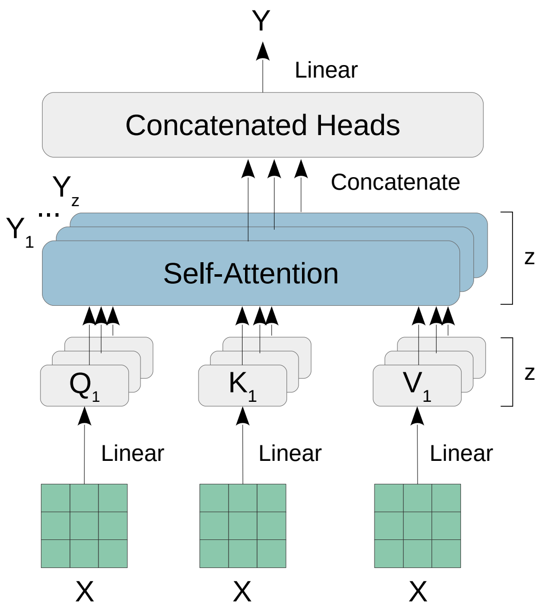


Figure 2.3: Multi-head attention is conducted in different representation subspaces of the input tokens $\mathbf{x}_i$, here packed into a matrix $X \in \mathbb{R}^{l \times d}$ for a sentence of length l (bottom). The token embeddings are first linearly projected. Self-attention then operates on each representation subspace separately. The resulting vectors (Y_1, ..., Y_z) are concatenated and mapped back to the original input dimensionality d by a linear layer.

$$Y = (Y_1 \circ Y_2 \circ ... \circ Y_z) \cdot W^o \qquad (2.7)$$

with $W^o \in \mathbb{R}^{z \cdot d_p \times d}$ and $Y \in \mathbb{R}^{l \times d}$. By applying these steps, each token embedding is adjusted by incorporating relevant other tokens into its own representation, ultimately leading to a more context-aware representation of the input tokens.

2.2.2 Architecture

The GP-Transformer architecture employed by Radford et al. (2018) for language modeling and subsequent finetuning utilizes multi-head self-attention as its key concept and is based on the original Transformer decoder component (Vaswani et al. 2017). As introduced in Liu et al. (2018), the encoder and decoder-encoder attention layer is removed from the model, leaving only multi-head self-attention and one fully connected feed-forward layer in a single *Transformer Block* (see Figure 2.4). Those Transformer Blocks are stacked on top of each other and the final model consists of $n = 12$ Blocks, each with one multi-head self-attention layer. Given a sequence of tokens $(t_1, t_2, ..., t_l)$, the tokens are mapped to their corresponding embedding vector by a lookup in the embedding matrix $E \in \mathbb{R}^{V \times d}$ (V denoting the vocabulary

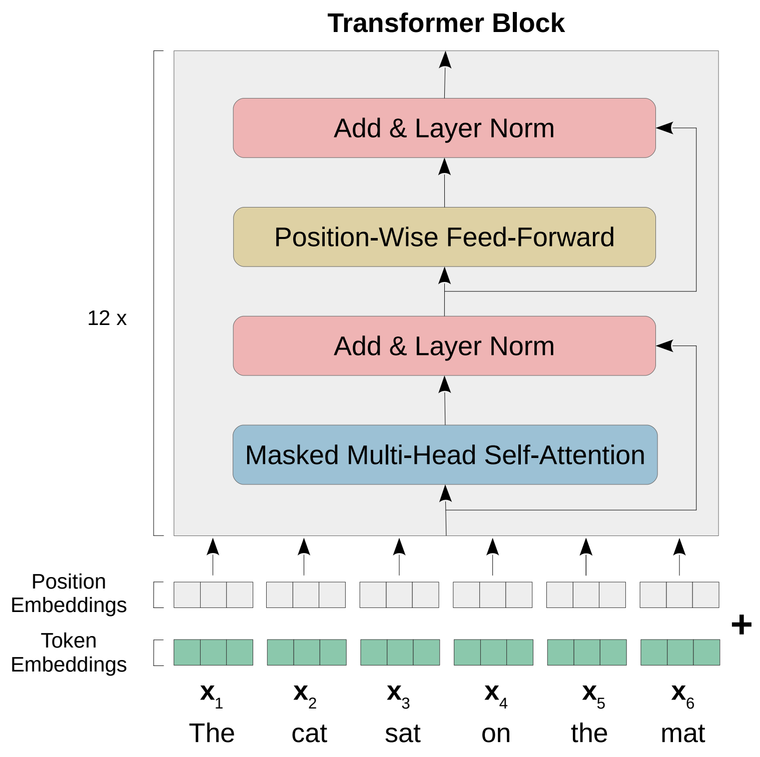


Figure 2.4: The GP-Transformer consists of 12 Transformer blocks and each block contains a masked multi-head self-attention and a position-wise fully-connected layer. Layer normalization is used throughout the model. Residual connections are also employed around the self-attention and feed-forward layers. The input token embeddings $\mathbf{x}_i$ are formed by adding position embeddings to the original token embeddings.

size) to $(\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_l)$. In contrast to recurrent neural networks, which are able to infer positional information through the successive processing of input data, the GP-Transformer naturally cannot utilize any information about the positional structure of the token sequence it operates on: In the self-attention layer, the token weights are calculated independent of their position in the sequence. However, the ordering of tokens plays an important role in many NLP tasks and also cannot be ignored in language modeling. To add positional information to the Transformer model, Radford et al. (2018) employ so-called position embeddings. For each absolute position in the input sequence, a separate position embedding $\mathbf{p}_i$ is learned during pre-training. For example, when the model operates on sequences of l tokens, l position embeddings are learned. These position embedding are simply added to the original token embeddings before the sequence is fed into the first Transformer block:

$$\mathbf{x}_i = \mathbf{u}_i + \mathbf{p}_i \tag{2.8}$$

With this extension, the Transformer model is able to infer the ordering of tokens in the sequence by the offset in the input vectors corresponding to each distinctive position. As described in Section 2.2.1, the resulting embeddings are then transformed by the multi-head self-attention layer of the first Transformer block, yielding new representations for each token. These are normalized by layer normalization (L. J. Ba, Kiros, and Hinton 2016) and then fed into a 2-layer, position-wise fully-connected feed-forward network with an inner *Gaussian Error Linear Unit* (GELU, see Hendrycks and Gimpel 2016) activation function:

$$\mathrm{FFN}(\mathbf{x}) = \mathrm{GELU}(\mathbf{x} \cdot W_1 + \mathbf{b}_1) \cdot W_2 + \mathbf{b}_2 \tag{2.9}$$

with $W_1 \in \mathbb{R}^{d \times d \cdot 4}$, $\mathbf{b}_1 \in \mathbb{R}^{d \cdot 4}$, $W_2 \in \mathbb{R}^{d \cdot 4 \times d}$ and $\mathbf{b}_2 \in \mathbb{R}^{d}$. The same transformations are applied for every token in the sequence, but differ in each Transformer block. Residual connections are added around the self-attention and fully-connected feed-forward layers. In addition to this, dropout is also applied after both the self-attention and feed-forward layer. This process is repeated for each Transformer Block, projecting the embeddings time and time again:

$$\begin{aligned} H^0 &= X \\ H^j &= \text{Transformer-Block}(H_{j-1}) \text{ for } j \in (1, ..., n) \end{aligned} \tag{2.10}$$

with $X, H^j \in \mathbb{R}^{l \times d}$. X refers to a matrix that contains all input embeddings $\mathbf{x}_i$, which are the sum of the corresponding token and position embeddings (Equation 2.8), for a sequence of length l.

Since the GP-Transformer is pre-trained on a large English corpus, Radford et al. apply language modeling after the last Transformer block. As already explained in Section 2.1, the forward language modeling objective predicts, for each sequence position, the likelihood of all tokens in the vocabulary to be the next token in the sequence. This is implemented by computing the dot-product between every token embedding $\mathbf{h}_i$, where $\mathbf{h}_i$ denotes the corresponding embedding after the final Transformer block, and the embeddings $\mathbf{u}_j$ of all tokens in the vocabulary. Then a softmax is applied row-wise to produce a probability distribution over the vocabulary:

$$P(t) = \mathrm{softmax}(H \cdot E^T) \tag{2.11}$$

with $H := H^n$, i.e. the token embeddings of an input sequence after the final Transformer block. To compare the predicted tokens at each position to the actual subsequent tokens, the GP-Transformer minimizes a negative log likelihood loss during pre-training. This loss is summed over all tokens of the context of length l:

$$\mathrm{Loss}_{\mathrm{LM}} = \sum_{i}^{l} -log\ P(t_i \mid t_{i-l}, ..., t_{i-1}) \tag{2.12}$$

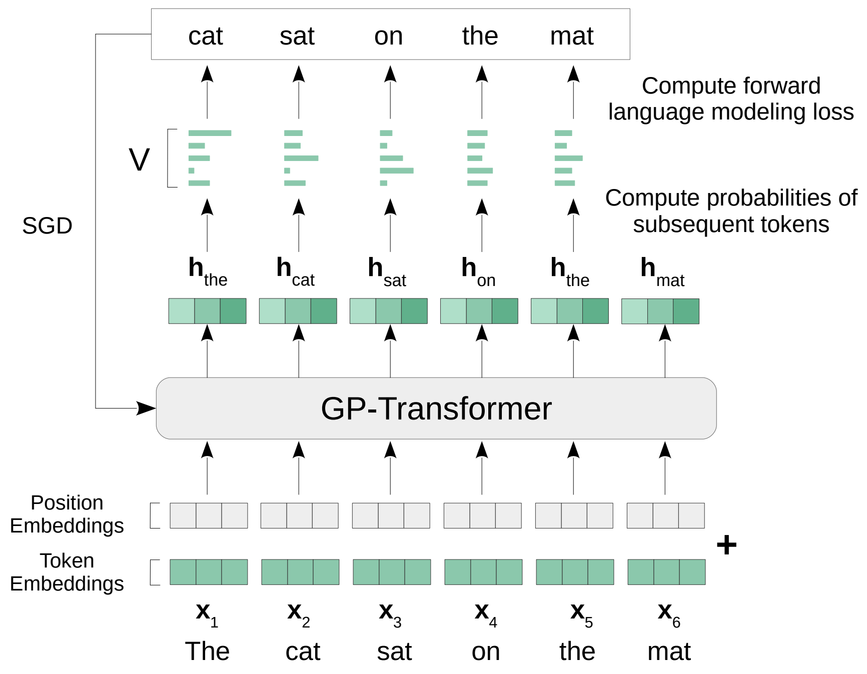

cat sat on the mat
Compute forward language modeling loss
V
SGD
Compute probabilities of subsequent tokens
h_{the} h_{cat} h_{sat} h_{on} h_{the} h_{mat}
GP-Transformer
Position Embeddings
Token Embeddings
+
x_1 x_2 x_3 x_4 x_5 x_6
The cat sat on the mat

Figure 2.5: Full GP-Transformer architecture. The GP-Transformer consists of 12 Transformer blocks (Figure 2.4). $\mathbf{h}_i$ (e.g. $\mathbf{h}_{\text{cat}}$)) denotes the context-ware embeddings after the final Transformer Block. During pre-training, a forward language modeling loss is minimized.

In the forward language modeling objective, each token is only allowed to focus on previous occurring tokens in the input, so the following tokens need to be masked out when computing the attention weights. This is done by setting the resulting dot-product scores (Equation 2.2) for all subsequent tokens to $-\infty$, leading to very small weights after the softmax application. This is also referred to as the *masked* multi-head self-attention (Figure 2.4). The complete model architecture is illustrated in Figure 2.5. Ultimately, the GP-Transformer learns to produce context-aware embeddings for the input tokens by pre-training it on the language modeling objective. These context-aware embeddings are also denoted as *Transformer embeddings* from now on.

2.2.3 Byte Pair Encoding

Traditional word embeddings like Word2Vec or GloVe encode each word of a predefined vocabulary independently. However, depending on the language, vocabulary sizes range from hundreds of thousands to even millions of words. For example, the vocabulary of everyday German language is estimated to contain up to 500.000 words (Duden 2017). When technical terms are also considered, this number easily grows to several million words. Because of that, machine learning developers face a tough decision when training word embeddings: To either only embed the most frequent words of a given language and map all other words to an unknown symbol, or to make the vocabulary as large as possible, therefore increasing the computational overhead and training duration. Especially in subword-rich languages like German, it is impossible to consider every possible word. Likewise mapping all out-of-vocabulary words to a single unknown symbol can lead to false predictions dependent on the target task.

Sennrich, Haddow, and Birch (2015) explored a new way of embedding out-of-vocabulary words by adapting the so-called *byte pair encoding* (BPE, see Gage 1994). Byte pair encoding is also employed in the GP-Transformer for vocabulary creation and to subsequently pre-process the input data. Originally, byte pair encoding is used for data compression by iteratively replacing the most frequent byte pair with a byte that is not part of the data. Similar to this, byte pair encoding for natural language processing iteratively replaces the most frequent consecutive symbols, starting with single characters, into a new symbol. So when starting with the sequence "ab dc ab" the symbols a and b are merged in the first iteration of the algorithm into a new symbol ab. This process is repeated till no symbols are left to be replaced, or a *fixed* vocabulary size V is reached. The resulting n-grams (n-gram denoting a contiguous sequence of n characters) then form the vocabulary and are usually embedded in the same way as vocabularies that consist of individual words. Ultimately, byte pair encoding leads to vocabularies that contain n-grams of flexible lengths, where frequent n-grams are merge more often and become longer. Common words therefore appear in the vocabulary as a single n-gram, while rare words are split into multiple n-grams.

To further illustrate the process, let us consider a set containing the three words $\mathcal{S}$={house*, mouse*, couch*}. Here, the token * marks the end of a word and is not part of the vocabulary. In the following example, the words should be byte pair encoded with five merge operations. In the beginning, the vocabulary consists only of the single characters that occur in $\mathcal{S}$, so $\mathcal{V}_{init} = \{c, e, h, m, o, s, u\}$. Then, the algorithm proceeds by merging the symbol pairs that occur most often into a single symbol:

Step 1 o u → ou {h **ou** s e *, m **ou** s e *, c **ou** c h *} (3 occurrences)
Step 2 s e → se {h ou **se** *, m ou **se** *, c ou c h *} (2 occurrences)
Step 3 ou se → ouse {h **ouse** *, m **ouse** *, c ou c h *} (2 occurrences)
Step 4 h ouse * → house * {**house** *, m ouse, c ou c h *} (1 occurrence)
Step 5 house * → house* {**house***, m ouse, c ou c h *} (1 occurrence)

After five merge operations, the final vocabulary is $\mathcal{V}_{final} = \{c, e, h, m, o, s, u, ou, se, ouse, house, house{*}\}$. Note that in case multiple pairs have the same frequency (as in step 2 for “s e” and “ou s”) it is an arbitrary decision which one to merge first. In the two most extreme cases, zero merge operations and a high number of merge operations, the vocabulary either consists of single characters only, or single characters combined with every dictionary word and previously merged n-grams.

After the creation of a byte pair vocabulary, token sequences (e.g. sentences) can be encoded into n-grams of the vocabulary. To do so, tokens (words) are first split into single characters. The learned operations are iteratively applied to merge characters into larger n-grams. In the above example, characters “o u” (step 1) are first merged, then characters “s e” (step 2) and so on. The process repeats till all learned operations have been applied and no n-grams are left to be merged.

By using byte pair encoding for the vocabulary creation, machine learning models such as the GP-Transformer are able to create representations of rare or unknown words by composing them based on their sub-words that are contained in the vocabulary. This is especially useful for languages such as German, where the composition of words is an integral component of the language: Even if a word like “Fahrkartenautomat” (travel ticket machine) is not part of the vocabulary, its meaning could possibly be inferred by utilizing the embeddings of “fahr”, “karte” and “automat”. This property makes token embeddings based on byte pair encoding, like the ones generated by the GP-Transformer, a good choice for applications that operate on domain specific vocabulary like company names or machine labels. Furthermore, models operating on byte pair encoded tokens are more robust to misspelled words, which would otherwise – in the case of typical word-embeddings – be mapped to an unknown token. Sennrich, Haddow, and Birch (2015) show that models that operate on byte pair encoded inputs outperform the baseline models on different machine translation tasks, especially for the translation of rare words.

2.3 The German GP-Transformer

The original English GP-Transformer was pre-trained by Radford et al. (2018) on the large English BooksCorpus (Zhu et al. 2015), which contains more than 7.000 unpublished books from different genres. By fine-tuning the model on a variety of target tasks, they achieved impressive results and outperformed the state-of-the-art in most cases. However, since the GP-Transformer is pre-trained on a purely English corpus, it is not possible to employ it for other languages with different vocabularies and syntax. In this work, the GP-Transformer is pre-trained on German language,

Corpus	#Documents	#Sentences	#Tokens	#BPE-Tokens
Wikipedia	2.055.300	49.237.266	978.820.539	1.185.976.120
Gutenberg	9345	25.937.791	616.639.208	697.738.025
FR	139.715	2.718.540	49.607.408	63.977.785
Total	2.204.360	77.893.597	1.645.067.155	1.947.691.930

Table 2.1: Statistics of the German corpora which the German GP-Transformer language model is pre-trained on. In total, the corpus consists of more than **1.6 billion** tokens and more than **1.9 billion** byte pair encoded tokens (see Section 2.3.1 for the creation of the German BPE vocabulary).

in order to be used for German joint entity and relation extraction (Chapter 6). The pre-training is implemented in Python[1] and uses the *PyTorch*[2] version of the GP-Transformer model by Hugging Face (2018). Note that the original OpenAI version (2018) was developed for TensorFlow[3] instead.

The German GP-Transformer is pre-trained on a combination of the *German Wikipedia*[4] corpus, the *Project Gutenberg*[5] corpus and the *Frankfurter Rundschau* corpus. The *German Wikipedia* dump was downloaded in July 2018 and includes over 2 million documents. The *Project Gutenberg* corpus contains over 58.000 free e-books in different languages, including German, from which 9345 documents are used in this work. Finally, the *Frankfurter Rundschau* corpus, a German news magazine, is composed of 139.715 news articles. Table 2.1 displays some statistics for each of the three corpora, including the sentence, token (e.g. words and punctuation marks) and final BPE token count (the creation of the German BPE vocabulary is described in Section 2.3.1). The *German Wikipedia* corpus is the largest of the three and contains nearly one billion tokens, followed by the *Gutenberg* corpus with more than 600 million tokens. Note that *Gutenberg*, while containing significantly less documents than the *German Wikipedia* dump, contains larger documents with 2775 sentences on average, compared to 23 sentences in Wikipedia. The smallest of the three corpora, *Frankfurter Rundschau*, contains roughly 50 million tokens and also features the fewest sentences per document (19 on average).

Besides size, the type of content differs between the corpora, with *Project Gutenberg* being the most distinct of the three. *Wikipedia* documents are very similar in structure and provide information about all kinds of topics, while the *Frankfurter Rundschau* news articles report current affairs. *Project Gutenberg* on the other hand contains

[1]https://www.python.org/
[2]https://pytorch.org/
[3]https://www.tensorflow.org/
[4]https://de.wikipedia.org
[5]https://www.gutenberg.org/

German Wikipedia *Bad Kreuznach ist eine Kurstadt und der Sitz der Kreisverwaltung des Landkreises Bad Kreuznach in Rheinland-Pfalz. Als Mittelzentrum mit Teilfunktionen eines Oberzentrums ist sie administratives, kulturelles und wirtschaftliches Zentrum einer Region mit mehr als 150.000 Einwohnern. Bad Kreuznach ist Sitz der Verbandsgemeinde Bad Kreuznach, gehört ihr als große kreisangehörige Stadt...*

Project Gutenberg *"Schön brav sein, Wotan, und sitzenbleiben!" sagte der gutgekleidete junge Mann und sah scheu nach den Vorübergehenden, ob sie etwa Verdacht schöpften. Wer kümmert sich drum, wenn einem Hund befohlen wird, daß er sich nicht rühren soll? – Der Neufundländer blickte seinen Herrn aus den restlos gutmütigen Augen traurig an, bettelte noch ein wenig mit der Pfote, fügte sich aber...*

Frankfurter Rundschau *PASSAU, 3. Juli (ap). An einer Demonstration gegen den geplanten weiteren Ausbau der Donau auf der Strecke zwischen Straubing und Vilshofen in Niederbayern haben sich am Wochenende in Passau etwa 1000 Menschen beteiligt. Sie kamen mit Traktoren und Fahrrädern zu der Kundgebung; der Vorsitzende des Bundes Naturschutz, Hubert Weinzierl, nannte den Ausbau des Flusses...*

Table 2.2: Text excerpts of each of the three corpora.

mostly literature, including novels, poets or theatre plays (e.g. the German translation of Shakespear's Hamlet). Table 2.2 contains text excerpts of each corpus.

The *PreTrainer* is developed in Python and is divided into three main components:

1. Corpora preprocessing and subsequent generation of a byte pair vocabulary.
2. Encoding of all documents and storage in a data structure for efficient access.
3. Actual training of the language model, including different kinds of evaluation.

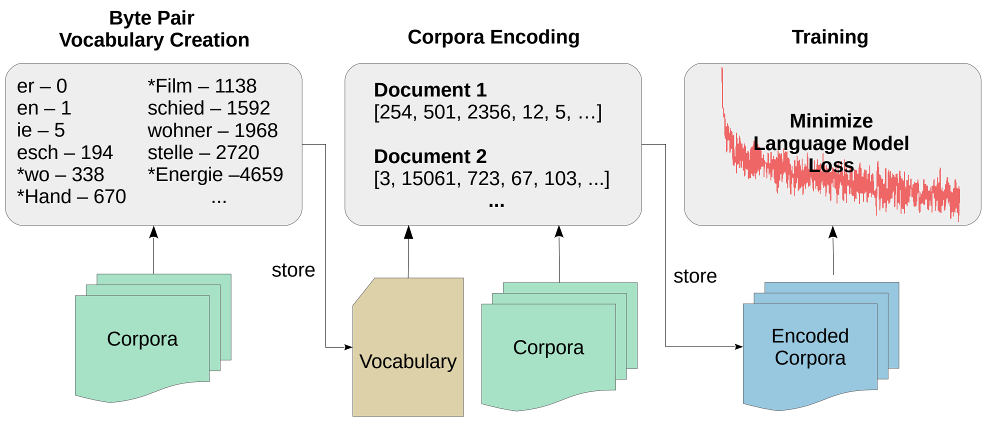


Figure 2.6: The three main components of the PreTrainer.

With the PreTrainer, it is possible to train a language model in any language by a command line interface. This includes all necessary steps, from the creation of a byte pair vocabulary, to corpora encoding and training. The main components of the PreTrainer and its utilization for German language model pre-training are described in this chapter.

2.3.1 Pre-Processing and Byte Pair Vocabulary Creation

To later transform the corpora text into common byte pair encoded n-grams, a German byte pair vocabulary must be created first. To do so, Google's SentencePiece[6] Python library is used in this work. SentencePiece is a language independent library for BPE vocabulary creation and subsequent encoding of sentences. For vocabulary generation, it expects a text file with one sentence per line as input. In a first step, the documents of each corpus are merged together and divided into single sentences by the NLP-Toolkit spacy[7]. Some documents, especially from Wikipedia, contain many characters that are not common in German, for example Chinese translations. According to a rough estimate, those characters accounted for over 20% of the final vocabulary. To avoid that, a whitelist of 476 common characters[8]) is used and any sentences are removed that contain characters not occurring in the list. The byte pair vocabulary was then created with the remaining sentences (about 60 million).

n-gram	ID
te	17
*hell	5057
*Hauptsache	18187
ietung	29810
ignale	32441

Table 2.3: N-gram examples.

Contrary to the English GP-Transformer, no lower-casing was applied to the text: Nouns and named entities usually start with a capital letter in German, so this information may be useful for joint entity and relation extraction to discriminate between verbs, which can hint at certain relations, and the entities between the relation exists. In contrast to the byte pair encoding that is used in the English GP-Transformer (described in Section 2.2.3), SentencePiece encodes the start of a word instead of the end. To achieve this, every whitespace character is replaced with a special symbol (visualized by a * in Table 2.3), and later merged with the subsequent token, if the respective n-gram occurs often enough. For the German GP-Transformer, the final vocabulary size V (including single characters) was set to 40k. SentencePiece assigns every n-gram of the vocabulary a unique ID (see Table 2.3). This ID is later used to retrieve the corresponding token embeddings.

[6]`https://github.com/google/sentencepiece`
[7]`https://spacy.io/`
[8]The single characters contained in the English BPE vocabulary employed in OpenAI (2018).

2.3.2 Corpus Encoding and Storage

Due to the large corpus size, the byte pair encoding of all documents demands a lot of time. Therefore, an on-the-fly encoding during pre-training would needlessly waste precious computing resources and time. Instead, each corpus is byte pair encoded before pre-training and saved in a HDF5[9] file. HDF5 is a hierarchical data format for storing large amounts of d-dimensional data. With the Python implementation PyTables[10], NumPy[11] arrays can be directly saved in and read from a HDF5 file.

In the encoding step, each corpus (*German Wikipedia*, *Project Gutenberg* and *Frankfurter Rundschau*) is byte pair encoded and stored in a separate HDF5 file. For every document in the corpus, a row with meta-information (document ID and token count) and a reference to the encoding NumPy array is created in the HDF5 file. The encoding array contains the n-gram IDs of the tokenized document (see Figure 2.7). To speed things up, a pool of processes reads documents from a queue and byte pair encodes them, while the main process writes the encoded documents to the HDF5 file. Figure 2.7 shows how an example sentence is split into common n-grams and converted into a sequence of vocabulary token IDs.

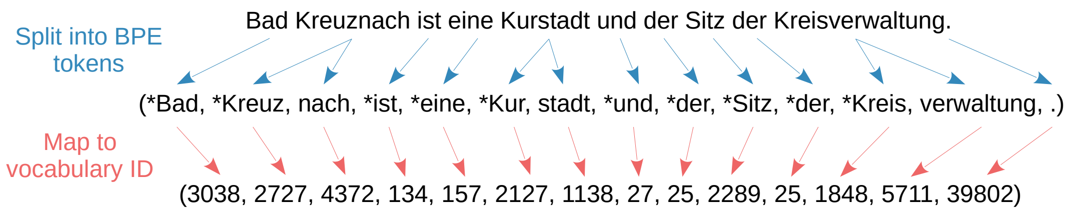

Split into BPE tokens
Map to vocabulary ID
Bad Kreuznach ist eine Kurstadt und der Sitz der Kreisverwaltung.
(*Bad, *Kreuz, nach, *ist, *eine, *Kur, stadt, *und, *der, *Sitz, *der, *Kreis, verwaltung, .)
(3038, 2727, 4372, 134, 157, 2127, 1138, 27, 25, 2289, 25, 1848, 5711, 39802)

Figure 2.7: Byte pair tokenization of an example sentence. Depicted is the actual tokenization performed by SentencePiece using the German BPE vocabulary (Section 2.3.1). The input sentence is first split into common BPE n-grams and then mapped to vocabulary IDs. During corpus encoding, each document is byte pair encoded and saved in a HDF5 file.

[9]`https://www.hdfgroup.org/`
[10]`https://www.pytables.org`
[11]`http://www.numpy.org/`

2.3.3 Language Model Training

After corpus encoding, the language modeling pre-training of the GP-Transformer can be conducted. This section describes the sampling of training token sequences as well as the training procedure, efficiency adjustments and the hyperparameters that where used during training.

Input Data

Language models, such as the GP-Transformer, are usually trained on batches of contiguous token sequences, commonly referred to as the context window, or just context. Let $C = \{D_1, D_2, ..., D_z\}$ denote the set of z byte pair encoded documents that the model is trained on and s_i the BPE token count of a document D_i. A document with s_i tokens contains $s_i - l + 1$ distinct contiguous sequences, where l denotes the context length and $s_i >= l$. The PreTrainer runs the training procedure for m epochs, with one epoch containing $k = \sum_{i=1}^{z} v_i$ samples of $v_i = \lfloor \frac{s_i}{l} \rfloor$ random contiguous sequences per document. In other words, k is the quantity of non-overlapping context windows across all documents. Note that the contiguous sequences are still randomly sampled, i.e. they begin at random offsets in the document.

In the beginning of pre-training a list of samples, which contain the corpus ID and the encoding reference for a particular document, is created, with v_i of such samples added to the list for each document D_i. Optionally, documents that contain fewer tokens than l are removed, otherwise v_i is set to 1 when $s_i < l$. In this case, the corresponding samples are later padded to the size of the context window. Next, random documents are removed from this list and put in a separate validation set. The validation set is later used to periodically compute the language modeling loss on sequences which were not part of the training data, and to generate text for visualization and evaluation purposes. The corpus and document IDs that are used for the training and validation set respectively are stored in *CSV* files for inspection and in case the training must be continued later on.

Training Procedure

The list of samples is shuffled in every epoch. During training, a separate producer process consumes the list, reads the encoding array from the corresponding HDF5 file for each sample and extracts a random contiguous sequence of length l from it. In case the size of the document is lower than l, the sequence starts at the beginning and is padded to the size of the context window by appending the special *Unknown* token, which is also used for real unknown tokens[12]. Otherwise the context window

[12] Since the padding tokens are always masked and no gradient is computed based on the padding, this introduces no problems regarding the embedding of real unknown tokens.

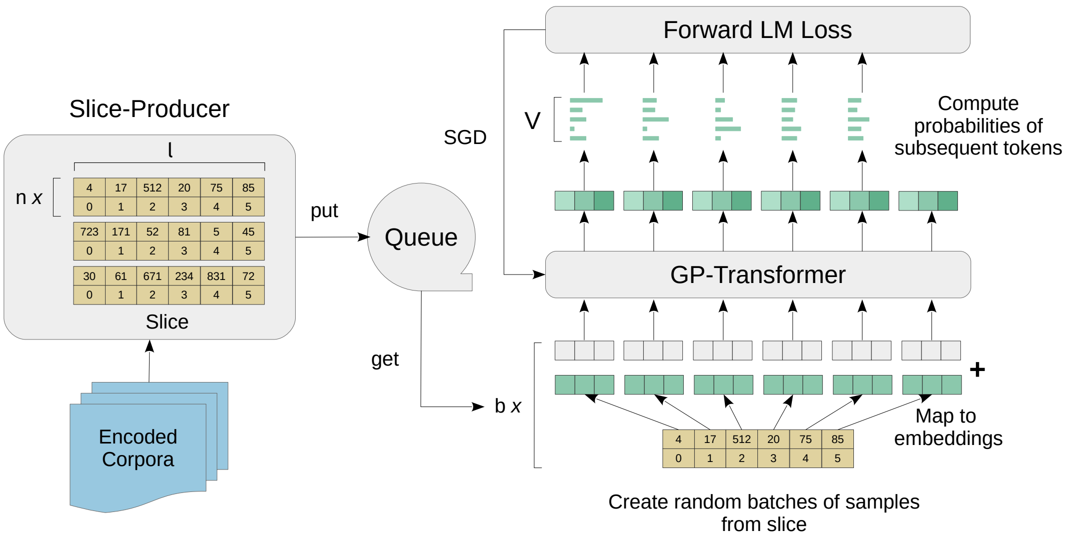


Figure 2.8: Illustration of the training process. A separate process (*Slice-Producer*) creates slices of n token sequences of the context size l (left). These are put into a multiprocessing queue and read in the training process (center). Here, batches of b samples are created and fed into the GP-Transformer (right). For every but the last token (since it cannot be compared to a subsequent token), the probability distribution that depicts the probability of each vocabulary token to be the subsequent token of the sequence is computed. The forward language modeling loss is computed as described in Section 2.2.2 and the model's weights are updated by stochastic gradient descent (SGD). Note that the right side (the model) is analogous to Figure 2.5.

is always a subsequence of the document. The resulting samples are packed together into a sample slice $W \in R^{n \times l\, 2}$ tensor, with n referring to the slice size. Besides the n-gram IDs, W contains the embedding matrix indices that correspond to the position embeddings of each token (see Section 2.2.2). In the implementation used in this work, the position embeddings are placed in the same embedding matrix E as the token embeddings. With the vocabulary size V (40k in case of the German GP-Transformer), the position embeddings correspond to the vectors V to $V + l$ in the embedding matrix $E \in \mathbb{R}^{V+l \times d}$ (with d again denoting word embedding size). Furthermore, a separate matrix $M \in \{0, 1\}^{n \times l}$ is created, which is later used to mask padding tokens in the language modeling loss calculation. The producer then writes each slice into the queue, while the main training procedure reads from it and creates random batches of size b from the slice. Note that the creation of slices is just a design decision and not necessarily required: By adjusting the slice size n, with $n = a \cdot b$ where $a \in \mathbb{N}$, one can control the quantity of queue read/writes (with $n = b$ leading to a batch-wise read/write). Next, the batch of contiguous BPE token sequences is fed into the GP-Transformer and the forward language modeling loss is computed as described in Section 2.2.2. The models weight's are then updated by stochastic gradient descent (SGD). The whole process is illustrated in Figure 2.8.

Adaption and Efficiency

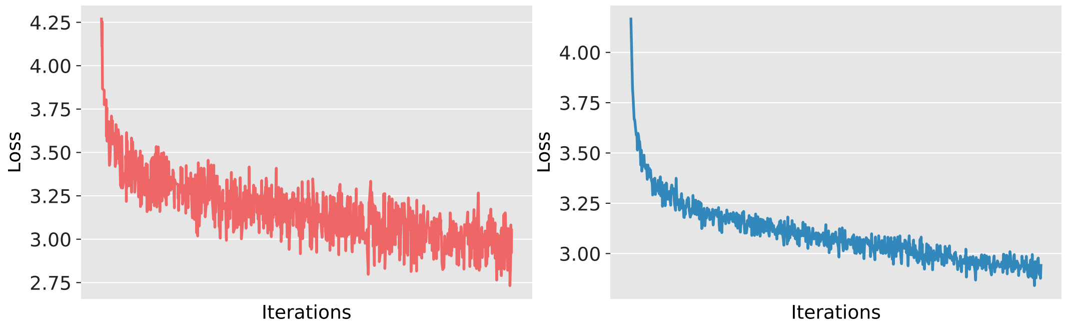
Because the pre-trained English GP-Transformer was indented to be used in a transfer learning setting, only the code that is necessary to fine-tune the model on a new task was released by OpenAI. In order to replicate the pre-training on a German text corpus, several adjustments had to be made to the model. First, since language modeling is used as an auxiliary objective during fine-tuning, the language modeling loss was separated from the classification loss calculation. Second, Radford et al. (2018) trained their model on a batch size of 64, however, due to less available resources this was not possible for the German GP-Transformer. With the purpose of mimicking the English pre-training, the model was tuned to allow for a higher effective batch size by employing an optimized tensor GPU distribution and gradient accumulation: In order to distribute a batch onto available CUDA devices, the model wrapper *DataParallel*, which splits the model's input along the batch dimension, is commonly utilized when working with PyTorch. But DataParallel has an important drawback: After the forward pass is conducted on each GPU in parallel, the outputs are gathered on a single GPU in order to compute the gradients based on the loss (Wolf 2018). This leads to a single graphic card doing the additional work of loss and gradient computation, requiring more memory than the other cards and slowing down the training process. Instead, the German GP-Transformer uses a different DataParallel implementation (Wolf 2018), which solves the problem by providing a distribution wrapper for the loss criterion. With this modification, the forward pass as well as the loss and gradient computation is executed on the available graphic cards in parallel. According to a rough estimate, this resulted in about 40% training speed-up compared to the original PyTorch implementation. In addition to this and due to restricted memory space, gradients where accumulated for multiple iterations and then back propagated. With this, the model can effectively be trained on large batches that do not fit on the available GPUs.

Hardware and Hyperparameters

The pre-training was conducted on eight graphic cards in parallel, two P600 cards with 24 GB memory each and six Nvidia GTX 1080 TI with 11 GB memory. In a multi GPU setup, the model must be loaded onto every graphic card in order to compute the forward pass independently. In this case, the six GTX 1080 GPUs are the limiting factor, because they have less than 50% memory space compared to the P600 GPUs. Since the GTX 1080 GPUs do not have a sufficient amount of memory, the batch size b was set to 32 instead of 64. With the purpose of adopting the hyperparameters of the English GP-Transformer, gradients were accumulated for every two batches in the training phase and then back propagated, effectively increasing the batch size to 64. Other hyperparameters were also adopted from the original GP-Transformer: The model contains 12 Transformer blocks and 12 attention heads per block. The embedding size was set to 768. The model weights are updated by an Adam Optimizer (Kingma and J. Ba 2014) with a maximum learning rate of $2.5 \cdot 10^{-4}$, which is increased over the first 2000 updates and then annealed to 0 following a cosine schedule. The context size l is set to 512 and dropout is applied throughout the model with a rate of 0.1. For further details, see Radford et al. (2018). In order to evaluate the model on a separate validation set, 0.1% of the documents of each corpus where put aside (140 documents for *Frankfurter Rundschau*, 10 documents for *Project gutenberg* and 2055 *German Wikipedia* documents). The sample slice size n was set to a relatively small value to reduce memory usage: In the final training run to 6400 and therefore 200 batches per slice. The German GP-Transformer was scheduled to run for 30 epochs (roughly 30 days). However, due to time restrictions, the training procedure was aborted after 22 epochs (1.510.600 training steps / iterations).

2.3.4 Language Model Evaluation

The German GP-Transformer was evaluated throughout the training process. The PreTrainer saves the model parameters at least every epoch and logs the training and validation loss redundantly to TensorBoard, a visualization tool of TensorFlow, and separate CSV files. The Python library tensorboardX[13] is used to log data into a file which can be read and interpreted by TensorBoard. For insights and reproducibility, the configuration (including hyperparameters) and the current Python code base[14] is also saved on disk for each pre-training run. Figure 2.9 shows the language modeling



(a) Trainings loss **(b)** Validation loss

Figure 2.9: Trainings loss and validation loss of the final pre-training run.

loss for the training (left) and validation set (right). The training loss is averaged

[13]`https://github.com/lanpa/tensorboardX`
[14]only .py files

over the last processed batch (of effective batch size 64 by gradient accumulation, see Section 2.3.3) while the validation loss is averaged over 512 random samples of the validation set. Note that both losses are very similar up to the end of training, indicating little to no overfitting on the validation set. As visible in the figure, both losses still decrease almost linearly, so there is probably room for improvement by training the language model for a longer duration.

Since language models inherently learn the syntax and semantics of language, they are widely used for text generation. For visualization purposes of the model's learning progress, the PreTrainer periodically generates text and stores it in TensorBoard and on disk. Text is generated by starting with a predetermined start token, which is fed into the GP-Transformer. All other values of the context are set to the padding token. Then the second token is determined by a *sampling strategy* based on the language model logits and concatenated with the start token, padded, and again fed into the model. This process is repeated till a predefined maximum text length is reached (Figure 2.10). The PreTrainer implements three different sampling strategies:

1. **text-max** Use the maximum value of LM logits over all vocabulary tokens as the next token.
2. **text-softmax** Sample next token from the softmax distribution over LM logits.
3. **text-gumbel-softmax** Sample next token from the gumbel softmax distribution over the LM logits.

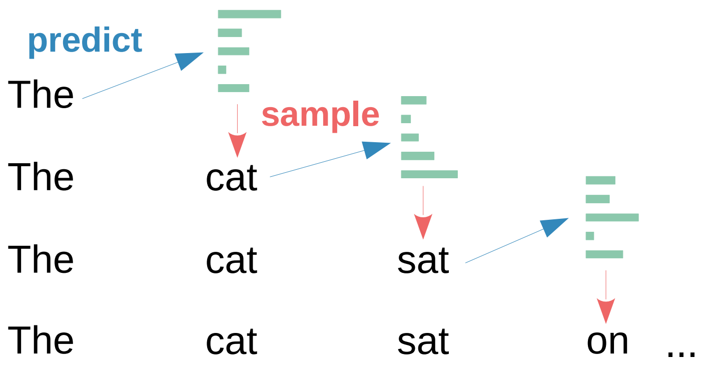


Figure 2.10: Text generation is conducted by feeding a start token (here “The”) into the GP-Transformer and by sampling the following token based on the language model logits. This token is then concatenated with the start token and again fed into the model. The process is repeated till a predefined sequence length is reached.

The gumbel softmax, introduced by Jang, Gu, and Poole (2017), is used in this work to add additional noise to text generation by adding samples from the Gumbel distribution during the softmax calculation. The so-called temperature parameter τ

was set to 1. The start token is either set to ".", forcing the model to begin with a new sentence (at least in most cases), or by sampling a random beginning from the validation documents. Table 2.4 shows how the quality of text generation progresses

step 0 WASHINGTON, 9. Mai (afp). stands Basen verteidigteführungsmöglichellos Fleiß sum bewachen Andre solcher Vaterland schulüchtige Must räumlichen Linken güt Mottopreuß Inkzykluseite...

step 24.000 WASHINGTON, 9. Mai (afp). Zwei verohrte Pflanzen , so die Inselloge , werden darin wohnbar sein. Das war klar. Die Sicherung im Park und die Hilfe von Feldpflanzen in Freiburg in der Schweiz und Münster , berichtet...

step 628.000 WASHINGTON, 9. Mai (afp). Das US - Verteidigungsministerium ist am Wochenende bemüht , von der Straße abzuspringen , um einen spektakulären Bustransfer der US - Armee in die zweite Heimat einzufristen.

step 1.480.000 WASHINGTON, 9. Mai (afp). US - Präsident Bill Clinton hat vor einer fünfjährigen weltpolitischen Demokratisierung von 16 Milliarden Dollar für Entwicklungspolitik wie für die Förderung des ökonomischen und kulturellen...

Table 2.4: Text generation over the training progress. The underlined text was fed into the model as the beginning, forcing the model to continue the text thereafter. Generated by softmax sampling. Note that the pre-training was conducted up to 22 epochs or 1.510.600 training iterations.

over the training duration. Starting with the initial and untrained GP-Transformer model, subsequent tokens are just guessed at random. At iteration 24.000, the model gets better at creating grammatically well-formed sentences but the continuation does not fit the news article beginning "WASHINGTON, 9. Mai (afp).". In later steps, the model adopts the style of news articles and even tends to continue the beginning in semantically plausible ways.

Table 2.5 contains several text generation examples. In the *text-max* setting, the model was frequently observed to repeat the generated text over and over again. Text repetition also occurred occasionally when the two other sampling strategies, *text-softmax* and *text-gumbel-softmax*, where used, but to a lesser extent. By training the language model on multiple, partly highly distinctive textual sources, it learns to adapt to different styles of writing: In the second example, the generated text resembles a dialogue sequence typical to literature, probably mimicking text from the *Project Gutenberg* corpus. The third example on the other hand resembles the style of factual writing often found in *German Wikipedia* articles.

Remarkably advanced is the capability of the model to semantically connect to previous sentences, in this case by referencing the fictional discipline of Vacca Ramirez (he is a catholic bishop in reality), walking, in every sentence. By also continuing the year dates in a plausible way, the produced text is almost indistinguishable from

(1) text-max *"Ich habe mich nicht getäuscht", sagte er, "ich bin der Sohn des Königs von Frankreich, und ich bin der Sohn des Königs von Frankreich, und ich bin der Sohn des Königs von Frankreich, und ich bin der Sohn des Königs von Frankreich, und ich bin der Sohn des Königs von Frankreich, und ich bin der Sohn des Königs..."*

(2) text-gumbel-softmax *"So heiße doch mal Juniinchen , " rief Hedwig böse. "Mein liederliches Gesichtchen ist wieder so besoffen wie früher", fügte sie heiser hinzu. "Büchse hat ihm 'nen Silberpinsel mitgebracht." Es sprang ein lärmend Wiehern um das Mädchen ,eine Erregung. "Was bist du für eine reine Tugend?" sagte Else. "Minze!" rief Albrecht. "Dein Mund ist dann noch mit Wasser angefüllt..."*

(3) text-softmax-completion *<u>Vacca Ramírez (Misael* 5. November</u> 1986 in San Carlos) ist ein moldawischer Geher. Bei den Jugendweltmeisterschaften 2006 wurde er Fünfter über dieselbe Distanz. Ein Jahr später nahm er bei den Leichtathletik-Weltmeisterschaften 2007 in Osaka teil und wurde Achter über 5000m. 2008 wurde er erstmals nationaler Meister im 20km Gehen. Eine weitere Silbermedaille errang er 2009 im 20km Gehen , jeweils auf der Kurzbahn.*

Table 2.5: Examples of text generation. In the *text-softmax-completion* setting, the underlined sequence marks the predefined beginning.

human written text. Note however, that these are hand-picked examples – the quality of the produced text, while tending to improve with each epoch, varies even near the end of training. As always, the well-known approach of more compute, data and longer training would probably lead to even better results. More examples of generated text can be found in Appendix A.

2.4 Giving Insights: Clustering of Embeddings

Machine learning models and especially neural networks are often described as a black box: A specific input leads to a specific output, but the internals of the model's decision are difficult to grasp. To be able to better understand how neural networks come to their decisions, several attempts have been made in various domains: From visualizing the focus point and captured features in image classification (Zeiler and Fergus 2013) to the illustration of attention weights in machine translation (Tang, Sennrich, and Nivre 2018). To give insights about the semantic relations that a language model learned on basis of unstructured text, word embeddings are frequently reduced in dimensionality or partionized by employing clustering methods like the well-known K-Means algorithm. Compared to using traditional fixed word embeddings like Word2Vec or GloVe, more complex interactions between words and additional context sensitive information can be utilized by transferring whole pre-trained models like the GP-Transformer to a new domain or task, as described in Section 2.1. In the GP-Transformer, word representations in higher layers are a function of previous words and the word itself. This is intended to produce context-aware word embeddings that include information about the specific preceding context. In this work, word embeddings produced by the English and German GP-Transformer are clustered and aligned with the intention of validating this assumption. This section describes how the Transformer embeddings are sampled and clustered based on the K-Medoids (Kaufmann and Rousseeuw 1987) algorithm. It also includes several examples of clusters that underline the model's capability to create context-aware word embeddings.

2.4.1 Sampling of Embeddings

In order to illustrate how the embeddings change from layer to layer, the embeddings that correspond to each token are captured in all Transformer blocks after the position-wise fully-connected layer (see Figure 2.5). Because clustering is executed on a large number of embeddings, the sampling step was decoupled from the actual clustering: By sampling the embeddings in advance and storing them on disk, clustering can be performed with different settings later on, without requiring a costly inference step each time. Embeddings are sampled in four essential steps till a maximum predefined embedding count z is exceeded:

1. Choose a random, not yet processed document.
2. Split the document into n sentences with spaCy and transform the sentences according to Section 2.3.3 into a matrix $W \in \mathbb{R}^{n \times l \times 2}$ (l refers to a predetermined context length).
3. Feed batches of sentences into the model. Record all Transformer block activations for each token per layer.
4. Sample a maximum amount of j random byte pair encoded token embeddings per sentence (all embeddings if $j >=$ sentence length), and store them per layer alongside the corresponding BPE token IDs.

The resulting embeddings are stored on disk per corpus/document in a hierarchal *tar* file. In this work, embeddings were recorded for the English and the German *Wikipedia* corpus, with the pre-trained model by OpenAI to record embeddings for the English corpus, and the German GP-Transformer for the German corpus respectively. One million embeddings per Transformer block were recorded for both text corpora and the maximum sentence embedding count j was set to 100. Note that the same BPE token is potentially sampled multiple times in different contexts, since the token's embedding is intended do vary between contexts.

2.4.2 Clustering and Assignment

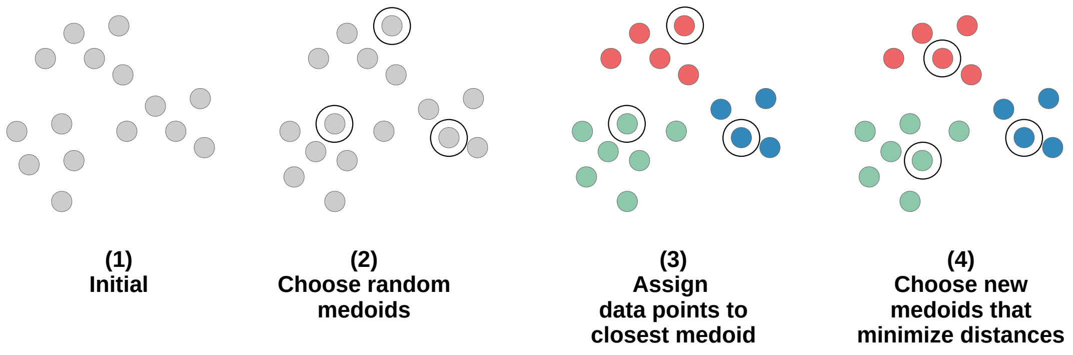

(1)
Initial
(2)
Choose random medoids
(3)
Assign data points to closest medoid
(4)
Choose new medoids that minimize distances

Figure 2.11: Procedure of the K-Medoids algorithm: Starting with the initial data points, K medoids are first chosen randomly. Then each data point is assigned to its nearest medoid based on an arbitrary distance metric. Finally, the data point that minimizes the distance to all other data points of the same cluster is chosen as the new medoid. Steps 3 and 4 are repeated for a set number of iterations or till the medoids do not change anymore.

The K-Means algorithm is a simple and frequently used clustering approach. However, it is designed to minimize the squared euclidean distance between data points and is therefore inappropriate for other distance functions. Since the GP-Transformer internally uses the dot product to compare word vectors, it seems natural to cluster the word embeddings regarding this metric. In contrast to K-Means, the K-Medoids clustering algorithm (Kaufmann and Rousseeuw 1987) is able to deal with arbitrary metrics. As well as the K-Means algorithm, it divides the data set into K predefined clusters. The points that have the minimal distance to all other points in their respective cluster are called *medoids* and serve as the cluster centers. The algorithm starts by initializing the medoids randomly and first assigns each data point to its nearest medoid. It proceeds with minimizing the summed distances between all data points in a cluster by choosing the data point as the new medoid that minimizes the distances to the other points of the respective cluster. This process is repeated till a specific number of iterations is exceeded or the clusters do not change from one iteration to the next (Figure 2.11).

In this work, the K-Medoids algorithm was implemented from-scratch in Python. In order to speed-up the clustering procedure on a graphic card, the fast tensor operations of PyTorch were utilized. Clustering is performed independently for each Transformer block. The algorithm (see Algorithm 1) start with initializing the K

Algorithm 1 K-Medoids clustering algorithm. The Pseudocode below resembles the style of mathematical coding in NumPy and PyTorch. $D \in \mathbb{R}^{n \times d}$ denotes a matrix of n data points $\mathbf{x} \in \mathbb{R}^{d}$. Sample returns random data points without replacement. Argmin(H, axis = 2) returns the indices of the minimum value per row of matrix H. Sum(E, axis = 2) returns the sum over all values per row of matrix E. $D[v == j]$ denotes the selection of data points assigned to cluster j. The algorithm is run for a total amount of I iterations or till the medoid's do not change between iterations.

```
1:  procedure CLUSTER
2:      $M \leftarrow \text{Sample}(D, K)$                ▷ Sample $K$ medoids from $P$ with $M \in \mathbb{R}^{K \times d}$
3:      while $M$ changed and i < I do
4:          $H \leftarrow D \cdot M^T \cdot (-1)$        ▷ Distances of medoids to all points ($H \in \mathbb{R}^{n \times K}$)
5:          $v \leftarrow \text{Argmin}(H, \text{axis} = 2)$        ▷ Indices of closest medoids ($v \in \mathbb{R}^{n}$)
6:          for $j = (1, ..., K)$ do        ▷ For each cluster $j$ of size $n_j$, do...
7:              $C \leftarrow D[v == j]$        ▷ Get points of respective cluster ($C \in \mathbb{R}^{n_j \times d}$)
8:              $E \leftarrow C \cdot C^T \cdot (-1)$        ▷ Distances between cluster points ($E \in \mathbb{R}^{n_j \times n_j}$)
9:              $\mathbf{s} \leftarrow \text{Sum}(E, \text{axis} = 2)$        ▷ Sum of distances ($s \in \mathbb{R}^{n_j}$)
10:             $b \leftarrow \text{Argmin}(s)$        ▷ Index of point that minimizes distances
11:             $M[j] \leftarrow$ C[b]        ▷ Set data point corresponding to index $b$ as new medoid
12:         $i \leftarrow i + 1$
13:     return M
```

medoids randomly from the set of available data points, the 768-dimensional token embeddings in this case (line 2). Next, each data point is assigned to its nearest medoid according to the (negative) dot product metric (lines 4 and 5). As mentioned before, the dot-product is used in this work to compare the token embeddings, since it is also used throughout the GP-Transformer to compute the relevance scores in the attention mechanism (see Section 2.2.1). The algorithm proceeds by computing the sum of distances between data points in a cluster (line 8 and 9). In line 10 and 11, the data point with the lowest distance to all other points is designated as the new medoid of the respective cluster. Lines 3-12 are repeated till the medoids do not change from one iteration to the next or the maximum number of iterations I is reached. Note that the cluster assignment (especially line 4) and medoid selection (especially line 9) had to be processed in chunks due to memory restrictions[15].

Figure 2.12 displays a simple clustering example with 100 2-dimensional data points sampled from a normal distribution ($\mu = 1; \sigma^2 = 1$). The data points where partitioned into $K = 5$ clusters. Because the points are clustered by the magnitude of the dot product, they tend to be assigned to the far outmost points. This leads to clusters of points that expand from the coordinate system's origin.

[15]Consider that all embeddings (e.g. 1M) are assigned to a single cluster. Then $E \in \mathbb{R}^{1.000.000 \times 1.000.000}$ is a matrix of 32-bit float values consuming 4000 GB memory.

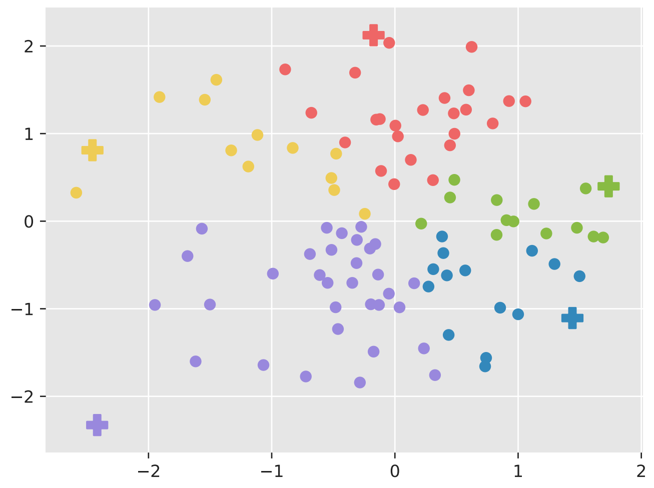


Figure 2.12: K-Medoids with 100 data points (samples from a normal distribution) and 5 clusters. A cross marks the medoid of the respective cluster.

The final clustering run on the Transformer embeddings was conducted for 10K iterations on the embeddings of 100K tokens, which were randomly sampled from the recorded embeddings. With the intention of tracing single embeddings later on, the same 100K tokens were used to cluster each Transformer block. The cluster count K was set to 10K. After the clustering procedure is finished, all embeddings (1M per block) were assigned to their nearest medoid for each Transformer block. The resulting clusters are again stored in a *tar* file on disc.

2.4.3 Inspection of Clusters

To gain insights into which words are grouped together and if the model is able to dissolve polysemes and homonyms by producing context-aware embeddings, some of the detected clusters were inspected manually. Usually, in embedding models like Word2Vec, words do only have one representation independent of their context, so individual words are grouped together with other vocabulary words when clustering is performed. The BPE token embeddings produced by the GP-Transformer on the other hand were obtained in dependence of their context, so the same token can be assigned to multiple clusters when it was extracted from different sentences.

Table 2.6 contains four clusters with exemplary sentences from the early Transformer blocks (layers) of the model. Which embedding was clustered in the corresponding sentence is depicted by the underlined word. The topmost cluster (layer 3, 131 assignments) contains only weekdays, while the middle cluster (second layer) comprises of more than 200 samples of countries and cities. Fore- and surnames of persons

Year 794 was a common year starting on	Saturday	,march 2, 2013...
The station does not air any news programs on	Sunday	.
At approximately 2:00 p.m. on	Monday	21 October 2013...
...Florida, Georgia, the Rocky Mountains,	Mexico	, and elsewhere made...
...Damilano (born 6 April 1957 in Scarnafigi,	Italy	) is an Italian former...
Poblet Monastery, one of the largest in	Spain	, is considered similarly...
Both he and	Scott	won the poetry society...
Richard Lindzen and his wife,	Nadine	, have two sons
Ann and her eldest brother,	Godfrey	, thus offered to...
...wrote about Sonic's lack of speed when	walking	on foot, which they...
...its street "Calle Uruguay", where locals	stroll	during weekend nights...
The bill hall trail is a	hiking	trail in Grand Canyon...

Table 2.6: Clusters of weekdays (layer 3), countries (layer 2), names (layer 3) and "walk" (layer 4).

reside in the third cluster (layer 3, 308 assignments). Furthermore, the last depicted cluster contains related words of "walk" (layer 4, 109 assignments).

...south of market (running parallel to , and a full...	block	south of Market Street)...
...then north on South Clinton Avenue a	block	later.
...matrix inversion and does not take advantage of the	block	form.
the minimum-norm solution is given by using the	block	matrix pseudoinverse...
it is a static DSL IP so a	block	or ban will not impact...
a week was a very short	block	in the first place IMO ...

Table 2.7: The illustrated clusters of the word "block" of layer 6 demonstrate the capabilities of the GP-Transformer to differentiate words based on their context. From top to bottom, the clusters contain street blocks, mathematical blocks and user blocks.

While these cluster examples are similar to those found for Word2Vec and GloVe embeddings, other clusters demonstrate the capabilities of the GP-Transformer in differentiating words based on their context. The word "block" can have different meanings depending on the particular context it appears in: From blocking a user of a message board, to a street block or a block in various mathematical terms. Table 2.7 shows three exemplary clusters (layer 6), with each of them representing a distinct meaning of "block".

In higher layers, the context-aware word embeddings tend to be distributed over many clusters and therefore single clusters that contain multiple mentions of the same word are harder to spot. However, one example can be found in table 2.8 (layer 12): By incorporating previous tokens into the vector representation of the homonym "bank", the GP-Transformer produces different embeddings for the financial (bank as an institution) or geological (riverside) meaning of the term.

An example of this is Banrisul, the largest	bank	in the south of Brazil...
...includes the largest commercial	bank	in Russia, Alfa-Bank...
the largest African development	bank	shareholder is Nigeria.
...was built for this occasion on the	bank	of the Volga river
...northwest of Montevideo, and on the east	bank	of the Río Uruguay across...
Frensham lies on the right	bank	of the river Wey

Table 2.8: Clusters of the homonym "bank" (layer 12): Financial banks (top) and river banks (bottom).

The importance of such a distinction for downstream tasks is obvious: The downside of regular fixed embeddings is that they can hardly encode the multiple, partly unrelated or even opposite (e.g. "oversight") senses of words. Yet, models such as the GP-Transformer are able to assign different embeddings to the same word based on the context and can therefore provide an appropriate representation to downstream models.

Figure 2.13 shows in how many clusters the words "make", "tree", "think" and "bank" are placed in the different Transformer blocks. Note that the lowest block, refers to the initial token embeddings, with only positional information added. The dotted line in Figure 2.13 illustrates the average cluster count that all tokens of the vocabulary where placed in.

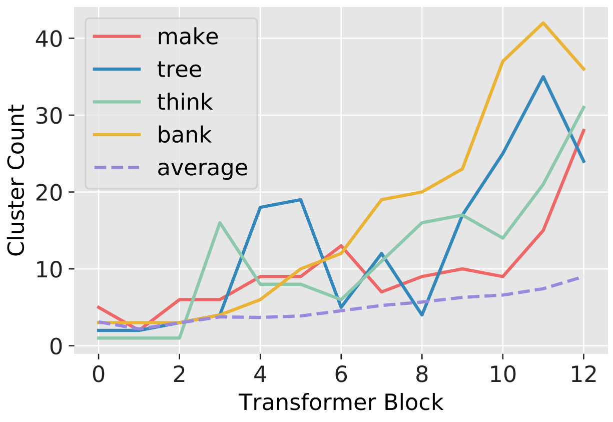


Figure 2.13: Cluster counts of the words "make", "tree", "think" and "bank" over each Transformer block (layer). The dotted line is the average cluster count of all vocabulary tokens, which appeared in the clustered corpus (25.861 of 40.000 tokens).

As visible in the figure, the cluster count tends to increase for each word from the lower to the higher layers. In average, the cluster count increases from 3.1 in block 0, to 9 in block 12[16]. Therefore, it is reasonable to assume that lower blocks

[16]Note that the average cluster count is lower than the cluster count of the examples due to infrequent words.

primarily capture the meaning of the embedded token itself, without spending much attention to the context. In higher blocks, token representations get frequently mixed up with the representations of other tokens due to the attention mechanism, so the context information becomes more and more pronounced in higher layers. This is comparable to findings in large convolutional neural networks, where lower layers focus on small, local features like edges or corners, while higher layers grasp more complex concepts, e.g. dog faces (Zeiler and Fergus 2013).

The GP-Transformer, however, has one limitation: Because the model is unidirectional, i.e. only attending to the previous tokens, it cannot bake the representations of subsequent tokens into an embedding. This behavior is especially bad for tokens that occur in the beginning of a sentence. Consider the two sentences “The Bank of Scotland is a commercial and clearing bank based in Edinburgh” and “Bank refers to a land alongside water”: Because the GP-Transformer does not consider subsequent tokens, it cannot produce a correct context-aware embedding of the homonym “bank”. On the other hand, a bidirectional model is able incorporate the full sentence into the “bank” embedding. Therefore it is save to assume that bidirectional models, such as ELMo or the bidirectional version of the GP-Transformer, BERT (see Section 2.1), are able to produce even better embeddings independently of the token positions.

Appendix B contains more examples of clusters, including those produced by the German GP-Transformer on the *German Wikipedia* corpus. The findings are similar to those of the English embeddings and the examples include, among others, clusters of year dates, fields of work and resolved homonyms like “Steuer”. Note however that the quality of clusters varies for both languages. For instance, many clusters only contain occurrences of the same word or sentence. Clustering in general can have largely different effects based on the choice of parameters, e.g. cluster count, distance metric or reinitializing of empty clusters, so other results and insights might be obtained with a different setup.

Chapter 3

Setup

Whereas pre-trained word embeddings like Word2Vec or GloVe are widely used for relation extraction, publications that fully transfer an existing, pre-trained, model to this task are scarce. Radford et al. (2018) obtained impressive results in various NLP tasks by converting an arbitrary and task-specific input into a token sequence, which is subsequently processed by the pre-trained GP-Transformer. By fine-tuning the GP-Transformer on the target task, they outperform other transfer learning approaches like ELMo, which require complex and task specific down-stream models.

In this work, the GP-Transformer is transferred to the task of relation extraction. Similar to Radford et al.'s work, the model is employed as an elaborate feature extractor that yields context-aware token embeddings. By adding simple and shallow down-stream models, the token stream is converted into a representation which conversely is processed to solve the three relation extraction subtasks explored in this work, namely "Relation Classification", "Few-Shot Relation Classification" and "Joint Entity and Relation Extraction". An overview about relation extraction as well as the two relation extraction datasets used in this work is given in this chapter.

3.1 Relation Extraction Overview

The extraction of semantic relations between entities in unstructured text is a complex problem that consists of several subtasks: This starts with the detection of named entities and ends with the classification of entity pairs into a set of relation types. Other methods like named entity disambiguation and coreference resolution might also be of relevance in practice. In relation extraction, the goal is to extract relation *triples* (head, relation, tail), which denote that a specific relation holds between a particular entity pair. Here the head entity represents the subject of the relation, while the tail entity depicts the relation's object. For example, the relation triple ([Nintendo]$_{\text{head}}$, Developer, [Super Mario 64]$_{\text{tail}}$) expresses that "Nintendo" developed the game "Super Mario 64", while ([Donald E. Knuth]$_{\text{head}}$, Occupation, [Computer Scientist]$_{\text{tail}}$) states that "Donald E. Knuth" works as a "computer scientist". In natural language, semantic relations can be expressed in a variety of different manifestations. Take for example the following sentence:

$$\underbrace{\text{Douglas Adams}}_{\text{Head Entity}}\text{ was an author of science-fiction novels who lived in }\underbrace{\text{Santa Barbara}}_{\text{Tail Entity}}.$$

Imagine that a system wants to classify the relation of the two entities of this sentence into one of three different relation: "Occupation", where a person is occupied at a specific job, "Residence" where a person lives at a specific location and "Work-Location", where a person works in a specific location. While "Occupation" can easily be excluded based on the entities alone because neither "Douglas Adams" nor "Santa Barbara" depict a job, the choice between "Residence" and "Work-Location" is not as simple. While certain entities restrict the choice of applicable relations, the relation is also dependent on the context surrounding the two corresponding entities. In this case, "Douglas Adams" and "Santa Barbara" could possibly also belong to the "Work-Location" relation ("Douglas Adams wrote science-fiction novels in his home in Santa Barbara.") based on their types (a person and a location) but the surrounding context strongly suggests that they are part of a "Residence" relation in this special case. Furthermore, the order of the two entities in the relation matters: It plays an important role if the context suggests that "Douglas Adams" lives in "Santa Barbara" or if "Santa Barbara" lives in "Douglas Adams". With this, the goal of relation extraction is usually to not only infer the relation type, but also the direction of the relation (at least for asymmetrical relations). Additionally, the ambiguity and multitude means of expression in human language is a core problem for relation extraction. The example sentence "Douglas Adams was an author of science-fiction novels who lived in Santa Barbara" can be rephrased in a variety of different ways, including:

$$\text{In 1999, }\underbrace{\text{Douglas Adams}}_{\text{Head Entity}}\text{ moved from London to }\underbrace{\text{Santa Barbara}}_{\text{Tail Entity}}.$$

$$\underbrace{\text{Santa Barbara}}_{\text{Tail Entity}}\text{ was the home of science-fiction author }\underbrace{\text{Douglas Adams}}_{\text{Head Entity}}.$$

$$\text{The English author }\underbrace{\text{Douglas Adams}}_{\text{Head Entity}}\text{ resided in a house in }\underbrace{\text{Santa Barbara}}_{\text{Tail Entity}}\text{, California.}$$

Therefore models employed for relation extraction are required to generalize to unknown entities as well as the variety of ways a particular relation can be expressed in a sentence. To do that, early relation extraction algorithms required an extensive amount of linguistic information and were therefore dependent on various NLP toolkits and hand-crafted features (see Hendrickx et al. 2010). This introduces more sources of error and makes the acquisition and choice of a suitable feature set a time-consuming challenge.

In contrast to this, current state-of-the-art models are based on neural networks, which are able to learn the defining patterns and features that suggest a particular relation automatically with a set of labeled samples. However, the manual annotation of samples is an expensive and time-consuming task and the set of available labeled samples for a target domain hardly cover the various aspects of semantic relations.

To aid generalization, a large variety of models which were used for relation extraction in the recent years employ pre-trained word embeddings, like those produced by Word2Vec (see 2.1), as the model's input (e.g. Miwa and Bansal (2016), Han et al. (2018a), Zheng et al. (2017)). Because these word representations encode semantic relatedness by their spatial distance, relations can also be extracted for samples which contain unknown entities or other words that the model was not trained on. With reference to the example above, the words "live" and "reside" may be close to each other in the embedding space because they often occur in similar contexts, so the model is able to generalize from "live" to "reside", even if the latter was not encountered during training. The same goes for new entities, for example by generalizing from familiar locations like "Santa Barbara" to unseen locations like "Bad Kreuznach". In addition to word embeddings, state-of-the art relation extraction models (e.g. Y. Zhang et al. 2017, L. Wang et al. 2016, Shen and Huang 2016, Gao et al. 2019), which are usually based on recurrent or convolutional neural networks, also incorporate the attention mechanism, which was described in Section 2.2.1. By the selective weighting of input words, attention enables the extraction of relations between entities that lie far apart in the corresponding sentence.

In this work, three relation extraction subtasks are explored, namely "Relation Classification", "Few-Shot Relation Classification" and "Joint Entity and Relation Extraction". In all three tasks the corresponding task-specific model is trained on a set of annotated sentences. Each sentence is labeled with a relation that is expressed between two entities of the sentence. Figure 3.1 (bottom) shows three exemplary relations alongside a set of sentences which are labeled with the corresponding relation. The three subtasks are outlined in Figure3.1 (top):

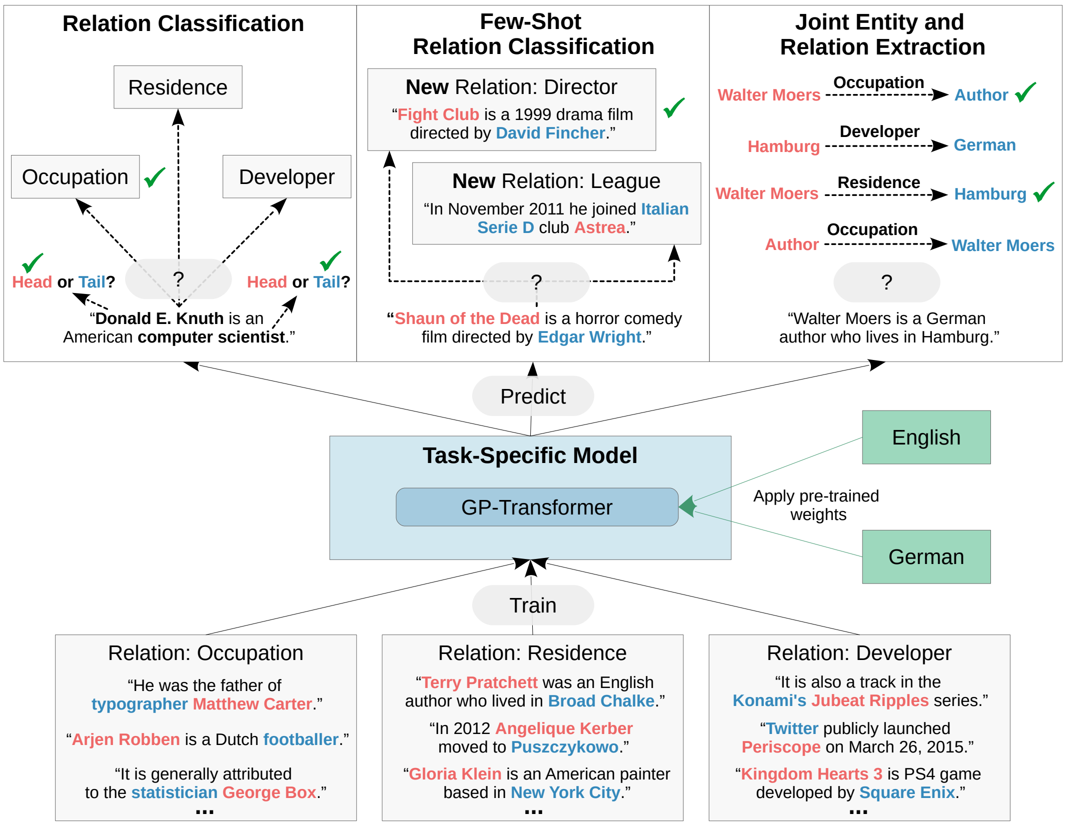

Relation Classification
Residence
Occupation
Developer
Head or Tail?
?
Head or Tail?
"Donald E. Knuth is an American computer scientist."
Few-Shot Relation Classification
New Relation: Director
"Fight Club is a 1999 drama film directed by David Fincher."
New Relation: League
"In November 2011 he joined Italian Serie D club Astrea."
?
"Shaun of the Dead is a horror comedy film directed by Edgar Wright."
Joint Entity and Relation Extraction
Walter Moers Occupation Author
Hamburg Developer German
Walter Moers Residence Hamburg
Author Occupation Walter Moers
?
"Walter Moers is a German author who lives in Hamburg."
Predict
Task-Specific Model
GP-Transformer
English
Apply pre-trained weights
German
Train
Relation: Occupation
"He was the father of typographer Matthew Carter."
"Arjen Robben is a Dutch footballer."
"It is generally attributed to the statistician George Box."
...
Relation: Residence
"Terry Pratchett was an English author who lived in Broad Chalke."
"In 2012 Angelique Kerber moved to Puszczykowo."
"Gloria Klein is an American painter based in New York City."
...
Relation: Developer
"It is also a track in the Konami's Jubeat Ripples series."
"Twitter publicly launched Periscope on March 26, 2015."
"Kingdom Hearts 3 is PS4 game developed by Square Enix."
...

Figure 3.1: Illustration of the three relation extraction subtasks explored in this work: "Relation classification", "Few-Shot Relation Classification" and "Joint Entity and Relation Extraction". A task-specific model is trained for each of these subtasks on a set of labeled sentences. The core of these models is formed by the language model pre-trained GP-Transformer (center, blue). In relation classification, a sample must be assigned to a specific relation type that the model encountered during training. In contrast to this, few-shot relation classification requires the model to generalize to unseen relations by providing a few samples (in this example only a single one) for each new class. In joint entity and relation extraction the target entities are unknown and the model needs to simultaneously detect both the entities and their relation that is expressed in the sentence.

Relation Classification In relation classification, the two target entities are given and the model is required to assign a specific relation, which is expressed in the sentence, to the entity pair. The directionality of the relation must also be inferred. As common for a classification objective, the model learns the defining patterns of a relation during training and is then required to generalize to unseen samples of these relations during inference.

Few-Shot Relation Classification Contrary to relation classification, the few-shot relation classification objective is to generalize to new relations that were not encountered during training. While the model is also trained on labeled relation triples that are expressed in a sentence, it is presented with new relation classes during inference. For each one of the new relations, only a *few* examples (*shots*) are given and the model needs to assign one of the new relations to a query based on the provided examples alone. The relation's directionality must not be predicted in the few-shot setting addressed in this work.

Joint Entity and Relation Extraction Whereas the two target entities are given in (few-shot) relation classification, they are unknown in joint entity and relation extraction and need to be predicted alongside the relation type and the directionality of the relation. So when a sentence is presented to the model, it is required to detect likely entities and their boundaries. A relation is then assigned to an entity pair when this relation is expressed in the sentence between the two entities.

Each of these subtasks is tackled with task-specific models (center of Figure 3.1). The GP-Transformer, which was introduced in Section 2, forms the core part of these models and is fine-tuned on the respective target task. In contrast to the aforementioned fixed word embeddings, which are frequently utilized in relation extraction, the pre-trained GP-Transformer is able to detect more complex interactions between words and produces context-aware embeddings, which may be beneficial for relation extraction: Take the word "apple" for example, which can express a fruit as well as a company name dependent on the context. The GP-Transformer learned how to consider the specific input context during pre-training and therefore the context-aware "apple" embedding may suggest a specific relation, for example "Occupation". Also, as explained before, many state-of-the-art models employ attention as an additional layer after an recurrent or convolutional neural network. Since the GP-Transformer is primarily based on attention, it may be well suited for relation extraction, which is explored in this work. Depending on the dataset, the weights of the GP-Transformer are either initialized with those of the language modeling pre-trained English GP-Transformer by OpenAI (Radford et al. 2018) or the German GP-Transformer (Section 2.3). This is illustrated in Figure 3.1 (center).

3.2 The SemEval Dataset

The most widely used dataset for relation classification is the manually annotated database from the SemEval-2010 Task 8 challenge (Hendrickx et al. 2010). In this dataset, entitiy pairs in sentences crawled from the web were assigned to a set of 9 different relation types. According to Hendrickx et al., the types were chosen to cover a broad number of relation mentions but at the same time offer a minimal semantic overlap. This resulted in rather abstract semantic relations like "Cause-Effect" and "Component-Whole".

Relation	**Freq**	**Pos**	**IAA**
Cause-Effect (CE)	1331 (12.4%)	91.2%	79.0%
Component-Whole (CW)	1253 (11.7%)	84.3%	70.0%
Entity-Destination (ED)	1137 (10.6%)	80.1%	75.2%
Entity-Origin (EO)	974 (9.1%)	69.2%	58.2%
Product-Producer (PP)	948 (8.8%)	66.3%	84.8%
Member-Collection (MC)	923 (8.6%)	74.7%	68.2%
Message-Topic (MT)	895 (8.4%)	74.4%	72.4%
Content-Container (CC)	732 (6.8%)	59.3%	95.8%
Instrument-Agency (IA)	660 (6.2%)	60.8%	65.0%
Other (O)	1864 (17.4%)	N/A	N/A
Total	10717 (100%)		

Table 3.1: Semantic relations of SemEval-2010 Task 8. Absolute and relative frequency (Freq), percentage of positive samples in the candidate set (Pos) and inter-annotator agreement (IAA). Original source: Hendrickx et al. (2010)

The task organizers also added an undirected “Other” class that contains all the entity pairs that do not fit the 9 relation categories. Because the order of the entities must be inferred alongside the relation type, this results in 19 classes in total ($2 \cdot 9 +$ “Other”). The task creators also defined a set of annotation guidelines, for example to exclude speculative or counterfactural scenarios and to use only common-noun heads as entities (for exhaustive annotation guidelines see Hendrickx et al. 2010).

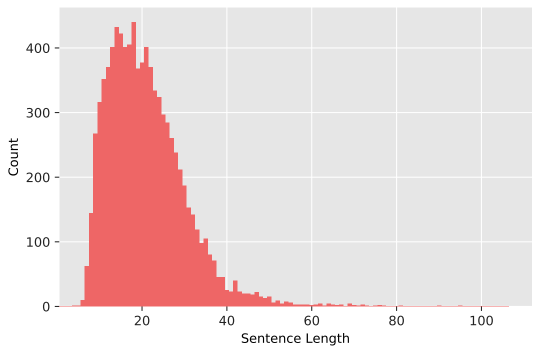
Relation	**Entities**
Entity-Destination[a]	**People** have been moving back into **downtown**.
Cause-Effect[b]	The **burst** has been caused by water hammer **pressure**.
Instrument-Agency[c]	A **programmer** uses a high level **language** to implement its algorithms.
Member-Collection[d]	I was attacked by a **flock** of **pigeons** today.
Component-Whole[e]	The oculomotor **nerve** rests in a **cistern** within the sinus roof.

Table 3.2: Example sentences of the SemEval dataset (head **red**, tail **blue**)

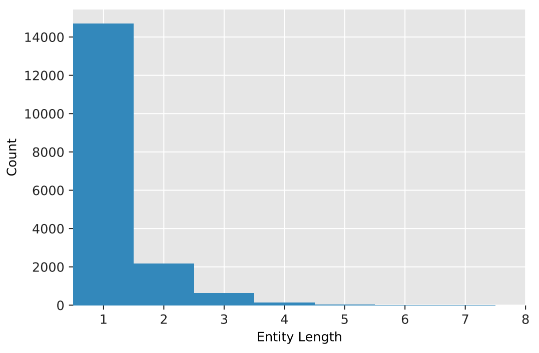
[a]“An entity is moving towards a destination” (Hendrickx et al. 2010)
[b]“An event or object leads to an effect” (Hendrickx et al. 2010)
[c]“An agent uses an instrument” (Hendrickx et al. 2010)
[d]“A member forms a nonfunctional part of a collection” (Hendrickx et al. 2010)
[e]“An object is a component of a larger whole” (Hendrickx et al. 2010)

The annotation process took place in three rounds: In the first round, 1200 sentence candidates per relation were manually crawled through a pattern-based web search. These sentences were then annotated by two independent annotators in the second round. In the third round, emerged conflicts between the annotations of the two annotators were resolved or corresponding sentences removed if no consensus was achieved. This approach yielded 10717 annotated relation mentions in total, distributed unequally over the 10 relation types. The highest inter-annotator agreement was reached for “Cause-Effect” (79.0%), and the lowest for “Instrument-Agency” (65%), which suggests that relation classification is in itself a hard and ambiguous problem, even for humans.

Some typical example sentences of the SemEval dataset are depicted in Table 3.2: Each sentence is annotated with the two target entities, their position in the relation triple (i.e. if an entity is the head or tail of the relation) and the corresponding relation. Figure 3.2 shows the byte pair encoded sentence lengths and entity lengths (in terms of byte pairs) of the SemEval dataset. As visible in the figure, the average sentence length is fairly low with 20.91 BPE tokens. Moreover, most entities comprise of only a single token (about 83%) and the average entity length is 1.23 BPE tokens.



(a) Sentence lengths



(b) Entity lengths

Figure 3.2: Byte pair encoded sentence lengths (left) and byte pair encoded entity lengths (right) of the whole SemEval dataset. The average sentence length is 20.91 BPE tokens and the average entity length 1.23 BPE tokens.

In total, SemEval comprises of 10.717 annotated relation examples. The dataset is split into an official training (including 8.000 sentences) and test set (including 2.717 sentences). Since the release of SemEval in 2010, the dataset is used in a wide variety of publications on the matter of standard relation classification. In this work, the dataset is is also employed for joint entity and relation extraction.

3.3 The FewRel Dataset

The FewRel dataset (Han et al. 2018a) was originally intended to be used in a few-shot setting (see Chapter 5), but is also employed for standard relation classification and joint entity and relation extraction in this work. It has the following benefits compared to other relation extraction datasets like SemEval 2010 or the NYT dataset (Riedel, Yao, and Mccallum 2010): FewRel includes 100 different relation types with 700 samples each, making it much larger and more diverse compared to the aforementioned dataset. Second, the dataset was created by aligning the Englisch Wikipedia Corpus with Wikidata[1], which is an open knowledge base containing structured data. Particularly, Wikidata already stores a large number of entities and the relations that exist between them in its database. For example, if you type *P740* into the search field of Wikidata, the relation type “Location of Formation” is returned, which is described as a “location where a group or organization was formed”. By entering “Q1299”, Wikidata displays several informations regarding the entity “The Beatles”, an “English rock band”: This includes a list of relations that exist between “The Beatles” and other entities, for example Liverpool (*Q24826*) in the relation “Location of Formation”.

Reasoning	**Example**
Simple Pattern	**Chris Bohjalian** graduated from **Amherst College** Summa Cum Laude, where he was a member of the Phi Beta Kappa Society.
Commonsense Reasoning	**James Alty** obtained a 1st class honours (Physics) at **Liverpool University**.
Logical Reasoning	He was a professor at **Reed College**, where he taught **Steve Jobs**, and replaced Lloyd J. Reynolds as the head of the calligraphy program.
Coreference Reasoning	He and **Cesare Borgia** were thought to be close friends since childhood, going on to ac- company one another during their studies at the **University of Pisa**.

Table 3.3: Relation mention examples of “Educated At” (*P69*), which require different forms of reasoning. **Red** indicates head, **blue** indicates tail entity. Original Source: Han et al. (2018a)

The distant supervision approach for relation extraction (Mintz et al. 2009), which Han et. al. rely on to build the FewRel dataset, is based on the following assumption: If two entities in a sentence are connected by a relation in a knowledge base, the sentence is assumed to indeed express the relation between those entities. While this assumption is obviously violated in some cases (e.g. “[The Beatles]$_{\text{head}}$ frequently played in The Cavern Club in [Liverpool]$_{\text{tail}}$” does not directly express “Location of Formation”), it is an established pre-selection step for the creation of relation classification datasets. To do so, Han et al. extracted entity mentions from Wikipedia by both using already referenced mentions of Wikidata entities and additional mentions

[1]`https://www.wikidata.org/`

detected by spaCys's NER tagger. Then, the distant supervision assumption was applied by linking every entity pair (e_1, e_2) in a sentence to a Wikidata relation r in case Wikidata contains the triple (e_1, r, e_2). Because models might tend to only consider the entities for classification without paying attention to the surrounding context if these entities often occur together in a specific relation, only one sample per entity pair was kept. After that, all relations that contained less than 1000 samples were removed.

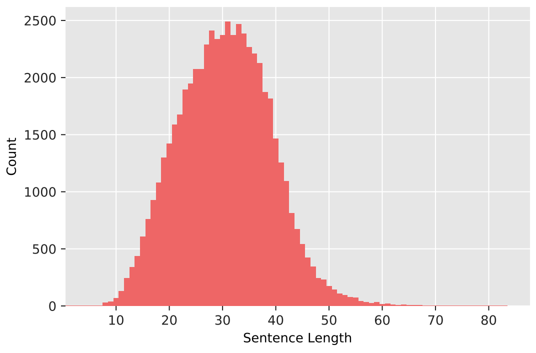
Relation	Entities
Speciality[a]	**Roberto Crivello** (born 14 September 1991) is an Italian footballer who plays as a **left back** for Frosinone.
Record Label[b]	The recording by American country singer **Sonny James** was released by **Capitol Records** as catalog number 3602.
Occupation[c]	**Benjamin Vermeulen** (born 15 July 1957) is a former Belgian **racing cyclist**.
Mother[d]	**Damon Elliott** was born on march 21, 1973 to **Dionne Warwick** and Bill Elliott.
Nominated for[e]	Sisters Olivia De Havilland and **Joan Fontaine** were both nominated for **Best Actress** in 1942, with Fontaine winning for "Suspicion".

Table 3.4: Example sentences of the FewRel dataset. The head entity of the relation is colored in **red** and the tail entity in **blue**.

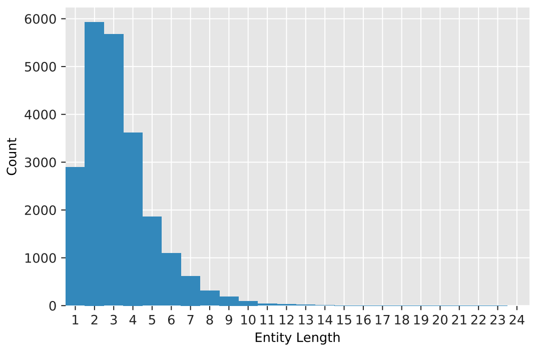
[a]"Position or specialism of a player on a team, e.g. Small Forward" (Wikidata)
[b]"Brand [...] associated with the marketing of subject music recordings and music videos" (Wikidata)
[c]"Occupation of a person" (Wikidata)
[d]"Female parent of the subject" (Wikidata)
[e]"Award nomination received by a person, organisation or creative work" (Wikidata)

From the remaining 122 relations 1000 samples were randomly selected. In a second step, similar to the SemEval acquisition process, each instance (e_1, r, e_2) from the candidate set was given to two annotators, who independently decided if the relation is correctly expressed in the sentence (see Han et al. 2018a for the full procedure). At the end, the annotation process yielded a balanced dataset of 100 relations, each containing 700 samples. Han et. al. argue that the most challenging aspect of their dataset is the diversity of relation expressions, whereby models require different and complex forms of reasoning in order to classify a specific sample (see Table 3.3).

Table 3.4 shows some typical samples from the FewRel dataset. Again, each sentence is labeled with the head and tail entity and the corresponding relation type. Compared to the rather generic relations of the SemEval dataset, which restrict the entity type to a lesser extent, the FewRel dataset contains mostly specific relations that narrow down the set of participating entities. For example, the relations “Location of Formation” is restricted to a location as the head entity, e.g. a city or country, while “Instrument” (*P1303*, “musical instrument that a person plays”) can only persist between an instrument and a person.



(a) Sentence lengths



(b) Entity lengths

Figure 3.3: Byte pair encoded sentence lengths (left) and byte pair encoded entity lengths (right) of the whole FewRel dataset. On average, FewRel contains longer sentences (30.47 BPE tokens) than the SemEval dataset (20.91 BPE tokens). The same goes for the byte pair entity length, with 3.24 (FewRel) compared to 1.23 tokens on average.

In comparison with the SemEval dataset, FewRel contains much longer sentences. This is illustrated in Figure 3.3: The byte pair encoded sentences contain 30.47 (FewRel) versus 20.91 (SemEval) tokens on average. Since longer sentences introduce additional noise, this makes the FewRel dataset more challenging. As illustrated in the right plot, the entities are also longer compared to the SemEval dataset (3.24 versus 1.23 BPE tokens on average). This makes the dataset especially challenging for joint entity and relation extraction, which is explored in Section 6, because the entity boundaries must also be detected in this task.

As mentioned before, FewRel was originally constructed for a few-shot scenario, so the full dataset is split into a train (74 relation types), validation (16 relation types) and a hidden, not publicly available, test (20 relation types) set of disjunctive relation types. In order to exploit the dataset for standard relation classification and joint entity and relation extraction, the training and validation set were merged, yielding a dataset of 80 different relations. The dataset is again divided into a training (600 samples per relation) and test dataset (100 samples per relation). This split is denoted as *standard classification split* from now on.

German Translation

In order to employ the FewRel dataset for German joint entity and relation extraction, a small part of the FewRel dataset was translated into German. The five relation types

"Country of Citizenship", "Director", "Instrument", "Architect" and "Participating Team" were selected. A set of 40 sentences were randomly chosen for reach relation. These samples were first automatically translated with Google's Cloud Translation API[2]. Next, the translated sentences were manually reviewed and corrected. After correction, each sentence was split into tokens with spaCy[3]. The two entities that are annotated in the original English FewRel dataset for each sentence were then manually assigned to the corresponding translated token sequence. Table 3.5 shows one example for each of the five relations. The German dataset was split into a training (30 samples per relation) and test dataset (10 samples per relation).

Relation	Entities
Country of Citizenship[a]	**Peter Veselovsk** (geboren 11. November 1964) spielte 1992 für die **Tschechoslowakei** in dem Team, dass die olympischen Bronzemedaille gewonnen hat.
Director[b]	Apurbas Debütfilm ist "**Gangster Returns**", der am 27. November 2015 unter der Regie von **Ashiqur Rahman** veröffentlicht wurde.
Instrument[c]	**Randy Meisner** spielte **Bass** und sang von 1961 bis 1965 mit einer lokalen Band namens The Dynamics.
Architect[d]	Er leitete den Bau des Hauptquartiers der Bank, des **Marine Trust Building**, entworfen von seinem Freund **Edward Brodhead Green** (1855-1950), einem bekannten Architekten aus Buffalo.
Participating Team[e]	Nieto stellte bei der **FIFA-Weltmeisterschaft** 2002 einen Rekord auf und teilte in einem Spiel zwischen **Deutschland** und Kamerun 14 gelbe und 2 rote Karten aus.

Table 3.5: Example sentence for each relation of the German FewRel dataset (head entity **red**, tail entity **blue**).

[a]"The object is a country that recognizes the subject as its citizen" (Wikidata)
[b]"Director(s) of film, TV-series, stageplay, video game or similar" (Wikidata)
[c]"Musical instrument that a person plays" (Wikidata)
[d]"Person or architectural firm that designed this building" (Wikidata)
[e]"[Team] (object) that actively takes/took part in an event or process (subject)" (Wikidata)

[2]`https://cloud.google.com/translate/`
[3]`https://spacy.io/`

Chapter 4

Relation Classification

In Chapter 2, the GP-Transformer was shown to learn the syntax and semantics of human language when pre-trained on a large corpus of unstructured text. By clustering the Transformer embeddings the model's capability in producing context-aware embeddings was confirmed. In this chapter, the knowledge of the pre-trained GP-Transformer is transferred to one of the key steps of relation extraction: The assignment of a relation to a pair of entities in a sentence. This task is commonly evaluated in a multi-way classification setting: Given a sentence s with labeled entities (e_1, e_2), the task is to complete the triple (e_1, *, e_2), i.e. to predict (e_1, $\underline{r}$, e_2), with a relation $r \in \mathcal{R}$ that is expressed in the sentence between the target entities. $\mathcal{R}$ refers to a finite set of relations (the classes in the multi-way classification setting) between which the model decides. Often, the directionality of the relation must be detected alongside the relation type: Is ($[e_1]_{\text{head}}$, $\underline{r}$, $[e_2]_{\text{tail}}$) or ($[e_2]_{\text{head}}$, $\underline{r}$, $[e_1]_{\text{tail}}$) expressed in the sentence? As explained in Section 3.1, the first entity is usually denoted as the *head* and the second entity as the *tail* of the relation. Traditionally, relation classification is tackled by extracting a variety of lexical features from the target sentence, e.g. WordNet hypernyms, grammatical relations, semantic roles or Levin classes (see Hendrickx et al. 2010 for example models). These features are then used to classify the relation. However, this approach requires the utilization of a large variety of different feature extractors and therefore an expensive and complex pre-processing pipeline. More recent publications take a different approach and train a single neural network on multi-way relation classification, getting rid of additional lexical features altogether. Today, competitive models are usually based on this new approach and range from recurrent neural networks like LSTMs (e.g. Xu, Mou, et al. 2015) to CNNs (e.g. Zeng et al. 2014a). Current state-of-the-art models combine RNNs or CNNs with an attention mechanism (e.g. Lee, Seo, and Choi 2019 or L. Wang et al. 2016) in order to identify the relevance of a word for a specific relation.

In this work, the pre-trained GP-Transformer, which is solely based on attention, is applied to the task of relation classification. In experiments on two datasets, the following questions are particularly addressed:

- How can previously applied methods to indicate the target entities of a sentence, like position indicators (D. Zhang and D. Wang 2015) and distance embeddings (Zeng et al. 2014a), be adapted for the GP-Transformer?
- As shown in Section 2.4, the embeddings produced by the GP-Transformer possess promising properties. How does the pre-trained GP-Transformer compare to other relation classification models, which usually employ fixed word embeddings as input?
- Is a comparably expensive fine-tuning of the GP-Transformer necessary or is a simple fully-connected feed-forward model, which operates on the context-aware embeddings produced by the pre-trained GP-Transformer, able to achieve similar results?

4.1 Related Work

This section gives an overview of other models published in the literature for relation classification. The section focuses on neural networks and is structured by the most common types: Recurrent neural networks, convolutional neural networks and models that incorporate or rely on the attention mechanism.

Recurrent/recursive neural networks Yin et al. (2017) compare several baseline architectures for relation classification, including a recurrent neural network with either an LSTM- (Hochreiter and Schmidhuber 1997) or GRU-Unit (Cho et al. 2014). The baseline models do not employ special modifications for relation classification or pre-trained word embeddings. These are just initialized randomly and refined during training. The output of the network's unit at the last time step is then used for relation classification. Their results suggests, that a simple model without pre-trained word embeddings is not sufficient for relation classification, and that adjustments targeted specifically at the relation classification task are required.

D. Zhang and D. Wang (2015) improve upon the baseline model with several modifications: They apply a bidirectional model with standard RNN units and introduce special position tokens that indicate the target entities. These entity indicators are added before and after each entity, resulting in four indicators in total. The relative positional information to the other words of the sentence can then be obtained through the recurrent processing. With these adjustment, the RNN is able to focus on relevant sections of the input sentence and is less prone to irrelevant noise.

Socher et al. (2012), who were one of the first to use neural networks for relation classification, introduced a matrix-vector (MV) Tree-RNN which operates on the syntax tree of a sentence: Each word or phrase is represented by both a matrix (modification of meaning of combined word/phrase) and a vector (meaning of component). Tree-RNNs combine words/phrases recursively bottom-up corresponding to the structure of a binarized constituency parse tree. According to Socher et al. (2012), the model is capable of capturing "semantic compositionality in a syntactically plausible way". For relation classification, Socher et al. employed the MV Tree-RNN only on the subtree that is spawned by the common ancestor node of the two entities of interest. According to the paper, this approach achieves a better performance because the relation type is mostly dependent on the context between the two entities.

Xu, Mou, et al. (2015) state that the shortest dependency path (SDP) between two entities in a dependency tree captures compactly the information that is most significant for the relation between these entities. As a consequence, they apply a LSTM only on the words of the sentence that are contained in the shortest dependency path. Because the direction of relations in a dependency tree matter, the authors apply two independent LSTM networks separately on the left and right subpath of the SDP. To do so, they split the SDP on the common ancestor node of the target entities. Xu et al. also show how the dropout of word embeddings benefits the generalization capabilities of a neural network for the relation classification task.

In a follow up paper, Xu, Jia, et al. (2016) improve upon the SDP-LSTM described above by building deep recurrent neural networks on the two subpaths instead of single layer LSTMs. They also introduce a novel data augmentation technique that increases the size of the training set by swapping the two subpaths of the SDP.

Furthermore Ebrahimi and Dou (2015) employ a tree-structured RNN on the shortest dependency path between two entities. Lastly, Miwa and Bansal (2016) stack bidirectional tree-structured LSTMs on bidirectional sequential LSTMs in order to infer the semantic relation between entities.

Convolutional neural networks Zeng et al. (2014a) employ a convolutional neural network for relation classification and establish a novel way of indicating the target entities, which subsequently published CNN-based models often rely on: Distance embeddings[1]. These features encode the relative distance of each position in the input sentence to the target entities. In the same way as the regular word embeddings, the

[1]Commonly denoted as position features. In this work, the term distance embeddings is used to distinguish them from the standard positional embeddings of the GP-Transformer.

relative distances are mapped to an embedding vector and adjusted during training. The final word representations, which serve as the CNN's input, are then obtained by concatenating the word embeddings with the two distance embeddings corresponding to the relative distance to the head and tail entity. Zeng et al. report an increase of almost 10% macro-F1 by using distance embeddings versus word embeddings alone.

Santos, Xiang, and B. Zhou (2015) add a pair-wise ranking loss to a CNN which is designed to diminish the negative impact of the noisy "Other " relation (SemEval dataset, see Section 3.2) on the classification accuracy. Similar to Zeng et al. (2014a), they indicate the target entities with distance embeddings. By employing the ranking loss instead of a regular softmax classifier, they report an improvement of 1.59 macro-F1.

Attention-based models The attention mechanism has recently been also applied to relation classification: P. Zhou, Shi, et al. (2016) combine a bidirectional LSTM with an attention function by computing the weighted sum over the LSTM's output vectors. They employ the entity indicators introduced by D. Zhang and D. Wang (2015) and report an improvement in F1 score by adding the attention mechanism.

Y. Zhang et al. (2017) propose position-aware attention, which is specifically designed for relation extraction. The position-aware attention is applied on top of a regular, unidirectional LSTM and uses learned entity position embeddings in addition to the LSTM's output vectors in order to compute the attention scores. They compare their architecture with some competitive baseline models like Xu, Mou, et al. (2015) and accomplish the best performance on the TACRED dataset (Y. Zhang et al. 2017). Moreover, Zhang et. al. demonstrate that a higher robustness for long sentence lengths can be achieved by incorporating position-aware attention into the model.

Lee, Seo, and Choi (2019) also apply a position-aware attention mechanism on top of a model that consists of both an self-attention layer (similar to the one applied in the GP-Transformer) and a bidirectional LSTM network. In contrast to Y. Zhang et al. (2017), they also incorporate entity types into the attention calculation and argue that these could provide powerful hints for relation classification.

The model introduced by Shen and Huang (2016) combines a single-layer convolutional neural network with an novel attention network, which operates on word-entity pairs: By computing the attention score based on the concatenation of word and entity embeddings, the network is able to weight each word of the input by its relevance to the target entities. The convolution vector and the two context-vectors (corresponding to the relevance to the head and tail entity respectively) are then concatenated and fed into a final fully-connected layer for classification.

L. Wang et al. (2016) employ a word-entity attention mechanism in a multi-level CNN. The attention score for each word-entity combination is computed by the inner product of the corresponding embeddings. In contrast to Shen and Huang (2016), each word is then weighted by the average of the attention score to the head and tail entity. Additionally, Wang et al. rely on an attention-based pooling after the convolutional layer to select only relevant sections for relation classification.

Finally, other Transformer-based models were recently applied to relation classification: The TRE (Transformer based Relation Extraction) model by Alt, Hübner, and Hennig (2019) is closely related to the one employed in this work. Alt et al. also utilize the GP-Transformer for relation classification (Radford et al. 2018). They make the model aware of the two target entities by copying them to the start of the sentence (separated by a special token), followed by the sentence itself. By doing so, Alt et al. report strong results on the SemEval and TACRED dataset.

Verga, Strubell, and McCallum (2018) employ a Transformer encoder to simultaneously classify the relations between all marked entities in a paragraph: The paragraph is only fed once through the model and entity-pair relation scores are obtained based on the Transformer embeddings that correspond to an entity mention. They report state-of-the-art results on a biomedical dataset. In contrast to this work, Verga et al. do not utilize language modeling pre-training. In a similar approach, H. Wang et al. (2019) also score all labeled entity mentions simultaneously. However, in contrast to Verga et al. they rely on the pre-trained bidirectional BERT (Devlin et al. 2018) model. By encoding the distance of a word to every entity in the self-attention layers, they achieve the current relation classification state-of-the-art on the SemEval dataset (see Section 3.2 for dataset description).

4.2 Approach

This work follows the approach employed by Radford et al. (2018) for other NLP classification tasks: To classify a sentence, special <*Start*> and <*End*> tokens are added to the beginning and end of the input sequence, respectively (see Figure 4.1). These tokens are added to the vocabulary and adjusted during training. Because the <*End*> token is able to attend to all previous words and therefore the whole input sentence, it represents the sentence's context. The Transformer embedding that corresponds to the *End* token, $\mathbf{h}_{\text{<End>}}$, is later used for classification.

In relation classification, it is important to know what the target entities are: Since several relations can be expressed in a single sentence simultaneously and words near the entities may be more significant for the expressed relation, state-of-the-art

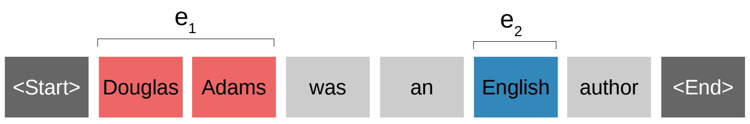


Figure 4.1: Radford et al. fine tune the GP-Transformer by converting an arbitrary input into a sequence. They also add special tokens at the start and end of the sequence. The vector representation of the last token, the <*End*> token, after the final Transformer block is then used for classification. In this figure, e_1 denotes the first entity of the sentence and e_2 the second entity.

models encode positional information to the target entities. Take the sentence “A team of students from Virginia Tech University have created a vehicle that allows vision impaired drivers to take control of the wheel” for example. Without the knowledge of the entities, which relation we want to classify, there is no way to infer if “Product-Producer”[2] ([*team of students*]$_{\text{head}}$, [*vehicle*]$_{\text{tail}}$) or “Agency-Instrument”[3] ([*drivers*]$_{\text{head}}$, [*wheel*]$_{\text{tail}}$) is the target relation. However, with the knowledge of the target entities, “drivers” and “wheel”, the model is able to focus on the relevant section of the input and to infer the correct relation. Two common approaches to mark the target entities are *entity indicators* (EI) (D. Zhang and D. Wang 2015) and *distance embeddings* (DE) (Zeng et al. 2014a)[4]. Entity indicators are frequently employed in recurrent models, while distance embeddings are usually found in convolutional neural networks. In this work, both approaches are applied to the GP-Transformer in order do evaluate which method performs better in the self-attention based model. Additionally, three other methods for entity indication that rely on an averaging of the final Transformer embeddings are also employed in this work and are compared to the aforementioned methods.

Method 1: Entity Indicators

Entity indicators (EI) were first proposed by D. Zhang and D. Wang (2015) and employed in a standard recurrent neural network. By feeding special position indicators into the network before and after the target entities, they improved the classification accuracy significantly by about 10% macro-F1. The approach is adapted for the non-recurrent GP-Transformer by inserting the indicators before and after the first entity (e_1) and second entity (e_2) into the sequence. The position indicators and the <*Start*> and <*End*> tokens are then added to the vocabulary and treated the same way as the BPE tokens. For example, if $V = 100$ (with V again denoting the vocabulary size), the vocabulary IDs from 101 to 106 are assigned to the six special tokens (see Figure 4.2). The embeddings that correspond to the

[2]“A producer causes a product to exist.” (Hendrickx et al. 2010)

[3]“An agent uses an instrument.” (Hendrickx et al. 2010)

[4]These are also commonly denoted as position indicators and position embeddings. Since the GP-Transformer already employs standard position embeddings, the two approaches are referred to as entity indicators and distance embeddings in this work

special tokens are then placed in the same embedding matrix $E \in \mathbb{R}^{V + l_{max} + 6}$ (with l_{max} denoting the maximum sentence length in the dataset) as those of the BPE token and regular position embeddings (see Section 2.2.2).

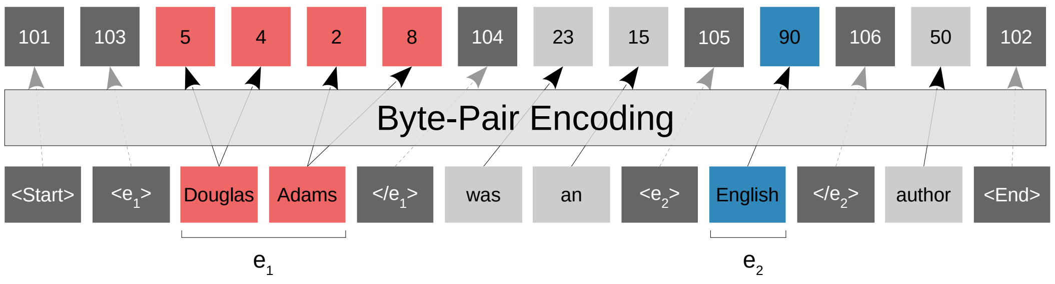


Figure 4.2: Tokens are byte pair encoded and mapped to a vocabulary ID. The special entity indicators as well as the <*Start*> and <*End*> tokens are assigned IDs 101-106, which did not exist in the vocabulary before. Here e_1 denotes the first entity in the sentence and e_2 the second entity.

Method 2: Distance Embeddings

Distance embeddings (DE), introduced by Zeng et al. (2014a), encode the relative distance of each word (or token) in the given sentence to the two entities. So there exists embeddings (..., $\mathbf{q}^1_{-2}$, $\mathbf{q}^1_{-1}$, $\mathbf{q}^1_0$, $\mathbf{q}^1_1$, ...) for the relative distances to e_1. Likewise the relative distances to e_2 are encoded, resulting in embeddings (..., $\mathbf{q}^2_{-2}$, $\mathbf{q}^2_{-1}$, $\mathbf{q}^2_0$, $\mathbf{q}^2_1$, ...). This procedure is visualized in Figure 4.3.

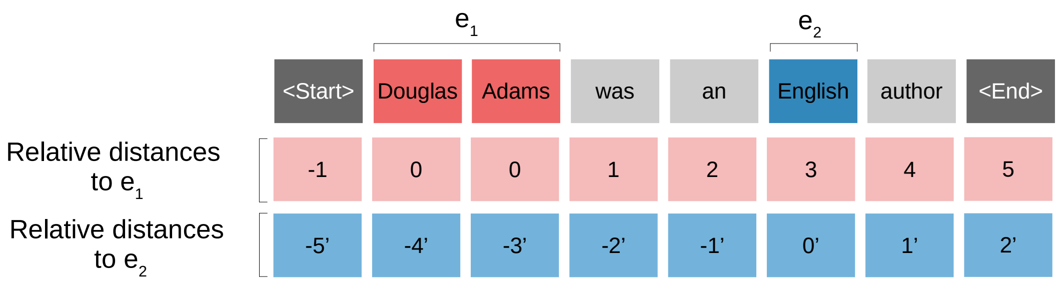


Figure 4.3: Distance embeddings encode the relative distance two the target entities. In this work, separate distinct embeddings are trained for the relative distance to the first and second entity independently.

Since the relative distance of each word is computed with respect to both entities and a single entity is composed of at minimum one token, the size of the set of required distance embeddings, Q, can be calculated by:

$$|Q| = (l \cdot 2 - 1) \cdot 2 \tag{4.1}$$

for an input sentence of l tokens. As before, the distance embeddings are placed in the same embedding matrix E as those of the BPE and regular position embeddings and adjusted during training.

In the original implementation by Zeng et al. the distance embeddings $\mathbf{d}_i^1, \mathbf{d}_i^2$ for a specific token t_i are concatenated with the corresponding token embedding u_i and fed into the CNN. If $\mathbf{u}_i \in \mathbb{R}^d$ and $\mathbf{q}_i^1, \mathbf{q}_i^2 \in \mathbb{R}^a$, this approach results in $d + 2 \cdot a$ dimensional input vectors $\mathbf{x}_i$. Since the GP-Transformer is pre-trained and the dimensionality of the model's layers is already fixed, distance embeddings cannot be applied without any adjustment. One obvious adaption is to concatenate the distance embeddings *after* the GP-Transformer and before the classifier to the context-aware embeddings. However, with this approach the additional information gained by distance embeddings cannot be utilized in the pre-trained Transformer layers. Another option might be to concatenate the vectors in advance and map them back to the original embedding space of the Transformer by a simple linear transformation, but this introduces additional parameters to the model. In this work, the distance embeddings $\mathbf{q}_i^1, \mathbf{q}_i^2 \in \mathbb{R}^d$ are instead added to those of the token $\mathbf{u}_i \in \mathbb{R}^d$ and position embeddings $\mathbf{p}_i \in \mathbb{R}^d$ before the GP-Transformer:

$$\mathbf{x}_i = \mathbf{u}_i + \mathbf{p}_i + \mathbf{q}_i^1 + \mathbf{q}_i^2 \tag{4.2}$$

With this approach, the model may be able to infer the entities and relative distances by the shift introduced by adding the distance embeddings. Moreover, no additional parameters are added to the model and the entity information can potentially be utilized throughout the model.

Method 3: Post-Indication

In addition to entity indicators and distance embeddings, three simple approaches were explored that indicate the target entities *after* the GP-Transformer. An input sentence is first encoded by the GP-Transformer, yielding context-aware Transformer embeddings $(\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3, ..., \mathbf{h}_l)$, with each $\mathbf{h}_i \in \mathbb{R}^d$, per BPE token for a sentence of length l. Based on an averaging of sentence segments an entity-aware encoding $\mathbf{m}$ of the input sentence is obtained. This sentence encoding is then used instead of $\mathbf{h}_{\text{<End>}}$ and is linearly mapped to the relations. The following sentence encodings where explored:

- The **entity average encoding** (*Entity-Avg*) averages the Transformer embeddings of both entities and concatenates them with the averaged context (the full sentence). When $(\mathbf{h}_1^{e_1}, \mathbf{h}_2^{e_1}, ..., \mathbf{h}_w^{e_1})$ correspond to the Transformer embeddings of the sentence's first entity e_1 and $(\mathbf{h}_1^{e_2}, \mathbf{h}_2^{e_2}, ..., \mathbf{h}_s^{e_2})$ to those of the second entity e_2, they are first averaged:

$$\begin{aligned}\mathbf{v}_{e_1} &= \frac{1}{w}\sum_{i=1}^{w}\mathbf{h}_i^{e_1} \\ \mathbf{v}_{e_2} &= \frac{1}{s}\sum_{i=1}^{s}\mathbf{h}_i^{e_2}\end{aligned} \tag{4.3}$$

with $\mathbf{v}_{e_1} \in \mathbb{R}^d$ and $\mathbf{v}_{e_2} \in \mathbb{R}^d$. The whole context is also averaged:

$$\mathbf{v}_c = \frac{1}{l}\sum_{i=1}^{l}\mathbf{h}_i \tag{4.4}$$

with $\mathbf{v}_c \in \mathbb{R}^d$. To obtain the entity-aware sentence encoding $\mathbf{m} \in \mathbb{R}^{d \cdot 3}$, the resulting vectors are concatenated:

$$\mathbf{m} = \mathbf{v}_{e_1} \circ \mathbf{v}_{e_2} \circ \mathbf{v}_c \tag{4.5}$$

No <*Start*> or <*End*> tokens are added to the sequence in the entity-avg encoding.

- The **segment average encoding** (*Segment-Avg*) averages every segment of the sequence independently and concatenates the resulting vectors. Segments are the context before e_1, e_1 itself, context between e_1 and e_2, e_2 itself and context after e_2. This results in a final sentence encoding $\mathbf{m} \in \mathbb{R}^{d \cdot 5}$. The procedure is visualized in Figure 4.4. Again no <*Start*> or <*End*> tokens are added to the sequence in this case.

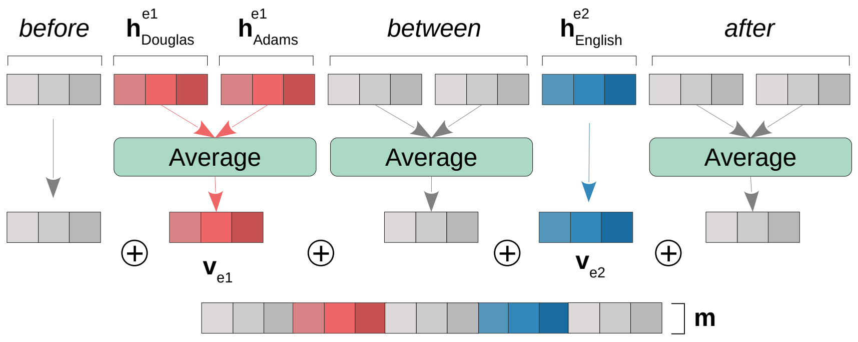


Figure 4.4: Example visualization to obtain the entity-aware sentence encoding **m** based on averaging of Transformer embeddings. Depicted is the segment average encoding for an example sentence with the two entities “Douglas Adams” and “English”.

- The **mixed encoding** (*Mixed*) is similar to the entity average encoding: The Transformer embeddings of both entities are averaged and concatenated. Instead of additionally concatenating the averaged context vector, the Transformer embedding of the final *<End>* token is used. So $\mathbf{m} = \mathbf{v}_{e_1} \circ \mathbf{v}_{e_2} \circ \mathbf{h}_{End}$ and $\mathbf{m} \in \mathbb{R}^{d \cdot 3}$.

Implementation wise, bit-masks are used to average entities (or segments) of the input sentence. These bit-masks are created as tensors in a preprocessing step for every segment (*before*, e_1, *between*, e_2, *after*) of the input sentence and are then applied during training and inference dependent on the used encoding.

Architecture

The model's architecture in regards to the three entity marking methods discussed previously is illustrated in Figure 4.5. A pre-trained GP-Transformer functions as the core part of the model. In a first step, the maximum sentence length l_{max} of all sentences in the dataset is determined. Special padding tokens are added to every sentence that is shorter than l_{max}. As for pre-training, the *Unknown* token is used for padding.

Given a padded sentence, all initial token embeddings (with position and potentially distance embeddings added) are packed into a matrix $X \in \mathbb{R}^{l_{max} \times d}$ where d again denotes the Transformer's embedding dimensionality. X is then fed through the GP-Transformer (see Section 2.2.2):

$$H = \text{GP-Transformer}(X) \tag{4.6}$$

with $H \in \mathbb{R}^{l_{max} \times d}$. Here H can be interpreted as a sequence $(\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_{l_{max}})$ of context-aware Transformer embeddings $\mathbf{h}_i \in \mathbb{R}^d$. Dependent on the entity marking method being used, let $\mathbf{g} \in \mathbb{R}^{d \cdot a}$ either denote the Transformer *<End>* token embedding[5], $\mathbf{h}_{<End>}$, or one of the post-indication encodings. Here a dependents on the method used for entity marking ($a = 1$ for entity indicators and distance embeddings, $a = 3$ for *Entity-Avg* and *Mixed*, $a = 5$ for *Segment-Avg*). For classification, this vector is then mapped to the relation classes:

$$\hat{\mathbf{y}} = \text{softmax}(W \cdot \mathbf{g} + \mathbf{b}) \tag{4.7}$$

where $W \in \mathbb{R}^{|\mathcal{R}| \cdot 2 \times d \cdot a}$, $\mathbf{b}, \hat{\mathbf{y}} \in \mathbb{R}^{|\mathcal{R}| \cdot 2}$. By applying the softmax function, a probability distribution over all relations is obtained. Since the directionality of the relation must also be inferred, the number of classes is doubled, therefore adding one output node that indicates the reverse directionality (tail occurs before the head in the sentence) for each relation.

[5]Since sentences are padded, this is usually not the last token in the sequence.

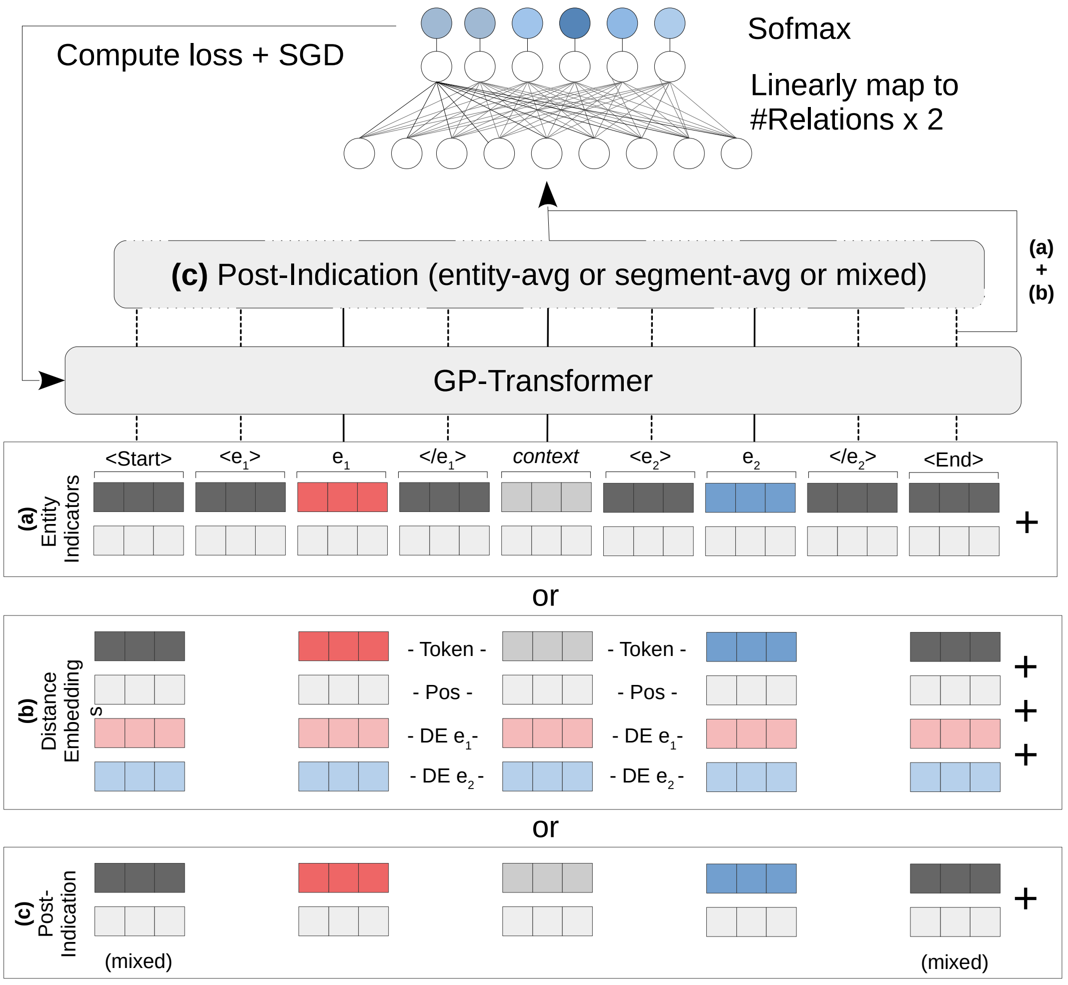


Figure 4.5: To make the GP-Transformer aware of the target entities, **(a)** insert special entity indicators, **(b)** add embeddings that encode the relative distance of each token to the entities or **(c)** post-indicate the entities by special entity-aware encodings. Note that only a simple example of three tokens (e_1, context, e_2) is depicted and the context (also to either side of the entities) is usually larger.

In contrast to pre-training (Chapter 2), the model is fine-tuned in a supervised manner on relation classification: Given a set of sentences with marked target entities and the relation that is expressed between the entities (as illustrated in Section 3.2 and 3.3) the model's weights are updated in each training iteration. To assess the ability of the model to correctly classify a sample, a negative log-likelihood loss of the predicted probability $\hat{\mathbf{y}}_r$, which corresponds to the ground truth relation r of the respective sample, is minimized by stochastic gradient descent:

$$\text{Loss}_{\text{task}} = -log\ \hat{\mathbf{y}}_r \tag{4.8}$$

Additionally, Radford et al. (2018) employ language modeling as an auxiliary objective during fine-tuning, which is adopted in this work:

$$\text{Loss} = \text{Loss}_{\text{task}} + \alpha \cdot \text{Loss}_{\text{LM}} \tag{4.9}$$

Here α controls the weight of the language modeling loss in the total loss. See Section 2.2.2 (especially Equation 2.12) for the description of the language modeling loss Loss_{LM}.

Note that in practice the model is trained on mini-batches of samples instead of single samples. In this case, the loss is summed over all samples of the batch. The model is trained for a set amounts of epochs. Here an epoch denotes a training run over the entire training set. The training procedure executes the following steps:

1. Dependent on the method used to mark target entities, (a) entity indicators, (b) distance embeddings, (c) post-indication, pre-process each sentence of the dataset:
   - (a) Add entity indicators and <*Start*> + <*End*> tokens to the sentences.
   - (b) Create, for each sample sentence, the sequence of embedding matrix indices that correspond to the distance embeddings.
   - (c) Only for *Mixed* encoding: Add <*Start*> + <*End*> tokens.
2. Randomly shuffle the training samples in the beginning of each epoch.
3. Create a mini-batch of samples.
4. Assignment of corresponding embedding vectors.
   - (a) Embed the BPE tokens, position embeddings and special tokens.
   - (b) Embed the BPE tokens, position embeddings and distance embeddings.
   - (c) Embed the BPE tokens and position embeddings (and <*Start*> + <*End*> special tokens in the *Mixed* encoding).
5. Sum the resulting embedding vectors.
6. Feed the batch through the GP-Transformer and obtain the context-aware embeddings after the last Transformer block.
   - (a+b) Extract the Transformer embedding $\mathbf{h}_{End}$ corresponding to the *End* token for each sample.

(c) Apply one of the previously (Section 4.2) described post-identification methods.

7. Apply Equation 4.7 to obtain a probability distribution over relations for each sample.

8. Calculate the loss (4.9) and update the model's weights using SGD.

Steps (3-8) are repeated for each mini-batch of the training data for a set amount of epochs.

4.3 Experiments

In the following experiments, the GP-Transformer's performance is assessed for relation classification. Particularly the focus is on evaluating the performance impact of the different entity marking methods described in Section 4.2 and to validate the effect of language model pre-training.

Setup

Because the pre-trained English GP-Transformer (Radford et al. 2018) is employed as the core component for relation classification, the model's architecture is fixed and most hyperparameters remain the default values (e.g. 12 Transformer blocks, 12 heads per block, 768 dimensional embeddings). Special tokens and the weights of the final classification layer are initialized with values drawn from a normal distribution ($\mu = 0$, $\sigma^2 = 0.02$). The final classification layer's bias is initialized with 0. Akin to Radford et al., a linear increasing and decreasing (warmup over 0.2% of samples) learning rate with a peak value of $6.25 \cdot 10^{-5}$ is applied during fine-tuning. The mini-batch size is set to 32 and α to 0.5 (see Equation 4.9). A dropout with a rate of 0.1 is applied to the classifier. In each setting, fine tuning the GP-Transformer for three epochs proved sufficient for the FewRel dataset. More epochs led to stagnating results and eventually overfitting on the training dataset. Because the SemEval dataset is rather small compared to FewRel, the loss is still decreasing after three epochs. Therefore, the GP-Transformer was fine tuned for five instead of three epochs on the SemEval dataset.

In order to evaluate the model, the accuracy, precision, recall and F1 scores (macro-averaged over relation classes) are computed for both the SemEval and the FewRel test dataset. The directionality of the relation must also be predicted for both datasets. By that, the number of classes is 160 ($80 \cdot 2$) for FewRel and 19 ($9 \cdot 2 + 1$) for SemEval (directionality cannot be inferred for the "Other" class). For SemEval, the official train/test split was used. Here the official Macro-F1 score which excludes the "Other" class and takes directionality into account[6] is also reported. For FewRel, the standard classification split is used (3.3).

A bidirectional recurrent neural network with gated recurrent units (see Cho et al. 2014 for GRU reference) is employed as a strong baseline model (BI-GRU). The model uses 300-dimensional GloVe[7] embeddings as input, which are pre-trained on the English Wikipedia corpus[8]. Entity indicators are fed into the GRU before and after each entity. The learning rate is set to 0.001 and weight decay is added with the regularization parameter set to 0.001. Training is conducted for 30 epochs with a batch size of 10. The Adam optimizer is used and a step learning rate decay is applied by reducing *lr* by half every 5 epochs. The embedding layer units are dropped out with a probability of 50% and weights were initialized by using Xavier initialization (Glorot and Bengio 2010), while the biases are initialized by a uniform distribution. The bidirectional GRU was optimized in previous work on the SemEval dataset.

Comparison with State-of-the-Art

Table 4.1 displays the evaluation results on the SemEval dataset. The averaged scores of five runs are reported for each model. As expected, the model performs substantially worse without entity markers (GPT) compared to adding entity indicators (GPT+EI) or distance embeddings (GPT+DE). While both methods of indicating the target entities perform well on the dataset, entity indicators outperform distance embeddings by about 4.8% official macro-F1. Moreover, the GP-Transformer surpasses the already strong bidirectional GRU baseline by about 3.7 official macro-F1.

Model	Accuracy	Precision	Recall	Macro-F1	O-Macro-F1
GPT + EI	**83.77**	**79.28**	**80.29**	**79.72**	**87.34**
GPT + DE	78.77	74.28	74.55	74.3	82.53
GPT + Mixed	82.38	78.7	77.87	78.12	86.11
GPT + Segment-Avg	82.66	79.48	78.48	78.81	86.52
GPT + Entity-Avg	82.75	79.32	78.24	78.51	86.3
GPT	74.52	70.77	69.75	70.14	78.42
Bi-GRU (Baseline)	79.85	75.69	75.83	75.55	83.64

Table 4.1: Average of five runs (SemEval) for each model. In the bottommost model (GPT), entities are not marked.

[6]How the directionality is taken into account is not further specified in Hendrickx et al. (2010). The official scorer is utilized in this work.

[7]from `https://nlp.stanford.edu/projects/glove/`

[8]6B tokens, 400K vocabulary size

Albeit their simplicity, the post-identification methods perform similarly well and rank only slight behind GPT+EI, even surpassing GPT+DE and the baseline. These results show, that while indicating the entities throughout all GP-Transformer layers (as in GPT+EI) improves the performance, competitive results are also obtained when the GPT-Transformer itself has no notion of the entities by indicating them only before the final classification layer. In this case, simply averaging the Transformer embeddings of the entities (or all segments as in Segment-Avg) before classification improves the official macro-F1 by about 8 (GPT to GPT+Segment-Avg).

As visible in Table 4.2 (right), training the GP-Transformer on the task is fairly stable over the five runs, resulting in a low standard deviation of 0.23. With an official macro-F1 score of 87.34, the GP-Transformer with entity indicators ranks third best behind the Multi-Level Attention CNN by L. Wang et al. (2016) and the entity-aware BERT (H. Wang et al. 2019) (Table 4.2 left). A more detailed list of results that were reported for other models in the literature can be found in Appendix D.1. The training loss curve as well as the test set official macro-F1 over epochs is included in Appendix C.1 for the best performing model (Run 4 in Table 4.2).

Model	M-F1
Entity-Aware BERT$_{SP}$ (H. Wang et al. 2019)	89.0
Multi-Level Attention (L. Wang et al. 2016)	88.0
GPT + EI **(own)**	87.34
TRE (Alt, Hübner, and Hennig 2019)	87.1
SDP deep RNN (Xu, Jia, et al. 2016)	86.10
Attention CNN (Shen and Huang 2016)	85.9

Run	Off-Macro-F1
1	87.10
2	87.15
3	87.43
4	87.66
5	87.34
Avg.	87.34
Stdev.	0.23
Best	87.71

Table 4.2: **(Left)** Selection of results reported on the SemEval dataset (Official-Macro-F1). The GP-Transformer with entity indicators ranks second best, slightly behind the Multi-Level Attention CNN by L. Wang et al. 2016. **(Right)** Official-Macro-F1 scores of the best performing architecture (GPT+EI) over five runs.

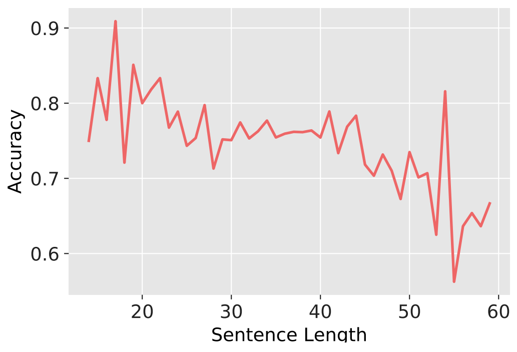
Table 4.3 displays the evaluation results of the FewRel dataset. Again, the average of five runs is reported for each model. The results are similar to those of the SemEval dataset, with entity indicators slightly outperforming distance embeddings. In comparison to SemEval, the difference between using no entity markers at all (GPT) to employing entity indicators is much larger (about 22% macro-F1 for FewRel and 10% macro-F1 for SemEval). Since FewRel contains a large variety of common

Model	Accuracy	Precision	Recall	Macro-F1
GPT + EI	**76.01**	**74.57**	**74.81**	**74.32**
GPT + DE	75.03	73.51	73.69	73.22
GPT + Mixed	74.08	72.74	72.93	72.52
GPT + Segment-Avg	73.93	72.47	72.43	72.14
GPT + Entity-Avg	73.55	72.3	72.09	71.85
GPT	55.77	52.68	53.12	52.15
Bi-GRU (Baseline)	74.17	72.77	72.48	72.03

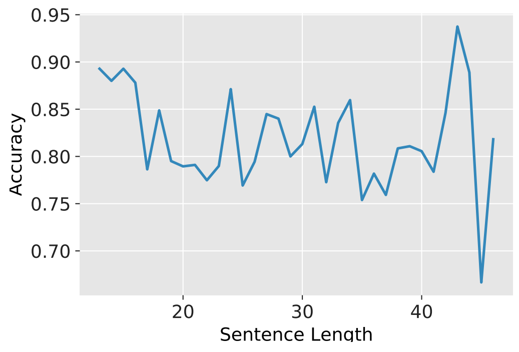
Table 4.3: Average of 5 runs (FewRel) for each model. In the bottommost model (GPT), entities are not marked.

relation types, it is more likely that multiple relations are expressed in a single sentence. Along with the fact that the FewRel dataset contains longer sentences on average (see 3.3), this introduces additional noise and makes the spotting of relevant phrases that suggest a specific relation more challenging without any indication of the target entities. As for SemEval, the post-indication methods perform similar well. The best post-indication method is GPT+Mixed with 72.52 macro-F1 (about 1.8 worse than GPT+EI). However, the GP-Transformer with distance embeddings (GPT+DE) outperforms the post indication methods on this dataset. One explanation is that the distance embeddings require a larger and more diverse dataset (such as FewRel) to be trained successfully. The training loss curve as well as the test set macro-F1 over epochs is included in Appendix C.2 for the best of the five GPT+EI runs (74.60 macro-F1).

Effect of Sentence Length



(a) FewRel



(b) SemEval

Figure 4.6: Influence of the sentence length on the classification quality for the FewRel (left) and SemEval (right) dataset. Only sentence lengths with more than 10 samples were taken into account.

The influence of longer sequences on the classification quality is depicted in Figure 4.6. The negative effect of longer sequences is especially obvious for the FewRel dataset, with the accuracy dropping after a sentence length of about 40 BPE tokens. These results indicate that more sophisticated methods may be necessary to diminish the negative influence of noise in relation classification.

Error Inspection

To gain an insight into common error sources, wrong relation assignments were manually inspected. Table 4.4 shows a selection of common error cases for the SemEval dataset[9]. Remarkably is the frequent confusion with the "Other" class: In about 74% of cases (321 of 433 misclassified samples), a sample was falsely assigned to the "Other" class or a sample from the "Other" class was assigned to one of the proper relations. It is reasonable to assume that the noisy nature of the "Other" class, containing entity pairs that are not related by one of the SemEval relations, is hardly distinguishable from the proper relations.

GT	Prediction	Sentence
Other	Instrument-Agency[a]	The author **index** was generated using the latex authorindex **package**.
Component-Whole[b]	Other	A more spare, less robust use of classical [**motifs**]$_{\text{head}}$ is evident in a [**ewer**]$_{\text{tail}}$ of 1784-85.
Other	Message-Topic[c]	However several **problems** were pointed out during the course of this feasibility **study**.
Component-Whole[b]	Member-Collection[d]	Now this [**laboratory**]$_{\text{head}}$ also is part of a larger [**organisation**]$_{\text{tail}}$.
Instrument-Agency[a]	Entity-Destination[e]	At my work, an [**electronic engineer**]$_{\text{tail}}$ has migrated into [**embedded software**]$_{\text{head}}$.
Component-Whole[b]	Component-Whole[b]	The [**chain**]$_{\text{tail}}$ of [**Hawaiian**]$_{\text{head}}$ islands provides evidence that the Pacific Plate moves to the northwest.

Table 4.4: Common error cases for the SemEval dataset: Confusion with "Other" class (top) and misclassifications between proper relations (bottom). [**Hawaiian**]$_{\text{head}}$ denotes that "Hawaiian" was predicted as the relation's tail, but is actually the head according to the ground truth.

[a]"An agent uses an instrument" (Hendrickx et al. 2010)
[b]"An object is a component of a larger whole" (Hendrickx et al. 2010)
[c]"A message, written or spoken, is about a topic" (Hendrickx et al. 2010)
[d]"A member forms a nonfunctional part of a collection" (Hendrickx et al. 2010)
[e]"An entity is moving towards a destination" (Hendrickx et al. 2010)

Table 4.4 illustrates common "error cases" for the FewRel dataset. Upon closer inspection, many "error cases" were found to be actually correctly classified samples and are mainly attributed to missing annotations in the FewRel dataset: Because just a single relation is annotated in FewRel for each sentence but multiple different rela-

[9]Predicted by the best performing model, run 4 in Table 4.2 (right)

GT	**Prediction**	**Sentence**
Notable Work[a]	Architect[b]	The design inspired [**Frei Otto**]$_{\textbf{head}}$'s arena designs for the [**Olympic Stadium**]$_{\textbf{tail}}$ in Munich.
Head of Government[c]	Country of Citizenship[d]	On 5 August 2011, Kazamias was appointed finance minister of [**Cyprus**]$_{\textbf{head}}$ by [**Demetris Christofias**]$_{\textbf{tail}}$ replacing Charilaos Stavrakis while still serving his term.
Residence[e]	Head of Government[c]	He was a candidate for governor of [**Connecticut**]$_{\textbf{tail}}$, losing to [**Abiram Chamberlai**]$_{\textbf{head}}$.

Table 4.5: In FewRel, many "error cases" are attributed to multiple relations that are expressed in the same sentence between the two target entities: While the model actually predicted correct relations in the above examples, the prediction is counted as an error due to missing annotations in the dataset. [**Frei Otto**]$_{\textbf{head}}$ denotes that "Frei Otto" was predicted as the relation's tail, but is actually the head according to the ground truth.

[a]"Notable scientific, artistic or literary work, or other work of significance among subject's works" (Wikidata)
[b]"Person or architectural firm that designed this building" (Wikidata)
[c]"Head of the executive power of this town, city, [...] or other governmental body" (Wikidata)
[d]"the object is a country that recognizes the subject as its citizen" (Wikidata)
[e]"The place where the person is or has been, resident" (Wikidata)

tions are potentially expressed between the same entity pair, a sample is frequently misclassified by the model according to the ground truth[10]. A typical example can be found in the middle of Table 4.4: Because "Head of Government" commonly entails "Country of Citizenship" the model frequently misclassifies a sample according to the ground truth despite actually predicting a correct relation. With this, the actual model's performance on FewRel is probably higher than reported in Table 4.3, but cannot fully be correctly assessed due to missing annotations.

Ablation Studies

To gain a better insight into the combination of relation classification and transfer learning, additional ablation studies are conducted on four different settings:

- **(freeze)** In order to verify if an expensive fine-tuning on the target domain is necessary, or if the model is able to extract relevant information from the context-aware embeddings alone, the GP-Transformer is not fine-tuned in this setting. Since there is no point in adding additional special tokens without fine-tuning, the *Segment-Avg* post-indication method is employed in this setting.

[10]This is not only a problem of the standard classification split used for relation classification (see Section 3.3), but also occurs in the original FewRel split that is used for few-shot classification later in this work: In various cases, multiple relations of the original train or validation set were found to be expressed between a single entity pair.

- **(entities)** As mentioned in Section 3.3, the FewRel dataset contains multiple relations that restrict the choice of possible entities. Because the GP-Transformer places words in close proximity in the embedding space when they share certain semantic similarities (like person names, see Section 2.4) it may be possible to predict a relation based on the two entities alone, which is assessed in this setting. Instead of inserting entity indicators into the sequence, the two entities in a sample are simply separated by a special *separate* token, which is shown in Figure 4.7. As before, the *End* token is used for classification.

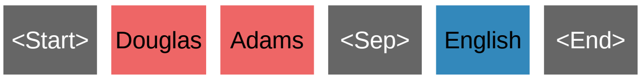


Figure 4.7: In the **entities** setting the participating entities are separated by a special *separate* token (*<Sep>*), while the surrounding context is omitted.

- **(from-scratch)** Learning properties of and dependencies between words in an unsupervised manner on plenty of data is the key benefit of employing pre-trained word embeddings or language models like the GP-Transformer. In order to check if pre-training is also beneficial for the rather large FewRel dataset, the GPT+EI model is trained from-scratch in this setting.

- **(BPE only)** The main argument for transferring whole models to a new domain instead of employing fixed word embeddings, is their ability to learn word representations dependent on the specific input context. By omitting the GP-Transformer's weights and only retaining the learned BPE token embeddings, the advantages of learning context-aware embeddings for relation classification can be assessed in this setting. This experiment is conducted with the GPT+EI model.

The results of the four ablation studies are depicted in Figure 4.8. *Original* refers to the fine-tuned GPT+EI model from Table 4.3: When the GP-Transformer is not fine-tuned on the target domain (*freeze* setting) and the default hyperparameters outlined in the beginning of this section (same as for *original*) are used, a considerable worse performance is achieved (54.03 to 74.32 macro-F1). Instead of using the default hyperparameters, the learning rate was additionally tuned for this setting by a manual search over different values: In contrast to fine-tuning (*original*), the *freeze* model performs considerably better with a higher learning rate of 0.001[11]. Still, the *freeze* model is by about 8.68 macro-F1 score behind the *original* model (65.64 to

[11]When fine-tuning is enabled (*original*), the model's loss does not decrease with such a high learning rate. Training the model for more epochs in the *freeze* setting amounts to roughly the same score as with a learning rate of 0.001 (tested with up to 12 epochs).

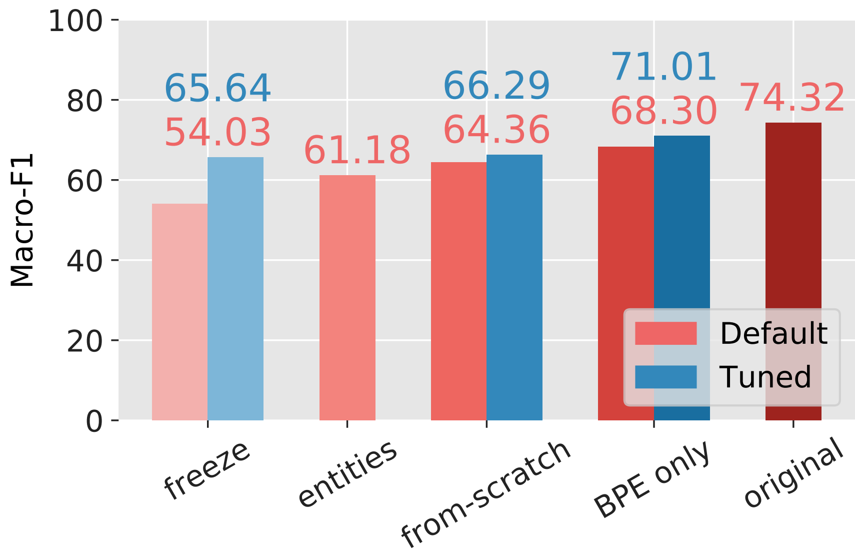


Figure 4.8: Ablation studies on the FewRel dataset: **(freeze)** The GP-Transformer is not fine-tuned on relation classification. **(entities)** Predict relationship solely with regards to the participating entities. **(from-scratch)** Omit weights and BPE embeddings of the pre-trained GP-Transformer and train it from-scratch. **(BPE only)** Only omit GP-Transformer weights but retain BPE embeddings. Ablation studies are either conducted with the default hyperparameters (as outlined in the beginning of this section) or specifically tuned hyperparameters.

74.32). This confirms that a fine-tuning of the GP-Transformer is essential to detect relations between entities or that more sophisticated downstream models like LSTMs are needed to reach a similar performance. However, this results in an even more expensive training process, whereas fine-tuning the attention-based GP-Transformer alone is able to achieve impressive results on the SemEval and FewRel dataset.

Although the context is omitted in the *entities* setting, a macro-F1 score of 61.18 is reached. While this is inferior to predicting the relation based on the whole sentence and using entity indicators (74.32 macro-F1), it outperforms the GPT without entity markers (52.15 in Table 4.3). This confirms the assumption that it is possible in many cases to predict the correct relation based on the entities alone, especially when the relation types are very specific like those in FewRel. Still, maintaining the context produces far better results but to indicate the target entities is important in this case due to the additional noise.

The *from-scratch* setting as well as the *BPE only* settings are conducted with the GPT+EI model. When trained with the default hyperparameters, the macro-F1 amounts to 64.36 in the *from-scratch* setting. Training the model for a longer duration (12 epochs[12]) improves the performance by 1.93 points to 66.29 macro-F1.

[12]Again manually tuned. More epochs lead to stagnating results. Increasing the learning rate instead lead to inferior results.

As expected, the model performs substantially worse when it is trained from-scratch on the relation classification task (66.29 to 74.32 macro-F1). This is in line with other findings on transfer learning and is also the reason why regular fixed word embeddings are dominantly employed in the recent years: Even a fairly large dataset like FewRel cannot fully capture the semantics of the large variety of words in human language. Since many words, especially named entities, differ between the train and test dataset, a pre-training on large corpora proves to be important for generalization. When the pre-trained BPE token embeddings are not omitted (*BPE only*), the model outperforms the *from-scratch* setting by about 4.72 points when also trained for 12 epochs (66.29 to 71.01). Still, transferring the whole model (74.32 macro-F1) yields superior results compared to only retaining the embedding matrix. This confirms that the pre-trained ability of the GP-Transformer to relate words based on the specific input context is important for relation classification.

4.4 Conclusions

In this chapter, the GP-Transformer was fine-tuned on relation classification. By training the model for only a few epochs, it achieves competitive performance compared to other state-of-the-art models. Taking the entities into account was shown to be critical for the model's performance, especially in the FewRel dataset. Here adding special entity indicators before and after the two entities works best. Post-indicating the entities was also shown to obtain good classification results. Furthermore, fine-tuning the GP-Transformer proved to be essential for relation classification, confirming that the self-attention mechanism is very well suited for this task.

However, not fine-tuning the complex GP-Transformer reduces training duration and makes the pre-trained Transformer embeddings more universally applicable. Findings in other deep neural networks suggest (e.g. in image classification, as shown in Junkert et al. 2017) that pre-trained model's tend to overadapt to the pre-training objective, in this case language modeling, in upper layers. An interesting direction for future work on this topic could be to validate this assumption for the GP-Transformer and use the context-aware embeddings of inner layers (or a combination of different layers as the ELMo model by Peters et al. 2018) when the model is not fine-tuned on the target task.

Chapter 5

Few-Shot Relation Classification

Machine learning algorithms, and in particular conventional relation extraction, usually require a huge amount of labeled data in order to identify the recurring patterns that define a specific problem. In practice, however, such labeled data is often scarce and can only be acquired at great expense: When dealing with a multi-way classification problem, samples must be annotated for every class (here, a relation) the system in question is designed to detect. So whenever a new class is added to the system, the costly process of data annotation starts all over again. In relation classification, classes can differ largely from domain to domain and can take very specific forms dependent on the target problem. For example, consider reports written by a service technician describing problems with "Machine X3Y":

Relation	**Sentence**
Liquid Leakage	Today, **Machine X3Y** leaked a concerning amount of **oil**.
Loose Part	The **lever** of **Machine X3Y** almost fell off when pulled.

Humans are known to be able to learn a new concept from only a few or even a single sample. So when the service technician describes a novel type of encountered problem, like "Noise", you would instantly be able to tell it apart from "Liquid Leakage" or "Loose Part" independent of it being worded as "The [red button]$_{tail}$ of [Machine X3Y]$_{head}$ makes a strange noise when pressed" or "The [red button]$_{tail}$ of [Machine X3Y]$_{head}$ squeaks when it is pushed down". Neural networks on the other hand are built to alter their internal weights only slightly when a new sample is encountered to avoid overfitting to certain features. So when only a single "Noise" relation instance is presented to the model it would gradually overfit to the sample with each iteration or a sufficiently high enough learning rate.

The task of *few-shot relation classification* is especially designed to solve the problem of data scarceness of never before encountered relations. Analogous to standard relation classification, the model is trained on a set of labeled relation samples. Here the quantity of train relations and corresponding samples can be large dependent on the dataset. The challenge is then to adapt the model to new relations based on a limited amount of examples per relation: Given a set of C relations (*ways*), which the model was not trained on, predict the correct relation based on only N examples (*shots*) per relation. The set of relation examples is commonly referred to as the *support set*, while the sample that the system must classify based on the support set is denoted as a *query*. Usually, N is very small, e.g. five or even a single example per relation (as visualized in Figure 5.1). The few-shot methodology has only recently been applied to relation classification, but many attempts to adapt a network trained on many objects to a few samples of new objects have already been made in other domains or modalities, especially in image classification (e.g. Vinyals et al. 2016, Snell, Swersky, and Zemel 2017, Finn, Abbeel, and Levine 2017).

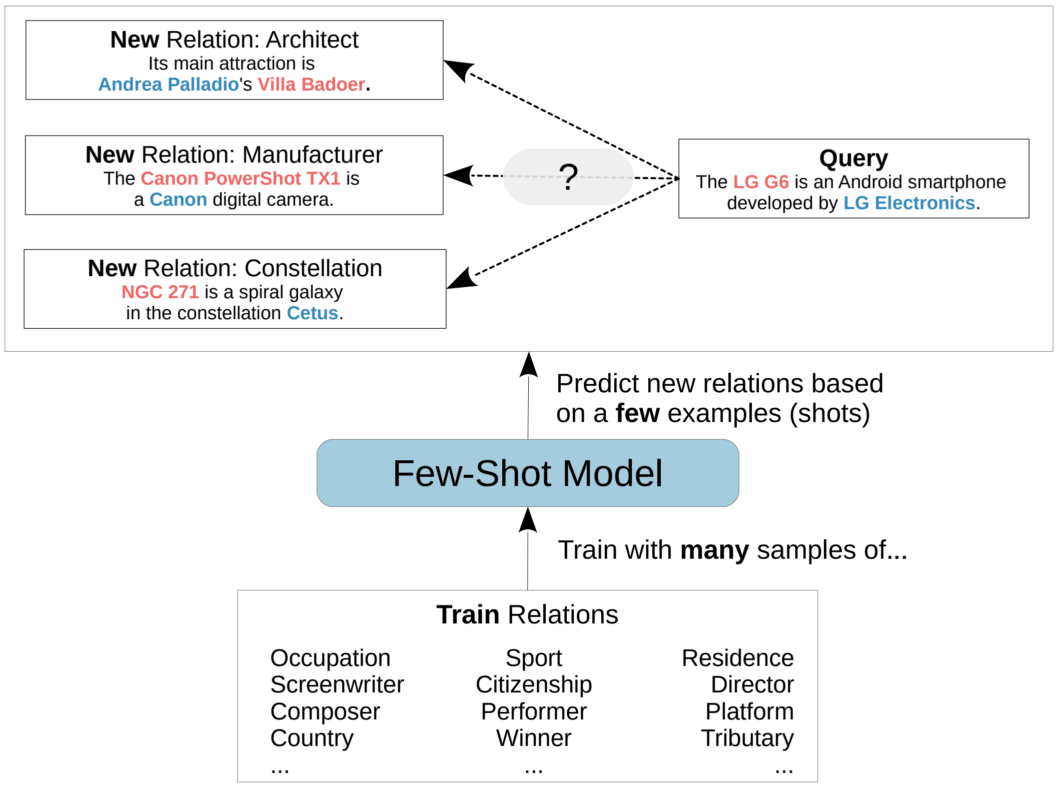

New Relation: Architect
Its main attraction is Andrea Palladio's Villa Badoer.
New Relation: Manufacturer
The Canon PowerShot TX1 is a Canon digital camera.
New Relation: Constellation
NGC 271 is a spiral galaxy in the constellation Cetus.
?
Query
The LG G6 is an Android smartphone developed by LG Electronics.
Predict new relations based on a few examples (shots)
Few-Shot Model
Train with many samples of...
Train Relations
Occupation
Screenwriter
Composer
Country
...
Sport
Citizenship
Performer
Winner
...
Residence
Director
Platform
Tributary
...

Figure 5.1: Few-shot classification: The model is trained on a dataset containing a large variety of different classes (here relations) and samples. During evaluation, the model must decide which one of unseen classes (relations), which the model was not trained on, is expressed in the query sentence, based on only a *few* examples per new class. Depicted is a 3-way (three new classes) 1-shot setting (one example per class).

This chapter builds upon results from Chapter 4, indicating that the GP-Transformer is a good option for relation classification. The prototypical network (Snell, Swersky, and Zemel 2017), which achieved state-of-the-art results in various few-shot tasks, is combined with the GP-Transformer for few-shot relation classification. Prototypical Networks create a comprehensive representation of each class (called *prototype*) in order to match it with the query. Here the GP-Transformer acts as an elaborate sentence encoder and yields vector representations of single sentences. These are later used to form the relation's prototype by a simple averaging over the support instances, which is described in detail in Section 5.2. In experiments on the FewRel dataset (discussed in Section 3.3), which is especially designed for few-shot relation classification, the following questions are addressed:

- How does the GP-Transformer compare to the model proposed by Han et al. (2018a), who employ a CNN with word embeddings as a sentence encoder in a prototypical network?
- Language modeling pre-training was found to be important in Chapter 4. Does the language model pre-training of the GP-Transformer has a greater impact on a low resource task like few-shot relation classification compared to standard relation classification?
- The few-shot scenario is normally evaluated on classes (here, relations) that the model did not observe during training. In practice however, new classes would most likely be added to the existing set of training classes, requiring the model to decide between training classes and the newly added classes. How does the model perform when a new class is matched with existing training classes?

5.1 Related Work

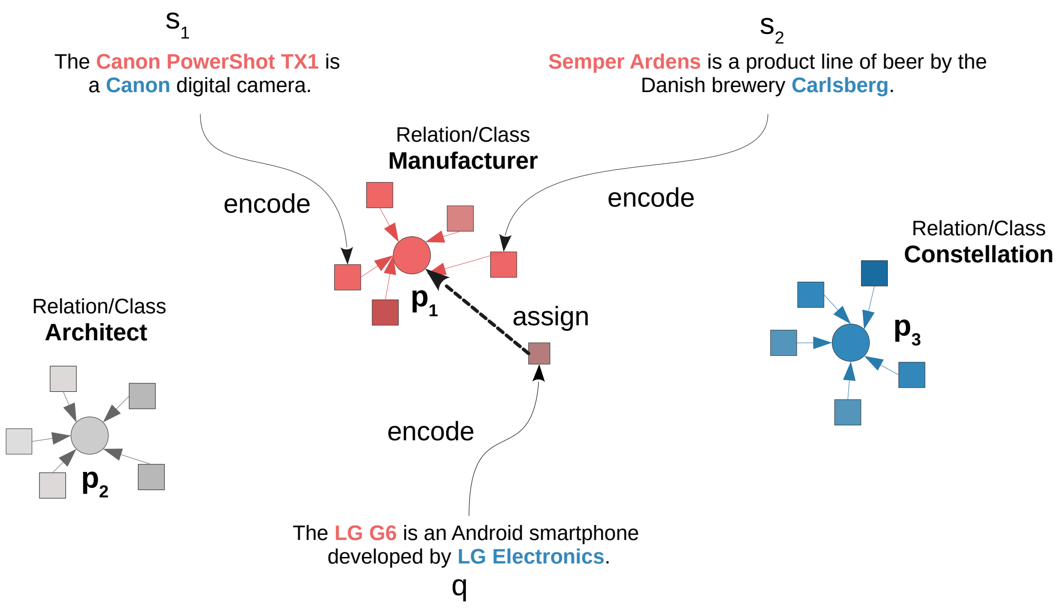
Since a variety of different architectures were proposed for few-shot classification in the recent years, especially in image classification, this section only contains a brief overview of approaches that were already applied to the relation classification domain.

Han et al. (2018a) evaluated several baseline methods for few-shot classification on the FewRel dataset, namely meta networks (Munkhdalai and Yu 2017), simple neural attentive learner (SNAIL, Mishra et al. 2017), graph neural networks (GNN, Garcia and Bruna 2017) and prototypical networks (Snell, Swersky, and Zemel 2017). Meta networks divide weights into fast and slow changing weights: While slow weights are trained as usual by minimizing the classification loss, fast weights are generated by a separate meta learner to allow for rapid generalization across individual samples. SNAIL concatenates all support instances for a given class into a sequence, followed by the query instance. It combines temporal convolutions with causal attention in order to learn from past experiences. Graph neural networks organize all encoded support and query instances in a fully connected graph. The labels of the support set (i.e. the specific relation type) are also embedded into the corresponding node. The model is then trained to learn the weights between the graph nodes in order to correctly match a query with the support instances. Prototypical networks on the other hand (described in detail in Section 5.2) learn to create comprehensive representations, called prototypes, from the support samples and match these with the query instance. Han et al. (2018a) employ the aforementioned few-shot algorithms to classify relations of the FewRel dataset. In each model, they use the output of a regular convolutional neural network as the encoding for a given sample. They also employ word embeddings and the entity distance embeddings, which are described in section 4.2. In addition to the elaborate few-shot models described above, Han et al. evaluate a simple k-NN approach and a fine-tuning baseline.

Gao et al. (2019) also employ a prototypical network with a CNN encoder, but extend the prototype creation with an attention mechanism. In contrast to conducting a simple averaging of the support instances to create the corresponding prototype, as in the original prototype network paper, Gao et al. propose an *instance-level* and *feature-level attention* over the support instances. Because prototypes are created based on only a few samples, they are prone to outliers and noise. The instance-level attention is designed to reduce the impact of noise by maintaining only those support instances which are similar to the query, i.e. by weighting the support instances with regards to their distance to the query instance. The feature-level attention on the other hand learns to weight the feature dimensions of the prototype representation differently when it is matched with the query encoding in order to retain only the most informative features.

5.2 Approach

In this work, a prototypical network is employed in conjunction with a pre-trained GP-Transformer, which functions as an elaborate sentence encoder. This model is also denoted as *prototypical GP-Transformer* from now on. Prototypical networks are specifically designed for a few-shot scenario and were proposed by Snell, Swersky, and Zemel (2017). Despite their simplicity, they achieve state-of-the-art results in a number of different tasks. Moreover, prototypical networks can be combined with



s1
The Canon PowerShot TX1 is a Canon digital camera.
s2
Semper Ardens is a product line of beer by the Danish brewery Carlsberg.
Relation/Class
Manufacturer
encode
encode
p1
assign
Relation/Class
Constellation
p3
Relation/Class
Architect
p2
encode
The LG G6 is an Android smartphone developed by LG Electronics.
q

Figure 5.2: In a prototypical network, support samples of a class (s_1, s_2) are first encoded by an instance encoder into a feature vector. For each class (three relations in the example), a prototype is created by simply averaging the corresponding feature vectors. A query sample q is also transformed into the same feature space by the instance encoder and assigned to its nearest prototype based on the euclidean distance.

any arbitrary model, which serves as an instance encoder and transforms a given sample into a d-dimensional feature vector h, without requiring any changes to the model's architecture. In the first part of this section, the model's architecture is described. The second part explains the training procedure of the model, which is based on *episodes* of samples that mimic the few-shot task's objective.

Architecture

As explained in Section 5, a support set of N labeled samples $\mathcal{S}_r = \{s_1, ..., s_N\}$ is given for each (relation) class $r \in \mathcal{R}$ in a few-shot setting. In few-shot relation classification, each of these samples correspond to sequences of byte pair encoded tokens $s_i = (t_1, ..., t_l)$ (with l again denoting the sentence's length), which express a certain relationship between two entities. The number of classes C, which the model is required to predict, is also referred to as the *way*. For example, in a 10-way 5-shot ($C = 10, N = 5$) scenario, the task is to assign 1 out of 10 classes (all unseen during training) to a query instance q, based on only 5 labeled samples per class. In order to do so, each sample s_i is first encoded by an instance encoder f_θ into a feature vector $\mathbf{m}_i \in \mathbb{R}^d$ (here d denotes the dimensionality of $\mathbf{m}_i$):

$$\mathbf{m}_i = f_\theta(s_i) \tag{5.1}$$

where θ denotes the set of learnable parameters of the instance encoder, in this work the GP-Transformer model. In order to match the support instances of each class with a query, the prototypical network first transforms these instances into a prototype $\mathbf{p}_r \in \mathbb{R}^d$, which represents the corresponding relation type. This is done by simply averaging the encoded support instances that belong to a class r:

$$\mathbf{p}_r = \frac{1}{N} \sum_{s_i \in \mathcal{S}_r} f_\theta(s_i) \tag{5.2}$$

Next, the query instance q is encoded in the same way as the support instances by the instance encoder f_θ:

$$\mathbf{m}_q = f_\theta(q) \tag{5.3}$$

In order to compare the query with a prototype, the euclidean distance between both feature vectors, $\mathbf{p}_r$ and $\mathbf{m}_q$, is computed by:

$$d(\mathbf{p}_r, \mathbf{m}_q) = \sqrt{(\mathbf{p}_r - \mathbf{m}_q) \cdot (\mathbf{p}_r - \mathbf{m}_q)} \tag{5.4}$$

To obtain a probability distribution over all classes, a softmax of the distances of the query to each prototype is computed next. Since the closest prototype must have the highest probability after the softmax application, the negative euclidean distance is employed:

$$P(y = r \mid q, f_\theta) = \frac{e^{-d(p_r, m_q)}}{\sum_{r' \in \mathcal{R}} e^{-d(p_{r'}, m_q)}} \tag{5.5}$$

Because the GP-Transformer with entity indicators yielded the best results in relation classification (see Chapter 4), the same model is applied as the instance encoder in the few-shot setting. Just like in relation classification, the Transformer embedding corresponding to the special <*End*> token, $\mathbf{h}_{End}$, is used as the feature vector $\mathbf{m}_i$ of a sample s_i. The whole procedure of support and query instance encoding, prototype computation and query assignment is also illustrated in Figure 5.3.

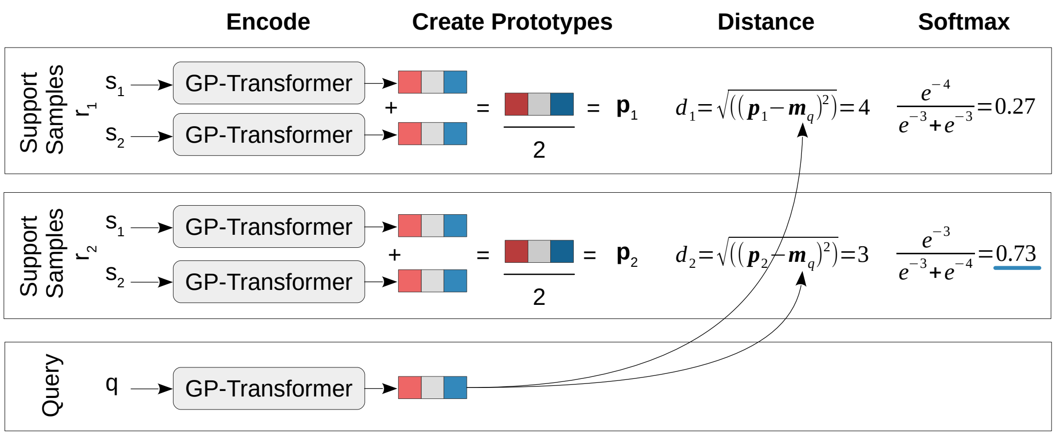


Figure 5.3: Prototypical Network, which uses the GP-Transformer as an instance encoder. Illustrated is a 2-way 2-shot setting. First, the support instances of both relations as well as the query instance are transformed into a feature vector by extracting the <*End*> token of the last Transformer block (see Section 4.2). Next, the prototypes $\mathbf{p}_1$, $\mathbf{p}_2$, which represent relations r_1 and r_2 respectively, are created by averaging the feature vectors of the corresponding support samples. The euclidean distance between the query feature vector and the prototypes is computed. A final softmax application over the negative distances (representing the similarity between the feature vectors) yields a probability distribution over the two classes. In the example, the query instance is classified as r_2, because it is the most likely classed based on the softmax probability.

Training

Just as in standard relation classification (see Section 4.2), the model is trained by minimizing the negative log-likelihood of the ground truth relation r based on a large set of training relations $\mathcal{R}_{\text{train}}$ and corresponding samples:

$$\text{Loss} = -log\ P(y = r \mid q, f_\theta) + \alpha \cdot \text{Loss}_{\text{LM}} \tag{5.6}$$

As in standard relation classification, samples of a batch are padded to the same length and language modeling is employed as an auxiliary objective in addition to the classification objective. However, in order to mimic the few-shot objective (predict the correct relation based on a few examples per relation) the model is trained on batches of episodes. Each episode is composed of C_{train} relation types and N_{train} support samples per relation type. Here C_{train} and N_{train} can mimic the evaluation settings, i.e. $C_{train} = C, N_{train} = N$, but also be set to other values. To speed up computation, M query instances per each of the C_{train} relations are assigned simultaneously in a single episode. Batches of episodes are created randomly in each of I iterations. During training, the negative log-likelihood loss is summed over all batches and every query instance within a batch. The training procedure for a single episode per iteration (batch size = 1) is depicted in Algorithm 2. Implementation

Algorithm 2 Training procedure: $\mathrm{C}_{\mathrm{train}}$ refers to the amount of training relations per episode and $\mathrm{N}_{\mathrm{train}}$ to the amount of support samples per training relation in the episode. $\mathcal{T}$ denotes the full training set, which contains k (e.g. $k = 1000$) samples per training relation $r \in \mathcal{R}_{\mathrm{train}}$ (e.g. $\mathcal{R}_{\mathrm{train}} = \{$occupation, composer, country, sport, winner, residence, ...$\}$). $\mathcal{T}_r$ denotes the set of training samples of relation r. Sample returns random data points without replacement. M denotes the number of queries per relation and I the number of training iterations.

```
1:  procedure CreateEpisode                                      ▷ Sampling of episodes per iteration
2:      $\mathcal{S} \leftarrow \{\}$
3:      $\mathcal{Q} \leftarrow \{\}$
4:      $\mathcal{R}_{\mathrm{episode}} \leftarrow \mathrm{Sample}(\mathcal{R}_{\mathrm{train}}, \mathrm{C}_{\mathrm{train}})$            ▷ Sample $\mathrm{C}_{\mathrm{train}}$ unique relations
5:      for $r \in \mathcal{R}_{\mathrm{episode}}$ do
6:          $\mathcal{S}_r \leftarrow \mathrm{Sample}(\mathcal{T}_r, \mathrm{N}_{\mathrm{train}})$              ▷ Sample $\mathrm{N}_{\mathrm{train}}$ unique support samples
7:          $\mathcal{Q}_r \leftarrow \mathrm{Sample}(\mathcal{T}_r \setminus \mathcal{S}_r, M)$                ▷ Sample $M$ unique query samples
8:          $\mathcal{S}.Add(\mathcal{S}_r)$
9:          $\mathcal{Q}.Add(\mathcal{Q}_r)$
10:     return $\mathcal{S}, \mathcal{Q}$
11:
12: procedure Train                                                  ▷ Training Routine
13:     while $i < I$ do
14:         Loss $\leftarrow 0$
15:         $\mathcal{S}, \mathcal{Q} \leftarrow$ CreateEpisode()
16:         for $\mathcal{S}_r \in \mathcal{S}$ do                                  ▷ Encode support sets
17:             $p_r \leftarrow \mathrm{CreatePrototype}(\mathcal{S}_r, f_\theta)$                      ▷ See Equation 5.2
18:         for $\mathcal{Q}_r \in \mathcal{S}$ do                                    ▷ Assign queries
19:             for $q \in \mathcal{Q}_r$ do
20:                 Loss $\leftarrow$ Loss $- \, logp(y = r \mid q, f_\theta) + \alpha \cdot \mathrm{Loss}_{\mathrm{LM}}$    ▷ See Equation 5.6
21:         SGD($f_\theta$, Loss)     ▷ Update the instance encoder's (GP-Transformer's) weights
```

wise, the GP-Transformer is only run once in each iteration by stacking the support and query samples into a single tensor, which is encoded by the GP-Transformer.

5.3 Experiments

The purpose of the following experiments is to evaluate if the GP-Transformer is suitable as the feature extractor in a prototypical network and how this implementation compares to state-of-the-art model's. The impact of language modeling pre-training is also assessed in this section for few-shot relation classification.

Setup

In order to compare the results of the prototypical GP-Transformer to the baseline models of the original FewRel paper (Han et al. 2018a), the model was trained and evaluated on the same few-shot scenarios, namely: 5-way 5-shot, 5-way 1-shot, 10-way 5-shot and 10-way 1-shot. As explained before (see Section 3.3), the set of training- (64 classes) and validation-relations (16 classes) are disjoint in the FewRel dataset, so the model is required to generalize to unseen relations. The baseline models by Han et al. were only evaluated on the hidden test set, thus no scores on the validation set were reported. To be able to make a fair comparison to the baseline models during development, the best performing baseline model (prototypical CNN) was re-trained and evaluated on the validation set with the FewRel framework, which is available on GitHub by Han et al. (2018b).

The hyperparameters of the GP-Transformer were set according to those outlined in Section 4.3 for the standard classification task. Again the English-language pre-trained version by Radford et al. (2018) is employed. Since the prototypes are computed by averaging the <*End*> token embedding vectors of each support sample after the last Transformer block, the prototypes are also 768-dimensional. As explained in Section 5.2, the prototypical GP-Transformer is trained on episodes of randomly selected relations and samples of the training set. The training set contains 64 relations and 700 samples per relation (see Section 3.3). Snell, Swersky, and Zemel (2017) report that setting the number of training relations per episode (C_{train}) to a higher value than during evaluation is beneficial for few-shot learning, so $C_{train} > C$. The number of shots N_{train} should be matched with those of the respective evaluation setting, so $N_{train} = N$. In light of this advice, the number of ways N_{train} was set to 20 during training for each of the four scenarios. Meanwhile, the number of shots mimic those encountered in the specific evaluation scenario during training. The number of queries M per relation was set to 5, while the batch size was set to 1 due to memory restrictions. Note that the loss still depends on multiple samples in a single episode, even if the number of episodes per batch is set to 1. For example, in a 5-way 5-shot setting the episode's loss is computed based on 25 query samples (5 per relation). 5000 training iterations were found to be sufficient for every few-shot setting.

Because Han et al. reported no hyperparameters for the prototypical CNN, the default values of the FewRel framework were used: A train way of 20, learning rate of 0.1, weight decay of 10^{-5}, hidden size of 230 and 30.000 train iterations. The CNN instance encoder uses 50-dimensional pre-trained GloVe word embeddings and 5-dimensional distance embeddings.

Comparison with State-of-the-Art

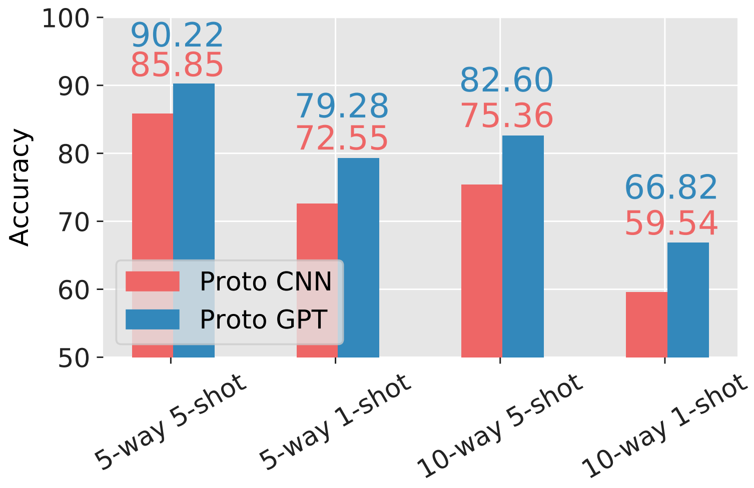


Figure 5.4: The prototypical GP-Transformer (blue bars) and the baseline prototypical CNN (red bars - Han et al. 2018a) were both evaluated on the official validation set. The graph shows the averaged results of 3 runs, each with 3000 random episodes and 5 queries per relation.

Figure 5.4 displays the results that were obtained by the baseline model (red) and the prototypical GP-Transformer (blue) for each of the four few-shot scenarios. To mimic the practice of the FewRel framework, both models were trained 3 times and evaluated on 3000 random episodes with 5 queries per relation (e.g. 75000 query samples per run in the 5-way settings). The averaged accuracy on the validation set over the 3 training runs is reported for each setting. As visible in the figure, the GP-Transformer outperforms the baseline model in every setting by a large margin. The smallest difference with about 4.37% accuracy was obtained in the 5-way 5-shot scenario, which is the easiest of the four, while the GP-Transformer outperforms the baseline model in the 5-way 1-shot setting by 6.73% accuracy. The difference is even higher in the 10-way settings: 7.24% accuracy in the 10-way 5-shot setting and 7.28% accuracy in the 10-way 1-shot setting. The training loss curve of the 10-way 1-shot setting and validation accuracy over iterations is included in Appendix C.3.

As to acquire the final results on the hidden test set, the best performing prototypical GP-Transformer was also submitted via CodaLab[1]. CodaLab allows the creation of so-called worksheets, which are built to run reproducible experiments: Since the bundles (e.g. python files or model checkpoints) are immutable, they can later be used to reproduce an experiment. The code that is required to evaluate the prototypical GP-Transformer was uploaded to CodaLab and the final bundles were then sent to the authors of the FewRel dataset, who executed the code on the hidden test set. The test set comprises of 20 relation types which do not occur in the training or validation set (see Section 3.3). The evaluation results of the prototypical GP-Transformer as well as the baseline models and the hybrid attention CNN by Gao et al. (2019) are listed in Table 5.1. Since Gao et al. developed the hybrid attention for multi-shot scenarios, no results for the 1-shot scenarios are reported. They also report results of an own implementation of a prototypical CNN without attention (Proto (CNN)* in Table 5.1), which performs better than the baseline prototypical CNN by Han et al. As visible in the table, a simple fine-tuning of a CNN or PCNN and a k-nearest-neighbor matching yield inferior results to the more elaborate few-shot models.

Model	**5-way 5-shot**	**5-way 1-shot**	**10-way 5-shot**	**10-way 1-shot**
Human	-	92.22	-	85.88
Proto (GPT + EI)	**92.11 ± 0.10**	**81.40 ± 0.21**	**86.03 ± 0.17**	**72.51 ± 0.28**
Proto-Attn (CNN)[a]	90.12 ± 0.04	-	83.05 ± 0.05	-
Proto (CNN)*[a]	89.05 ± 0.16	-	81.46 ± 0.13	-
Proto (CNN)[b]	84.79 ± 0.16	69.20 ± 0.20	75.55 ± 0.19	56.44 ± 0.22
SNAIL (CNN)[b]	79.40 ± 0.22	67.29 ± 0.26	68.33 ± 0.259	53.28 ± 0.27
GNN (CNN)[b]	81.28 ± 0.62	66.23 ± 0.75	64.02 ± 0.77	46.27 ± 0.80
Meta-Net (CNN)[b]	80.57 ± 0.48	64.46 ± 0.54	69.23 ± 0.52	53.96 ± 0.56
kNN (PCNN)[b]	72.41 ± 0.39	60.28 ± 0.43	59.11 ± 0.30	46.15 ± 0.31
kNN (CNN)[b]	68.77 ± 0.41	54.67 ± 0.44	55.87 ± 0.31	41.24 ± 0.31
Finetune (CNN)[b]	68.66 ± 0.41	44.21 ± 0.44	55.04 ± 0.31	27.30 ± 0.28
Finetune (PCNN)[b]	57.86 ± 0.61	45.64 ± 0.62	37.43 ± 0.42	29.65 ± 0.40

Table 5.1: Performances on the hidden FewRel test set. Included are the baseline models by Han et al. (2018a) and the hybrid-attention model, Proto-Attn (CNN), by Gao et al. (2019). Proto (GPT + EI) denotes the model employed in this work.
Source (retrieved march 2019)[c]: `http://www.zhuhao.me/fewrel/`

[a]Reported in Gao et al. (2019)

[b]Reported in Han et al. (2018a).

[c]The leaderboard also contains two better performing models. However, since these models are unlabeled and therefore unknown, a fair comparison cannot be made and they are not reported in this table.

Out of the four baseline models Han et al. 2018a have evaluated on the FewRel dataset, the prototypical CNN performed best. In the 5-way 5-shot and 10-way

[1]`http://codalab.org/`

5-shot setting, the hybrid-attention model outperforms the prototypical baseline CNN by 5.33% and 7.5%, respectively. These results are again outperformed by the prototypical GP-Transformer by 1.99% and 2.98% accuracy. In the 5-way 1-shot and 10-way 1-shot settings, the difference to the prototypical baseline CNN is even as big as 12.20% and 16.07%. Note that the prototypical GP-Transformer was only trained on the official training set: By retraining it on a combination of the training and the validation set an even higher performance may be reached. Adding the hybrid attention mechanism, which was proposed by Gao et al., might also improve the performance, but was not explored in this work.

Ablation Studies

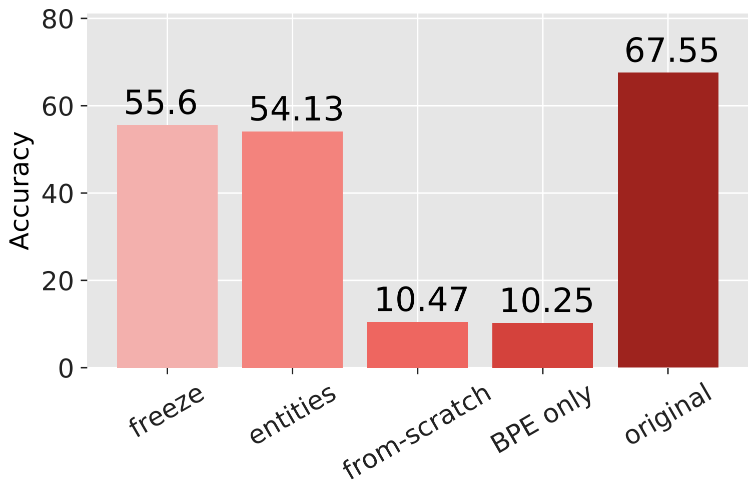
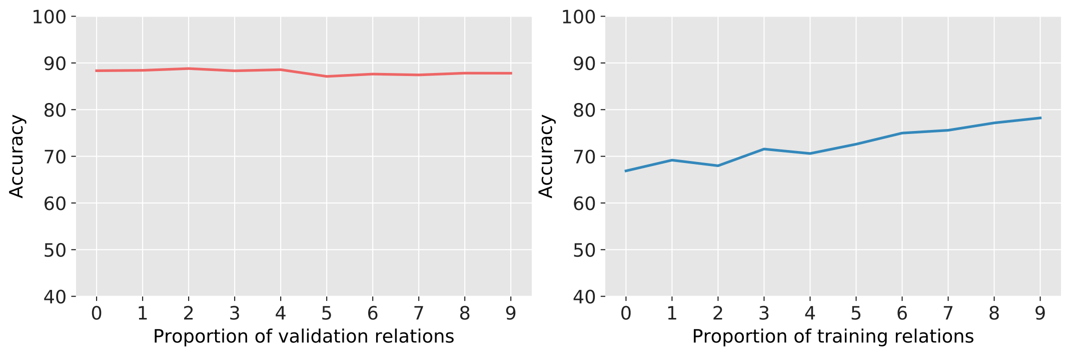


Figure 5.5: Ablation studies were conducted on four different settings in the 10-way 1-shot scenario: **(freeze)** The GP-Transformer is not fine-tuned on relation classification. **(entities)** Predict correct relation solely with regards to the participating entities. **(from-scratch)** Omit weights and BPE embeddings of the pre-trained GP-Transformer and train it from-scratch. **(BPE only)** Only omit GP-Transformer weights but retain BPE embeddings.

In order to validate if the model behaves differently in the low-resource few-shot task, the same ablation studies as for relation classification, which were described in Section4.3, were conducted. Since the ablation studies are only run once, each setting was evaluated on the same 10k samples in a 10-way 1-shot scenario. The prototypical GP-Transformer (Proto GPT + EI) is also re-evaluated on these samples (denoted as *original* in Figure 5.5). With the exception of the *freeze* setting, the prototypical GP-Transformer with entity indicators was also employed for the ablation studies. In the *freeze* setting, the *Segment-Avg* entity indication method was employed (see Section 4.2), instead of using $\mathbf{h}_{\text{<End>}}$ as the feature vector. An additional linear fully-connected layer was also added in this setting, which maps the *Segment-Avg* encoding to size $d = 768$ [2]. As visible in Figure 5.5 and similar to the observations made for standard relation classification, the model performs inferior when it is not fine-tuned (*freeze* setting) on the target task (55.6% to 67.55%). Again, a more elaborate downstream model like a LSTM might be able to achieve a better performance. When only the two entities of a sentence serve as the input of the model and the surrounding context is omitted, the model's performance decreases by about 13.42%. While the two entities already contain useful information for few-shot classification, the surrounding context is therefore also of high relevance. Surprisingly, and in contrast to the results reported for standard relation classification (Section4.3), the model performs far worse when trained from-scratch or with omitted GP-Transformer weights and only reaches an accuracy that would also be obtained by random-guessing a relation. In this case, better results may be obtained by further adjusting the learning rate or other hyperparameters, which was not explored in this work.

Influence of Training Relations



(a) Training **(b)** Validation

Figure 5.6: To assess the capability of the model to discriminate between observed (training) and unseen (validation) relations, it has been evaluated on two different settings: In the *training* setting (left), a sample of one of the training relations serves as the query, while the ratio of unseen validation relations in the set of the 10 target relations switches from 0 (10 training relations, 0 validation) to 9 (1 training relation, 9 validation relations). Similarly, in the *validation* setting, an unseen validation sample serves as the query, and the ratio of training relations is adjusted.

In the few-shot setting, the neural network must learn to classify completely unseen relations based on only a few (or even a single) examples per relation type. To check how the performance is effected when the model must decide between new relations and already observed relations (the training relations) and if the model prefers relations it was trained on, an additional experiment was conducted. This experiment is visualized in Figure 5.6. Two settings were evaluated on the 10-way

[2] In standard relation classification a higher learning rate improved the results in the *freeze* setting (Section 4.2). For few-shot relation classification, increasing the learning rate was not observed to have a positive impact.

1-shot scenario: In the *training* setting (red line), a sample from the training relations serves as the query, while the ratio of unseen validation relations in the set of the 10 target relations switches from 0 (10 training relations, 0 validation relations) to 9 (1 training relation, 9 validation relations). In the *validation* setting (blue line), a sample from the validation set (unseen relations) serves as the query, while the ratio of training relations in the set of the 10 target relations switches from 0 to 9. Both settings were evaluated on 5000 random 10-way 1-shot episodes, with one query sample per episode. As visible in Figure 5.6, the model performs incrementally better when the ratio of training relations is increased in the *validation* setting. This shows, that the prototypical GP-Transformer learned to create good embeddings (prototypes) of the training relations, which makes the distinction to unseen relations easier. In the *training* setting on the other hand, the model's performance declines only sightly when unseen relations are added to the mix.

5.4 Conclusions

In this chapter, the pre-trained GP-Transformer was employed in a few-shot relation classification setting. Here the model is used to encode sentences, which then form a prototypical representation of the corresponding relation. In experiments on four settings (5-way 5-shot, 5-way 1-shot, 10-way 5-shot, 10-way 1-shot), the model outperforms the baseline, which uses a CNN in conjunction with a prototypical network, by a maximum of 16% accuracy on the test dataset (10-way 1-shot). Additionally, the prototypical GP-Transformer outperforms the sophisticated hybrid-attention model by Gao et al. (2019). The prototypical GP-Transformer employed in this work is especially dominant in the 1-shot settings, indicating that language modeling pre-training is important for generalizing from only a few samples. In addition to this and in conjunction with observations of Chapter 4, the attention mechanism was confirmed to be a good fit for (few-shot) relation classification: To form a prototype, the model can specifically attend to those features that are most discriminative for the corresponding relation.

Albeit the prototypical GP-Transformer obtains a fairly high accuracy when provided with a single sample per relation (1-shot settings), the model must decide between just a few classes (10 at maximum). Here the model's performance already decreases by more than 6% accuracy when the number of relations is doubled (5-way to 10-way). In practice, the number of relations between the model is required to decide may even grow higher than this, depending on the domain. Further research could therefore concentrate on better separating the relation prototype's and be less prone to a higher relation count.

Chapter 6

Joint Entity and Relation Extraction

The task of relation extraction is composed of several subtasks and often tackled by employing different models, each solving a single one of these subtasks, in a pipeline. This includes the extraction of named entities by NER taggers, the decision if a relation between an entity pair is expressed in the sentence and ends with the actual classification of this entity pair into a certain relation. However, common NER taggers like the *Stanford Named Entity Recognizer*[1] and spaCy's *Named Entity Recognition* are only able to detect a limited amount of entity types. Moreover, since these models are trained on specific samples, they usually must be updated when employed in a new domain which contains unknown entities. In practice, training multiple distinct models for relation extraction is expensive, time consuming and hard to maintain. Ideally, a single model can be trained on the available data, which is both able to detect probable entities and their boundaries as well as the relations that hold between them. In contrast to the sole classification of relations, as explored in Section 4, the task of joint entity and relation extraction (also denoted as *end-to-end relation extraction* in this work) is to predict correct relation triples (*head*, *relation*, *tail*) with a joint model, given only the input sentence. In this case, entities and relations are correlated: To detect a relation, the target entities must be known. On the other hand, knowing the target relation may guide the model towards certain entities.

In this work, relations between entities are extracted by jointly training a single model end-to-end to detect entities and the relations that persist between them. Like in Chapter 4 and Chapter 5 the GP-Transformer acts as an elaborate feature extractor, which yields context-aware embeddings per byte pair encoded token of a sentence. Entities and relations are then detected by a fast exhaustive search over all plausible

[1]`https://nlp.stanford.edu/software/CRF-NER.html`

entity candidate pairs. Because each sentence is only fed *once* through the entire GP-Transformer to obtain the context-aware embeddings, the fast exhaustive search is feasible even on a large search space of entity candidate pairs. Here an entity candidate is solely required to consist of arbitrary adjacent tokens, as illustrated in Figure 6.1. The presented approach is inspired by methods in object detection, which were shown to efficiently classify object proposals: The *Fast R-CNN* (Girshick 2015) feds an image once through a convolution neural network to generate a feature map, which is then used to classify a set of region proposals. As in this work, an exhaustive search over all potential region (here, entity) candidates is conducted. By doing that, Girshick demonstrates that the model achieves a high accuracy in object detection while still requiring a reasonable amount of training and prediction time.

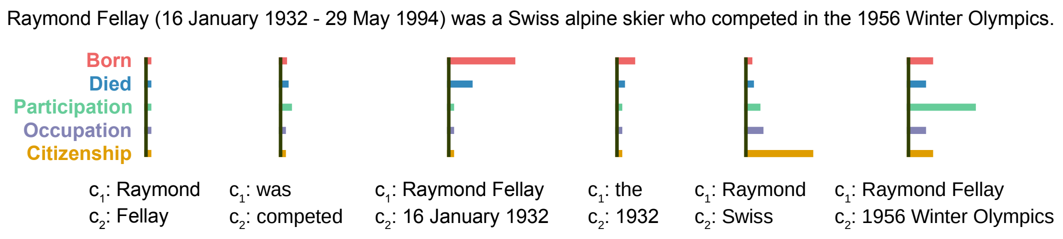


Figure 6.1: Joint entity and relation extraction aims at training a single model end-to-end to extract all entity pairs and the relations between them from a given sentence. In contrast to the standard relation classification objective, which was evaluated in Section 4, the entity boundaries are not known in advance and must also be detected in conjunction with the relation for each entity pair. In this work, a fast exhaustive search over all entity candidate pairs is performed by computing the probability that each relation is expressed in the sentence between the candidate pair.

Since many relations can exist between different entities in a single sentence, the model which is employed in this work scores each entity candidate pair independently (as visualized in Figure 6.1). To facilitate a fast exhaustive search over all candidate pairs, several adjustments have been made to the model compared to the one employed for relation classification. In experiments on both the SemEval and FewRel dataset the following questions are particularly addressed:

- Since a singe sample should only run once through the GP-Transformer in order to extract important features, how to best indicate the target entities?
- How well does the fast exhaustive search perform on joint entity and relation extraction?
- The amount of possible entity candidates can grow significantly with the sentence length. How to best optimize the fast exhaustive search to reduce evaluation speed and required resources?

- Because the model is required to assign a low score to entities that are not related in a sentence, how can the model be enabled to express that a candidate pair is not related?

6.1 Related Work

Various different models have been published for joint entity and relation extraction, especially in the recent years. This section focuses on competitive related approaches and differentiates them from the fast exhaustive search based approach proposed in this work.

Kate and Mooney (2010) arrange entities and relations as nodes in a pyramid-structured graph. The entities form the leaf nodes of the graph and are connected by nodes which express a potential relationship between the child entities[2]. Unlike in this work, Kate et al. assume the entity boundaries to be given in advance, e.g. by including all noun phrases in a preprocessing step. They then classify each node in the pyramid graph with two separate SVM's, one that classifies the leaf node entities and the other one to classify the relation at each inner node. They also include a "No Relation" class to express that an entity pair is not related in the given sentence.

Miwa and Sasaki (2014) tackle joint entity and relation extraction as a table-filling problem, where each cell of the table corresponds to a word pair of the sentence. The diagonal of the table is to be filled with the entity type of the word itself and the off-diagonal cells with the relations between the respective word pair. Named entities are extracted by tagging them according to the BILOU scheme: The BILOU (*Begin*, *Inside*, *Last*, *Outside*, *Unit*) scheme is commonly used for named entity recognition to predict the boundaries of entities. Here each token is assigned one of the BILOU tags dependent of it being the begin, last or any other token (*Inside*) of the entity. In case the entity only consists of a single token, it is assigned the *Unit* tag, while tokens that are not part of an entity are labeled with *Outside*. Miwa and Sakasi predict relations for the last word of the corresponding entity (*Last* or *Unit* tag according to BILOU). The table is filled with a label (BILOU + entity tag or relation type) for each cell by minimizing a scoring function which is based on local (e.g. POS tags or simple word types) and global features (e.g. combination of two entity labels, combination of two relation labels...). Miwa et al. then employ a beam search over the table cells to find an optimal table-filling solution.

[2]The pyramid-graph is similar to a binary tree, but some nodes have two parents instead of a single one.

Gupta, Schütze, and Andrassy (2016) also formulate joint entity and relation extraction as a table-filling problem. Unlike Miwa et al. they employ a bidirectional recurrent neural network to label each word pair.

The stacked model of Miwa and Bansal (2016) is also used for joint end-to-end relation extraction besides relation classification (as already mentioned in Chapter 4.1). In this case, the bidirectional sequential LSTM first tags the entities of the respective sample according to the BILOU scheme. Finally, the bidirectional tree-structured RNN operates on the dependency parse tree between an entity pair to predict their relation type.

P. Zhou, Zheng, et al. (2017) utilize a combination of a bidirectional LSTM and a CNN to extract a high level feature representation of the input sentence. A sigmoid layer is used to predict all relations that are expressed in the sentence. For each relation, a sequential LSTM then labels the entities according to the BILOU scheme. Since named entity extraction is only performed for the most likely relations, the approach by Zhou et al. needs to predict a lower number of labels compared to the table-filling approaches.

Zheng et al. (2017) first encode each word of the input sentence with a bidirectional LSTM. A sequential LSTM then operates on each encoded word representation and outputs the entity boundaries (akin to BILOU scheme) alongside their relation type and their position in the relation, i.e. if the entity is the head or tail of the relation. Entities that are tagged with the same relation type are then combined to obtain the final relation triples. Unlike the other approaches mentioned above and similar to this work, Zheng et al. do not utilize entity types for the joint extraction. However, conditions where one entity is related to multiple other entities are not considered, which limits the practical applications of their approach.

Finally, Bekoulis et al. (2018) also employ a bidirectional LSTM to encode each word of the sentence. They use character embeddings alongside Word2Vec embeddings to obtain the input word-level representations. Entity boundaries and tags are extracted with a conditional random field, which is able to account for neighboring tags to infer the most likely BIO[3] sequence. A sigmoid layer then outputs the probability that a specific relation is expressed between two words that belong to an entity with regards to the BIO scheme. In contrast to Zheng et al. (2017), Bekoulis et al. are also able to detect cases in which a single entity is related to multiple other entities with their approach.

In addition to the aforementioned joint models, the relation classification models by Verga, Strubell, and McCallum (2018) and H. Wang et al. (2019), which were

[3]BIO (Begin, Inside, Outside) is similar to the BILOU scheme but omits the additional Last and Unit tags.

already discussed in Section 4.1, are also related to the approach followed in this work: Both models feed a paragraph only once through an attention-based model and employ the resulting embeddings to classify a given set of labeled entity mentions of the paragraph. In contrast to this, the fast exhaustive search approach of this work does not rely on labeled entities and jointly detects both entities and their relations.

In comparison to other joint models, the exhaustive search approach presented in this work comes with the following fundamental benefit: The aforementioned models tag entities (e.g. by BIO/BILOU scheme) in a separate step before classifying the relation between them. However, these models are not able to correctly classify sentences in which entities overlap:

The VW Golf Engine is manufactured in the Shanghai Plant.

In the above example, (VW Golf Engine, Part of, VW Golf) as well as (VW Golf Engine, Manufacturer, Shanghai Plant) and (Plant, Part of, Shanghai) are all correct relation triples. Here the entity “VW Golf Engine” also contains another entity: “VW Golf”. The same goes for “Shanghai Plant”, which includes “Shanghai” as well as “Plant”. By employing an exhaustive search over every entity candidate combination, the model is able to detect all possible relation triples, even between overlapping entities.

6.2 Approach

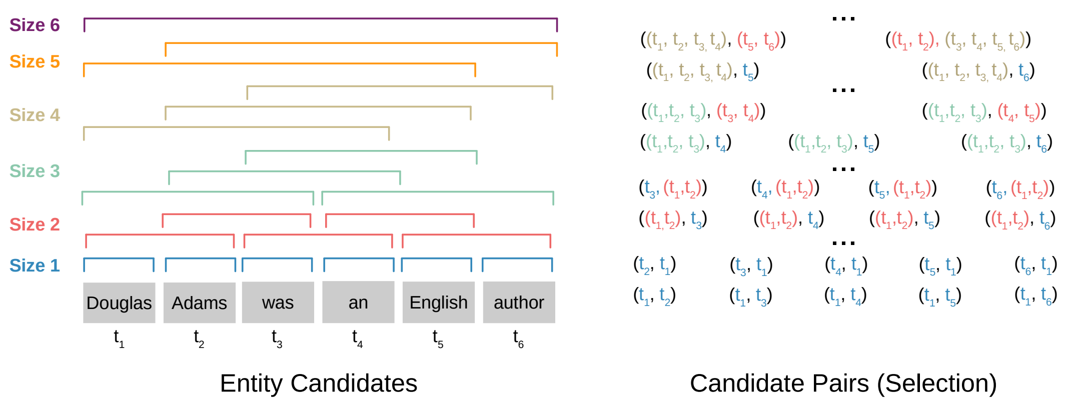


Figure 6.2: An *entity candidate* consists of adjacent byte pair encoded tokens (left). *Candidate pairs* are every combination of entity candidates (right).

In this work, joint entity and relation extraction is performed by exploring the whole space of potential relation triple hypotheses. The amount of potential hypotheses that the model is required to examine can either be restricted with the application of domain specific heuristics or completely left open. In the most generic case (no restrictions), a valid entity candidate $c \in C$ is solely required to be a subsequence of adjacent BPE tokens in a sentence s. Given a sentence of l byte pair encoded tokens $s = (t_1, t_2, ..., t_l)$, an entity candidate is therefore formally characterized by:

$$c = t_{n_1}, t_{n_1+1}, t_{n_1+2}, ..., t_{n_2} \quad \text{with} \quad n_1 < n_1 + 1 < n_1 + 2 < ... < n_2 \tag{6.1}$$

In contrast to many previous approaches, likely entity boundaries and their relations are detected in a single step by computing a score $\phi(c_1, r, c_2)$ for each candidate pair (c_1, c_2) with respect to a relation class $r \in \mathcal{R}$ independently. With this, it is not necessary to extract entities in advance, e.g. by tagging words with respect to the BILOU scheme.

Figure 6.2 shows an example of all possible entity candidates in a sentence of six tokens and a selection of resulting candidate pairs. In practice, entity candidates can also consist of sub-word structures, since the input of the GP-Transformer is byte pair encoded (see Section 2.2.3). The size of the entity candidate set C_l for a sentence of length l can be calculated by:

$$|C_l| = \sum_{i=1}^{l} i = \frac{l \cdot (l+1)}{2} \tag{6.2}$$

Note that the set of entity candidates may be narrowed by applying several heuristics, which depend on the target domain. This can range from only taking entities up to a specific size into account to excluding entities that span the entire sentence. It is even possible to employ a separate NER-Tagger in advance in order to limit the entity candidates. The size of the set $\mathcal{P}_l$, which contains all candidate pairs, can be calculated by:

$$|\mathcal{P}_l| = |C_l|^2 = \Big(\sum_{i=1}^{l} i\Big)^2 = \frac{l^4 + 2l^3 + l^2}{4} \tag{6.3}$$

Analogous to the amount of possible entity candidates, the set of candidate pairs can also be restricted by applying several heuristics, e.g. an entity cannot be related to itself or intersecting entities cannot be related to one another. However, these heuristics depend on the specific problem where the model is applied to and may be appropriate in some domains and violated in others. The advantage of employing an exhaustive search for joint relation extraction is that every combination of entities and relations is theoretically computable. For example, the commonly used BILOU based approaches that were referenced in Chapter 6.1 cannot account for relations that exist between overlapping entities. When applicable, simple heuristic filtering can then be used to narrow down the search space and speed up computation.

As shown in Equation 6.3, the amount of candidate pairs increases significantly (*quartically*) with the sentence length. For example, the sentence in Figure 6.2, which contains only six byte pair tokens, amounts to 21 entity candidates and 441 candidate pairs when no heuristic filtering is applied. A single sentence of the FewRel dataset ($\approx$ 30 BPE tokens on average) therefore contains 465 candidates and 216225 candidate pairs on average. Table 6.1 shows some examples of candidate/candidate pair counts dependent on the sentence length. As shortly mentioned before, each relation triple (c_1, r, c_2) is ranked by a scoring function ϕ during the exhaustive search. The score expresses the likelihood that two entity candidates are related by r and that these candidates are also in the correct order, i.e. the head is placed first in the triple.

Speedup

Sentence Length	Candidates	Candidate Pairs
5	15	225
10	55	3.025
20	210	44.100
40	820	672.400
60	1830	3.348.900

Table 6.1: The amount of candidate pairs, which the model is required to score, grows significantly with the sentence length.

A naive approach could score a relation triple by employing the best performing relation classification model (GPT+EI), which was presented in Section 4.2. However, since the candidate pair search space can grow significantly with the length of the input sequence, it is not possible to insert entity indicators for every candidate pair and run it through the GP-Transformer: With a batch size of 600[4], it takes about one second to classify 274 samples (0.0037 seconds per sample) on a single GPU[5]. In order to evaluate a *single sentence* with 300.000 candidate pair hypotheses[6], it would therefore take more than 18 minutes to get a result for the sentence, which is far too long for most applications.

Instead, a more suitable approach is to feed the sample only once through the GP-Transformer and employ the resulting context-aware embeddings in the fast exhaustive search over the candidate space. In this case, neither entity indicators nor distance embeddings can be added to the input sequence *before* the GP-Transformer in order to indicate the entity candidates. Instead, the post-indication approaches which where introduced in Section 4.2 provide a good alternative: They perform only slightly worse than the GPT+EI model and can be applied efficiently *after* the GP-

[4]The maximum that fits on a Nvidia Quadro P6000

[5]A Nvidia Quadro P6000

[6]In this work, the fast exhaustive search is executed on even more candidate pairs

Transformer. As discussed in Section 4.3, the three evaluated methods (*Entity-Avg*, *Segment-Avg*, *Mixed*) performed similar well. However, averaging each segment of the input (*Segment-Avg*) is more expensive to compute compared to the two other variants. Since the End token in the *Mixed* sentence encoding (concatenation of averaged entities and the special <*End*> token) is able to additionally weight each token, this method was preferred over *Entity-Avg* and is used in an adjusted version in the end-to-end model.

Architecture

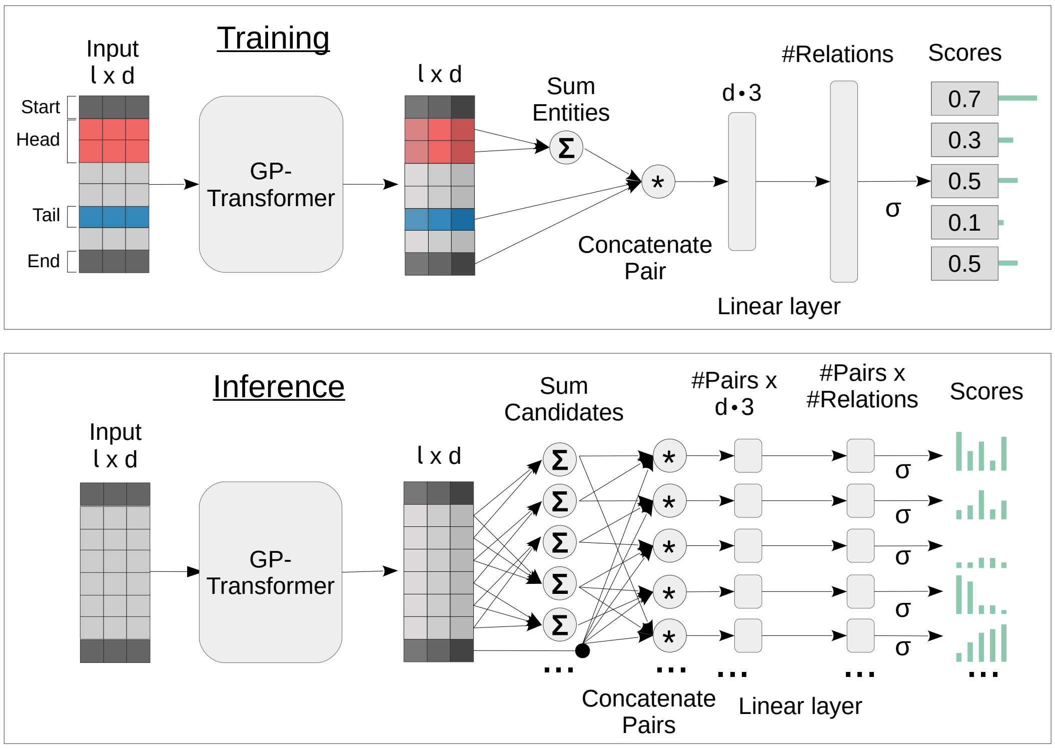


Figure 6.3: The end-to-end model that is employed for joint entity and relation extraction operates differently in the training and inference stage: During training (top) the model is trained to minimize its loss based on positive and negative samples. Here *l* refers to the sentence length. The Transformer embeddings of size *d*, which correspond to the entities, are summed and concatenated with the <*End*> token embedding, which represents the sentence context. The resulting vector $\mathbf{m} \in \mathbb{R}^{d \cdot 3}$ is then mapped to the relation classes. After that, the score for each class is computed by applying a sigmoid activation function. During inference (bottom), a single sample is fed once through the GP-Transformer and each entity candidate pair is scored independently based on the corresponding Transformer embeddings.

Figure 6.3 illustrates the end-to-end model's architecture for training and inference. In comparison to the model employed for relation classification, two main changes had to be made to the model's architecture. First, the *Mixed* sentence encoding was adjusted to make the model aware of the length of the entities or entity candidates.

Because the variance increases when less token embeddings are averaged, short entity candidates tend to attain extreme scores. Therefore the model cannot infer the boundaries correctly and was found to prefer sub-words of entities. For joint extraction, the entities are simply summed instead of averaged to retain length information. With this, the model obtains information on the entity length based on the magnitude of the corresponding summed embedding vector. Second, the softmax function, which was employed for relation classification, was replaced with a sigmoid to be able to model instances where no relation or multiple relations are expressed between a candidate pair. By using a sigmoid instead of a softmax, each relation class is independently assigned a value in the range [0, 1], which expresses the score or likelihood that the candidate pair is related by the respective relation.

Instead of doubling the number of relation types to predict the directionality of the relation, the head and tail of the relation are always placed at the same position of the entity-aware *Mixed* sentence encoding during training. So if $\mathbf{v}_{e_1}$ and $\mathbf{v}_{e_2}$ correspond to the summed Transformer embeddings of the ground truth entity e_1, which occurs first in the sentence, and e_2, the second entity, and $\mathbf{h}_{\text{<End>}}$ denotes the Transformer embedding of the *<End>* token, $\mathbf{v}_{e_1}$ and $\mathbf{v}_{e_2}$ are always placed in the same order dependent on being the tail or head of the relation ($\circ$ denotes concatenation):

$$\mathbf{m} = \begin{cases} \mathbf{v}_{e_1} \circ \mathbf{v}_{e_2} \circ \mathbf{v}_{\text{<End>}}, & \text{if } e_1 = \textit{head} \\ \mathbf{v}_{e_2} \circ \mathbf{v}_{e_1} \circ \mathbf{v}_{\text{<End>}}, & \text{otherwise} \end{cases} \tag{6.4}$$

with $\mathbf{m} \in \mathbb{R}^{d \cdot 3}$ and d denoting the embedding dimensionality of the GP-Transformer. By retaining this order during training, the model learns to score candidate pairs higher when the order in $\mathbf{m}$ is correct. During inference, each relation is then scored with respect to (c_1, r, c_2) as well as (c_2, r, c_1), where c_1 and c_2 again denote two arbitrary candidate entities. Given the entity-aware sentence encoding $\mathbf{m}$, the final scores for each relation are then computed by:

$$\hat{\mathbf{y}} = \textit{sigmoid}(W \cdot \mathbf{m} + \mathbf{b}) \tag{6.5}$$

where $W \in \mathbb{R}^{|\mathcal{R}| \times d \cdot 3}$, $\mathbf{b} \in \mathbb{R}^{|\mathcal{R}|}$ and $\hat{\mathbf{y}} \in \mathbb{R}^{|\mathcal{R}|}$.

Training

In the training stage (top of Figure 6.3), the model is trained to minimize its loss based on batches of positive and negative samples. To asses the ability of the model to correctly score an entity candidate pair, a binary cross entropy loss is used instead of the negative log-likelihood loss, which was used for standard classification (Chapter

4). The binary cross entropy loss is better suited for tasks where a single sample can have multiple labels (relations in this case) or no at all. This loss is minimized during training using stochastic gradient descent. A binary cross entropy loss can be formulated as:

$$\text{Loss} = -\sum_{i=1}^{|\mathcal{R}|} \mathbf{y}_i \cdot log(\hat{\mathbf{y}}_i) + (1 - \mathbf{y}_i) \cdot log(1 - \hat{\mathbf{y}}_i) \tag{6.6}$$

where $\mathbf{y} \in \{0, 1\}^{|\mathcal{R}|}$ refers to the target vector, i.e. a vector that only contains zeros and ones, indicating which relations are expressed. $\hat{\mathbf{y}}$ on the other hand denotes the output of the neural network (Equation 6.5). As in (few-shot) relation classification, this loss is jointly optimized with the language modeling loss.

When the model is only trained on correct relation triples (positive samples), it does not learn to assign a low score to unrelated candidate pairs. Consider the sentence "[Douglas Adams]$_{\text{head}}$ was an [English]$_{\text{tail}}$ author": When the end-to-end model is only trained to assign a high score to (Douglas Adams, Citizenship, English), there is no way to know how to handle (Douglas Adams, Citizenship, author) or (an, Citizenship, was). To correctly separate real relation triples from unrelated candidate pairs, the model is therefore trained on batches of positive and negative samples. Here a negative sample denotes every combination of tokens that is, according to the ground truth, not related in the respective sample. Figure 6.4 shows a selection of different negative samples for an example sentence.

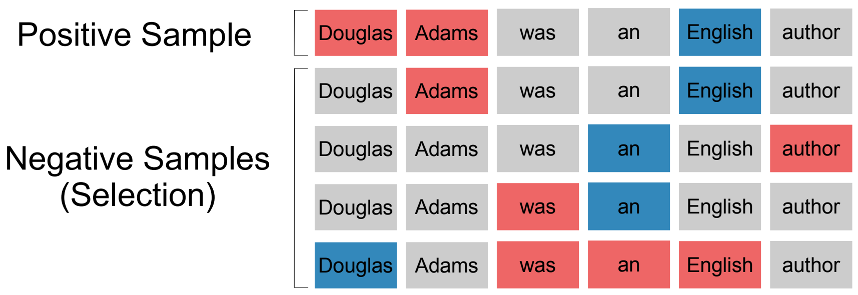

Positive Sample
Negative Samples (Selection)
Douglas Adams was an English author
Douglas Adams was an English author
Douglas Adams was an English author
Douglas Adams was an English author
Douglas Adams was an English author

Figure 6.4: Selected examples of negative samples for the annotated ground truth "[Douglas Adams]$_{head}$ was an [English]$_{tail}$ author". A negative sample is solely required to not match the ground truth relation triple.

For each negative sample, the target vector **y** is filled with zeros. The hyperparameter β controls the proportion of negative samples in relation to the positive samples (i.e. the annotated trainings set) in each epoch. So if β is set to 1, negative samples are just as many as positive samples, if β is set to 2, negative samples are twice as many as positive samples and so on. In each epoch, β random sentences are drawn from the training data for each positive sample and negative entities are marked which do not match the ground truth entities. As for positive samples, bit-masks are created in a preprocessing step to be able to sum the corresponding (negative) entities after the GP-Transformer. Positive and negative samples for all sentences are then mixed randomly to form the final batches, which are fed into the model. The whole process of negative sampling is efficiently implemented in PyTorch based on tensor operations.

Inference

During inference (bottom of Figure 6.3), the model has no notion of the entity boundaries or the relations that are expressed in a sentence. In a preprocessing step, entity candidate bit-masks are created for each sentence length l occurring in the dataset. As explained before, a candidate can be every combination of adjacent tokens (Figure 6.2). However, additional heuristics can be applied at this step to further reduce the candidate count. With this, each candidate has a corresponding bit-mask, which is placed in a matrix $B_l \in \{0,1\}^{|C_l| \times l}$. These candidate masks are referenced by their index in B to obtain the bit-masks which correspond to a candidate pair. The matrix $V_l \in \mathbb{N}_0^{|\mathcal{P}_l| \times 2}$ contains the indices of all possible candidate pairs. Again, these can be reduced by applying several heuristics at this step. Figure 6.5 shows an excerpt of the bit-mask and candidate pair matrices for a sentence length of 5 BPE tokens.

$$B_5 = \begin{array}{c}\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \end{bmatrix}\end{array} \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ \dots \end{array} \qquad V_5 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 2 & 1 \\ 3 & 4 \\ 5 & 1 \\ 1 & 3 \\ \dots & \dots \end{bmatrix}$$

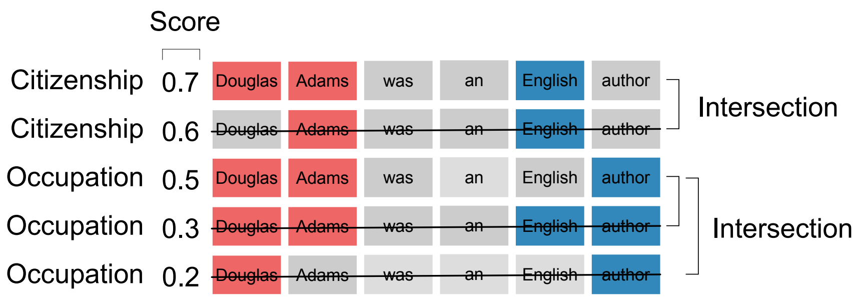
Figure 6.5: Example matrices that contain the entity candidate bit-masks (B_5) and selection of candidate pair indices (V_5) for a sentence of length 5. The indices in V_5 refer to rows in B_5 and are used to retrieve the Transformer embeddings which belong to a specific candidate pair.

Each sample is fed once through the GP-Transformer to obtain the final Transformer embeddings. Then, each entity candidate is extracted by summing the corresponding Transformer embeddings based on the candidate bit-masks, yielding a matrix $U \in \mathbb{R}^{|C_l| \times d}$. The Transformer embedding that corresponds to the <*End*> token is also extracted at this step. Next, each possible pair of candidates is concatenated with the <*End*> token to obtain the final entity-aware sentence encodings $M \in \mathbb{R}^{|\mathcal{P}_l| \times d \cdot 3}$ (with $|\mathcal{P}_l|$ again referring to the amount of candidate pairs for a sentence of length l)

by accessing U with the indices of candidate pairs in V_l. Analogous to the training stage, each entity-aware *Mixed* encoding $\mathbf{m}$ is linearly mapped to the number of relations $|\mathcal{R}|$. Finally, the sigmoid function is applied to the resulting values in order to acquire a score for each candidate pair with regards to a specific relation.

Because of the large amount of possible candidate combinations during the exhaustive search, not every inference step can be processed in a single tensor operation. For example, 1.000.000 candidate pairs would result in a matrix $M \in \mathbb{R}^{1.000.000 \times 768 \cdot 3}$ of 32-bit float values consuming about 9 GB of memory space. This number is even higher because the model operates on batches of samples instead of single samples to speed up processing. For example, with a batch size of 32, M would consume about 288 GB of (GPU) memory. To still be able to process all candidate pairs, the inference stage is processed in chunks in critical places to reduce the memory footprint. Additionally, in order to speed up the evaluation of a large set of sentences (e.g. the test dataset), samples are sorted by length and only candidates (pairs) are evaluated that do not exceed the maximum sentence length in a padded batch (i.e. by accessing the corresponding matrices $B_{l_{\max}}$ and $V_{l_{\max}}$ for a batch with a maximum sentence size $l_{\max}$). Candidate pairs of a batch, where one candidate includes the <*End*> or a padding token, are later masked by setting the score $\phi(c_1, r, c_2)$ for all relations r to 0.

Prediction Filtering



Score
Citizenship 0.7 Douglas Adams was an English author
Citizenship 0.6 ~~Douglas Adams was an English author~~
Occupation 0.5 Douglas Adams was an English author
Occupation 0.3 ~~Douglas Adams was an English author~~
Occupation 0.2 ~~Douglas Adams was an English author~~
Intersection
Intersection

Figure 6.6: Intersecting candidate relation triples are removed in a filtering step. To do this, triple predictions are first sorted by their score. The algorithm begins with the highest scored triple and removes each triple with intersecting entity boundaries and matching relation from the set. At the end, only the highest scored triple of a set of candidate triples that refer to the same ground truth triple is retained.

After inference, $|\mathcal{P}_l| \cdot |\mathcal{R}|$ scores belong to a single sentence of length l, i.e. one sore for each “candidate pair / relation” combination. To obtain the final relation triples that are expressed in a sample, two filtering steps are applied. First, each triple with $\phi(c_1, r, c_2) < \gamma$ is removed from the set of relation triples, where γ is a hyperparameter that controls the minimum required prediction score. Despite this filtering, the set of predictions was still found to contain many candidate triples that refer to the same ground truth triple, e.g. by containing only sub-words of the corresponding entities (e.g. “Adams” instead of “Douglas Adams” in Figure 6.6). To remove the “duplicate” predictions, all candidate triples are first sorted in descending order by their respective score. The algorithm begins with the highest scored triple and removes all triples that *intersect* from the set of triples. Two candidate triples (c_1, r_1, c_2), (c_3, r_2, c_4) intersect if they express the same relation type $(r_1 = r_2)$ and the tokens of the corresponding head and tail predictions intersect, i.e. c_1 intersects c_3 and c_2 intersects c_4. The routine proceeds with the next remaining triple until all intersecting triples are removed. This process is visualized in Figure 6.6. Note that this filtering step does not work in every situation, for example when the entity candidate boundaries do not intersect directly but still refer to the same triple. For example, if the triple (Douglas Adams, Citizenship, English) is assigned a lower score than (Douglas, Citizenship, English) and (Adams, Citizenship, English), the two latter triples would be retrained although referring to the same triple. A more elaborate solution is left for future work.

6.3 Experiments

Whether the fast exhaustive search over all candidate pairs is a viable choice for joint entity and relation extraction depends on the actual model’s performance as well as the inference speed. The purpose of the experiments, which are presented in this section, is to evaluate both critical points, performance and speed, and to also asses the effect of different hyperparameters introduced in Section 6.2, like prediction filtering and the negative sampling rate. Moreover, the German GP-Transformer, which was introduced in Section 2.3, is employed for joint entity and relation extraction on a small-scale German dataset. Here it is assessed if the model scales to a very small training set and how the German-language pre-training aids generalization to new samples.

Setup

Experiments for joint entity and relation extraction were conducted on both the SemEval and FewRel datasets. For SemEval, the official split into a train and test set was used. All samples that are labeled with the “Other” class were removed. For FewRel, the standard classification train/test split, which was described in Section 3.3, was used to evaluate the model. Because the end-to-end model can also incorrectly predict the entity boundaries in addition to the relation type and the direction, four different settings were evaluated:

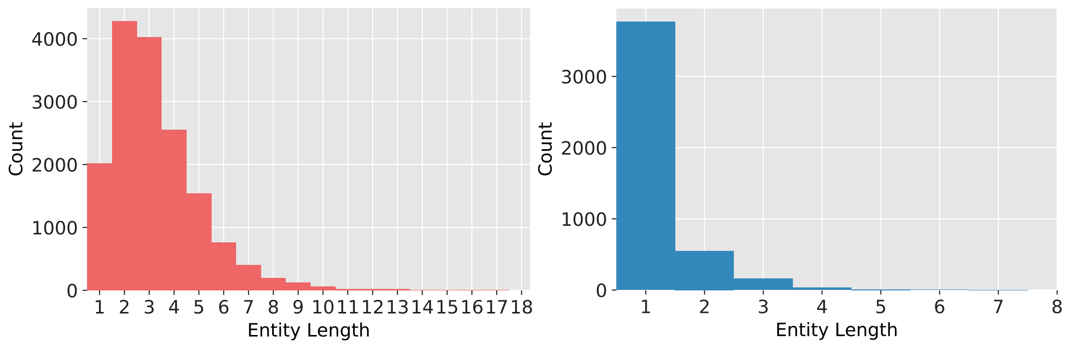
- **(plain)** A candidate triple (c_1, r_c, c_2) is judged to be correct when the relation class coincides with the relation of the ground truth triple (e_1, r, e_2), so when $r_c = r$. The predicted entity boundaries and direction play no role in the plain setting.
- **(inside)** In the *inside* setting, a triple is deemed to be correct when the relation class fits the ground truth relation class and the predicted entity tokens lie inside (or are equal) of the ground truth boundaries. The directionality must also be predicted correctly, i.e. the head must be placed first in the triple. For example, "Ford [Prefect]$_{\text{head}}$ is a car which was manufactured by [Ford UK]$_{\text{tail}}$" is correct when the ground truth triple is "[Ford Prefect]$_{\text{head}}$ is a car which was manufactured by [Ford UK]$_{\text{tail}}$", while "Ford [Prefect is]$_{\text{head}}$ a car which was manufactured by [Ford UK]$_{\text{tail}}$" is not.
- **(jaccard)** The Jaccard index, also called Intersection over Union, is a similarity measure between two sets A, B and formally described as:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{6.7}$$

  In the context of this work, the Jaccard index measures the overlap between the boundaries of the entity candidates and the ground truth entities. A sentence is predicted correctly when $J(c^s_{\text{head}}, e^s_{\text{head}}) \geq 0.5$ and $J(c^s_{\text{tail}}, e^s_{\text{tail}}) \geq 0.5$, where c^s_{head}, e^s_{head}, c^s_{tail} and e^s_{tail} refer to the set of token indices of the respective entity (candidate) boundaries. Again, the directionality and relation must also be predicted correctly. For example, "Ford [Prefect]$_{head}$ is a car which was [manufactured by Ford UK]$_{tail}$" is correct (Jaccard index of 0.5 for both candidates), while "[Ford Prefect is a car]$_{head}$ which was manufactured by [Ford]$_{tail}UK$" (Jaccard index of 0.4 and 0.5 respectively) is not.
- **(strict)** In the *strict* setting, a triple is only deemed to be correctly predicted when it matches the ground truth triple exactly. This includes the boundaries of both entities (equivalent to $J(c^s_{\text{head}}, e^s_{\text{head}}) = 1$ and $J(c^s_{\text{tail}}, e^s_{\text{tail}}) = 1$), the directionality and the relation.

Since the two datasets do not contain overlapping entities, heuristic filtering of candidate pairs is applied by not considering candidate pairs whose entities overlap in order to speed up evaluation, unless noted otherwise. However, since detecting relations between intersecting candidate pairs may be important in practice (as described in the introduction of this chapter), results obtained without heuristic filtering are also presented in an ablation studies section. In both the FewRel and

SemEval dataset, a single relation triple is annotated for each sample. In most experiments in this section, the model is evaluated to correctly predict the ground truth triple and therefore no prediction filtering is applied. The performance is usually measured for the top prediction, i.e. the triple with the highest score, but a top-k evaluation is also included. Since multiple relations can be expressed in a single sample, prediction filtering is applied in later experiments to obtain the final triples, which are manually inspected. The model was trained for three epochs for both the FewRel and SemEval train dataset. The proportion of negative samples, β, is set to 6 for both datasets (6x the amount of training samples).



(a) FewRel **(b)** SemEval

Figure 6.7: Histogram of entity BPE token lengths for the FewRel (left) and SemEval (right) test dataset. The maximum entity length is 17 for FewRel and 7 for SemEval. For FewRel, only samples whose entities are smaller than 7 byte pair encoded tokens are considered (black dotted line), unless noted otherwise. For SemEval, no samples were removed. Average BPE token length of the remaining test samples is 3.07 for FewRel and 1.23 for SemEval.

Figure 6.7 shows the byte pair encoded entity lengths of annotated ground truth samples for the FewRel and SemEval test dataset. The maximum entity length is 17 (1 entity) for FewRel and 7 for SemEval (1 entity). To speed up evaluation, all samples of the FewRel test dataset that contain an entity which is longer than 7 BPE tokens were removed. The remaining samples correspond to 95% of the original test dataset (7584 / 8000). The average entity length of these samples is 3.07 BPE tokens (3.24 for the full FewRel dataset, see Section 3.3). For SemEval, all test samples are evaluated. The average entity length of the SemEval test dataset is the same as for the full dataset with 1.23 BPE tokens (see Section 3.2). Note that training was conducted on the full training set for both datasets.

The other hyperparameters remain the same as described in Section 4.3 for relation classification. The Englisch GP-Transformer, which was pre-trained by OpenAI, is employed for the English FewRel dataset, while the German GP-Transformer together with the German BPE tokenizer is employed for the translated FewRel subset (see Section 3.3 for the acquisition of the German dataset).

Evaluation

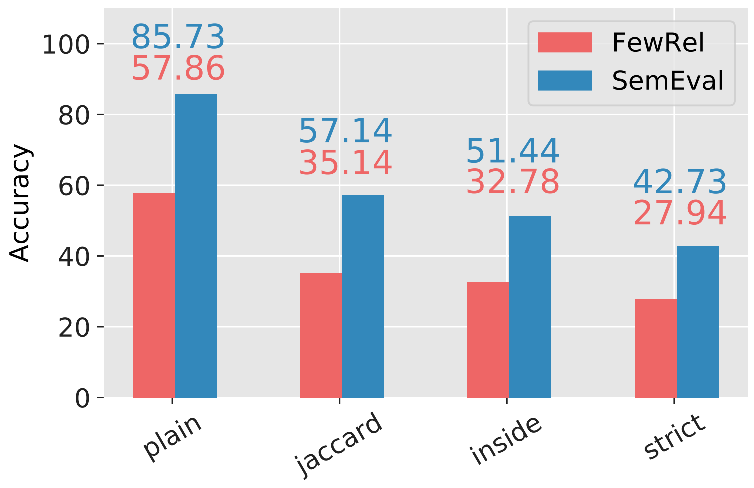


Figure 6.8: Accuracy of the top prediction (the relation triple that was ranked the highest by the model) of each sample in the four settings *plain*, *jaccard*, *inside* and *strict*. FewRel dataset colored in red, SemEval in blue.

Figure 6.8 shows the accuracy of predicting the correct ground truth triple for the four settings, namely *plain*, *jaccard*, *inside* and *strict*. For each sample, the top predicted triple, i.e. the prediction with the highest score, was compared to the ground truth triple. The bars corresponding to the FewRel dataset are colored in red and for the SemEval dataset in blue. As visible in the figure, the ground truth relation (*plain* setting) is predicted correctly in 57.86% (FewRel) and 85.73% (SemEval) of the samples. Note that the SemEval dataset contains significantly fewer relations (9) than the FewRel dataset (80), so predicting the correct class is harder for the latter. Unsurprisingly, out of the three settings that take entity boundaries into account, the *strict* setting performs worst with 27.94% and 42.73% accuracy, respectively. As described before, this setting is very rigorous by only accepting exact matches of relation, entity boundaries and direction. In the more relaxed settings, *jaccard* and *inside*, the accuracy increases to 32.78% and 51.44% (inside) and 35.14% and 57.14% (jaccard). The training loss curves are included in Appendix C.4.

Table 6.2 shows three examples (ground truth top, prediction bottom) where the entity boundaries are not *strictly* predicted correctly by the model, but assessed as correct in the *jaccard* setting. The head entity is colored in red and the tail entity in blue. The topmost prediction shows a typical example where the model did not correctly predict the full entity boundaries ("Blake" is missing from the head entity) but still guessed the right entities. The same applies to the second example: Here the word "book" was predicted to be part of the tail entity while it is not. In some of the manually inspected predictions, the ground truth annotation was also debatable, as

	Relation	Entities
(1)	Military Rank[a]	**Major general** **David Valentine Jardine Blake** (10 November 1887-1965) was a senior commander of the Australian army who served in both World Wars. **Major general** **David Valentine Jardine** Blake (10 November 1887-1965) was a senior commander of the Australian army who served in both World Wars
(2)	Notable Work[b]	the group named itself after the popular children 's book **Harold and the Purple Crayon** by **Crockett Johnson**... the group named itself after the popular children 's **book Harold and the Purple Crayon** by **Crockett Johnson**...
(3)	Licensed to Broadcast to[c]	Most of his reputation comes from his stint, from 1997 to 2005, as morning man with radio station **Choi-FM** in **Quebec city**, Quebec, Canada. Most of his reputation comes from his stint, from 1997 to 2005, as morning man with radio station **Choi-FM** in **Quebec city, Quebec**, Canada.
(4)	Place Served by Transport Hub[d]	In **Porto Alegre** on June 24, around 200 protesters gathered in the city center and traveled toward **the airport**. In **Porto Alegre** on June 24, around 200 protesters gathered in the city center and traveled toward the **airport**.

Table 6.2: FewRel predictions that are not correct in the *strict* setting but in the *jaccard* setting (as well as the *inside* setting for the first and fourth example). The head of the relation is colored in red and the tail in blue. The ground truth is at the top and the prediction at the bottom in each example. In the first and second example, the boundaries do not perfectly match the correct ground truth boundaries, but the entities are correctly guessed. The third and fourth example on the other hand show wrongly or at least debatable annotated ground truth entities.

[a]"military rank achieved by a person ([...]), or military rank associated with a position" (Wikidata)
[b]"notable scientific, artistic [...], or other work of significance among subject's works" (Wikidata)
[c]"place that a radio/TV station is licensed/required to broadcast to" (Wikidata)
[d]"territorial entity or entities served by this transport hub (airport, train station, etc.)" (Wikidata)

shown in the third and bottommost (4) example. In both cases, the model is wrong in the strict setting, but correct in the jaccard setting. However, in case of the second example, "Quebec city, Quebec" (the prediction) as well as "Quebec city, Quebec, Canada" could also be correct dependent on the state or country being included in the entity, which is not specified in the Wikidata description of the relation. The third example is clearly not perfectly annotated, with "airport" (the prediction) being rather correct than the ground truth "the airport".

A closer manual inspection revealed another problem: Since neither FewRel nor SemEval are specifically designed for end-to-end relation extraction, only a single triple is annotated for each sentence, although many relations are potentially expressed in this sentence. Therefore the model was frequently observed to predict a correct

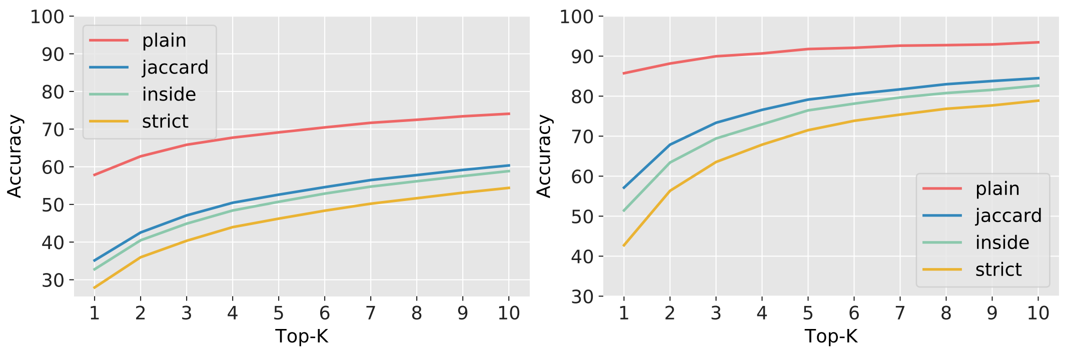
	Relation	Entities
(1)	Component-Whole[a]	The grip is fitted over a rear part of a core of the **helve** of the **hammer**.
	Component-whole[a]	The **grip** is fitted over a rear part of a core of the helve of the **hammer**.
(2)	Applies to Jursidiction[b]	In early 2007, Don Stewart, a retired **supreme court** judge, called for a royal commission into **Victorian** police corruption.
	Occupation[c]	In early 2007, **Don Stewart**, a retired supreme court **judge**, called for a royal commission into Victorian police corruption.
(3)	Voice Type[d]	**William Zakariasen** (August 19 , 1930 - September 4, 2004) was an American operatic **tenor** and music critic.
	Country of Citizenship[e]	**William Zakariasen** (August 19 , 1930 - September 4, 2004) was an **American** operatic tenor and music critic.
(4)	Sport[f]	**Joanne Henke** (born 5 November 1958) is a former Australian **alpine skier** who represented Australia at the 1976 Winter Olympics.
	Participant of[g]	**Joanne Henke** (born 5 November 1958) is a former Australian alpine skier who represented Australia at the **1976 Winter Olympics**.

Table 6.3: Selection of predicted triples that do not match the ground truth but are actually correct (ground truth top, prediction bottom). The first example is extracted from the SemEval dataset and the three other examples from the FewRel dataset. In the first example, a different entity pair is related in the same sentence, but with the same relation type ("Component-Whole"). The other examples depict cases where an entirely different triple (including the relation) occurs in the sentence besides the annotated ground truth.

[a]"An object is a component of a larger whole" (Hendrickx et al. 2010)
[b]"The item ([...]) or statement belongs [...] to the value" (Wikidata)
[c]"Occupation of a person" (Wikidata)
[d]" Person's voice type" (Wikidata)
[e]"The object is a country that recognizes the subject as its citizen" (Wikidata)
[f]" Sport in which the subject participates or belongs to" (Wikidata)
[g]"Event a person or an organization was/is a participant in" (Wikidata)

triple that is not annotated in the ground truth. This is also partly the reason why the model performs better on the SemEval dataset (27.94% versus 42.73% accuracy in strict mode): SemEval contains shorter sentences on average (30.42 BPE tokens versus 20.91 BPE tokens, as depicted in Chapter 3) and fewer relations, so it is more unlikely that multiple relations are expressed in a single sentence.

Table 6.3 shows four examples in which the top prediction does not match the ground truth triple but is a correct relation triple nevertheless. In the first example (SemEval dataset), the sentence contains multiple "Component-Whole" relations to the tail entity "hammer", but the prediction (grip, Component-Whole, hammer) does not match the ground truth (helve, Component-Whole, hammer) triple. Examples 2, 3 and 4 show predictions from the FewRel dataset, where a single sentence contains multiple relations: In the bottommost example (4), “Joanne Henke” is described as an alpine skier as well as a participant of the “1976 Winter Olympics”. Because such cases are very common, especially in FewRel, a manual evaluation of 100 random predictions was conducted for FewRel regarding the *strict* setting. Out of the 100 predictions, 70 where judged to be correct, i.e. to be a valid relation triple that is expressed in the corresponding sentence. The accuracy obtained by comparing the predicted triple with the ground truth was 28% in this case, which is pretty close to the 27.94% strict accuracy over the whole dataset (Figure 6.8). This shows that the fairly low accuracy of 27.94% is partially owed to missing annotations and that the accuracy regarding the top prediction of each sample would be far higher if all valid relation triples were annotated.



(a) FewRel **(b)** SemEval

Figure 6.9: Top-k evaluation of the FewRel and SemEval dataset. The number of considered predictions per sample increases from 1 (only top prediction) to 10.

Figure 6.9 shows a top-k evaluation of the FewRel and SemEval dataset. When the ground truth is only checked to be under the top-10 predictions, the *strict* accuracy reaches 54.4% (60.35% jaccard) for FewRel and 78.88% (84.5% jaccard) for SemEval. Note that this only partially accounts for the aforementioned annotation problems, since subparts of entities are frequently under the top predictions, as shown in Table 6.4: While the correct triple (2011, Sports Season of, Copa do Brasil) is also the top prediction with a score of 99.96 (on a scale from 0 to 100), other triples that contain subparts or neighboring BPE tokens are also assigned a high score. For example, (2011, Sports Season of, pa do Brasil) has a prediction score of 99.18 and (2011, Sports Season of, the Copa do Brasil) a prediction score of 98.96. This confirms that a filtering step like the one described in Section 6.2 is necessary to obtain the final relation triples of a sentence.

Score	Prediction
	São José competed in the **Copa do Brasil** in **2011**, reaching the quarterfinals of the competition [...]
99.96	São José competed in the **Copa do Brasil** in **2011**, reaching the quarterfinals of the competition [...]
99.18	São José competed in the co**pa do Brasil** in **2011**, reaching the quarterfinals of the competition [...]
98.96	São José competed in **the Copa do Brasil** in **2011**, reaching the quarterfinals of the competition [...]
95.48	São José competed in the **Copa do Bra**sil in **2011**, reaching the quarterfinals of the competition [...]
94.38	São José competed in the **Copa do Brasil** **in 2011**, reaching the quarterfinals of the competition [...]

Table 6.4: Ground truth (top) and a selection of sorted predictions (relation "Sports Season of League or Competition"[a]). As shown in this example, the model frequently assigns a high score to subparts of entities or sequences that include neighboring words in addition to the entities.

[a]"Property that shows the competition of which the item is a season" (Wikidata)

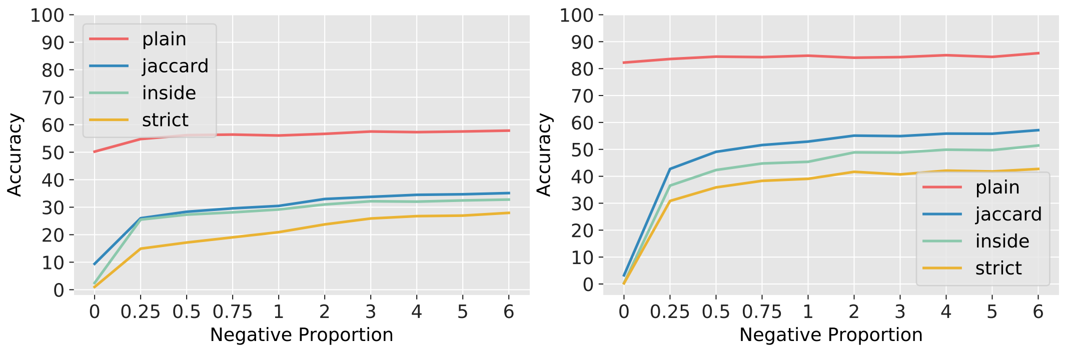
Effect of Negative Sampling

Adding negative samples, as described in Section 6.2, is important to make the model aware of token sequences that express correct entities and to reduce the model's confidence in unrelated entity pairs or pairs whose parts do not constitute actual entities. Table 6.10 shows the effect of an gradually increased negative sampling rate β on the *strict* accuracy: Without negative sampling, the accuracy is barely 1% for the FewRel and 0.27% for the SemEval dataset. By setting the proportion of negative samples to 0.25 (25% of the amount of training samples), the accuracy increases to 14.92% and 41.67%, respectively. The model especially performs better up to an negative proportion of 3 (FewRel, 3x the amount of training samples) and 2 (SemEval) and then only marginally improves. Note however that negative sampling also has a significant impact on the training duration.

Ablation Studies

To asses the performance influence of some of the decisions which were made regarding the model's architecture and choice of different hyperparameters, a set of several ablation studies was conducted on the FewRel dataset:

- (**average**) For the task of joint entity and relation extraction, the model is required to determine the exact boundaries of entities. In order to provide information about the entity lengths, the Transformer embeddings which belong to the candidate entities were summed instead of averaged. This experiment



(a) FewRel **(b)** SemEval

Figure 6.10: Influence of negative sampling for the FewRel (left) and SemEval (right) dataset. In both datasets, the top-1 accuracy increases drastically when negative sampling is performed, even in the lowest setting, $\beta = 0.25$. Out of the tested values for β, the best performance was achieved with $\beta = 6$, although the accuracy increases to a lesser extend after $\beta = 3$ (FewRel) and $\beta = 2$ (SemEval) (e.g. from 25.90% to 27.94% for FewRel).

assesses the impact on the model's performance, when an averaging of embeddings is performed instead.

- (**generalization**) The set of samples which the model was evaluated on, contains 4072 entities that also occur in the training set (according to the Wikidata ID, which is included in the annotated samples for each entity, and text matching). Overall, 7892 entities are distinct to the validation set. Since generalization to unseen entities is important for joint entity and relation extraction, the model was also evaluated on a test set which does not include any entity of the training set. This leaves a test set with 1731 out of 8000 samples (21.64%).

- (**no-filter**) Neither the SemEval nor the FewRel dataset contain overlapping entities in a sentence. Still, detecting relations between overlapping entities may be important in practice. With the purpose of validating if the addition of overlapping candidate pairs during influence has a negative impact on the model's performance, no heuristic filtering of candidate pairs was applied in this setting.

- (**full**) To speed up evaluation, the model was only tested on samples that contain entities up to a specific size, as described before. In this setting, the full test dataset is evaluated and candidate entities up to a length of 17 BPE tokens (which is the maximum entity length of the test dataset) are considered. Candidate pairs with overlapping entities are also not filtered in this setting before inference.

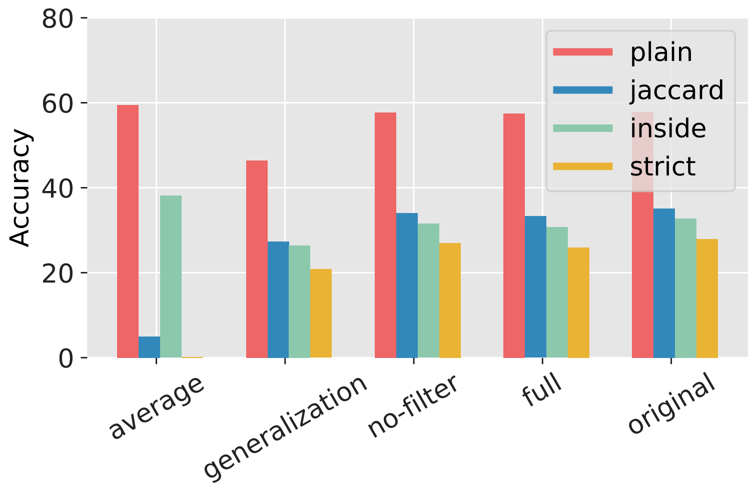


Figure 6.11: Top-1 accuracy for the five different ablation studies: (**average**) Conduct averaging instead of summing of the Transformer embeddings that correspond to the entities. (**generalization**) Only include samples whose entities do not occur in the training set. (**no-filter**) Omit heuristic filtering for candidate pairs. (**full**) Do not filter samples that contain an entity which is longer than 7 BPE tokens. Evaluation is instead conducted on the full test set. (**original**) The model of Figure 6.8, i.e. with heuristic filtering and entities up to 7 BPE tokens.

The first three settings (*average*, *generalization*, *no-filter*) are evaluated on the same test set as the *original* model (Table 6.8 and far right of Table 6.11)), which includes only entities up to a length of 7 BPE tokens. Also, since the settings *generalization*, *no-filter* and *full* only have an impact on the influence step and not on training, the model was not re-trained in this case. These settings were just evaluated with the trained *original* model. The results obtained under the different ablation studies are illustrated in Figure 6.11.

Average: By averaging the Transformer embeddings that correspond to the entity candidates (*average* study), the model performs better on the *plain* (59.49% versus 57.86%) and especially the *inside* setting (38.10% versus 32.78%) but substantially worse in the *jaccard* (5.01% versus 35.14%) and *strict* setting (0.17% versus 27.94%). Since the model is not aware of entity lengths when the embeddings are averaged, it tends to prefer single BPE tokens: For example in the sentence "[Nintendo]$_{tail}$ released '[Cobra Triangle]$_{head}$' in July 1989" (relation:Publisher[7]) the top prediction of the average model is "Ninten[do]$_{tail}$ released '[Cobra]$_{head}$ Triangle' in July 1989", while the *original* model (sum of embeddings) predicted the correct triple. This behavior is also evident when looking at the average entity lengths of the top prediction: 3.16 BPE tokens for the *original* model and exactly 1 BPE token for the *average* model. So while the *original* model is pretty close to the average annotated

[7]"Organization or person responsible for publishing books, periodicals, games or software" (Wikidata)

entity length of the test dataset (3.07 BPE tokens, see Figure 6.7), the *average* model prefers single tokens and is therefore inappropriate for the task.

Generalization: In the *generalization* study, the model proves to be able to generalize to completely unseen entities: Here the *strict* top prediction *strict* accuracy is 20.79% percent compared to 27.94% when train entities are included (*original*). Again, the actual top prediction accuracy is most likely way higher, as shown by the manual inspection before. Since the entities were not observed during training, the unsupervised pre-training of the GP-Transformer is probably an important factor for generalization under this setting.

No-filter and full: Also including overlapping candidate pairs (*no-filter*) during inference has only a minor impact on the performance (26.96% versus 27.94% *strict* accuracy and 34.02% versus 35.14% *jaccard* accuracy), although overlapping pairs do not occur in the two datasets. When the model is evaluated on the complete test dataset (*full* setting), the performance decreases from 27.94% to 25.95% (*strict*). Considering that the search space drastically grows when longer entities and intersecting pairs are included (e.g. from 223.636 pairs to 1.258.884 pairs for the longest test sentence), this also has only a minor impact on the model's prediction accuracy.

Exhaustive Search Speed

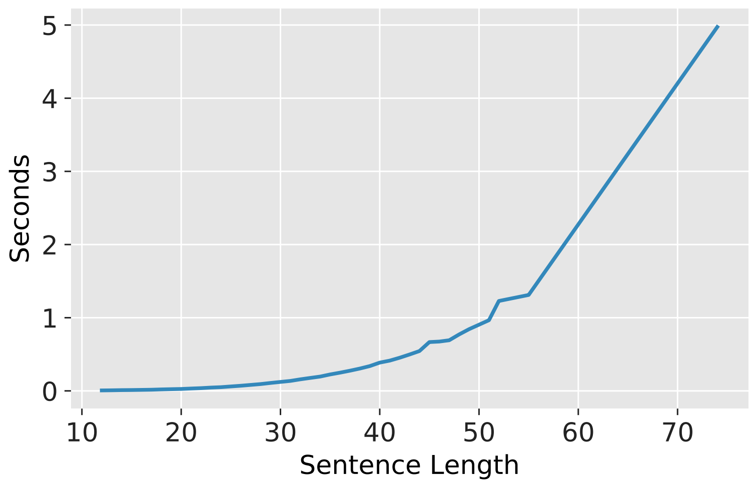


Figure 6.12: Inference duration for sentences of a specific length. The duration is displayed in seconds for single sentences of the test dataset.

Since a reasonable prediction speed is a core requirement for the practical capability of the exhaustive search approach, the model's speed with respect to scoring all candidate pairs (inference step in Figure 6.3) of a sentence was evaluated on the complete FewRel test dataset. For speed evaluation, no heuristic filtering was applied. Moreover, the size of entity candidates is not capped, so candidates up to the size of each sentence are included. With this, the count of candidate pairs can be calculated by Equation 6.3 of Section 6.2: The shortest evaluated sentence of the test dataset (12 BPE tokens) contains 6084 candidate pairs and the longest sentence (74 BPE tokens) 7.700.625 candidate pairs. Each single one of those candidate pairs is assigned a prediction score with respect to every relation. The speed evaluation was conduced on batch sizes of 32, the maximum that can fit on a NVIDIA Quadro P6000 GPU.

Figure 6.12 shows the inference duration in seconds for single samples of a specific sentence length. For a short sentence of 14 BPE tokens, the model needs about 0.006 seconds on average and for the longest sentence (74 BPE tokens) 4.97 seconds. The time that is required to run 300.000 candidate pair hypotheses through the *entire* GP-Transformer was approximated in Section 6.2: 18 minutes per sentence. In contrast to this, the model which is employed for the fast exhaustive search only needs 0.177 seconds for a sentence with 33 BPE tokens (314.721 candidate pairs). Extracting candidate pairs *after* the GP-Transformer, as introduced in this work, makes the fast exhaustive search therefore feasible in practice.

Prediction Filtering

Relation	Entities
Participant of[a]	**Joanne Henke** (born 5 November 1958) is a former Australian alpine skier who represented Australia at the **1976 Winter Olympics**.
Participant of[a]	Joanne Henke (born 5 November 1958) is a former Australian alpine skier who represented **Australia** at the **1976 Winter Olympics**.
Sport[b]	**Joanne Henke** (born 5 November 1958) is a former Australian **alpine skier** who represented Australia at the 1976 Winter Olympic.
Country of Citizenship[c]	**Joanne Henke** (born 5 November 1958) is a former Australian alpine skier who represented **Australia** at the 1976 Winter Olympics.
Country of Citizenship[c]	**Joanne Henke** (born 5 November 1958) is a former **Australian** alpine skier who represented Australia at the 1976 Winter Olympics.

Table 6.5: Example of final triples obtained by prediction filtering ($\gamma = 0.8$) on a sentence that contains multiple relations between different entities.

[a]"Event a person or an organization was/is a participant in" (Wikidata)
[b]" Sport in which the subject participates or belongs to" (Wikidata)
[c]"The object is a country that recognizes the subject as its citizen" (Wikidata)

As already illustrated in Table 6.4, subparts of entities and sequences that span across entities were frequently observed to be assigned a high score. In practice, the final relation triples that are expressed in a sentence are usually of high relevance. In this case, an additional filtering step like the one described in Section 6.2 must be conducted. Since neither FewRel nor SemEval are fully annotated with every relation triple, prediction filtering was manually evaluated based on some samples. In the following examples, the prediction filtering threshold γ is set to 0.8, which was observed to be a fairly good compromise between precision and recall. Table 6.5 shows an example sentence with the final triples that where predicted by the model with prediction filtering (see Section 6.2). In this example, all predicted triples are correct, with "Joanne Henke" being a participant of the "1976 Winter Olympics", an "alpine skier" and an "Australian" citizen. The triple (Australia, Participant Of, 1976 Winter Olympics) on the other hand is also correct according to the Wikidata description and the other ground truth samples of the FewRel dataset for this relation.

	Relation	**Entities**
(1)	Religion[a]	**Giovanni Maria Gabrielli** (January 10, 1654 - September 17, 1711) was an Italian **catholic church** 's cardinal.
		Giovanni Maria Gabrielli (January 10, 1654 - September 17, 1711) was an Italian **catholic church** 's **cardinal**.
		Giovanni Maria Gabrielli (January 10, 1654 - September 17, 1711) was an **Italian** **catholic church** 's cardinal.
		Giovanni Maria Gabrielli (January 10, **16**54 - September 17, 1711) was an Italian **catholic church** 's cardinal.
		Giovanni Maria Gabrielli (January 10, 1654 - September 17, **17**11) was an Italian **catholic church** 's cardinal.
(2)	Instrument[b]	While in Chicago, he learned the **blues harp** from Little Walter and began an association with pianist **Eddie Boyd**.
(3)	Father[c]	She and **Kawelo'okalani** had no children, although one source says that Kaukuna Kahekili was the son of **Kawelo'okalani** and Peleuli...

Table 6.6: Typical error cases: (1) Selection of final triples after prediction filtering of the given sentence. As in this example, the model was frequently observed to assign a high score (> 95 in this case) to unrelated entity pairs, such as (16, Religion, catholic church) and (17, Religion, catholic church). (2+3) Both examples show predicted entity pairs which might appear in the predicted relation based on the entity types (e.g. a person and an instrument), but the relation is not expressed in the sentence between the entities.

[a]" Religion of a person, organization or religious building, or associated with this subject" (Wikidata)
[b]" Musical instrument that a person plays" (Wikidata)
[c]"Male parent of the subject" (Wikidata)

However, the overall quality of final triples varied between the inspected examples. Table 6.6 shows a selection of typical error cases. In the topmost example (1), the top prediction coincides with the ground truth and the second and third prediction may be correct according to the vague Wikidata description. The fourth and fifth prediction on the other hand also have a high prediction score (> 95), but the corresponding entities are clearly not related in the sample. Such cases are common: The model frequently assigns a fairly high score to relation triples where the relation and only a single entity is predicted correctly but the other part does not constitute an actual entity or is unrelated. This makes the selection of the filtering threshold γ a tough choice. In this example, a higher count of and more elaborately selected negative samples might improve the accuracy. The two other examples (2+3) of Table 6.6 show instances where both entities could theoretically be related according to their entity type (e.g. person and instrument) but are actually not in the corresponding sentence: “Little Walter”, not “Eddie Boyd”, plays the blues harp and “Kawelo’okalani” is certainly not the father of himself. In such cases, the model pays especially attention to the participating entities independent of their location in the sentence. While not explored in this work due to time restrictions, adding additional information about the sentence syntactic structure, for example with a dependency parse of the sentence, may be a useful addition to the model.

German Evaluation

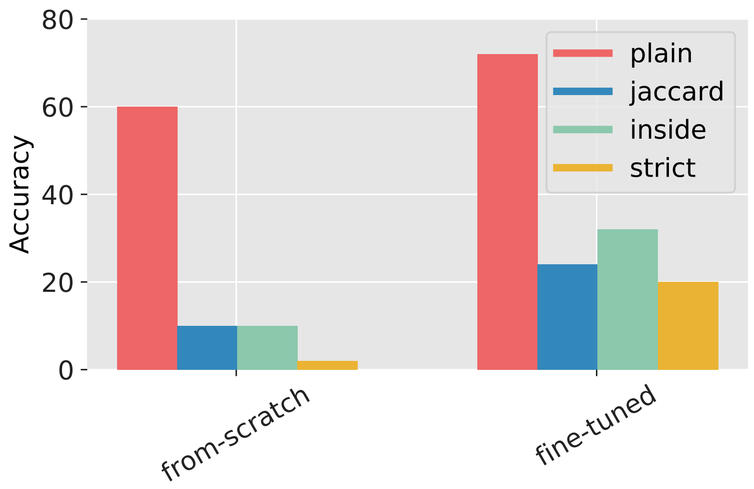


Figure 6.13: Accuracy regarding the four settings *plain*, *jaccard*, *inside* and *strict* on a German subset derived from FewRel. The German GP-Transformer is used for feature extraction during the fast exhaustive search instead of the English version. It is either fine-tuned on the target domain (left) or trained from-scratch (right). The fine-tuned model outperforms the model trained from-scratch in all settings, especially on those where entity boundaries are considered. When fine-tuned it reaches a top prediction performance of 20% *strict* accuracy.

The German GP-Transformer, which was introduced in Section 2.3, is employed in this work for joint entity and relation extraction on the small German relation extraction dataset (see Section 3.3). The dataset contains the five relation classes “Director”, “Country of Citizenship”, “Architect”, “Participating Team” and “Instrument”. Since only 40 training samples are available for each class, the end-to-end model is required to generalize from very little training data. The architecture used for training and inference remains the same as described in Section 6.2, but the English GP-Transformer is replaced with the German version. The model is then either fine-tuned on the end-to-end relation extraction task or trained from-scratch instead (no pre-training). In both cases, the model was trained for 10 epochs and evaluated on a separate test dataset (see Section 3.3), which contains 50 samples in total. Training the model for more than 10 epochs had no positive impact on the performance.

Figure 6.13 shows the accuracy regarding the top prediction of all test samples. As visible in the figure, the model predicts the ground truth relation triple in only 2% percent of the cases when it is trained from-scratch. While the model performs better in predicting only the relation type (60%), it fails in detecting correct entity boundaries. When the pre-trained German GP-Transformer is instead employed as the feature extractor and fine-tuned on end-to-end relation extraction, the model reaches an accuracy of 20% in the *strict* setting (24% *jaccard*, 32% *inside*). The accuracy of predicting the correct relation class (*plain*) also increases to 72%. This demonstrates the importance of unsupervised pre-training: Even with only a small amount of samples from the target task, the German GP-Transformer's outperforms the model which was trained from-scratch by 18% *strict* accuracy. Here the unsupervised pre-training is especially beneficial for generalizing to unseen entities.

	Relation	Entities
(1)	Director[a]	**Trapped and Deceived** ist ein Fernsehfilm aus dem Jahr 1994 von **Robert Iscove**. Trapp**ed and Deceived** ist ein Fernsehfilm aus dem Jahr 1994 von **Robert Iscove**.
(2)	Participating Team[b]	In einer **Europa-League-Partie** gegen **Benfica** gab er am selben Abend sein Debüt für den VfB. In einer **Europa-League-Partie** gegen Benf**ica** gab er am selben Abend sein Debüt für den VfB.
(3)	Country of Citizenship[c]	**Gaetano Cicognani** ([...]) war ein **italienischer** Kardinal der katholischen Kirche. Gaetano Cicogn**ani** ([...]) war ein **italienischer** Kardinal der katholischen Kirche.

Table 6.7: Examples of predicted triples, which refer to the correct entities but contain only subsequences of the ground truth entities. The annotated ground truth is at the top and the prediction at the bottom of each example.

[a]"Director(s) of film, TV-series, stageplay, video game or similar" (Wikidata)
[b]"[Team] that actively takes/took part in an event or process" (Wikidata)
[c]"The object is a country that recognizes the subject as its citizen" (Wikidata)

As explained before, other relations besides the ground truth relation are frequently expressed in a single sentence. Just like for the English dataset, the results where manually inspected and evaluated. In 36% of all samples (18/50), the top prediction was judged to be a correct relation triple which is expressed in the corresponding sentence. Furthermore, the model was frequently observed to correctly guess the entities and their relations, but to not fully match the correct boundaries. Regarding the *inside* setting, the model therefore reached a higher top prediction of 64% upon manual inspection. Some examples of this behavior are include in Table 6.7: Here the model predicts the triple (ed and Deceived, director, Robert Iscove) while (Trapped and Deceived, Director, Robert Iscove) is actually correct. Similarly, in the bottommost example the top predictions is (ani, Country of Citizenship, italienischer) although (Gaetano Cicognani, Country of Citizenship, italienischer) is the right triple. However, jointly detecting entities and their relations is a tough problem and promising results where obtained by fine-tuning the German GP-Transformer on a small amount of data. It is therefore save to assume that more German training data leads to better results and predicted entity boundaries that better match the actual entities.

6.4 Conclusions

In this chapter, the GP-Transformer was employed for the joint extraction of entities and their relations by training a single model end-to-end on this task. By either using the pre-trained OpenAI model or the German GP-Transformer as an elaborate feature extractor in a fast exhaustive search, promising results were obtained for both the English and German language. Moreover, it was demonstrated that a fast exhaustive search is feasible even on a large search space of several thousand or even several million candidate pairs. Making the model aware of entities that are not related during training by adding negative samples to the training set was shown to substantially improve the performance. In addition to that, a simple filtering approach that removes intersecting triples was proposed to obtain the final relation triples. However, since the two evaluated datasets only contain a single annotated ground truth relation per sentence, the model's performance could not fully be assessed. Besides evaluation, missing annotated triples also have a bad impact on the training stage, since correct relation triples can be presented as negative samples.

Several interesting extensions to the model can be examined in future work on this topic: For example, adding a separate classifier could enable the model to also extract the named entity tags of participating entities, which in turn may be beneficial to detect the correct relation. By extracting and scoring the negative triples of a sentence *after* the GP-Transformer (similar to the inference stage) the training stage could be speed up even in case of a large negative sampling rate. The model was observed to score triples high when the relation and only a single entity is predicted correctly but the other part does not constitute an actual entity or is unrelated. In this case, more carefully selected negative samples or an adjusted loss that specifically incorporates the correctness of both entities may mitigate this problem. Manual inspection of test samples also revealed that the model has difficulties in discriminating which entities of appropriate types fit an expressed relation, especially in long sentences. This could potentially be improved by making the model aware of the syntactic sentence structure, for example by adding information of the sentence's dependency parse. Finally, a more elaborate filtering approach than the one used in this work could be employed to obtain all final relation triples that are expressed in a sentence.

Chapter 7

Conclusions

In this work, transfer learning was explored for the extraction of relations between pairs of entities in a sentence. Here the GP-Transformer, a deep and attention based model which was pre-trained in an unsupervised manner on an English-language corpus (Radford et al. 2018), is fine-tuned on the relation extraction task. To also obtain knowledge about the syntax and semantics of the German language, the GP-Transformer was pre-trained in this work on a large and diverse German corpus. It was demonstrated by a clustering based approach that the model is able to generate word representations that are dependent on the specific input context. In experiments on three relation extraction subtasks, task-specific models that include the GP-Transformer as its core part were able to achieve competitive performance in (few-shot) relation classification and yield promising results in a novel fast exhaustive search based approach for joint entity and relation extraction.

Transferring whole pre-trained models such as the GP-Transformer to a new domain or task was already shown to significantly outperform previous models, which were trained from-scratch and often employ fixed word embeddings as input, in various NLP tasks (e.g. Radford et al. 2018, Peters et al. 2018, Howard and Ruder 2018, Devlin et al. 2018). In this work, the strength of transfer learning was also confirmed for the relation extraction task: The employed task-specific models were shown to converge quickly and to obtain strong generalization capabilities through the unsupervised pre-training. This is especially evident when the model is required to learn from only a small amount of data, as shown in few-shot relation classification and a German joint entity and relation extraction task with little training data. Here the language modeling pre-training proved to be very effective in reducing the amount of target domain samples that are necessary to correctly predict the relation between an entity pair. Since scarcity of labeled data is a core problem for relation extraction and other NLP tasks in practice, it is save to assume that transfer learning will be an important factor for machine learning development in the years ahead.

Many interesting questions have arisen in this work which can be addressed in future work on this topic: As observed in the explored relation extraction tasks, the performance drastically decreased when the GP-Transformer was not fine-tuned on the target task. Still, since training of large models is expensive, further research could be targeted at the question if there are ways to bypass fine-tuning and use the rich Transformer embeddings in shallow and fast to train models. Furthermore, the unidirectionality of the GP-Transformer, which only attends to previous occurring tokens, is an inherent limitation of the model. The bidirectional model BERT outperforms its unidirectional counterpart in a number of NLP tasks (Devlin et al. 2018). Therefore using BERT instead of the GP-Transformer as the core of the task-specific models proposed in this work could further increase the performance. While a multilingual BERT model already exists, it is interesting to see how it compares to a pure German pre-training, as conducted in this work with the GP-Transformer. Although the models that were employed for the three relation extraction subtasks can all be extended in various ways, the fast exhaustive search, which was proposed for jointly extracting entities and their relations, is a very promising candidate for further research. Here the model should especially be evaluated on datasets that are more suitable for joint entity and relation extraction and compared to state-of-the-art models. Since the model performs well but not yet perfect, especially for sentences with lots of clutter between entities, several interesting extensions can be examined in future work. This includes adding information about the sentence's syntactic structure or more complex prediction filtering approaches.

Bibliography

Alt, Christoph, Marc Hübner, and Leonhard Hennig (2019). “Improving Relation Extraction by Pre-trained Language Representations”. In: *Submitted to Automated Knowledge Base Construction*. under review. URL: https://openreview.net/forum?id=BJgrxbqp67.

Ba, Lei Jimmy, Ryan Kiros, and Geoffrey E. Hinton (2016). “Layer Normalization”. In: *CoRR* abs/1607.06450. arXiv: 1607.06450. URL: http://arxiv.org/abs/1607.06450.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2014). “Neural Machine Translation by Jointly Learning to Align and Translate”. In: *CoRR* abs/1409.0473. arXiv: 1409.0473. URL: http://arxiv.org/abs/1409.0473.

Bekoulis, Giannis, Johannes Deleu, Thomas Demeester, and Chris Develder (2018). “Joint entity recognition and relation extraction as a multi-head selection problem”. In: *CoRR* abs/1804.07847. arXiv: 1804.07847. URL: http://arxiv.org/abs/1804.07847.

Byrne, Kate (2006). “Relation Extraction for Ontology Construction”. In:

Cho, Kyunghyun, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio (2014). “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation”. In: *CoRR* abs/1406.1078. arXiv: 1406.1078. URL: http://arxiv.org/abs/1406.1078.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *CoRR* abs/1810.04805. arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

Duden (2017). “Duden. Deutsches Universalwörterbuch”. In: *Dudenverlag, Mannheim / Leipzig / Wien / Zürich*.

Ebrahimi, Javid and Dejing Dou (2015). “Chain Based RNN for Relation Classification”. In: *HLT-NAACL*.

Fellbaum, Christiane, ed. (1998). *WordNet: an electronic lexical database*. MIT Press.

Finn, Chelsea, Pieter Abbeel, and Sergey Levine (2017). “Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks”. In: *CoRR* abs / 1703.03400. arXiv: 1703.03400. URL: http://arxiv.org/abs/1703.03400.

Gage, Philip (1994). “A New Algorithm for Data Compression”. In: *C Users J.* 12.2, pp. 23–38. ISSN: 0898-9788. URL: http://dl.acm.org/citation.cfm?id=177910.177914.

Gao, Tianyu, Xu Han, Zhiyuan Liu, and Maosong Sun (2019). “Hybrid Attention-based Prototypical Networks for Noisy Few-Shot Relation Classification”. In: *The 33rd AAAI Conference on Artificial Intelligence (AAAI 2019).*

Garcia, Victor and Joan Bruna (2017). “Few-Shot Learning with Graph Neural Networks”. In: *CoRR* abs/1711.04043.

Girshick, Ross B. (2015). “Fast R-CNN”. In: *CoRR* abs/1504.08083. arXiv: 1504.08083. URL: http://arxiv.org/abs/1504.08083.

Glorot, Xavier and Yoshua Bengio (2010). “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Yee Whye Teh and Mike Titterington. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, pp. 249–256. URL: http://proceedings.mlr.press/v9/glorot10a.html.

Gupta, Pankaj, Hinrich Schütze, and Bernt Andrassy (2016). “Table Filling Multi-Task Recurrent Neural Network for Joint Entity and Relation Extraction”. In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan: The COLING 2016 Organizing Committee, pp. 2537–2547. URL: http://aclweb.org/anthology/C16-1239.

Han, Xu, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun (2018a). “FewRel: A Large-Scale Supervised Few-Shot Relation Classification Dataset with State-of-the-Art Evaluation”. In: *CoRR* abs/1810.10147. arXiv: 1810.10147. URL: http://arxiv.org/abs/1810.10147.

— (2018b). *FewRel Dataset, Toolkits and Baseline Models*. https://github.com/ProKil/FewRel. [Online; uploaded 29. August 2018; last accessed on 17. April 2019].

Hendrickx, Iris, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó. Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz (2010). “SemEval-2010 Task 8: Multi-way Classification of Semantic Relations Between Pairs of Nominals”. In: *Proceedings of the 5th International Workshop on Semantic Evaluation*. SemEval ’10. Los Angeles, California: Association for Computational Linguistics, pp. 33–38. URL: http://dl.acm.org/citation.cfm?id=1859664.1859670.

Hendrycks, Dan and Kevin Gimpel (2016). "Gaussian Error Linear Units (GELUs)". In: *CoRR* abs/1606.08415. arXiv: `1606.08415`. URL: `http://arxiv.org/abs/1606.08415`.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780.

Howard, Jeremy and Sebastian Ruder (2018). "Fine-tuned Language Models for Text Classification". In: *CoRR* abs/1801.06146. arXiv: `1801.06146`. URL: `http://arxiv.org/abs/1801.06146`.

Hugging Face (2018). *PyTorch implementation of OpenAI's Finetuned Transformer Language Model*. `https://github.com/huggingface/pytorch-openai-transformer-lm`. [Online; uploaded 13. June 2018; last accessed on 16. April 2019].

Jang, Eric, Shixiang Gu, and Ben Poole (2017). "Categorical Reparameterization with Gumbel-Softmax". In: URL: `https://arxiv.org/abs/1611.01144`.

Junkert, Fabian, Markus Eberts, Adrian Ulges, and Ulrich Schwanecke (2017). "Cross-modal Image-Graphics Retrieval by Neural Transfer Learning". In: *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*. ICMR '17. Bucharest, Romania: ACM, pp. 330–337. ISBN: 978-1-4503-4701-3. DOI: `10.1145/3078971.3078994`. URL: `http://doi.acm.org/10.1145/3078971.3078994`.

Kate, Rohit J. and Raymond J. Mooney (2010). "Joint Entity and Relation Extraction Using Card-pyramid Parsing". In: *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*. CoNLL '10. Uppsala, Sweden: Association for Computational Linguistics, pp. 203–212. ISBN: 978-1-932432-83-1. URL: `http://dl.acm.org/citation.cfm?id=1870568.1870592`.

Kaufmann, Leonard and Peter Rousseeuw (1987). "Clustering by Means of Medoids". In: *Data Analysis based on the L1-Norm and Related Methods*, pp. 405–416.

Kingma, Diederik P. and Jimmy Ba (2014). "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980. arXiv: `1412.6980`. URL: `http://arxiv.org/abs/1412.6980`.

Kobayashi, Nozomi, Kentaro Inui, and Yuji Matsumoto (2007). "Extracting Aspect-Evaluation and Aspect-Of Relations in Opinion Mining". In: *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL*.

Lee, Joohong, Sangwoo Seo, and Yong Suk Choi (2019). "Semantic Relation Classification via Bidirectional LSTM Networks with Entity-aware Attention using Latent Entity Typing". In: *CoRR* abs/1901.08163. arXiv: `1901.08163`. URL: `http://arxiv.org/abs/1901.08163`.

Liu, Peter J., Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer (2018). "Generating Wikipedia by Summarizing Long Sequences". In: *CoRR* abs/1801.10198. arXiv: `1801.10198`. URL: `http://arxiv.org/abs/1801.10198`.

Merity, Stephen, Caiming Xiong, James Bradbury, and Richard Socher (2016). "Pointer Sentinel Mixture Models". In: *CoRR* abs/1609.07843. arXiv: `1609.07843`. URL: `http://arxiv.org/abs/1609.07843`.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean (2013). "Efficient Estimation of Word Representations in Vector Space". In: *CoRR* abs/1301.3781. arXiv: `1301.3781`. URL: `http://arxiv.org/abs/1301.3781`.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean (2013). "Distributed Representations of Words and Phrases and their Compositionality". In: *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger. Curran Associates, Inc., pp. 3111–3119. URL: `http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf`.

Mintz, Mike, Steven Bills, Rion Snow, and Dan Jurafsky (2009). "Distant Supervision for Relation Extraction Without Labeled Data". In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*. ACL '09. Suntec, Singapore: Association for Computational Linguistics, pp. 1003–1011. ISBN: 978-1-932432-46-6. URL: `http://dl.acm.org/citation.cfm?id=1690219.1690287`.

Mishra, Nikhil, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel (2017). "Meta-Learning with Temporal Convolutions". In: *CoRR* abs/1707 . 03141. arXiv: `1707.03141`. URL: `http://arxiv.org/abs/1707.03141`.

Miwa, Makoto and Mohit Bansal (2016). "End-to-end Relation Extraction using LSTMs on Sequences and Tree Structures". In: *CoRR* abs/1601.00770. arXiv: `1601.00770`. URL: `http://arxiv.org/abs/1601.00770`.

Miwa, Makoto and Yutaka Sasaki (2014). "Modeling Joint Entity and Relation Extraction with Table Representation". In: *EMNLP*.

Munkhdalai, Tsendsuren and Hong Yu (2017). "Meta Networks". In: *CoRR* abs/1703.00837. arXiv: `1703.00837`. URL: `http://arxiv.org/abs/1703.00837`.

OpenAI (2018). *Finetune-Transformer-LM*. `https://github.com/openai/finetune-transformer-lm`. [Online; uploaded 11. June 2018; last accessed on 16. April 2019].

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning (2014). "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543. URL: `http://www.aclweb.org/anthology/D14-1162`.

Peters, Matthew E., Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer (2018). "Deep contextualized word representations". In: *CoRR* abs/1802.05365. arXiv: `1802.05365`. URL: `http://arxiv.org/abs/1802.05365`.

Radford, Alec, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever (2018). “Improving Language Understanding by Generative Pre-Training”. In:

Riedel, Sebastian, Limin Yao, and Andrew Mccallum (2010). “Modeling Relations and Their Mentions without Labeled Text”. In: pp. 148–163. DOI: `10.1007/978-3-642-15939-8_10`.

Santos, Cicero Nogueira dos, Bing Xiang, and Bowen Zhou (2015). “Classifying Relations by Ranking with Convolutional Neural Networks”. In: *CoRR* abs/1504.06580. arXiv: `1504.06580`. URL: `http://arxiv.org/abs/1504.06580`.

Sennrich, Rico, Barry Haddow, and Alexandra Birch (2015). “Neural Machine Translation of Rare Words with Subword Units”. In: *CoRR* abs/1508.07909. arXiv: `1508.07909`. URL: `http://arxiv.org/abs/1508.07909`.

Shen, Yatian and Xuanjing Huang (2016). “Attention-Based Convolutional Neural Network for Semantic Relation Extraction”. In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan: The COLING 2016 Organizing Committee, pp. 2526–2536. URL: `http://aclweb.org/anthology/C16-1238`.

Snell, Jake, Kevin Swersky, and Richard S. Zemel (2017). “Prototypical Networks for Few-shot Learning”. In: *CoRR* abs/1703.05175. arXiv: `1703.05175`. URL: `http://arxiv.org/abs/1703.05175`.

Socher, Richard, Brody Huval, Christopher D. Manning, and Andrew Y. Ng (2012). “Semantic Compositionality Through Recursive Matrix-vector Spaces”. In: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. EMNLP-CoNLL ’12. Jeju Island, Korea: Association for Computational Linguistics, pp. 1201–1211. URL: `http://dl.acm.org/citation.cfm?id=2390948.2391084`.

Tang, Gongbo, Rico Sennrich, and Joakim Nivre (2018). “An Analysis of Attention Mechanisms: The Case of Word Sense Disambiguation in Neural Machine Translation”. In: *CoRR* abs/1810.07595. arXiv: `1810.07595`. URL: `http://arxiv.org/abs/1810.07595`.

Toutanova, Kristina, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon (2015). “Representing Text for Joint Embedding of Text and Knowledge Bases”. In: *EMNLP*.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin (2017). “Attention Is All You Need”. In: *CoRR* abs/1706.03762. arXiv: `1706.03762`. URL: `http://arxiv.org/abs/1706.03762`.

Verga, Patrick, Emma Strubell, and Andrew McCallum (2018). “Simultaneously Self-Attending to All Mentions for Full-Abstract Biological Relation Extraction”. In: *CoRR* abs / 1802.10569. arXiv: `1802.10569`. URL: `http://arxiv.org/abs/1802.10569`.

Vinyals, Oriol, Charles Blundell, Timothy P. Lillicrap, Koray Kavukcuoglu, and Daan Wierstra (2016). “Matching Networks for One Shot Learning”. In: *CoRR* abs / 1606.04080. arXiv: 1606.04080. URL: http://arxiv.org/abs/1606.04080.

Wang, C., A. Kalyanpur, J. Fan, B. K. Boguraev, and D. C. Gondek (2012). “Relation Extraction and Scoring in DeepQA”. In: *IBM J. Res. Dev.* 56.3, pp. 339–350. ISSN: 0018-8646. DOI: 10.1147/JRD.2012.2187239. URL: http://dx.doi.org/10.1147/JRD.2012.2187239.

Wang, Chang and James Fan (2014). *Medical Relation Extraction with Manifold Models*.

Wang, Haoyu, Ming Tan, Mo Yu, Shiyu Chang, Dakuo Wang, Kun Xu, Xiaoxiao Guo, and Saloni Potdar (2019). “Extracting Multiple-Relations in One-Pass with Pre-Trained Transformers”. In: *CoRR* abs/1902.01030. arXiv: 1902.01030. URL: http://arxiv.org/abs/1902.01030.

Wang, Linlin, Zhu Cao, Gerard de Melo, and Zhiyuan Liu (2016). “Relation Classification via Multi-Level Attention CNNs”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 1298–1307. DOI: 10.18653/v1/P16-1123. URL: http://aclweb.org/anthology/P16-1123.

Wolf, Thomas (2018). *Training Neural Nets on Larger Batches: Practical Tips for 1-GPU, Multi-GPU & Distributed setups*. [Online; posted 25. October 2018; last accessed on 14. April 2019]. URL: https://link.medium.com/8LwNYLjFMV.

Xu, Yan, Ran Jia, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu, and Zhi Jin (2016). “Improved Relation Classification by Deep Recurrent Neural Networks with Data Augmentation”. In: *CoRR* abs/1601.03651. arXiv: 1601.03651. URL: http://arxiv.org/abs/1601.03651.

Xu, Yan, Ge Li, Lili Mou, and Yangyang Lu (2014). “Learning Non-Taxonomic Relations on Demand for Ontology Extension”. In: 24, pp. 1159–1175.

Xu, Yan, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin (2015). “Classifying Relations via Long Short Term Memory Networks along Shortest Dependency Path”. In: *CoRR* abs/1508.03720. arXiv: 1508.03720. URL: http://arxiv.org/abs/1508.03720.

Yin, Wenpeng, Katharina Kann, Mo Yu, and Hinrich Schütze (2017). “Comparative Study of CNN and RNN for Natural Language Processing”. In: *CoRR* abs/1702.01923. arXiv: 1702.01923. URL: http://arxiv.org/abs/1702.01923.

Zeiler, Matthew D. and Rob Fergus (2013). “Visualizing and Understanding Convolutional Networks”. In: *CoRR* abs/1311.2901. arXiv: 1311.2901. URL: http://arxiv.org/abs/1311.2901.

Zeng, Daojian, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao (2014a). “Relation Classification via Convolutional Deep Neural Network”. In: *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*.

Dublin, Ireland: Dublin City University and Association for Computational Linguistics, pp. 2335–2344. URL: http://aclweb.org/anthology/C14-1220.

Zeng, Daojian, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao (2014b). "Relation Classification via Convolutional Deep Neural Network". In: *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin, Ireland: Dublin City University and Association for Computational Linguistics, pp. 2335–2344. URL: http://www.aclweb.org/anthology/C14-1220.

Zhang, Dongxu and Dong Wang (2015). "Relation Classification via Recurrent Neural Network". In: *CoRR* abs/1508.01006. arXiv: 1508.01006. URL: http://arxiv.org/abs/1508.01006.

Zhang, Yuhao, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning (2017). "Position-aware Attention and Supervised Data Improve Slot Filling". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, pp. 35–45. DOI: 10.18653/v1/D17-1004. URL: http://aclweb.org/anthology/D17-1004.

Zheng, Suncong, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu (2017). "Joint Extraction of Entities and Relations Based on a Novel Tagging Scheme". In: *CoRR* abs/1706.05075. arXiv: 1706.05075. URL: http://arxiv.org/abs/1706.05075.

Zhou, Peng, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu (2016). "Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification". In: *ACL*.

Zhou, Peng, Suncong Zheng, Jiaming Xu, Zhenyu Qi, Hongyun Bao, and Bo Xu (2017). "Joint Extraction of Multiple Relations and Entities by Using a Hybrid Neural Network". In: pp. 135–146. ISBN: 978-3-319-69004-9. DOI: 10.1007/978-3-319-69005-6_12.

Zhu, Yukun, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler (2015). "Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books". In: *CoRR* abs/1506.06724. arXiv: 1506.06724. URL: http://arxiv.org/abs/1506.06724.

Appendices

A Text Generation

240.000, gumbel completion Prager Schinken (auch: Prager Hermmann , Pragertrics , Pragerisch , Pragertricsk) sind die Ausstellungsreihen der böhmischen Speiseeis - und Nahrungsmittelindustrie. Die Märkte und Gastronomieketten im Zentrum von Prag stehen in Konkurrenz mit Partnern , die sich über die Tätigkeit in Osteuropa unterhalten. Neben tschechischen Wienern ergänzen auch tschechische Hersteller gemeinsam mit dem russischen "Sowason" und anderen in den Wiener Ortsteil Penzing - hier befinden sich Wein - und Küchenbetriebe - den Großteil des Angebotes dieser Art in Deutschland.
920.000, softmax completion Siegfried Echterhoff (* 1960 in Neuenkirchen) ist ein deutscher Mathematiker. Er ist seit 2014 Professor am Amt für Mathematik und Naturwissenschaften am Institut für Mathematik der Universität Erlangen - Nürnberg. Siegfried Echterhoff ist ein Sohn des Psychologen Heinz Echterhoff und dessen Ehefrau Luise geb. Hentzer. Er war ab 1982 an der Leibniz Universität Hannover tätig und seit 1986 bei der Straßenbaudirektion Nordrhein - Westfalen in Münster , später an der Bergischen Universität Wuppertal. Bis 1997 war er Zeichenlehrer an dem Born Gymnasium Leverkusen und absolvierte 1997 - 2001 ein Studium im Masterstudium der Mathematik , Physik und...
293.000, softmax completion Misael Vacca Ramírez (* 5. November 1990 in Cointree County) ist eine US - amerikanische Tennisspielerin. Ramírez spielt hauptsächlich auf dem ITF Women's Circuit , wo sie bis jetzt acht Einzel - und zwölf Doppeltitel gewann. Ihre besten Weltranglistenpositionen erreichte sie im September 2013 in Luxemburg und seither oft im Doppel. Vacca Ramírez spielte 2014 und 2015 im Einzel sowie 2016 im Doppel. Ihre beste Einzelplatzierung in der Tennis - Weltrangliste erreichte sie im Oktober 2015 mit Platz 343...
1.484.000, softmax random Außerdem entstanden im Laufe der Zeit weitere Manufakturen und große Fabriken in den umliegenden Dörfern, wie 1863 die Glaswerke Lichtenfels im Zeiselseler Stadtgebiet. Der bekannteste Reliefgestaltungsversuch bei der Anlage von Steinbrüchen ist ein Areal in Minghua (Nanna-dong) am Standort Südhang des Donduping-Sees im Gebiet rund um Minghua (Donduping), USA, der 1876 eröffnet wurde. In der Nachkriegszeit wurde die große Produktion nach Langfang...
1.444.000, softmax random Na gut ! aber das ist doch der einzige Grund dafür , warum diese Aussage auf Wikipedia stehen darf. Du weißt , dass bist Du ja , hast mehr Erfahrung mit der Materie ; -) Was genau meinst Du damit. Ich du sage doch , hast Du da viel zu tun ? Was ich vor einiger Zeit sagte: Sie sollen üben , wie gesagt , nicht solche Themen wo es dieselben Unterschiede gibt , oder ??? Ich weiß nicht , ob du richtig meldest , aber vielleicht sollten wir mal aufpassen ob es...

Table A.1: German text generation examples.

B Clustering

...his intervention did not cause the	increase	in police on site...
...contributing factor for the rapid	rise	of the free software...
...impact or contribution towards the	advancement	of the Chinese American...
...here have also been layoffs and	reduction	in pages ...
The animals most susceptible to the large	decrease	in water temperature include...
...whose despotic rule clearly marked the	decline	of the empire...

Table B.1: Synonyms and antonyms of "increase" (layer 6, 38 assignments).

...Quarry at the small mesa known as Table	Rock	.
...into the downtown haver	Hill	area, Route 125 crosses...
energy recommended approval of Yucca	Mountain	for development...
...North London adjacent to Hornsey, Muswell	hill	and wood green .
Tehama and the other volcanoes near Lassen	Peak	were produced by subduction...
toward glam and progressive	rock	, and he was particularly
The song was nominated for best hard	rock	performance during...
...Motown sound and psychedelic	rock	sound resulted in...
appointed him editor of the	music	journal he had founded
...but eventually transformed into a blues	recording	.

Table B.2: Layer 8: The word "rock" as part of a mountain's name and related words (top, 330 assignments) and "rock" as a musical genre, again clustered together with related words (bottom, 620 assignments). Note that the English GP-Transformer is case-insensitive, so "rock" is solely clustered based on the word itself and the preceding context.

...extended north in 1904 making its new	address	64-70 broadway.
...have been spammed on my email	address	that links to my user page...
contributor to wikipedia 's IP	address	to his username.
...this tactic was used to	address	a variety of political themes...
...time adjusting the steering to	address	a problem of tyre scuffing...
hopefully someone will rise to	address	the issue.

Table B.3: Two clusters in which the homonym "address" is placed in layer 6. Address as a location (top, 36 assignments) and as in "speak to" (bottom, 66 assignments).

...elevation of 264 feet (80m) above	mean	sea level.
with an area of, a	mean	depth of, and a volume...
mean that in spite of the	mean	temperature of the warmest
Just because gods sent the signs did n't	mean	that mesopotamians believed
However, this did not	mean	that she retired from public life
...needles can be marketed and does not	mean	that acupuncture is the...

Table B.4: Clusters of layer 6 that contain the word "mean" in the sense of averaging something (top, 23 assignments) and as implicating something (bottom, 4 assignments).

Er war Herausgeber des Archivs für	Mathematik	und Physik und Mitarbeiter.
...studierte er an der Universität Cambridge	Theologie	und wurde nach...
...im März 1587 zum Doktor der	Medizin	promoviert wurde.
...Verdienste hatte er sich auf dem Gebiet der	Anatomie	erworben, als er...
Schumacher studierte nach dem Abitur	Architektur	an der Technischen...

Table B.5: Clusters of fields of work/study obtained by the German GP-Transformer (layer 8, 229 assignments).

Sie wurde	2002	wiederentdeckt und ist äußerst selten.
Im Dezember	2006	wurde vom Flughafen Dresden aus Atommüll...
Im Dezember	2006	wurde vom Flughafen Dresden aus Atommüll...
Seit September	2002	nutzt das Maxim-Gorki-Theater einen etwa 80 Meter...
Seit	1992	ist die Stadt wieder Sitz des Landesamtes...
Von Figura war von	2005	bis 2010 Präsident der Georg-August- Universität zu Göttingen.
Im Frühjahr	2005	gewann er die Poreč Trophy in Kroatien.

Table B.6: Year numbers occurring mostly in the beginning of sentences (layer 5, 391 assignments).

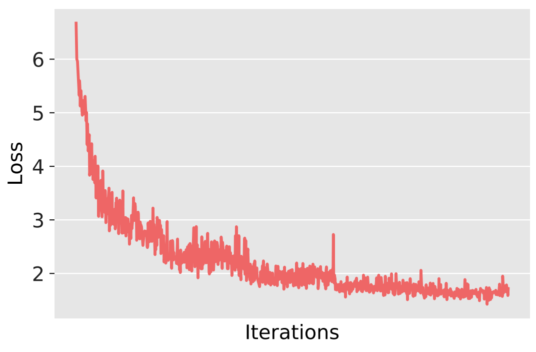
Bei der Sanierung des Mutterer Tunnels und der	Mutter	er Brücke (Mühlgrabenviadukt)...
...Haltestelle Burgstall passierend, zum Bahnhof	Mutter	s.
...frequentiert wird, da sie die Talstation der	Mutter	eralmbahn gut erschließt.
...mit seinen Brüdern und seiner	Mutter	Anna Wimmer in der Kinobranche...
...Alexander Ogarkow wird von seiner	Mutter	Ljudmila Ogarkowa...
Nach dem Krieg arbeitete die	Mutter	als Schneiderin und Verkäuferin...

Table B.7: Two different clusters (layer 9), containing the term "Mutter" (mother) as part of the Austrian town "Mutters" (top, 425 assignments) and as a mother of children (bottom, 590 assignments).

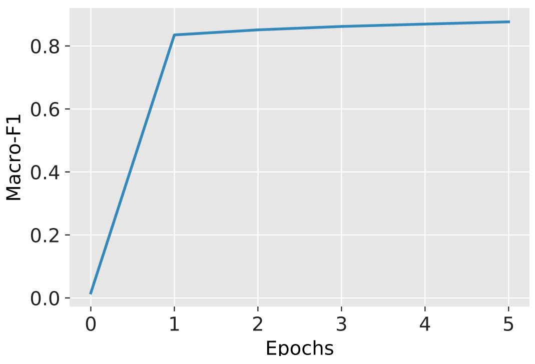
4 Taler Pacht jährlich und 6 Groschen	Steuer	an die Gemeinde...
das Rittergut Walda um einen	Steuer	erlass für die...
Erträgen ihres Eigentums und den	Steuer	einnahmen als Inhaber...
...wurde im Verlauf des Tages auf der	Steuer	bordseite montiert.
...dieses System eine getrennte	Steuer	- und Hauptbremsleitung ...
...konnte sich auch zeitweise hinter das	Steuer	eines Rennwagens setzen.

Table B.8: Two clusters of layer 7, containing the word "Steuer" mostly in the "tax" sense (top, 1444 assignments of mostly related words) and mostly in the "controller" sense (bottom, 53 assignments)

C Loss and Prediction Plots

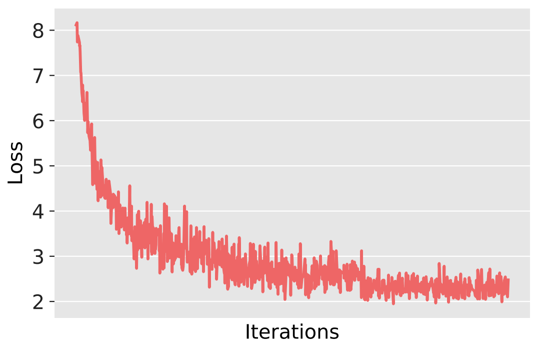


(a) Training Loss

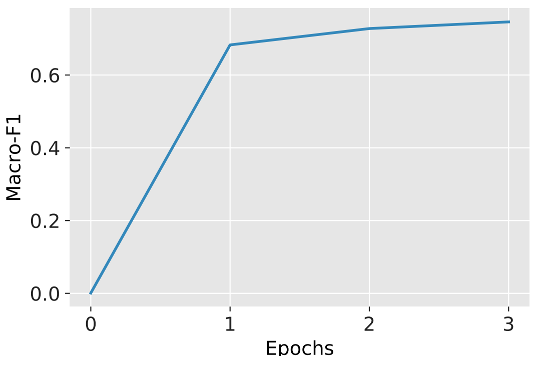


(b) Test Official-Macro-F1

Figure C.1: Relation classification trainings loss (left, averaged over each batch) and test set official macro-F1 (right) of the SemEval dataset, plotted after each epoch.

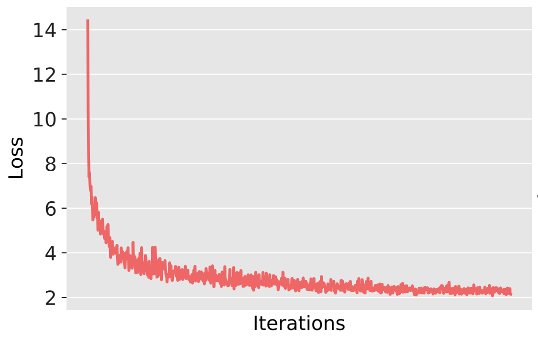


(a) Training Loss

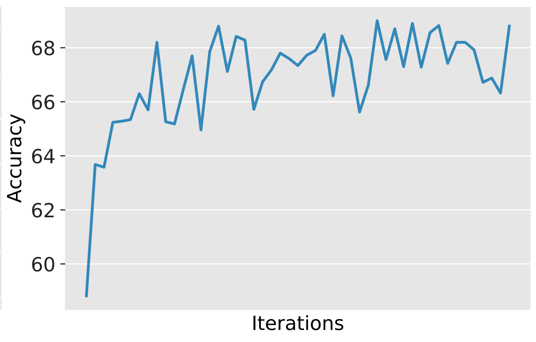


(b) Test Macro-F1

Figure C.2: Relation classification trainings loss (left, averaged over each batch) and test set macro-F1 (right) of the FewRel dataset, plotted after each epoch.

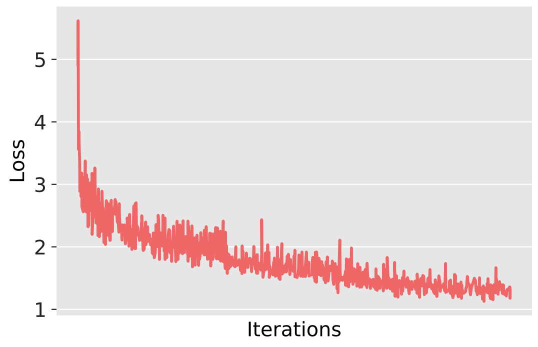


(a) Training Loss

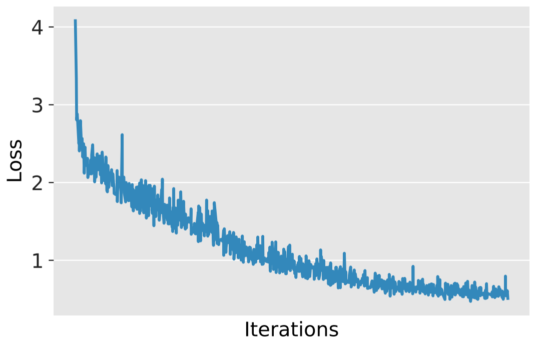


(b) Validation accuracy

Figure C.3: Few-Shot relation classification (10-way 1-shot setting) training loss over iterations, averaged over the respective batch, and validation set accuracy (right), measured over 100 episodes at each step.



(a) FewRel



(b) SemEval

Figure C.4: Joint entity and relation extraction training loss. FewRel dataset (left) and SemEval dataset (right), plotted over iterations and averaged over the respective batch.

D SemEval Scores

Model	M-F1
Entity-Aware BERT$_{SP}$ (H. Wang et al. 2019)	89.0
Multi-Level Attention (L. Wang et al. 2016)	88.0
GPT + EI (**own**)	87.34
TRE (Alt, Hübner, and Hennig 2019)	87.1
SDP deep RNN (Xu, Jia, et al. 2016)	86.10
Attention CNN (Shen and Huang 2016)	85.9
Tree+Seq RNN (Miwa and Bansal 2016)	85.50
Entity Attention Bi-LSTM (Lee, Seo, and Choi 2019)	85.2
CR-CNN (Santos et al. 2015)	84.10
SDP-LSTM (Xu, Mou, et al. 2015)	83.70
DE-CNN (Zeng et al. 2014b)	82.70
MV Tree-RNN (Socher et al. 2012)	82.40

Table D.1: Relation classification results on the SemEval dataset (Hendrickx et al. 2010).