University of Zurich^UZH

Knowledge Graph Driven Text Generation Using Transformers

Master Thesis
September 30, 2020

David Lay
of Mainz MZ, Germany

Student-ID: 14-707-939
david.lay@uzh.ch

Advisor: **Matthias Baumgartner**

Prof. Abraham Bernstein, PhD
Institut für Informatik
Universität Zürich
https://www.ifi.uzh.ch/ddis

Acknowledgements

I would like to express my very great appreciation to my advisor Matthias Baumgartner for his extraordinary guidance throughout the process of this thesis. Without his patience, thoughtful inputs and critiques, the objectives of this thesis could not have been realized.

Additionally, I want to show my gratitude to Professor Abraham Bernstein for accepting me as one of his students allowing me to write this thesis at the Dynamic and Distributed Information Systems Group of the University of Zurich.

At last I want to thank all my family and friends who helped me focusing on this thesis for the last six months and keeping me on track during this extraordinary time.

Zusammenfassung

Die Informationsgewinnung aus Knowledge Graphen ist aufgrund der Semantik und Komplexität für ungeübte Benutzer schwierig. Um dies zu vereinfachen, untersuchen wir die automatisierte Erstellung von Sätzen, basierend auf einer Sequenz von Knowledge Graph Entitäten. Während in früheren Arbeiten hauptsächlich Vorlagen oder spezialisierte Encoder-Decoder-Modelle trainiert wurden, konzentrieren wir uns auf die Verwendung von Transformern und bereits vortrainierten Modellen. Die Daten, die für das Training dieser Modelle verwendet werden, entspringen dem frei verfügbaren T-REx Datenset. Dieses verbindet Wikidata Entitäten, inklusive deren Labels und IDs, mit Wikipediaartikeln. Basierend auf diesen Daten untersuchen wir über 60 Modelle auf ihre Fähigkeit der Satzgenerierung. Zudem wird der Einfluss einer Inputsequenz, basierend auf Wikidata IDs anstelle von Wikidata Labels, untersucht. Unsere Resultate zeigen, dass vortrainierte Modelle unsere eigens trainierten Modelle übertreffen. Ebenfalls ist das Erzeugen natürlicher Sätze basierend auf Input IDs deutlich schwieriger als jenes basierend auf einer Sequenz aus Labeln.

Abstract

Understanding the semantics and interpreting the information inside a knowledge graph is challenging for an untrained user. To ease the access to this knowledge, we investigate how natural language-like sentences can be generated from a sequence of knowledge graph entities and relations between them. Whereas early work is based on template-like architectures or specialized encoder-decoder architectures, this work focuses on the use of Transformers and large pretrained language models. To deal with real-world knowledge graphs and text across many different domains we incorporate the T-REx dataset aligning Wikidata entities and relations with Wikipedia articles. We compare the performance between baseline models and finetuned large pretrained language models on the task of generating Wikipedia alike sentences. Additionally, we show the impact of using an input sequence of Wikidata IDs over an input sequence of the corresponding labels. By training over 60 different model configurations, we do an exhaustive parameter search to investigate our models. Results suggest that finetuning a pretrained language model outperforms the trained baseline model with respect to generating natural language-like sentences. Furthermore, we show that training using entity IDs over their respective labels requires task-specific adaptions with which the proposed models have difficulties.

Contents

1

Introduction

Since the introduction of the RDF standard in 2004 and Google's Knowledge Graph in 2012, several knowldege graphs have emerged, including but not limited to DBPedia, Wikidata, Freebase, and Google's Knowledge Vault. Even though architectures between knowledge graphs differ, the essence of having data modeled as a subject-predicate-object triple can be seen across implementations. Subject and object in this context describe real-world objects (entities) and the predicate describes the relationship between these entities. Having many triples in a single knowledge graph thus results in an extensive network of facts. The public can use the factual knowledge of such a graph for fast and precise knowledge retrieval about real-world objects and their relations.

But even though knowledge graphs are defined by precise terms and a well-defined structure, they are hard to interpret for untrained users. Thus, understanding the semantics of a knowledge graph becomes a challenge for the untrained user, which leads to a limited group of users having direct access to the knowledge hidden inside a knowledge graph itself.

To deal with this issue, many knowledge graphs include human-readable labels or short descriptions of entities and their relations. However, creating textual components is time consuming so that many entities and larger graph structures do not contain any textual descriptions. Which in turn increases the barrier for the public to use the data stored inside a knowledge graph. To achieve this on a large scale demands methods that can solve this objective automatically.

The task of creating natural language-like descriptions from graph data has been approached from different perspectives over the years. Early work mainly focused on completing handcrafted or learned templates with entity information based on an OWL ontology [Bernstein and Kaufmann, 2006, Galanis and Androutsopoulos, 2007, Third et al., 2011, Stevens et al., 2011]. This template-based approach shifted as new methods of dealing with natural language-like tasks emerged. Thus, research started focusing on machine learning tools such as encoder-decoder strategies to construct short, single-sentence summaries, and entity descriptions [Lebret et al., 2016, Kaffee et al., 2018, Wiseman et al., 2018].

With the recent development of Transformer [Vaswani et al., 2017] based architectures such as BERT [Devlin et al., 2018], research focused on the use of pretrained models. These models are pretrained on vast amounts of unlabeled data and have been shown to improve model performance across different domains, e.g. in combination with images or knowledge graphs [Qi et al., 2020, Yao et al., 2019]. Having such a system in place opens up the use for new applications that can generate human-readable knowledge based on available graph structures or given input sequences. Combined with the knowledge graph's truthfulness, the generated sentences can represent real-world data that is not limited to already explicitly documented knowledge.

Wikidata Labels	Barack Obama spouse Michelle Obama occupation lawyer
Wikidata IDs	Q76 P26 Q13133 P106 Q40348
Target Sentence	Barack Obama is married to his wife Michelle who is working as a lawyer.

Table 1.1: Example task of creating a natural language-like sentence on a given sequence of Wikidata labels or IDs

In this thesis, we investigate how sentences can be generated from a knowledge graph sequence by using Transformer-based architectures and pretrained language models. More precisely, we focus on the data-to-text generation for a given sequence of Wikidata entities and their connecting relations. This sequence does not necessarily need to be a graph walk in Wikidata but can also be a sequence consisting of entities having no relations in the graph. An example sequence of Wikidata entity labels with their associated IDs are shown in Table 1.1. The corresponding target sentence is the objective our models are trained on. We follow our goal by trying to understand the differences in the generation between the training of a baseline model and the finetuning of a large pretrained language model, namely BART [Lewis et al., 2019]. Additionally, we compare the natural language-like sentence generation given two different input sequences: IDs and labels. Entity labels and entity IDs either require specific model modifications to be made to BART or the training of a whole new model. We hypothesize that pretrained language models can outperform baseline models on our task and that using entity IDs as an input source can leverage the results of a label-based approach.

Following our goals, we train different models depending on the given task and conduct exhaustive parameter searches over the models. Overall, we train more than 60 models based on six different architectures. Our training data comes from the publicly available T-REx dataset containing 6.2 million Wikipedia sentences aligned with the corresponding Wikidata entities and predicates. Processing the dataset into input sequences associated with their target sentence leaves us with over 4 million sequence-sentence pairs across many different Wikipedia domains.

Results show that the task of generating natural language-like sentences based on labels can be achieved by baseline models, but are highly outperformed by a task-specific finetuned BART model. Manual inspections indicate that the produced sentences are indeed human-readable and, due to the training, very Wikipedia alike. Training the models based on available Wikidata labels allows us to use already pretrained embedding layers. However, we notice that this embedding layer is hard to train from scratch and becomes especially harder when training with Wikidata IDs, since the input dimension of such a training grows with each new entity.

The remainder of this Thesis presents the implemented and conducted experiments in more detail. Chapter 2 describes the related work and how it adapted to new methods and models over time. It is followed by more detailed background information and explanations of model architectures in Chapter 3. The implemented models can be found in Chapter 4 and the experimental setup of each model is described in Chapter 5. In Chapter 6 the results are analyzed both quantitatively and qualitatively. Finally, we conclude our work in Chapter 7 and make mentions of future work.

2

Related Work

Generating natural language-like descriptions or single sentences given graph data is well known, but methods have changed as architectures improved over the years. While earlier approaches focused on implementing template-based architectures and neural models trained from scratch, more recent approaches try to incorporate already learned knowledge into their models. While most work focuses on selecting facts from a single entity and generating natural language-like descriptions over these facts, this work concentrates on the generation given an already specified sequence of entities.

2.1 Template-Based Approaches

Early template-based approaches can be domain-related implementations designed to solve a specific problem. CORAL [Dale et al., 2003] can generate natural language-like descriptions for navigational assistance, whereas SUMTIME-MOUSAM [Sripada et al., 2003] is able to produce textual weather forecasts given a weather data time series.

Trying to close the gap for novice users to utilize the semantic webs knowledge GINO (guided input natural language ontology) was introduced by [Bernstein and Kaufmann, 2006]. GINO provides users with an editor that allows the querying and editing of an OWL ontology using natural language. That means that the tool can specify, complete, and parse sentences to modify its ontology. To do so, a basic grammar consisting of around 120 rules is extended with the underlying ontology's structure and vocabulary.

NaturalOWL introduced by [Galanis and Androutsopoulos, 2007] is based on ILEX [O'DONNELL et al., 2001] and M-PIRO [Androutsopoulos et al., 2001] and can generate natural language-like descriptions for single OWL classes in a restricted domain. It further implements the generation in English and Greek and is based on a four-step process: content selection, document planning, microplanning, and surface realization. Whereas content selection chooses the most important facts about a target class, document planning orders the selected facts to build the document structure. Microplanning then creates sentences using templates, including slots used as placeholders. The slots are filled with verb and noun phrases defined and provided by the OWL ontology or in separate files. Sentences are finally aggregated into a single natural language-like text using a domain-independent grammar. Additional words come from the domain-specific vocabulary.

Similar to NaturalOWL [Stevens et al., 2011], suggest a prototype that is able to generate textual definitions for OWL classes. Their approach aims to improve the fluency of the output without the need for additional user input. Like NaturalOWL the prototype collects axioms concerning one class, creates a sentence for each group, and merges these sentences to a final paragraph. An additional conducted survey shows that while the created paragraphs are acceptable, they cannot replace human written definitions.

[Third et al., 2011] provide a tool to verbalize OWL ontologies implemented in the SWAT project. Compared to previous approaches, they do not generate a definition for an OWL class but rather generate sentences for each class's logical axiom. [Third et al., 2011]. Again, the process follows a clear template: lexicon construction followed by document structuring and verbalization using a fixed grammar. This fixed grammar is then extended with the constructed lexicon. The evaluation suggests that tasks on the verbalized ontology were preferred and are found to be easier to complete [Third et al., 2011].

2.2 Neural Approaches

With the success of neural-language models such as Recurrent Neural Networks the focus of generating natural language-like text based on structured data shifted towards a neural approach. These neural approaches are comparable to the approach followed in this thesis.

[Lebret et al., 2016] implement a neural-language model able to generate natural language-like text based on Wikipedia infoboxes. To follow their goal, they additionally introduce the WIKIBIO dataset consisting of over 700'000 Wikipedia articles. All articles are in the biography domain and contain a Wikipedia infobox aligned with the Wikipedia article's first sentence. The first sentence is then predicted by a conditional neural-language model using local and global conditioning mechanisms. These conditioning mechanisms allow the model to use knowledge from the infoboxes. Additionally, they use a copy mechanism passing words from the input infobox to the sentence without having the specific word in the predefined fixed vocabulary. Similar to most neural-language models both infoboxes and words are embedded and aggregated into a single embedding. As in our work, results are analyzed using BLEU and ROUGE scores. Their best model reaches a generation score of 34.7 BLEU and outperforms their baseline by around 15 BLEU.

Comparable to [Lebret et al., 2016], [Chisholm et al., 2017] introduce a model to generate one-sentence biographies. Instead of infoboxes the input is taken from slot-value pairs of Wikidata and aligned with the corresponding Wikipedia article's first sentence. The focus relies on the 15 most occurring facts in the biography domain. Furthermore, the authors train a recurrent neural network including GRUs and treat the task similar to a machine translation task using a shared vocabulary. In addition, a second objective is introduced. This objective aims to recreate the input facts, which has the effect of generating text consisting of the input facts instead of more randomly added facts Inspections show that the secondary objective indeed outperforms the simpler architecture by nearly 8 BLEU, having a total of 41 BLEU on the test set. Additional inspections show that sentences generated by the model are preferred over Wikipedia sentences by human judges.

[Vougiouklis et al., 2018] take the problem a step further and generate textual summaries based on entity triples. The triples are taken from either Wikidata or DBpedia and are aligned with the respective Wikipedia summary. Again, the focus lies on the biography domain only. To limit the vocabulary size, the Wikipedia summaries are modified to replace rare or unknown tokens with special placeholders. Similarly, input triples containing years are remodeled into multiple triples containing only the year or month. During postprocessing, these placeholders are then replaced again with the original input tokens. The model is based on an encoder-decoder architecture having a feed-forward neural network on the encoder side and a recurrent neural network implementing either an LSTM or GRU on the decoder site. Whereas the encoder transforms the input triples into a concatenated output vector, the decoder is responsible for generating the textual summary based on the encoder's output. Their generation during testing uses a beam search to optimize the final summaries. The model's performance is measured utilizing a combination of perplexity, BLEU, and ROUGE scores combined with human judgment. The best model based on GRUs reaches a BLEU of 41.5 and is comparable to human judgment.

Combining neural models with copy actions, [Kaffee et al., 2018] propose a model based on the one introduced by [Vougiouklis et al., 2018]. Its architecture is again composed of a feed-forward encoder and a recurrent neural network decoder based on GRUs. However, the model implements the copy mechanism similar to [Lebret et al., 2016]. During preprocessing, rare entities are replaced with *property placeholders* and are again replaced after generation. The placeholder represents a property between the subject of the sentence and the object, whereas the object is a rare entity and will be replaced with its label. This mechanism again allows the authors to keep the vocabulary smaller and focus on the most occurring tokens. They test their implementation of multiple languages by aligning the first sentence of a Wikipedia article with its Wikidata entry. Only triples where either the object or subject occur in the sentence are used as input. Final generations show strong results across multiple domains in the used languages and outperform the baseline models of the research.

Due to reasoning that the neural encoder-decoder models are hard to interpret and challenging to control, [Wiseman et al., 2018] propose a neural hidden semi Markov model decoder. The architecture is chosen to use attention mechanisms and LSTMs while still being based on a hidden semi Markov model. They train the model using objects resembling templates and on data from the WikiBio dataset by [Lebret et al., 2016] and the E2E dataset by [Novikova et al., 2017]. Learned templates then again allows to control the final generation and are easier to interpret than their neural counterparts. Results are comparable to neural models but cannot reach state-of-the-art results. In the WikiBio domain the proposed model reaches a score of 9 BLEU lower than the compared best model.

More recent work addresses shortcomings of neural text generation regarding the content selection and ordering of facts shown by [Wiseman et al., 2017]. [Puduppully et al., 2019] introduce a model where the textual summary is conditioned on a learned content plan. The approach follows a structure comparable to template-based methods in a sense that it can be split into three steps: content selection, content planning, and text generation. Content selection and planning build the model's encoder site, outputting the mentioned content plan as a vector. The decoder is then responsible for the text generation given the content plan of the encoder. On a high-level, this architecture is more interpretable and reduces the amount of repeated output [Puduppully et al., 2019]. Results on the ROTOWIRE dataset [Wiseman et al., 2017] show better performance on generation quality and order of facts in the output text compared to previous methods and baselines.

All work presented in this section can be compared to our work in a sense that it uses linearized facts to predict single sentences or summaries in a natural language-like text. Some work relies on infoboxes, whereas other work uses knowledge base triples as input. However, all of the models shown have to select and order the content to verbalize the input before text generation. Our approach does not need these first two steps. This thesis' models expect an already selected and ordered input and generate text-based on this input. Instead, our model is not limited to a single domain such as the WikiBio domain but is trained on all of Wikipedia.

2.3 Transfer Learning

The Transformer architecture [Vaswani et al., 2017], which only relies on attention mechanisms and dispenses recurrent layers, reached state-of-the-art results in machine translation tasks. Transferring learned data from one task to another is crucial for deep learning tasks that lack annotated data. Multiple models trained on huge amounts of data improving the natural language understanding and generation exist throughout literature.

[Radford et al., 2018] improve the transfer learning from a word-level domain to a domain where supervised finetuning can be used for downstream tasks. Their generative pretrained model (GPT) is a unidirectional (left-to-right) Transformer based architecture trained on massive amounts of unlabeled data. The learned parameters of the pretrained model resemble a broad language understanding. Task-specific input adaptions allow the finetuning of the pretrained parameters on downstream tasks resulting in state-of-the-art results in 9 out of 12 tasks.

Arguing that the limitation of language models such as GPT is their unidirectionality, [Devlin et al., 2018] introduce BERT (Bidirectional Encoder Representations from Transformers). Using a masked language model (similar to the Cloze task), the pretrained BERT model can embed the left and right context together by predicting masked tokens during pretraining. BERTs architecture allows the finetuning of multiple downstream tasks with minimal architectural changes. These downstream tasks can include question answering or single sentence prediction and reach state-of-the-art results on eleven NLP tasks. Yet it cannot easily be used for natural language generation.

To deal with the lack of BERT's effectiveness for natural language generation tasks, [Lewis et al., 2019] present BART in (2019). Compared to BERT's masked language model approach, BART's pretraining is composed of a text noising stage and a reconstruction stage. During the noising stage, the input to BART is mixed using different noising functions. BART then learns the reconstruction of the noised input. They combine the auto-regression of GPT with the bidirectionality of BERT and show that BARTs performance can outmatch previous models in natural language understanding tasks. But especially in natural language generation tasks.

Most recently and similar to our work, [Ribeiro et al., 2020] explore the knowledge-graph-to-text generation using pretrained language models. The authors compare the generation of a previously finetuned BART and T5 [Raffel et al., 2019] model on the WebNLG dataset. Their work shows that even though the graph structure of the input is not needed directly, the pretrained model and its contained knowledge outperform specifically trained graph-to-text models. Since the knowledge-graph-to-text task is not the same as our task, they prepare the dataset differently by prepending special tokens to each entity and relation of the input sequence. Furthermore, they are limited to using the WebNLG dataset labels and thus focus only on the generation from labels to text, whereas our work also investigates the learning based on entity IDs.

3

Background

This chapter describes some essential background aspects used in this thesis in more detail. First, we state the formal definition of a knowledge graph and give an introduction to Wikidata. We further describe the Transformer architecture and its training method in depth. Based upon the Transformer, we illustrate BERT as well as BART, both being pretrained language models.

3.1 Knowledge Graphs

The definition of a knowledge graph differs across research and is often confused with knowledge bases. [Ehrlinger and Wöß, 2016] compare different definitions of knowledge graphs and related architectures. They define a knowledge graph as follows:

> *A knowledge graph acquires and integrates information into an ontology and applies a reasoner to derive new knowledge.*

This definition requires two concepts to be explained in more detail. (1) A knowledge base (e.g. ontology) being a pure semantic storage incorporating different kinds of information such as rules or facts about its world. In addition, a knowledge base can also hold instances separating knowledge bases from being database schemes [Ehrlinger and Wöß, 2016]. (2) The addition of a reasoner that can automatically collect, extract, and integrate new knowledge into an already existing knowledge graph. Thus, a reasoner can be seen as a tool or software that allows the computer to conduct automated analysis using a set of inference rules [Mishra and Kumar, 2011].

Additionally, the concept of having linked data modeled in a subject-predicate-object triple can be seen across different implementations and definitions. In such a triple, the subject and object refer to real-world concepts. The predicate describes the relationship between these two concepts.

One of the most famous knowledge graphs is Google's *Knowledge Graph* introduced in 2012. It is built upon freebase and further tuned on users' search queries containing more than 500 million objects and facts connecting these objects [Singhal, 2012]. This knowledge enhances Google's search engine from matching queries to matching real-world objects like people or electronic devices.

This thesis focuses on the use of Wikidata[1]. Wikidata, by its definition, is a free and open knowledge base containing around 89 million entities and more than 1 billion statements about these entities[2]. Like its sister project Wikipedia, Wikidata is edited and maintained collaboratively but is also extended by many bots. Due to these automatic extensions and its available reasoner[3] Wikidata can be seen as more than just a knowledge base. By the previously given definition, Wikidata can be interpreted as a knowledge graph incorporating a huge amount of real-world knowledge.

In Wikidata the subject-predicate-object triple (s, p, o) is a combination of two entities that are linked via a specific predicate. The subject, as well as the object, are elements of all entities included in the knowledge graph. In the case of Wikidata, these entities resemble real-world instances like persons or buildings. Additionally, the object of the triple can become a single value (e.g. a date or proper names) instead of an entity. The predicate connecting these entities is also called relation or property and is an element of the property set. Linking entities over predicates results in a linked data format, displayed as a graph similar to Figure 3.1. In the graph, each entity is represented as a node, including its label (the title of the entity) and its QID (the unique ID of a Wikidata entity, starting with "Q"). Predicates are visualized as edges between the entities and include their label and property ID. Some entities can also contain a so-called qualifier (e.g. life expectancy of the United States of America) contextualizing the item.

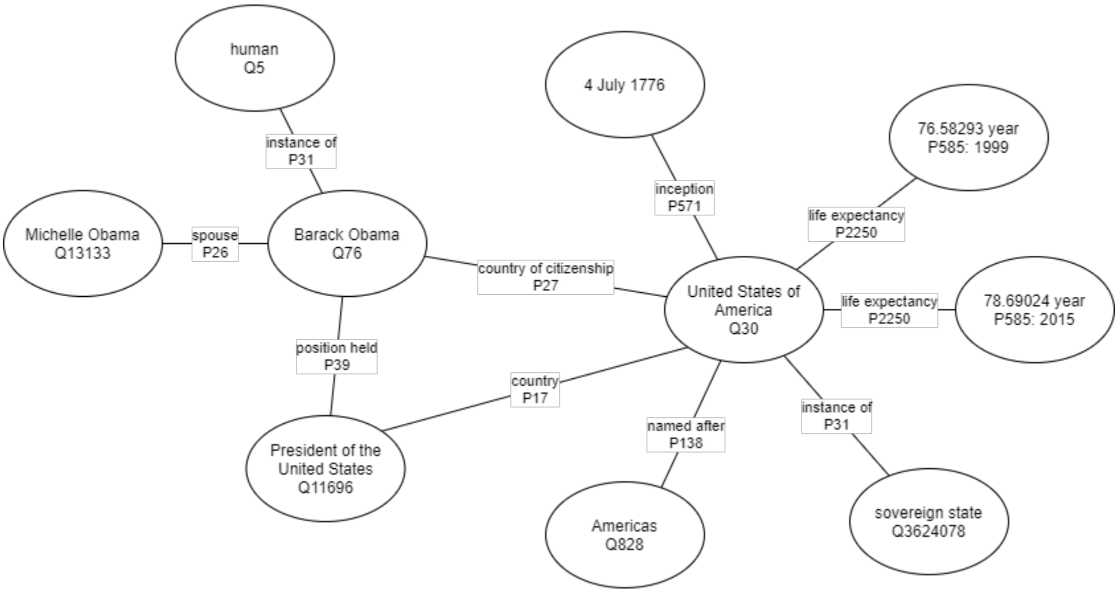


Figure 3.1: Extraction of a sample knowledge graph from Wikidata (not complete)

[1] *https://www.wikidata.org/wiki/Wikidata:Main_Page*
[2] *https://wikidata-todo.toolforge.org/stats.php*
[3] *https://reasonator.toolforge.org/*

3.2 Transformer

The Transformer was first introduced by [Vaswani et al., 2017] and replaced traditional sequence-to-sequence models' recurrence by its self-attention mechanism. Due to this replacement, machine translation improved not only in quality, but also in training time due to being highly parallelizable [Vaswani et al., 2017]. Similar to other natural language models, the Transformer is an encoder-decoder based architecture. Each the encoder and decoder consist of 6 layers having the same composition inside all layers. Whereas one encoder layer consists of two sub-layers, the decoder consists of three. This can be seen in Figure 3.2.

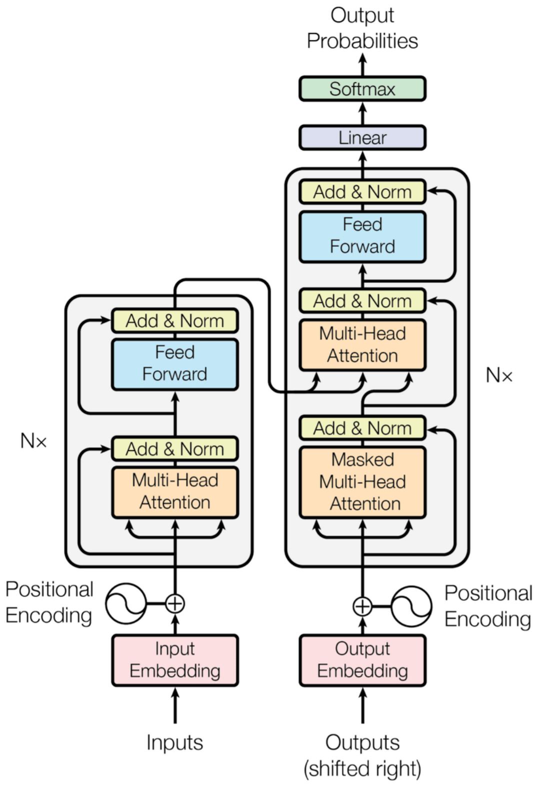


Figure 3.2: Basic Transformer architecture as introduced by [Vaswani et al., 2017]

Encoder To deal with the textual input sequence, the embedding layer learns to transform each word (or sub-word) into a vector of size d. This learned embedding is then added to a positional embedding of the same dimension helping the model compensate for the lack of recursion. Like the word embedding, the positional embedding can be learned but are fixed to a combination of sine and cosine functions in the original implementation.

Once the input embedding and positional embedding are summed up, they are fed to the multi-head attention layer of the encoder's first layer. The calculated self-attention can be seen as how the model tries to encode the relevance of surrounding words based on the currently processed word [Alammar, 2018]. Self-attention or originally *Scaled Dot-Product Attention* can be fully calculated with the use of optimized matrix multiplication code and is defined as [Vaswani et al., 2017]:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^{\mathrm{T}}}{\sqrt{d_k}})V \tag{3.1}$$

Where Q, K, and V are the query, key, and value matrices and are stacks of query, key, and value vectors respectively. d_k represents the dimensions of the query and key vectors. Thus, the self-attention is a weighted sum of the value vectors. To calculate Q, K, and V, the summed input embeddings are each multiplied with their own weight matrix $W^Q \in \mathbb{R}^{d \times d_k}$, $W^K \in \mathbb{R}^{d \times d_k}$, and $W^V \in \mathbb{R}^{d \times d_v}$.

The Transformer makes use of multi-head attention combining h different attention heads. Each head initializes its weight matrix differently so that the model is able to incorporate "different representation subspaces at different positions" [Vaswani et al., 2017]. During computation, attention is calculated for each head and the resulting matrices are concatenated. The concatenated matrix is multiplied with an additional weight matrix $W_O \in \mathbb{R}^{hd \times d_k}$.

Each sub-layer of the encoder implements a residual connection so that the input and output of each sub-layer can be normalized using layer normalization [Ba et al., 2016]. Next to the multi-head attention layer, an encoder layer implements a 2-layer, ReLU activated, and fully connected feed-forward network. This transformation can be applied to each input position in parallel but differs across multiple layers. The normalized output of one encoder layer is then used as the input for the next layer. Finally, the last encoder layer's output gets transformed into its key and value matrices used by the multi-head attention sub-layer of each decoder layer.

Decoder Like the encoder, the decoder uses an embedding and positional encoding layer to embed its input sequence. If a shared vocabulary is used, weights of the encoder and decoder embedding layer are shared as well. However, the input sequence needs to be shifted one position to the right to prevent the decoder from predicting its input tokens one by one. Compared to the encoder's multi-head attention layer, the decoder implements a masked multi-head attention layer, followed by a multi-head attention layer. Masked multi-head attention works comparable to multi-head attention. Since the decoder is only allowed to look at positions in front of its current position, right-ward content is masked. The mask is applied before the softmax calculation by setting the right-ward input to $-\infty$. The decoder's multi-head attention layer receives its key and value matrices from the encoder and its query matrix from its own attention layer. This allows the decoder to use all information from the encoded input sequence of the encoder as well as the left-ward input of the decoder to reason about the output.

Finally, the linear layer of the decoder transforms the output of the decoder's last stacked layer to a logits layer using the learned weights of the embedding layer. This results in a vector of the same size as the input vocabulary containing each word's prediction scores. These scores are turned into probabilities using a softmax layer so that all scores sum to one.

By virtue of to the Transformer's architecture, training can be completed in parallel. Originally the Transformer uses the Adam optimizer [Kingma and Ba, 2014] with an increasing learning rate before the warmup phase and a decreasing learning rate after. The loss is calculated using cross-entropy over the true and the predicted probability distributions.

3.3 Transfer Learning

BERT BERT (Bidirectional Encoder Representations from Transformers) is a language representation model that allows finetuning on multiple NLP tasks thanks to intensive pretraining. It improves upon the restrictions of unidirectionality of previous models like GPT by training deep bidirectional representations. The architecture is based on a stack of multiple Transformer Encoder layers and is further defined by the number of attention heads as well as the number of hidden dimensions. In its large implementation (24 encoder layers, 16 attention heads, 1024 hidden dimensions), BERT reaches 340 million parameters in total. The large model was pretrained over four days on 16 Cloud TPUs [Devlin et al., 2018]. The full pretraining architecture can be seen in Figure 3.3.

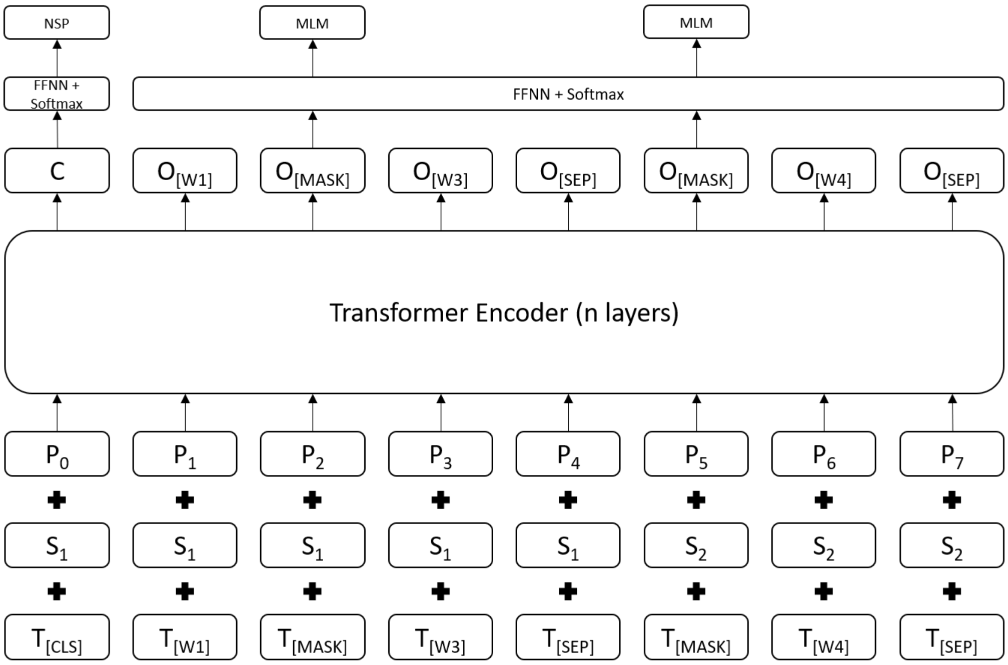


Figure 3.3: BERTs architecture including the input transformations and pretraining tasks [Devlin et al., 2018]

BERT's pretraining input sequence can either be a single sentence or a combination of two sentences. A [CLS] token in the beginning and [SEP] tokens at the end of the sequence are added to the original input. Same as in the original Transformer, the input sequence is then embedded using token and positional embeddings (T and P in Figure 3.3). Additionally, a learned sentence embedding (S in Figure 3.3) is added, which only differs between tokens of different sentences. The final embedding of each input token, including special tokens, is then fed into the first layer of BERT's encoder.

BERT is pretrained in two unsupervised tasks: Masked Language Model (MLM) and Next Sentence Prediction (NSP). In the first task, 15% of all input tokens of each sequence are replaced with the special [MASK] token. Only masking the randomly chosen token would lead to lower performance of the finetuned model, since the [MASK] token does not appear during finetuning. Thus, the chosen token is only replaced by the [MASK] token in 80% of all cases. In 10% it is replaced by another randomly chosen token and not replaced at all in the remaining 10%. The last layer's final output is then used to calculate the cross-entropy loss for the chosen tokens. Masking individual tokens essentially enables BERT to do a bidirectional pretraining since the masked tokens are predicted using the left and right context.

In the Next Sentence Prediction task, BERT tries to learn whether a sentence is indeed followed by another sentence and therefore can learn the relationship between the two sentences. To follow the task, BERT's input consists of 50% true sentence pairs (sentence 2 actually follows sentence 1) and 50% false sentence pairs. The [CLS] token's output is then again used to calculate the next sentence probability using a simple classification layer. Different from the original Transformer, GeLU activation is used over the ReLU activation. Both pretraining tasks are trained together so that the summed loss over both tasks is minimized.

Thanks to both pretraining tasks, finetuning can be done easily and inexpensive for different tasks. Finetuning tasks can include question answering, sentence pair classification, or single sentence tagging tasks. Depending on the task additional output layers might be required. Even though pretraining is very time- and energy-consuming, finetuning can be done in several hours using conventional GPUs [Devlin et al., 2018].

BART BART can be seen as a mixture of the original Transformer and BERT. Specifically, it is an encoder-decoder model like the Transformer, following pretraining strategies that slightly differ from those of BERT. These two aspects allow a task-specific finetuning, especially effective for text generation tasks like data-to-text generation.

BART's pretraining follows BERT's pretraining by corrupting the input text and learning a sequence-to-sequence model by reconstructing the original input. However, corrupting the input text is not limited to token masking alone, but can be any arbitrary corruption function. The paper compares the token masking task to four other corruption functions: token deletion, text infilling, sentence permutation, and document rotation. Results suggest that token masking is crucial and can be further enhanced by token deletion. Token deletion can significantly improve the performance of BART on generation tasks.

Apart from standard text-generation tasks, BART can be extended to perform machine translation using the same pretrained model. To do so, BART implements a randomly initialized encoder which replaces BART's original input layer. The whole model is therefore able to use different input and output vocabularies and tries to learn the mapping between the two languages in a two-step process. First, only the new encoder is trained together with BART's positional encoding as well as the attention layer of the first encoder layer [Lewis et al., 2019]. Second, the complete model is trained end-to-end, further improving the performance of the translation. Both steps try to minimize the cross-entropy loss like the standard Transformer does.

Generation During generation, both the Transformer and BART need to generate tokens one after another as done by traditional sequence-to-sequence models. The encoder works the same as during training and encodes the whole input sequence as one. It passes the transformed key and value matrices to the decoder. Meanwhile, the decoder only receives a single embedded start token and the encoder's output. Based on that, it has to generate the output sequence iteratively, which means that the predicted output token is used as an additional input to reason on and predict the next token. More tokens are predicted until either an end of sentence token is predicted, or the max sentence length is reached.

Two methods of decoding are used: greedy decoding and beam search. Greedy decoding is taking the token with the highest predicted probability from the probability distribution. This token is then appended to the sequence of already predicted tokens and can be used to further reason on. Beam search allows to hold on to multiple highly probable tokens at the same time step. The most probable sequence is then chosen over the other hypotheses at each time step, allowing to find whole sequences with higher probabilities. The number of held-on tokens can be chosen freely and can be used as a parameter during the generation.

4

Methods

Related work focuses on producing abstracts or single sentences based on multiple knowledge base triples. Given a specific input, the models have to choose individual triples and include them into a natural language-like text. In this work, we want to generate natural language-like sentences based on an already given sequence of knowledge graph entities and predicates taken from Wikidata. More specifically, we want to use already trained language models incorporating knowledge from many different domains. We further want to analyze whether the training and generation based on entity IDs or the corresponding labels differ. To do so, we need to implement and train different models to compare their generation on unseen data. The models are based on two already existing ones: the Transformer and BART. We use the Transformer as a baseline model and for the model translating IDs to labels. BART is finetuned specifically to our task. Altogether, we build and train six different models, some of which share their input and expected output sequence type. Others are trained to fulfill a certain sub-task incorporated in larger models. An overview of the different models and their input and outputs are shown in Table 4.1.

This chapter will explain the architectures of the models used, what input they require, and the output expected to be generated. The models are all implemented in Python and can be found in the project's repository[4].

Model	Input	Output
Baseline	Entity/Predicate IDs	Natural Sentence
Finetuned BART	Entity/Predicate Labels	Natural Sentence
BART for MT	Entity/Predicate IDs	Natural Sentence
IDs to Context Labels	Entity/Predicate IDs	Entity/Predicate Labels
Translation-BART	Entity/Predicate IDs	Natural Sentence
Translation-Middle-BART	Entity/Predicate IDs	Natural Sentence

Table 4.1: Summary of models with the input and expected output sequences

[4] *https://gitlab.ifi.uzh.ch/baumgartner/kg-text-generation*

Baseline Model We implement a baseline model because, to the best of our knowledge, there is no previous work on the same task to which we could compare our models. To generate sentences based on Wikidata entities, we use a machine translation model with two different embedding vocabularies. Thus, our baseline model follows the implementation of the original Transformer by [Vaswani et al., 2017] described in Section 3.2. We adapt the original implementation to be closer to the implementation of BERT by replacing fixed positional embeddings with learned ones. Having two different vocabulary embeddings allows us to train the generation of a target sentence based on the given entities end-to-end.

In addition to the baseline model trained on a sequence of Wikidata entity IDs, we train a model on a sequence of the corresponding entity labels. Both models have the exact same specifications during training. However, the model trained on labels uses the same input vocabulary for both the encoder and decoder. This model will help to understand the differences that finetuning a large pretrained model can make.

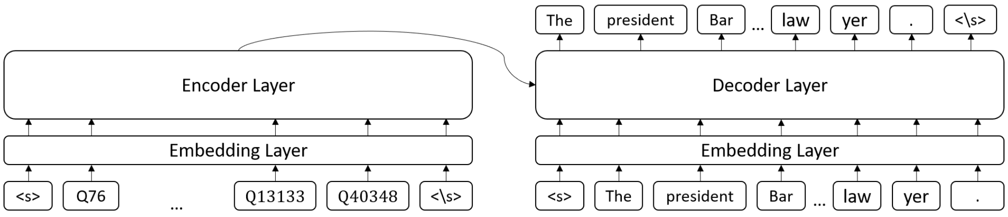


Figure 4.1: Baseline model built upon a basic Transformer architecture. The input is based on a sequence of Wikidata entity and predicate IDs. The target is the natural language-like sentence corresponding to the input.

BART Since already pretrained BART models exist, we can make use of one and finetune it for our task. A pretrained BART model, together with its Python implementation needed for finetuning, is made available by the Transformers[5] library. This pretrained BART model has 12 layers each on the encoder and decoder side, implementing a hidden size of 1024, resulting in over 400 million parameters. It is pretrained on a combination of corpora, including the CC-NEWS, BOOKCORPUS, OPENWEBTEXT, and STORIES dataset summing up to over 160GB of pretraining data [Liu et al., 2019]. Overall, the model was pretrained over a period of 500'000 steps having a batch size of 8000.

To use this pretrained model together with its embedding layer, we need to adopt the same tokenization strategy followed during pretraining. The tokenizer uses the byte pair encoding of the GPT-2 model [Radford et al., 2019], allowing to have a reduced vocabulary size without having unknown tokens. Essentially, this results in a splitting of infrequent words into their most frequent sub-words. Using that specific tokenization, we end up with a vocabulary of size 50'265, which is also the first embedding layer's input size.

[5] *https://huggingface.co/transformers/model_doc/bart.html*

The architectural choice of the pretrained model limits us to use an input that can be understood by BART. This would not be the case if we used a sequence of entity IDs. Tokenizing a string of these IDs would result in byte pair encodings consisting of single letters and numbers. The numbers itself would again split up into the most frequent numbers of the dataset. Due to this split-up, the initial ID representation and thus the meaning of an ID itself gets lost. For example the IDs *Q30* and *Q3030* get tokenized into *["Q", "30"]* and *["Q", "30", "30"]*. After the tokenization, the second ID would receive certain information the first ID contains too, even though the two IDs have nothing in common. Therefore, we cannot train BART directly using entity IDs.

We decide to finetune BART using the labels connected to Wikidata entity IDs. These labels do not face the same problem and can directly be interpreted by the tokenizer and BART's embedding layer. Each tokenized input sentence additionally receives a start and end of sentence token by the encoder. The target sentence does not get an end of sentence token to train the model on predicting the end of a sentence. Sentences can be trained in batches using an additional padding token in order to have the same token length for all the sentences of a single batch. The architecture together with a single tokenized sample sentence can be seen in Figure 4.2

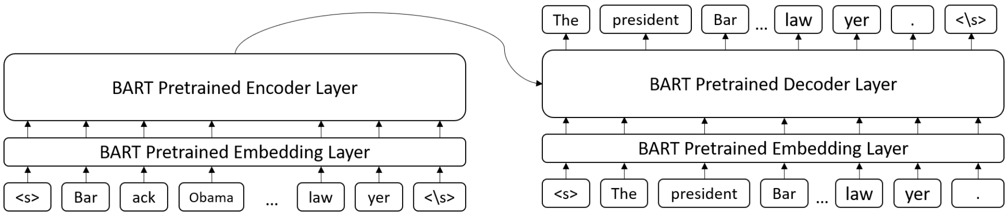

The
president
Bar
...
law
yer
.
<\s>
BART Pretrained Encoder Layer
BART Pretrained Decoder Layer
BART Pretrained Embedding Layer
BART Pretrained Embedding Layer
<s>
Bar
ack
Obama
...
law
yer
<\s>
<s>
The
president
Bar
...
law
yer
.

Figure 4.2: BARTs finetuning model using Wikidata entity and predicate labels as input to the pretrained encoder. The target is the corresponding Wikipedia sentence.

BART for Machine Translation Since the original BART model does not allow us to use a different embedding layer than the one implemented by the pretrained model, it cannot be finetuned on Wikidata IDs. In order to do so and still use a pretrained model, we can use the approach of treating the problem as a machine translation task. The original paper showed this approach, and we described it briefly in Section 3.3.

We follow that implementation by replacing the original BART encoder's embedding layer with a new BART encoder. This new BART encoder could have any dimensions regarding the number of encoding layers or the hidden dimension size. But to keep things simple between the two encoders, we fix the hidden dimension size to 1024, which is also the input embedding size of the original BART model. Thus, there will be no transformation needed between the two encoders. Each token embedding of the first encoder can simply be passed on to the second already pretrained and finetuned encoder. This pretrained and finetuned encoder comes from the already finetuned BART model that matches a sequence of Wikidata labels to natural language-like sentences.

The new encoder implements an embedding layer only dependent on the input vocabulary. It is then trained to learn the mapping between the newly introduced vocabulary and the original vocabulary in the described two-step process. This input vocabulary can consist of Wikidata IDs, and we can compare the generation directly to our baseline model having the same input and output sequences.

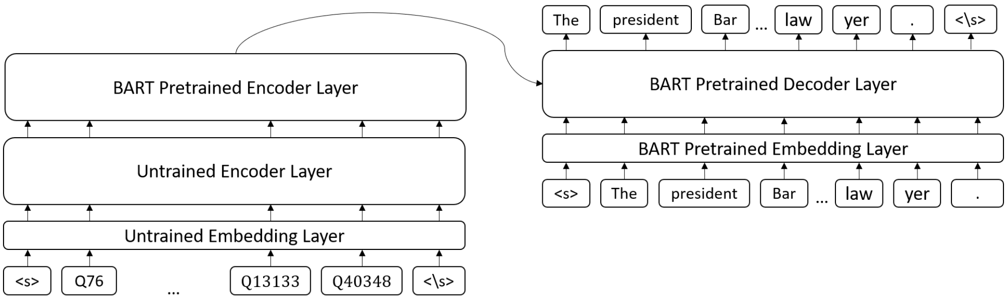

The
president
Bar
...
law
yer
.
<\s>
BART Pretrained Encoder Layer
BART Pretrained Decoder Layer
Untrained Encoder Layer
BART Pretrained Embedding Layer
Untrained Embedding Layer
<s>
The
president
Bar
...
law
yer
.
<s>
Q76
...
Q13133
Q40348
<\s>

Figure 4.3: BART model with an additional encoder able to perform machine translation tasks

ID-Label Translation We additionally focus on the task of translating entity IDs to their surface form. Learning such a mapping can be seen as an important intermediate step to our final target. If labels can be translated into their surface form depending on their surroundings, it will make the task of generating a sentence easier. Additionally, a pretrained model like BART could be used upon the output of such a model without training a new encoder.

To learn these contextualized mappings, we make use of the Transformer. Compared to our baseline, the model's goal is not to generate a sentence but only to learn contextualized labels for the Wikidata entity IDs. Thus, the input sequences are again Wikidata IDs, but the target output sequences are the entities' surface forms and predicates. Since a single ID can consist of multiple surface tokens, we need a sequence-to-sequence model able to reason on the input and predict a sequence, independent of the input length.

This translation model's architecture and training follow the implementation of our baseline model. Dimensions and number of layers on both the encoding and decoding sides can be varied.

Translation Prior to BART Having a functional translation from IDs to surface forms allows the generation of a label-like sequence. Thus, we can use a translation model followed by a pretrained and already finetuned BART model built into a sequence of two models. BART has to be finetuned on the sentence generation task since available models are not trained for it. This allows again the comparison to our baseline model, which is trained directly from entity IDs to sentences.

Figure 4.4 shows this sequential architecture. The two models need to be trained independently and are only connected during generation. To make the models compati-

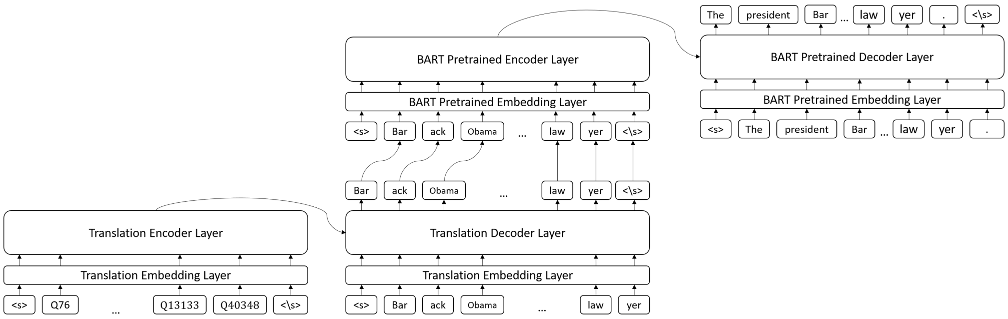

The
president
Bar
...
law
yer
.
<\s>
BART Pretrained Encoder Layer
BART Pretrained Decoder Layer
BART Pretrained Embedding Layer
BART Pretrained Embedding Layer
<s>
Bar
ack
Obama
...
law
yer
<\s>
<s>
The
president
Bar
...
law
yer
.
Bar
ack
Obama
...
law
yer
<\s>
Translation Encoder Layer
Translation Decoder Layer
Translation Embedding Layer
Translation Embedding Layer
<s>
Q76
...
Q13133
Q40348
<\s>
<s>
Bar
ack
Obama
...
law
yer

Figure 4.4: Translation model followed by BART used for generation only

ble, we need to prepend an additional start of sentence token to the translation output. Other than that, both models generate the sentences as they would on their own.

Translation-Middle-BART Instead of having two independent models after each other, it is possible to connect them directly. We do so by training a middle layer between the translation and the BART model. This middle layer transforms the translation models' output embeddings into the input embeddings of BART's encoder. Thus, it replaces the output layer of the translation Transformer and the embedding layer of BART. The full architecture can be seen in Figure 4.5.

The middle layer is a simple 2-layer feed-forward neural network similar to the one inside a single encoder layer. It implements the GeLU activation function. In the case where the output embedding dimension of the translation model does not fit the input dimension of BART (1024), we first up- or down-scale the output to fit the required input dimension.

Training the middle layer requires the input and target sequence of the translation training. This is due to the Transformer's decoder working with a masked target sequence to produce the output. Otherwise, the training could not be done in parallel and would require a prediction of output embeddings based on previous outputs. All parameters except the ones of the middle layer are kept fixed during training.

During generation however the translation model first generates its output recursively. The output embeddings are then fed through the middle layer and the BART encoder. BART's decoder works as usual based on the encoder's produced output.

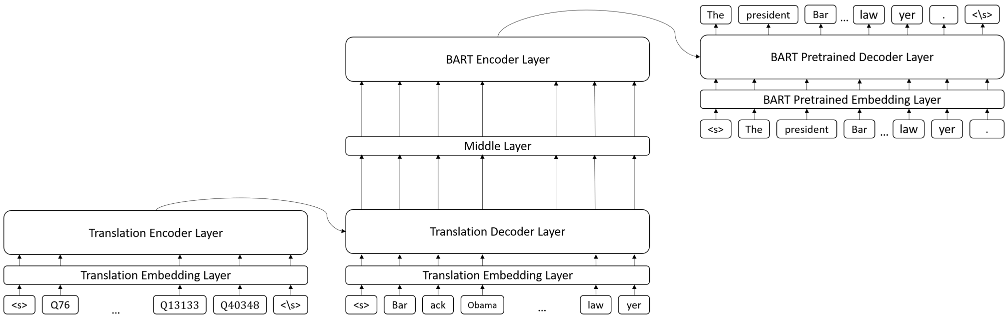

The
president
Bar
...
law
yer
.
<\s>
BART Encoder Layer
BART Pretrained Decoder Layer
BART Pretrained Embedding Layer
<s>
The
president
Bar
...
law
yer
.
Middle Layer
Translation Encoder Layer
Translation Decoder Layer
Translation Embedding Layer
Translation Embedding Layer
<s>
Q76
...
Q13133
Q40348
<\s>
<s>
Bar
ack
Obama
...
law
yer

Figure 4.5: Translation model followed by BART. Combines the output of the translations embedding layer with BARTs input layer via a feed forward neural network (middle layer).

5

Setup

This chapter describes the used dataset and how we processed it for our task in two steps. In the Training Setup section, we state how we train our presented models of Chapter 4 and what parameters we experiment with. At the end of this chapter, we state our evaluation methods.

5.1 Dataset

To train our models, we require sequences of knowledge graph entities aligned with annotated sentences. Several datasets in this direction exist, such as the WikiBio [Lebret et al., 2016], DAWT (densely annotated Wikipedia texts) [Spasojevic et al., 2017], and T-REx [Elsahar et al., 2019] dataset. However, the WikiBio dataset does not align single sentences with triples in the specific order of occurrence, and the DAWT dataset does not capture the relations between Wikidata items in a sentence. Thus, we decided to use the T-REx dataset for this thesis.

The T-REx dataset[6] is made available under a Creative Commons Attribution-ShareAlike 4.0 International License. It aligns 3.09 million Wikipedia abstracts with 11 million Wikidata triples containing over 600 different Wikidata predicates. This results in a total of 6.02 million aligned sentences. Though, more critical for our work are the Wikidata entities and predicates annotated in the sentences directly. These annotations can directly be used to train the models relevant to our research. The Python scripts able to parse the T-REx dataset are made available in this thesis repository[7].

We note that within this dataset not all Wikidata entities and predicate are annotated perfectly. Some sentences miss annotations or entities are wrongly annotated. This irregularity in the training data can make the model training harder since wrong facts are incorporated.

Preprocessing We process the data and adapt the format to our needs by parsing the input. Each page of the dataset is first loaded into our data model, allowing it to be handled in an object-oriented manner. Once the page is loaded, it gets propagated

[6] *https://hadyelsahar.github.io/t-rex/*

[7] *https://gitlab.ifi.uzh.ch/baumgartner/kg-text-generation*

Original Sentence	Star Science Fiction Stories No.1 is the first book in the anthology series, Star Science Fiction Stories, edited by Frederik Pohl.
Entity/Predicate	<Q7600878, Star Science Fiction Stories No.1><Q23653, anthology series><Q7600878, Star Science Fiction Stories><P98, edited by><Q312641, Frederik Pohl>
Output Sentence	Star Science Fiction Stories No.1 is the first book in the anthology series, Star Science Fiction Stories, edited by Frederik Pohl.
Original Sentence	It was first published in 1953 by Ballantine Books, without numeration, and was reprinted in 1972 as ”No. 1”.
Entity/Predicate	<Q7600878, Star Science Fiction Stories No.1><P1433, published in><XMLSchema#dateTime, 1953><Q2881141, Ballantine Books><XMLSchema#dateTime, 1972>
Output Sentence	Star Science Fiction Stories No.1 was first published in 1953 by Ballantine Books , without numeration , and was reprinted in 1972 as ”No. 1”.

Table 5.1: Two sample Wikipedia sentences followed by each other. Each sentence is aligned with its Wikidata entities annotated in the T-REx dataset. The final output sentence will be the target sentence to our models.

through a spaCy[8] pipeline. Tokenization, sentence segmentation, and named entities are all provided by the T-REx dataset and are kept as they are. However, spaCy allows the filtering and retokenization of named entities. This is necessary since the dataset can contain more than one detected Wikidata entity over the same token span. For example, the token span *”canton of Zurich”* might be annotated multiple times containing the entities *(<Q11943, canton of Zurich>, <Q72, Zurich>)*. If this is the case, we only keep the annotated Wikidata entity with the longer token span in the source text. Additionally, some entities are annotated over sentence boundaries requiring the deletion of these annotated entities.

Paragraphs of Wikipedia tend to talk about the same subject. This subject might be mentioned in its full form only once at the beginning of the first sentence and might be referred to via pronouns in the rest of the paragraph. The T-REx dataset algorithm annotates these references and marks them accordingly. If such a reference is found, we replace it with the title of the Wikipedia article. An example can be seen in the second sentence of Table 5.1. This will make the input to our models more concrete as pronouns will be harder to predict based on a given Wikidata entity.

The annotated Wikidata entities with their aligned text are written to a new set of files that will be used for postprocessing. Sample input and output sentences of this first processing step can be seen in Table 5.1.

[8] *https://spacy.io*

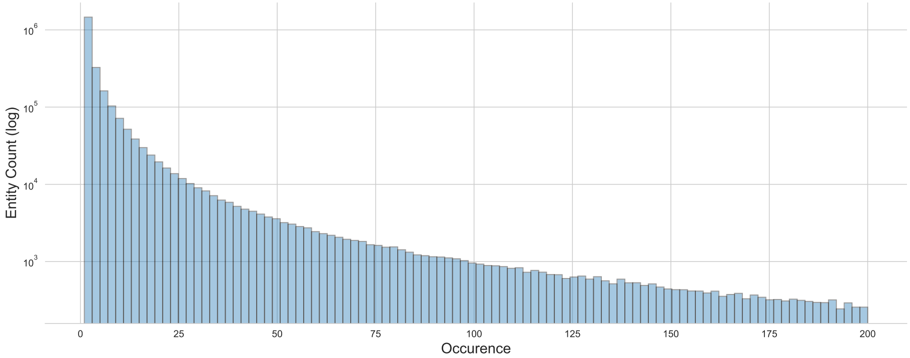


Figure 5.1: Histogram showing how many entities occur in how many sentences

Postprocessing After having the alignment fixed, we filter the dataset to contain only the most occurring entities and most densely annotated sentences. The resulting dataset contains nearly 2.5 million different Wikidata entities. The histogram of how many entities occur how often in the dataset can be seen in Figure 5.1. It shows the entities that occur less than 200 times, counting each occurrence in all sentences. Entities appearing more often are not displayed. We observe that many of the dataset entities are rare, and more than half of the entities occur less than three times over the whole dataset. The models that learn based on entity IDs cannot deal with these underpopulated entities, as a significant proportion of unknown labels in an input sequence can lead to poor generations [Luong et al., 2014]. As shown in Chapter 2, other approaches use the copy mechanism that cannot be used on a sequence of input IDs. Therefore, we decide on discarding all entities that occur less than 50 times across the dataset, leaving a vocabulary size of 99'142 entities. This vocabulary is only used for the models requiring an ID input sequence. Models based on a sequence of entity labels can use all entities. However, for comparison, we make sure that both model types receive the same amount of information, and only labels of the in-vocabulary-words are used as input.

On the other hand, some entities and especially predicates occur more often than others. For example, the entity <*Q30, United States of America*> occurs more than 700'000 times over the whole dataset. To deal with those entities, we allow the models to subsample from the input words, following the approach of [Mikolov et al., 2013]. This subsampling removes entities from the input sentence given some threshold t and the number of occurrences of the entity in the dataset f. The probability that a word is removed from the input sequence is then defined by $p = 1 - \sqrt{\frac{t}{f}}$. We choose a threshold t of 1000 to keep the importance of the most frequent entities.

We further remove sentences having less than two entities linked to it or having less than five tokens overall. Sentences having too few entities compared to their length are removed as well. More specifically, if the ratio of linked tokens is smaller than 0.35, the sentence is removed from the dataset. This results in a dataset consisting of

4.5 million densely annotated sentences that can be used for all our models (ID and label-based sequences). For later evaluation, we split the remaining data into a train (80%), validation (10%), and test (10%) set. This allows checking the models' ability to generalize on unseen data.

The final aligned sequences are either label- or ID-based and will be tokenized differently by the models as stated in Chapter 4. Whereas the ID-based models use simple space tokenization to tokenize the ID input sequence, the models based on labels use a byte pair encoding. Additionally, all dates are replaced with the special token <*XMLSchema#dateTime*>. The replacement is done for all dates regardless of their format. This is done to keep the input between the two approaches comparable and not rely on copy mechanisms.

5.2 Training Setup

We train all our models defined in Section 4 on their specific tasks using the required data. All models are trained in batches and are entirely parallelizable during training. In this section, we again go through all of our models, defining their training procedures, and state the adapted hyper-parameters of each model. Overall we train 60 different model configurations based on our six trainable model architectures. All different model configurations can be found in A.

Every model configuration was trained or finetuned on the same machine using the same proportion of available hardware[9]. The memory limitation of the GPU also restrains some parameter configurations (e.g. batch size or hidden dimensions) of our models.

Baseline Since one of the baseline models uses Wikidata IDs as input, we have different embedding layers on the encoder and decoder side. Whereas the encoder embedding has a vocabulary size of 99'142, the decoder embedding layer has a size of 50'265. The baseline model trained on Wikidata labels shares its vocabulary for both the encoder and decoder embedding layer and has a size of 50'265. Both the encoder and decoder implement a hidden dimension of 1024 and have 8 attention heads.

The parameter weights are initialized using a Xavier initialization [Glorot and Bengio, 2010]. We train the model with the Adam optimizer and use an Adam epsilon of 1e-8. Training is conducted over 10 epochs using a batch size of 64. The training data is not downsampled, leaving all IDs in the input sequence without removing the most frequent ones. We do not make use of the full dataset due to training time limitations. Thus, we train our model on 10% of the full dataset. No warmup phase is implemented.

For the training of the baseline model, we only change the learning rate. We use fixed learning rates of 1e-3, 3e-4, and 1e-5 to train the configurations of our baseline model.

[9] GeForce GTX Titan X, 12 GB memory

BART We finetune the pretrained BART model on Wikidata labels using the same encoder and decoder embedding layers as the pretrained model. Both embedding layers share the same vocabulary of size 50'265. The hidden dimension of 1024 is kept during finetuning and cannot be changed.

Next to the standard Adam optimizer, we finetune the model with the Adam fixed weight decay regularization as implemented by the Transformers library and introduced by [Loshchilov and Hutter, 2017]. The Adam epsilon of 1e-8 is kept fixed over the different configurations. Finetuning is conducted over 10 epochs using a batch size of 16. We change the amount of training data using only percentages of the available sentences. This is done to understand the finetuning process and the needed data better. Together with dataset downscaling, we use different learning rates.

Table 5.2 shows the parameters we are alternating and their values. Using every combination of the parameters would result into 32 finetuned BART configurations, which would be time and energy consuming. Thus, we only train on certain combinations of these parameters, always building up on already discovered knowledge. For example, we do not train a new model configuration on more data together with a higher learning rate, if a model configuration using less data having the same learning rate already performed poorly.

Dataset Size
1%
10%
20%
50%

Learning Rate
1e-4
3e-5
1e-5
5e-6

Optimizer
Adam
AdamW

Table 5.2: Parameters used to finetune BART on labels

Additionally, we train a BART model based on the surface forms of the labels in the text. Meaning that the input sequence contains the entities as they occur in the sentence. This model only needs to fill in blanks between the entities and does not have to change the entities' appearance in the text. We hypothesize that such a training will create an upper bound performance no other model can reach.

BART for Machine Translation This addition to the BART model implements a new encoder which needs additional training on mapping Wikidata IDs to labels. As our baseline, this model uses an encoder embedding layer of size 99'142 and a decoder embedding layer of size 50'265. Even though the new encoder could have any hidden dimension size, we keep it at 1024. We vary the number of encoder layers of the new encoder.

Before training, we load the weights of our best finetuned BART model so that the model is already able to perform the sentence generation task. The new encoder weights are then initialized using the Xavier initialization. We train the model with the Adam optimizer and use an Adam epsilon of 1e-8. Training is conducted over 10 epochs using a batch size of 16. The model configurations are trained on different dataset sizes. Additionally, we experiment with sentences where most frequent IDs were downsampled. This leads to the set of parameters listed in Table 5.3.

Dataset Size
1%
30%
100%

Learning Rate
1e-3
3e-4
1e-4
5e-5
1e-5

Downsampled IDs
TRUE
FALSE

Encoder Layers
3
6

Table 5.3: Parameters used to train BART for machine translation

ID-Label Translation To translate IDs to labels based on their context, we train a Transformer using several adjustable parameters. As with our baseline, we fix the number of encoder and decoder layers to 3. However, we use the size of hidden dimensions as a parameter for this model. The encoder's input vocabulary has a size of 50'265, and the decoder uses a vocabulary of the size 99'142.

The Adam optimizer is used for training with an Adam epsilon of 1e-8. Training is performed over 10 epochs and using a batch size of 64. We adjust the learning rate and size of the dataset across this model's configuration. The first results suggested that downsampling frequent Wikidata IDs is needed for this model to work so that all experiments use this approach. This leads to the parameters under investigation shown in Table 5.4

Dataset Size
10%
100%

Learning Rate
1e-3
5e-4
1e-4

Hidden Dimensions
64
128
256
1024

Table 5.4: Parameters used to train translation from IDs to labels

Translation to BART using a Middle Layer We use our previously trained translation and finetuned BART models and combine them via a middle layer. This middle layer itself consists out of two feed-forward layers. It is trained using the Adam optimizer with an Adam epsilon of 1e-8, a batch size of 64, and a hidden dimension size of either 1024 or 512. We also adapt the learning rate to be 1e-3, 3e-4, and 1e-5 respectively.

5.3 Evaluation

We perform a qualitative and quantitative evaluation. A qualitative evaluation focuses on human evaluation and is not done automatically. The quantitative evaluation can be done automatically and compares the generated sentence to the actual target sentence using different scoring functions.

We follow previous work and make use of the BLEU [Papineni et al., 2002] and ROUGE [Lin, 2004] scores. BLEU was originally implemented to compare the output of a machine translation model to given reference sentences. Each output sentence is scored with respect to its references between 0 and 100, where 100 would be an exact match between the predicted and actual target sentences. In its essence, it calculates the overlap of words between prediction and target sentences and can be seen as a precision measurement. However, the BLEU score calculation can be parameterized and thus can differ between implementations, as shown by [Post, 2018]. Therefore, we follow the introduced approach by calculating the BLEU score using *SacreBLEU*[10] and its Python implementation.

Using the ROUGE scores, we can further complement the automatic evaluation. Compared to BLEU, ROUGE can be seen as a recall measurement. It calculates the overlap between the generated and target sequences based on how many n-grams of the target also occur in the generation. In our evaluation we report ROUGE-1, ROUGE-2, and ROUGE-L scores calculated using the *rouge*[11] library implemented in Python. Whereas ROUGE-1 and ROUGE-2 are referring to a unigram and bigram overlap, ROUGE-L refers to the longest common subsequence.

Having BLEU and ROUGE scores for a quantitative analysis allows us to compare the different models and their configurations between each other quickly. However, these scores are focused on using unigrams or bigrams, and if not the same words occur in both sequences, these scores might be poor. Translations can be good even though not the exact same words are used. One can deal with the problem by having many human-generated reference sentences to which the prediction is compared to. Since this is not the case for our dataset, we additionally have to conduct a qualitative analysis of predicted sentences. This analysis is performed by creating a set of sample Wikidata entity sequences matching the input structure of each model. Sentences are then generated based on these input sequences and compared to each other across the models.

[10] *https://github.com/mjpost/sacrebleu*
[11] *https://github.com/pltrdy/rouge*

6

Results

In this chapter, we present our results in-depth and for each research objective separately. Our research objectives are both determined by a goal to be achieved, supplement research questions, and a hypothesis. The experiments we conduct try to support these hypotheses based on the identified research questions. Next to the set hypotheses, which in general compare different model architectures, we also inspect the results of the hyper-parameter adjustments introduced in Section 5.2.

Thus, this chapter is separated into two sections. Each section first describe the specific goal, which is followed by the experiment results. We focus on the most essential and differentiable hyper-parameter results for each model architecture. The results of all model hyper-parameter configurations, together with some generated sentences, can be found in Appendix A.

6.1 Sentence Generation Based on Wikidata Labels

Our overall objective is to generate natural language-like sentences based on a sequence of Knowledge Graph entities, by using a sequence of labels collected from Wikidata entities and try to predict a sentence based on these labels. More specifically, we want to use Transformer based architectures and large pretrained language models to fulfill this task. Thus, we state our first goal as follows:

Goal 1: Comparing the natural language-like sentence generation based on a sequence of Wikidata labels between a baseline model and finetuned pretrained language models, to improve the performance of such generative models.

Based on Goal 1 we can derive our research questions (RQ) and hypothesis as follows:

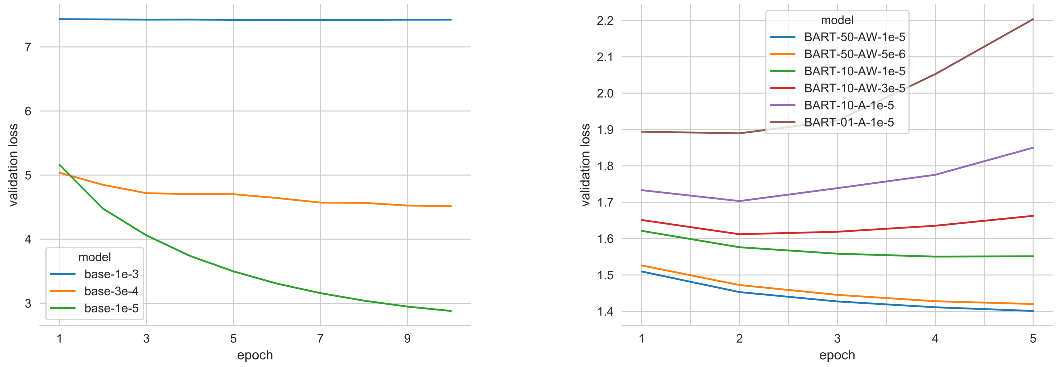
RQ 1: How can pretrained language models be finetuned to improve the natural language-like sentence generation based on a Wikidata entity sequence?

- RQ 1.1: How can a baseline model be trained to perform the generation of a sentence based on an entity sequence?
- RQ 1.2: How can a pretrained language model be finetuned to perform the generation of a sentence based on an entity sequence?

- RQ 1.3: How can the two models be compared to each other quantitatively as well as qualitatively?

Hypothesis H_1: A finetuned pretrained language model outperforms a trained baseline model on the task of generating a natural language-like sentence given a sequence of Wikidata labels.

To achieve the mentioned objective, we compare our baseline model results to the results from finetuning a pretrained BART model. Since the original BART model can only deal with a sequence of labels, the baseline model is also trained on labels. We first analyze the results of our baseline model, followed by the results of a finetuned BART model. The results displayed come from the generation of sentences based on our validation set, forcing the model to generate on previously unseen data. In the last step, we compare the two models to each other.



(a) Baseline

(b) BART

Figure 6.1: Loss on validation set of the trained baseline and finetuned BART model

Baseline Approach The results of our trained baseline model on the generation task can be found in Table 6.2. It can be seen that the two configurations having higher learning rates perform worse compared to the configuration having the lowest learning rate. By checking the generated sentences of these two models, we see that almost nothing is learned in both cases. This becomes clearer when checking the validation loss over different training epochs as seen in Figure 6.1. The model configuration with the highest learning rate of 1e-3 only outputs empty sequences, which leads to a BLEU score of 0. On the other hand, the configuration with a learning rate of 1e-4 generates a choice between a few sentences, which can be seen as underfitting on the data. Our best baseline model reaches a BLEU score of 15.34 and a ROUGE-L score of 45.83.

	Samples	Optimizer	LR	BLEU	ROUGE-1	ROUGE-2	ROUGE-L
baseline	10%	Adam	1e-3	0.00000	0.00000	0.00000	0.00000
	10%	Adam	3e-4	1.12376	19.47139	5.69398	19.14092
	10%	Adam	1e-5	**15.34037**	**45.95129**	**28.58206**	**45.82768**
BART	50%	AdamW	5e-6	42.06480	64.99611	48.58221	64.36054
	50%	AdamW	1e-5	**42.92034**	**65.68538**	**49.39549**	**64.97764**
	10%	AdamW	1e-5	39.62515	63.51268	46.54062	62.74191
	10%	AdamW	3e-5	38.45631	62.68675	45.23251	61.97608
	10%	Adam	1e-5	36.23535	60.68445	42.93500	60.01353
	1%	Adam	1e-5	30.43606	56.11096	37.52875	55.34202

Table 6.1: Results of baseline and BART model trained on labels

To check the generated sentences of the best baseline configuration in more detail, we conduct a qualitative evaluation of the generated sentences. 5 input sequences next to their generated sentences are shown in Table 6.2. These input sequences were built by hand and are not part of any of the used datasets. They do not represent a graph walk and thus do not represent real-world truthfulness.

Inspecting the generated sentences, we can see that the model already learned to build sentences in the correct format. However, the input label sequence is only partially used. Sentence 1 gets many elements right and switches the label of *spouse* to the tokens *married to*. It also replaces *Michelle Obama* with the surface form *Michelle* sounding more fluent in this specific context.

Other generated sentences like sentences 2 and 4 miss some of the labels entirely. In sentence 2, *Donald Trump* is replaced with *The Bill*, and the generation of sentence 4 is completely off, having nothing in common with the input sequence. A repetition of words can be seen in sentence 5, where the label *Mount Everest* is replaced with a combination of wrong surface forms.

Finetuned BART We finetune the pretrained BART model as mentioned in Section 5.2 and take the model checkpoint with the the lowest validation loss across its training epochs. The sentence generation results of chosen BART model configurations can be seen in Table 6.2. Using these configurations, we can show the most important performance differences.

The best performing model is finetuned on 50% of all data, using the adapted BART optimizer, and a learning rate of 1e-5. Finetuning on 50% of all data took 64h on a single GPU. It reaches a BLEU score of 42.92 and outperforms its sister configuration (42.06 BLEU) finetuned on a lower learning rate of 5e-6. Using a smaller portion of the available dataset leads to lower BLEU and ROUGE scores, as seen on the configurations trained on 1% and 10% of the data. Additionally, using a smaller dataset leads to a much faster overfitting, as shown in Figure 6.1 showing the validation loss of the different finetuned model configurations.

	Input Sequence	Generated Sentence
1	Barack Obama spouse Michelle Obama occupation lawyer	Barack Obama married to his wife Michelle, and his career.
2	Donald Trump founded by Facebook owned by Microsoft	The Bill is the founder and a Microsoft Windows owned by Microsoft.
3	Switzerland instance of sovereign state part of Europe instance of continent	The Swiss national and is a public country that is part of Europe, and is an annual continent.
4	Lion subclass of felidae	The film is a type of the Indian Ocean and the 2002 Indian Ocean.
5	Earth highest point Mount Everest continent Asia	Earth's highest point is the highest point in Mount Mount Mount Desert in Asia.

Table 6.2: Generated sentences from on labels trained baseline model

From the results, we can take that the optimization using the Adam optimizer, including a fixed weight decay regularization, outperforms the Adam optimizer's standard implementation. This can be seen in the result table by comparing the model configurations trained on 10% of all data and using a learning rate of 1e-5.

For the translation models and to get an idea of an upper boundary, we also finetune a model based on the labels' surface forms. This model reaches a score of 51.30 BLEU, outperforming the best finetuned BART configuration by 8 BLEU.

Our qualitative analysis is performed on the best finetuned BART model from Table 6.1. It suggests that the finetuned model indeed can produce fluent, natural language-like sentences based on Wikidata labels. Our set of example sentences can be seen in Table 6.3 next to the generated sentences by the finetuned BART model. All generated sentences incorporate the knowledge from the input sequence, including punctuation. Sentence 2 and 4 produce the expected output while not mentioning more facts than given by the input sequence. Still, sentence 3 produces the span *centre of European politics*, which is not given in the input sequence. As shown in sentence 2, the finetuned BART model can produce sentences that are based on wrong facts. Thus, it can be argued that the input sequence gets more attention than the incorporated world knowledge of the pretrained model.

Additionally, we inspect BART's sentence generation with varying entities and predicates occurring in an input sentence, starting with sentence 6 in Table 6.3. Sentence 6 has many entities and predicates linked to the start entity of Germany. Nonetheless, BART can generate a correct output sentence based on the given input. The objects linked to Germany and the object linked to Berlin all reference their specific subject. BART generates a new output sequence by changing the order of the input labels, as can be seen in sentence 7. We note that the generated sentence is highly dependent on the input order, meaning that BART does not select statements from the input sequence first and orders the statements accordingly. By switching the subject and object of a triple, BART mostly reasons on that new order, so that sentences are generated that do not describe true statements anymore. A counterexample can be seen in sentence 9. BART generates a sentence describing that Berlin is the capital of Germany even though the input sequence of *Berlin capital Germany* specifies that Germany is the capital of Berlin. The correct predicate in this case can be seen in sentence 8. Thus, we can argue that BART mainly generates sentences in the order of the given input sequence but can still do some reasoning about the entities and their relations to each other. More of these sequence and triple modifications are shown in Table A.4.

	Input Sequence	Generated Sentence
1	Barack Obama spouse Michelle Obama occupation lawyer	Barack Obama and his wife Michelle chose to pursue a career as an attorney.
2	Donald Trump founded by Facebook owned by Microsoft	Trump is the founder of Facebook, which is owned by Microsoft.
3	Switzerland instance of sovereign state part of Europe instance of continent	Switzerland is a sovereign country in the western part of Europe, and is a centre of European politics and commerce in the continent.
4	Lion subclass of felidae	The lion is a type of felidae.
5	Earth highest point Mount Everest continent Asia	The highest mountain on Earth is the highest peak, Mount Everest, on the continent of Asia.
6	Germany is a country part of Western Europe capital Berlin inception XMLSchema#dateTime located in or next to body of water Spree	Germany is a country in the northern part of Western Europe, with its capital city Berlin, which was founded on the shore of the Spree.
7	Germany capital Berlin located in or next to body of water Spree inception XMLSchema#dateTime part of Western Europe	Germany's capital, Berlin, on the shore of the Spree, was founded on May 1, 1801, and is the oldest continuously inhabited part of Western Europe.
8	Berlin capital of Germany inception XMLSchema#dateTime	Berlin, the capital of Germany, was founded on April 1, 1771.
9	Berlin capital Germany inception XMLSchema#dateTime	Berlin, the capital city of Germany, was founded on April 1, 1793.

Table 6.3: Generated sentences from the on labels finetuned BART model

Since the qualitative inspection suggested better performance than the finetuned BART model's scores, we analyze the generation in more detail. We find two characteristics that lower the scores of the finetuned BART model, namely dates as well as missing labels in the dataset. Dates are replaced in the dataset with a special token. Even though we could finetune BART using numbers, we decided against it to keep it comparable to our second objective. This decision highly reduces the generated sentences' accuracy since dates like birthdays in autobiographies or construction dates of buildings are almost never generated correctly. Even though the date format is generated in an accurate manner, the exact numbers are not, lowering the scores of the generated sentence. Examples of wrong date insertions can also be seen in sentences 6-9 of Table 6.3.

Another issue is missing labels in the input sequence of our datasets. Many target sentences are mentioning facts not stated in the input sequence. The model still can create a natural language-like sentence for these input sequences but might be off the target sentence by several tokens. This is commonly seen in examples where the input sequence starts with a predicate label missing an entity at its first position. To follow that lead, we explore sentences starting with the predicate *<P31, instance of>* in more detail. We can see that the model often generates an entity related to the true target entity but is not the same entity (e.g., a neighboring city of the actual one, or another natural phenomenon). Since these wrongly chosen entities mostly consist of more tokens, sentences tend to have lower scores. As mentioned in Section 5.1, some input sequences are labeled falsely, aligning wrong entities to the target sentence. This again makes the task of generation harder for the finetuned model.

Comparison Answering RQ 1.1, we showed that the baseline model can produce natural language-like sentences while reasoning on the input to a certain extent. We further demonstrate that a finetuned BART is able to produce natural language-like sentences based on an input sequence of Wikidata labels which answers RQ 1.2. Comparing the two models, we can say that the finetuned BART model outperforms our baseline model. It reaches a score of 24 BLEU higher than our baseline model when trained on the same amount of data (10%). The different ROUGE scores of the finetuned model are also considerably higher.

Additionally, the finetuned BART model's generated sentences are more fluent and incorporate more knowledge from the input sequence than the baseline model. We thus accept our hypothesis H_1 by saying that BART outperforms our baseline model on the same data, both quantitatively as well as qualitatively.

6.2 Label-Based vs. ID-Based Generation

Our second goal is to compare different input sequence types between each other: IDs and labels. We hypothesize that encoding entity IDs as an input sequence can leverage results since knowledge between different Wikidata entities can be embedded. Making use of that embedded knowledge between IDs could improve the performance of sentence generation based on Wikidata entities. Our second goal, its research questions, and hypothesis are the following:

Goal 2: Comparing the generation of natural language-like sentences based on a sequence of Wikidata IDs to the generation based on their respective labels, to understand the difference between the two approaches.

RQ 2: Can the generation of natural language-like sentences be improved by choosing entity IDs over labels as an input sequence?

- RQ 2.1: How can a baseline model be trained to generate a sentence given an input sequence consisting of entity IDs?
- RQ 2.2: How can a model be finetuned to generate a sentence given an input sequence consisting of entity IDs?
- RQ 2.3: How can the baseline model be compared to the finetuned model?
- RQ 2.4: How can the model trained on entity IDS be compared to the one trained on labels quantitatively as well as qualitatively?

Hypothesis H_2: A language model trained on the generation of natural language-like sentences based on Wikidata IDs outperforms a model trained on the respective Wikidata labels.

An essential part of the objective is to train a model that is able to transform a sequence of entity IDs into a natural language-like sentence. However, comparing this ID-based model to a label-based model is the overall objective. To do so, we use the finetuned BART model discussed in Section 6.1

Baseline Our baseline trains a Transformer on a sequence of Wikidata IDs aligned with a natural language-like sentence. The results of the baseline model are shown in Table 6.4. It can be seen that the model using the largest learning rate has a BLEU and ROUGE score of 0. That specific model configuration generates sentences consisting of empty strings and has not learned anything at all. Having lower learning rates, the scores slightly improve from almost 2 BLEU to 11.5 BLEU. The best baseline model configuration uses a learning rate of 1e-5 and reaches a ROUGE-1 and ROUGE-L score of around 40. Inspecting the high ROUGE scores, we can argue that the model already generates a large portion of the target sentences' actual tokens.

Learning Rate	BLEU	ROUGE-1	ROUGE-2	ROUGE-L
1e-3	0.00000	0.00000	0.00000	0.00000
3e-4	1.90713	16.55452	5.33299	16.38419
1e-5	**11.54714**	**41.01040**	**23.69338**	**40.74124**

Table 6.4: Results of baseline model trained on IDs

To not rely on the scores alone, we check generated sentences by the baseline model qualitatively. Our example input sequences, together with the generated sentences by the baseline model, can be seen in Table 6.5. Inspecting the generated sentences, we see that the overall format of generating sentences is learned correctly. In sentence 1, the generated sentence also incorporates all input entities correctly with their regarding surface form. Other sentences however do not work that well. For example, sentence 2 transforms the ID *Q22686* to the surface form *Robert Smith* even though the entity is about *Donald Trump*. Sentence 3, 4, and 5 have the same problem of not being able to map certain entity IDs to its corresponding surface form.

	Input Sequence	Generated Sentence
1	Q76 P26 Q13133 P106 Q40348 (Barack Obama spouse Michelle Obama occupation lawyer)	Barack Obama and his wife Michelle Obama, who had a career as an attorney.
2	Q22686 P112 Q355 P127 Q1213 (Donald Trump founded by Facebook owned by Microsoft)	Robert Smith was the founder and Facebook owner of the City of Wagon.
3	Q39 P31 Q3624078 P361 Q46 P31 Q5107 (Switzerland instance of sovereign state part of Europe instance of continent)	The Swiss Cross is a country system of Europe that is a part of the City of Gersh
4	Q140 P279 Q25265 (Lion subclass of felidae)	"The lion is a type of "" fine ""."
5	Q2 P610 Q513 P30 Q48 (Earth highest point Mount Everest continent Asia)	Earth's highest point is the highest point in Mount Everest, named after the continent of Asia.

Table 6.5: Generated sentences from the best baseline model trained on IDs

BART for Machine Translation The results of the trained BART model having an additional encoder are displayed in Table 6.6. As can be seen, the model configurations perform poorly no matter the training data, the learning rate or if input sentences are downsampled. Additionally, one can observe that the BLEU and ROUGE scores are the same across different model configurations. This indicates that all model configurations learned to generate the same output. By checking the generated sentences, we note that the model only produces a small variation of sentences. This confirms that the model only learned to generate the most often occurring entities.

To better understand the model's behavior, we additionally train several configurations with all combinations of parameters. This includes changing the number of the encoder layers of the added encoder. However, the results stay the same. Inspecting the single

Samples	Downsampled	LR	BLEU	ROUGE-1	ROUGE-2	ROUGE-L
1%	FALSE	1e-4	0.015506	11.6161	0.049183	11.34021
30%	FALSE	1e-4	0.00841	7.600451	0.012089	8.0874
30%	FALSE	5e-5	0.188195	10.23326	0.486224	11.50871
30%	TRUE	1e-5	0.188195	10.23326	0.486224	11.50871
100%	TRUE	1e-5	0.188195	10.23326	0.486224	11.50871

Table 6.6: Results of BART for machine translation trained on IDs

layers of the model one by one, we notice that the last encoder layer's output produces the same embeddings. These embeddings are equal regardless of the input from the newly trained encoder. This means that the previously finetuned BART encoder did not learn how to handle the new input accordingly.

To exclude a bug in our implementation, we check our model's ability to translate German sentences into English sentences similar to the original BART paper. Even though the results are not particularly good, the model is able to learn a translation. This means that the task of translating a sequence of entity IDs into a sequence of labels is harder than a usual translation task, regarding the data used. Additionally, learning to translate two similar length sequences might be easier than learning multiple tokens from one single ID.

Translation To pursue whether other models are able to generate natural language-like sentences from a sequence of Wikidata IDs, we first investigate if labels can be translated from IDs. Doing so will allow us to use contextualized labels and their embeddings over a simple ID to label mapping. These contextualized labels lead to more knowledge included in the model. The translation model we use is the Transformer described in Chapter 4. Results of model configurations are presented in Table 6.7. It can be seen that using a larger amount of data improves the results. Our best translation model uses all the available data, a hidden dimension of 1024, and a learning rate of 1e-4. It reaches a BLEU score of 53.37 and a ROUGE-1 score of 72.64, showing that many tokens from the target sequence also appear in the prediction.

Samples	Hid. Dim.	LR	BLEU	ROUGE-1	ROUGE-2	ROUGE-L
10%	1024	1e-4	17.32067	35.87757	22.15259	37.54379
100%	1024	1e-4	**53.37352**	**72.64361**	**58.32042**	**73.73855**
10%	256	1e-4	2.89304	11.63534	4.634781	12.38123
100%	256	1e-4	50.58442	71.03164	56.24096	71.83572
100%	256	5e-4	16.43847	33.67716	20.46892	33.65074
100%	128	5e-4	42.26839	63.08361	47.51500	64.01119
100%	64	1e-3	35.40920	57.11204	40.98437	58.23007

Table 6.7: Results of the translation model trained on IDs

Reducing the hidden dimension from 1024 while keeping the learning rate fixed results in lower scores. Additionally, a good balance has to be found between the hidden dimensions and the learning rate. This can be seen using the two model configurations of Figure 6.7 having a learning rate of 5e-4. While the one with 256 hidden dimensions performs poorly with only a score of 16.44 BLEU, the one with 128 hidden dimension scores 42.27 BLEU. However, when trained with a smaller learning rate, the model configuration with 256 hidden dimensions can outperform the other.

Translation to BART Our model transforming contextualized translated IDs into a natural language-like sentence uses two previously introduced and trained models. The Transformer translating IDs into contextualized labels and the BART model finetuned on a sequence of contextualized labels. We use the best performing configuration from both models to train the complete model connected via a trainable middle layer. The results are displayed in Table 6.8.

From the results, we can observe that only model configurations using a low learning rate of 1e-5 can produce results worth mentioning. The best model configuration uses 10% of all data having a hidden dimension size of 512 in the trainable middle layer. Thus, the best configuration first reduces the dimensions in the trainable middle layer instead of expanding them. It reaches a BLEU score of 11.98 and a ROUGE-1 score of 36.38, thus having a higher recall than precision. The model configuration increasing the layer size to 2048, while keeping the data and learning rate fixed, performs slightly worse. It reaches a BLEU score of 11.76.

Additionally, we generate natural language-like sentences on the ID-based input sequence by using a sequential approach. First, translating the sequence of input IDs into contextualized labels and feeding this output directly to the finetuned BART results in higher scores than the approach using a middle layer. This sequential approach reaches a BLEU score of 35.74, outperforming the middle layer approach by nearly 24 BLEU. Again, this supports the fact that translating embeddings from one task to another is harder than we expected.

Samples	Hid. Dim.	LR	BLEU	ROUGE-1	ROUGE-2	ROUGE-L
1%	2048	3e-4	0.16423	12.57257	1.186031	12.41652
1%	2048	1e-5	9.867016	32.76723	15.93545	32.51841
10%	2048	1e-5	0.191977	13.22733	1.014918	12.75595
10%	2048	1e-5	11.75899	35.72452	18.38085	35.55134
10%	512	1e-5	**11.98177**	**36.37637**	**18.96035**	**36.25002**

Table 6.8: Results of the translation to BART model trained on IDs

We qualitatively analyze the translation model and the models built on it together. The results of our sample sentences can be seen in Table 6.9. Checking the translation of our ID to label model, we see that for our sample sentences, the generated output sequences are moderate. In sentence 1 the ID *Q76* gets translated to *President Obama* whereas the ID for *Michelle Obama* gets translated to *Lady Obama*. One might argue that these surface forms can make sense in a certain context, but together with the translated predicate *husband*, the generated label sequence does not seem to be particularly good. Sentence 2 does correctly translate *Donald Trump*, *Facebook*, and their connecting predicate but does fail on the translation of *Q1213* to *Microsoft*. The same holds for sentence 3, where the first part of the sequence sounds correct, but the continent *Europe* is translated into *Mirren*. By checking more translations generated on our test set, we can observe that the translations shown in Table 6.9 are worse than the average generated translations.

	Input IDs	Q76 P26 Q13133 P106 Q40348
1	ID-Label Trans.	President Obama husband Lady Obama career attorney
	Trans.-BART-Middle	President Barack Obama and his wife, Nancy, and former First Lady Nancy Obama, while he was still a career federal prosecutor.
	Trans. Sequential	President Obama and his husband, First Lady Obama, are both career lawyers and attorney.
	Input IDs	Q22686 P112 Q355 P127 Q1213
2	ID-Label Trans.	Donald Trump founder Facebook belongs to Indian Reservation
	Trans.-BART-Middle	The company was founded in 1997 and belongs to the International Federation of the Manufacturers of Automobiles and Motorcycles (FIM).
	Trans. Sequential	Donald Trump, founder of Facebook, belongs to the Tukwila Indian Reservation.
	Input IDs	Q39 P31 Q3624078 P361 Q46 P31 Q5107
3	ID-Label Trans.	Switzerland is a sovereign sovereign country part of Europe is a Mirren
	Trans.-BART-Middle	Switzerland is a country in the northern part of Europe and is a component of the European Economic Area.
	Trans. Sequential	Switzerland is a sovereign sovereign country, part of Europe , and is a constituent of the European Community (EEC).

Table 6.9: Generated sentences from the best translation models

The model using a middle layer to transform embedding outputs of the translation into understandable input for BART generates fluent but mostly incorrect sentences. The span *Michelle Obama* for example became *Nancy Obama* who is no relative or connected to her at all. Sentence 2 does not contain any information from the input IDs except for the relations *founded by* and *belongs to.*

The sequential model makes better use of the translated IDs and can build fluent sentences around them. However, it also incorporates false translations into its generated sentences. This can mostly be seen in sentence 3, where the translation model generates *sovereign* twice in a row. The finetuned BART model takes the input sequence as-is and does not change the repeating word. Thus, it is blindly using the input sequence.

Comparison Concluding our second objective, we can see that the task of learning natural language-like sentences from an input sequence of Wikidata IDs is difficult. We show that our baseline model can produce such natural language-like sentences to a certain extent, reaching a score of 11 BLEU. A model combination of pretrained translation and generation connected via a trainable middle layer is on par with our baseline model, based on scores alone. However, the generated sentences seem worse than our baseline, and thus we cannot clearly say which one performs better or worse. The translation model alone can generate contextualized labels based on Wikidata entity IDs but makes mistakes during the generation. These mistakes are carried on to the finetuned BART model, generating a sentence based on the translated IDs. Even though some mistakes are carried on, a sequence of these two models can improve upon our baseline by around 20 BLEU. The generated sentences also seem more fluent and grammatically correct.

Moreover, training and finetuning BART with an additional encoder does not work as expected on any of the trained model configurations. This shows us that the problem of translating ID embeddings into understandable embeddings for BART is hard. A single encoder is not capable of this task. We argue that the main issue of this task is having many different Wikidata IDs building our vocabulary. This vocabulary can only be reduced by removing more IDs from the vocabulary, as shown in Section 5.1. Then again, reducing the vocabulary leads to less training data as sentences with too few tagged entities would have to be removed. Additionally, sentences can then solely be generated on the Wikidata IDs available in the vocabulary leading to a smaller domain of possible sentences to be generated. These issues together makes the training of our models harder.

To compare BART finetuned on a sequence of labels to the models trained on IDs, we need to finetune BART once more on different data. Since BART can use all Wikidata labels, we trained it for our first objective using all available labels. However, having much more input information at hand would not be fair compared to an approach using a reduced set of available entities. Thus, we reduce the data to finetune BART to the same entities we used to train the ID-based models.

Comparing this newly finetuned BART to the models of this section still shows clear results. The BART model finetuned on a sequence of labels reaches a BLEU score of 41.22, whereas the best model trained on a sequence of entity IDs reaches a score of 35.74 BLEU. However, the model trained on IDs does not directly incorporate the ID embeddings but translates the IDs first into labels, so that that the gained knowledge from embedding the Wikidata IDs is lost except for the context given with contextualizing the labels.

Our model, which transforms the translated label embeddings directly into an input understandable by BART, only reaches 11 BLEU and is outperformed by the BART model directly trained on labels. Thus, we conclude that our Hypothesis H_2 does not hold for the models we trained.

7

Conclusions

In this thesis, we studied the generation of natural language-like sentences based on a sequence of input facts. These input facts consist of Wikidata entities and their connecting predicates but do not necessarily need to be connected via a path in the Wikidata knowledge graph. We trained several models using a sequence of these facts to generate the matching natural language-like-like sentence. More specifically, we compared six different model architectures either trained from scratch or already pretrained but finetuned on the same task. All the trained models are based on the Transformer architecture, and we used BART as a pretrained model. Additionally, we compared two types of input sequences: a Wikidata label and a Wikidata ID-based input format.

Following these goals, we processed the freely available T-REx dataset. The dataset annotates Wikidata entities in the related Wikipedia sentences. We ended up with a training dataset containing over 4.2 million aligned sentences usable for both input sequence formats.

The first objective of this thesis focused on creating natural language-like sentences from a sequence of Wikidata labels and compared a baseline model to the finetuned BART model. Results show that it is indeed possible for both models to generate such Wikipedia like sentences. Nevertheless, finetuning the pretrained BART model on this task can outperform our baseline model by far. The finetuned model can produce more fluent sentences based on the given input facts and incorporates the given knowledge better, while making fewer errors. It is also possible to generate sentences across all Wikipedia domains and articles, not limited to the first sentence. Thus, we accept our Hypothesis that finetuning a pretrained model on our task can outperform the baseline model.

However, the finetuned model is not perfect. Missing annotated entities in the dataset and translating dates into the same input token can reduce the performance model. It should be noted that the BART model can be trained on additional input facts that are not necessarily part of a knowledge graph. These facts could be any kind of strings like proper names or, as in this case, dates.

In our second objective, we compared the sentence generation based on an input sequence of Wikidata IDs to a label-based input sequence. Since this change in the input sequence requires a change in any model's embedding layer, the embedding layer has to be replaced or retrained entirely but cannot be finetuned directly. Thus, the training on IDs makes the task harder for the trained models. Our results show that this task is indeed harder, as none of our trained models can clearly outperform our baseline. The most promising model, BART adapted for machine translation, cannot cope with the task, and none of the configurations trained learns the generation task at all. We showed that it is possible to train a Transformer based translation of IDs into their contextualized surface forms. Using the output of this translation can then be fed again into a finetuned BART model. This sequence of models performs better than the models trained on the single task of training to generate a sentence based on a sequence of Wikidata IDs. Therefore, we reject our hypothesis that the use of an ID-based input sequence can improve the sentence generation based on a label-based approach.

This conclusion contradicts the broad assumption that end-to-end trained models outperform models having a modular design. However, our tasks of translating IDs into a sequence of labels and the task of generating natural language-like sentences are both not trivial. This combination of the two tasks might lead to an overall task too hard for the models under consideration. Thus, the modular alternative, where each model is trained separately, could be the right approach to the problem.

We argue that the increase in difficulty relies on the change of the vocabulary size vital to deal with Wikidata IDs. Using IDs as the input increases the vocabulary size by each new ID in the dataset so that a limitation to specific most occurring IDs, especially for large knowledge graphs, is required. Limiting these IDs makes training sparser, resulting in less training data and a smaller domain.

Outlook We finetuned a pretrained language model on the task of generating a single sentence over a sequence of Wikidata entities. Whereas our label-based approach worked well, the ID-based approach did not. Future work should focus on the limitations of the ID-based approach in more detail. Thus, the embedded information between entities and relations of a knowledge graph could be taken into account. Such an approach might use previously constructed graph embeddings or ways to deal with larger vocabularies.

Additionally, the use of a cleaner, more densely annotated dataset might help to improve the performance. Such datasets exist but are domain-specific. This would require a new annotation algorithm that aligns knowledge graph triples with text. Being finally able to generate sentences more accurately also allows the building of applications. These applications can then be used together with a knowledge graph's truthfulness to generate true sentences over given graph paths.

References

[Alammar, 2018] Alammar, J. (2018). The illustrated transformer.

[Androutsopoulos et al., 2001] Androutsopoulos, I., Kokkinaki, V., Dimitromanolaki, A., Calder, J., Oberlander, J., and Not, E. (2001). Generating multilingual personalized descriptions of museum exhibits-the m-piro project. *arXiv preprint cs/0110057*.

[Ba et al., 2016] Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

[Bernstein and Kaufmann, 2006] Bernstein, A. and Kaufmann, E. (2006). Gino–a guided input natural language ontology editor. In *The Semantic Web - ISWC 2006*, volume 4273, pages 144–157. Springer Berlin Heidelberg.

[Chisholm et al., 2017] Chisholm, A., Radford, W., and Hachey, B. (2017). Learning to generate one-sentence biographies from wikidata. *arXiv preprint arXiv:1702.06235*.

[Dale et al., 2003] Dale, R., Geldof, S., and Prost, J.-P. (2003). Coral: Using natural language generation for navigational assistance. In *Proceedings of the 26th Australasian computer science conference-Volume 16*, pages 35–44.

[Devlin et al., 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. N. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

[Ehrlinger and Wöß, 2016] Ehrlinger, L. and Wöß, W. (2016). Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)*, 48:1–4.

[Elsahar et al., 2019] Elsahar, H., Vougiouklis, P., Remaci, A., Gravier, C., Hare, J., Simperl, E., and Laforest, F. (2019). T-rex: A large scale alignment of natural language with knowledge base triples. In *LREC 2018 - 11th International Conference on Language Resources and Evaluation*, pages 3448–3452.

[Galanis and Androutsopoulos, 2007] Galanis, D. and Androutsopoulos, I. (2007). Generating multilingual descriptions from linguistically annotated owl ontologies: the naturalowl system. In *Proceedings of the Eleventh European Workshop on Natural Language Generation (ENLG 07)*, pages 143–146.

[Glorot and Bengio, 2010] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

[Kaffee et al., 2018] Kaffee, L.-A., Elsahar, H., Vougiouklis, P., Gravier, C., Laforest, F., Hare, J., and Simperl, E. (2018). Learning to generate wikipedia summaries for underserved languages from wikidata. *arXiv preprint arXiv:1803.07116*.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Lebret et al., 2016] Lebret, R., Grangier, D., and Auli, M. (2016). Neural text generation from structured data with application to the biography domain. *arXiv preprint arXiv:1603.07771*.

[Lewis et al., 2019] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2019). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

[Lin, 2004] Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

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

[Loshchilov and Hutter, 2017] Loshchilov, I. and Hutter, F. (2017). Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101.

[Luong et al., 2014] Luong, M.-T., Sutskever, I., Le, Q. V., Vinyals, O., and Zaremba, W. (2014). Addressing the rare word problem in neural machine translation. *arXiv preprint arXiv:1410.8206*.

[Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G. S., and Dean, J. (2013). Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

[Mishra and Kumar, 2011] Mishra, R. B. and Kumar, S. (2011). Semantic web reasoners and languages. *Artificial Intelligence Review*, 35(4):339–368.

[Novikova et al., 2017] Novikova, J., Dušek, O., and Rieser, V. (2017). The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206, Saarbrücken, Germany. Association for Computational Linguistics.

[O'DONNELL et al., 2001] O'DONNELL, M., Mellish, C., Oberlander, J., and Knott, A. (2001). Ilex: an architecture for a dynamic hypertext generation system. *Natural Language Engineering*, 7(3):225–250.

[Papineni et al., 2002] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

[Post, 2018] Post, M. (2018). A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

[Puduppully et al., 2019] Puduppully, R., Dong, L., and Lapata, M. (2019). Data-to-text generation with content selection and planning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):6908–6915. Number: 01.

[Qi et al., 2020] Qi, D., Su, L., Song, J., Cui, E., Bharti, T., and Sacheti, A. (2020). Imagebert: Cross-modal pre-training with large-scale weak-supervised image-text data. *arXiv preprint arXiv:2001.07966.*

[Radford et al., 2018] Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training.

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

[Raffel et al., 2019] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683.*

[Ribeiro et al., 2020] Ribeiro, L. F., Schmitt, M., Schütze, H., and Gurevych, I. (2020). Investigating pretrained language models for graph-to-text generation. *arXiv preprint arXiv:2007.08426.*

[Singhal, 2012] Singhal, A. (2012). Introducing the Knowledge Graph: things, not strings.

[Spasojevic et al., 2017] Spasojevic, N., Bhargava, P., and Hu, G. (2017). Dawt: Densely annotated wikipedia texts across multiple languages. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17 Companion, pages 1655–1662, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

[Sripada et al., 2003] Sripada, S., Reiter, E., and Davy, I. (2003). Sumtime-mousam: Configurable marine weather forecast generator. *Expert Update*, 6(3):4–10.

[Stevens et al., 2011] Stevens, R., Malone, J., Williams, S., Power, R., and Third, A. (2011). Automating generation of textual class definitions from owl to english. In *Journal of Biomedical Semantics*, volume 2, page S5. Springer.

[Third et al., 2011] Third, A., Williams, S., and Power, R. (2011). Owl to english: a tool for generating organised easily-navigated hypertexts from ontologies. In *10th International Semantic Web Conference (ISWC 2011)*, pages 23–27.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

[Vougiouklis et al., 2018] Vougiouklis, P., Elsahar, H., Kaffee, L.-A., Gravier, C., Laforest, F., Hare, J., and Simperl, E. (2018). Neural wikipedian: Generating textual summaries from knowledge base triples. *Journal of Web Semantics*, 52:1–15.

[Wiseman et al., 2017] Wiseman, S., Shieber, S. M., and Rush, A. M. (2017). Challenges in Data-to-Document Generation. *arXiv:1707.08052 [cs]*. arXiv: 1707.08052.

[Wiseman et al., 2018] Wiseman, S., Shieber, S. M., and Rush, A. M. (2018). Learning neural templates for text generation. *arXiv preprint arXiv:1808.10122*.

[Yao et al., 2019] Yao, L., Mao, C., and Luo, Y. (2019). Kg-bert: Bert for knowledge graph completion. *arXiv preprint arXiv:1909.03193*.

A

Appendix

On the following pages we show additional information summarized in large tables. This includes the hyper-parameter adaptions the models were trained on as well as the associated results.

Setup We present all hyper-parameter settings in Table A.1. The column *In* specifies the input vocabulary size whereas the column *Out* specifies the size of the target vocabulary. *Hid. Dim.* indicates the number of hidden dimensions. The hidden dimension might be defined differently regarding different models and should be checked by using Section 5.2. Column *Downs.* reports if IDs or labels in the training sentences were downsampled. The *ID* of each model will be used in the results table as well.

Results The results of the model variations are shown in Table A.2. As in Chapter 6 we state the BLEU and ROUGE scores. The models can be linked to the hyper-parameter settings via the *ID* column. A * in a score field indicates that we could not calculate the score due to an error in the ROUGE score calculation.

Generated Sentences We present more generated sentences in Table A.3. The sentences are chosen randomly from our test set and are presented next to the original target sentence. We use the best performing model of each model category to generate the output sentence. Additionally, we show 20 different sentence variations based on the same entities and predicates in Table A.4.

Model	ID	Batch Size	Optimizer	Learning Rate	In	Out	Hid. Dim.	Samples	Downs.
Baseline Labels	bl-1	64	Adam	0.001	99146	50265	1024	10%	FALSE
	bl-2	64	Adam	0.0003	99146	50265	1024	10%	FALSE
	bl-3	64	Adam	0.00001	99146	50265	1024	10%	FALSE
Baseline Ids	bi-1	64	Adam	0.001	99146	50265	1024	10%	FALSE
	bi-2	64	Adam	0.0003	99146	50265	1024	10%	FALSE
	bi-3	64	Adam	0.00001	99146	50265	1024	10%	FALSE
BART	B-1	16	Adam	0.0001	50265	50265	1024	1%	FALSE
	B-2	16	Adam	0.00001	50265	50265	1024	1%	FALSE
	B-3	16	Adam	0.00003	50265	50265	1024	1%	FALSE
	B-4	16	Adam	0.0001	50265	50265	1024	10%	FALSE
	B-5	16	Adam	0.00003	50265	50265	1024	10%	FALSE
	B-6	16	Adam	0.00001	50265	50265	1024	10%	FALSE
	B-8	16	AdamW	0.00003	50265	50265	1024	10%	FALSE
	B-9	16	AdamW	0.00001	50265	50265	1024	10%	FALSE
	B-10	16	AdamW	0.00001	50265	50265	1024	20%	FALSE
	B-11	16	AdamW	0.000005	50265	50265	1024	20%	FALSE
	B-12	16	AdamW	0.00001	50265	50265	1024	50%	FALSE
	B-13	16	AdamW	0.000005	50265	50265	1024	50%	FALSE
BART-MT	BMT-1	16	Adam	0.001	99146	50265	1024	1%	FALSE
	BMT-2	16	Adam	0.0003	99146	50265	1024	1%	FALSE
	BMT-3	16	Adam	0.0001	99146	50265	1024	1%	FALSE
	BMT-4	16	Adam	0.001	99146	50265	1024	30%	TRUE
	BMT-5	16	Adam	0.0003	99146	50265	1024	30%	TRUE
	BMT-6	16	Adam	0.0001	99146	50265	1024	30%	TRUE
	BMT-7	16	Adam	0.00005	99146	50265	1024	30%	TRUE
	BMT-8	16	Adam	0.00001	99146	50265	1024	30%	TRUE
	BMT-9	16	Adam	0.001	99146	50265	1024	100%	TRUE

	BMT-10	16	Adam	0.0003	99146	50265	1024	100%	TRUE
	BMT-11	16	Adam	0.0001	99146	50265	1024	100%	TRUE
	BMT-12	16	Adam	0.00005	99146	50265	1024	100%	TRUE
	BMT-13	16	Adam	0.00001	99146	50265	1024	100%	TRUE
Translation	T-1	64	Adam	0.001	99146	50265	64	10%	TRUE
	T-2	64	Adam	0.0005	99146	50265	64	10%	TRUE
	T-3	64	Adam	0.0001	99146	50265	64	10%	TRUE
	T-4	64	Adam	0.001	99146	50265	128	10%	TRUE
	T-5	64	Adam	0.0005	99146	50265	128	10%	TRUE
	T-6	64	Adam	0.0001	99146	50265	128	10%	TRUE
	T-7	64	Adam	0.001	99146	50265	256	10%	TRUE
	T-8	64	Adam	0.0005	99146	50265	256	10%	TRUE
	T-9	64	Adam	0.0001	99146	50265	256	10%	TRUE
	T-10	64	Adam	0.001	99146	50265	1024	10%	TRUE
	T-11	64	Adam	0.0005	99146	50265	1024	10%	TRUE
	T-12	64	Adam	0.0001	99146	50265	1024	10%	TRUE
	T-13	64	Adam	0.001	99146	50265	64	100%	TRUE
	T-14	64	Adam	0.0005	99146	50265	64	100%	TRUE
	T-15	64	Adam	0.0001	99146	50265	64	100%	TRUE
	T-16	64	Adam	0.001	99146	50265	128	100%	TRUE
	T-17	64	Adam	0.0005	99146	50265	128	100%	TRUE
	T-18	64	Adam	0.0001	99146	50265	128	100%	TRUE
	T-19	64	Adam	0.001	99146	50265	256	100%	TRUE
	T-20	64	Adam	0.0005	99146	50265	256	100%	TRUE
	T-21	64	Adam	0.0001	99146	50265	256	100%	TRUE
	T-22	64	Adam	0.001	99146	50265	1024	100%	TRUE
	T-23	64	Adam	0.0005	99146	50265	1024	100%	TRUE
	T-24	64	Adam	0.0001	99146	50265	1024	100%	TRUE
Translation-BART-M	TBM-1	64	Adam	0.001	99146	50265	1024	1%	FALSE
	TBM-2	64	Adam	0.0003	99146	50265	1024	1%	FALSE

TBM-3	64	Adam	0.00001	99146	50265	1024	1%	FALSE
TBM-4	64	Adam	0.001	99146	50265	1024	10%	FALSE
TBM-5	64	Adam	0.0003	99146	50265	1024	10%	FALSE
TBM-6	64	Adam	0.00001	99146	50265	1024	10%	FALSE
TBM-7	64	Adam	0.00001	99146	50265	512	10%	FALSE

Table A.1: Parameter settings of all models

Model	ID	BLEU	ROUGE-1	ROUGE-2	ROUGE-l
Baseline Labels	bl-1	0	0	0	0
	bl-2	1.123765	19.47139	5.693975	19.14092
	bl-3	15.34044	45.95129	28.58206	45.82768
Baseline Ids	bi-1	0	0	0	0
	bi-2	1.907134	16.55452	5.332989	16.38419
	bi-3	11.54714	41.0104	23.69338	40.74124
BART	B-1	0.02532	13.73772	0.360986	16.61815
	B-2	28.9671	55.36641	36.33771	54.67537
	B-3	30.43606	56.11096	37.52874	55.34203
	B-4	0.00438	4.976551	0.003736	8.756074
	B-5	0.157377	*	*	*
	B-6	36.23535	60.68445	42.935	60.01353
	B-8	37.95499	62.31538	44.89822	61.72198
	B-9	39.62515	63.51268	46.54062	62.74191
	B-10	38.45631	62.68675	45.23251	61.97608
	B-11	39.79686	63.66637	46.63661	62.90538
	B-12	42.92034	65.68538	49.39549	64.97764
	B-13	42.0648	64.99611	48.58221	64.36054
BART-MT	BMT-1	0.015506	11.6161	0.049183	11.34021
	BMT-2	0.188195	10.23326	0.486224	11.50871
	BMT-3	0.015506	11.6161	0.049183	11.34021
	BMT-4	0.188195	10.23326	0.486224	11.50871
	BMT-5	0.188195	10.23326	0.486224	11.50871
	BMT-6	0.00841	7.600451	0.012089	8.0874
	BMT-7	0.188195	10.23326	0.486224	11.50871
	BMT-8	0.1077	13.98067	0.227796	12.54011
	BMT-9	0.188195	10.23326	0.486224	11.50871
	BMT-10	0.188195	10.23326	0.486224	11.50871
	BMT-11	0.00841	7.600451	0.012089	8.0874
	BMT-12	0.188195	10.23326	0.486224	11.50871
	BMT-13	0.1077	13.98067	0.227796	12.54011
Translation	T-1	1.564703	4.917149	1.93553	5.115813
	T-2	1.984104	6.690321	2.867255	7.004961
	T-3	0.029495	*	*	*
	T-4	0	0	0	0
	T-5	2.834947	8.42017	3.740959	8.810606
	T-6	0.895954	5.28394	1.601399	6.423045
	T-7	0	0	0	0
	T-8	1.629299	5.813638	2.347951	6.363033
	T-9	2.893042	11.63534	4.634781	12.38123

	T-10	0	0	0	0
	T-11	0	0	0	0
	T-12	17.32067	35.87757	22.15259	37.54379
	T-13	35.4092	57.11204	40.98437	58.23007
	T-14	38.24787	59.85558	43.71784	60.70243
	T-15	19.4245	39.38167	24.03838	40.05311
	T-16	7.163043	17.59693	9.415023	17.86017
	T-17	42.26839	63.08361	47.515	64.01119
	T-18	41.67328	63.21315	47.14553	63.95228
	T-19	0	0	0	0
	T-20	16.43847	33.67716	20.46892	33.65074
	T-21	50.58442	71.03164	56.24096	71.83572
	T-22	0	0	0	0
	T-23	0	0	0	0
	T-24	53.37352	72.64361	58.32042	73.73855
Translation-BART-M	TBM-1	0.026357	11.09486	0.22098	10.72951
	TBM-2	0.16423	12.57257	1.186031	12.41652
	TBM-3	9.867016	32.76723	15.93545	32.51841
	TBM-4	0.106863	13.05058	0.79484	12.36823
	TBM-5	0.191977	13.22733	1.014918	12.75595
	TBM-6	11.75899	35.72452	18.38085	35.55134
	TBM-7	11.98177	36.37637	18.96035	36.25002

Table A.2: BLEU and ROUGE scores of all model variations

Source IDs	Q950205 Q81960 Q2736 Q1384 Q1916087
Source Labels	West New York Robert Burns association football New York National Association Football League
Target	The West New York Burns Club were a professional soccer club from New York which played in the National Association Football League .
bl-3	The West New York Football Association season was the first season of association football in the National Football League, the New York Football Association, the National Football Association, and the National Football Association.
B-12	Born in West New York, New York , Burns played professional football in New York in the National Association Football League .
bi-3	The New York State Legislature is the oldest football club in New York, and the National Association Association Association.
T-24	West New York, New York Robert Burns football New York national football team
BM-7	Born Born in West New York, New York, he attended the University of New York at West New York, where he played soccer for the New York State soccer team and the New York State national soccer
Trans. - BART	Born in West New York, New York , Robert Burns played football for the New York national football team .
Source IDs	P527 Q235858 Q332953 Q1164775 Q1427551 Q525314 Q1145652
Source Labels	has part folk music Wizz Jones Bert Jansch Danny Thompson Pentangle Gerry Conway Fotheringay Dave Mattacks Fairport Convention
Target	Guest musicians include folk pioneers Wizz Jones ; Bert Jansch and Danny Thompson from Pentangle ; Gerry Conway from Fotheringay ; and Dave Mattacks from Fairport Convention .
bl-3	Other influences include folk, Wans Jones, Bert Jones, Peter Jones, Peter Jones, David Jones, David Jones, David Jones, David Jones, and Dave Fairport.
B-12	The members of the folk trio were Wizz Jones , Bert Jansch , Danny Thompson , Pentangle , Gerry Conway , Fotheringay , Dave Mattacks and Fairport Convention .
bi-3	Other notable musicians include folk, Jans, Jans, Jans, Jansi Lee, John Lee, John Lee Lee, John Lee Lee, John Lee Lee Lee Lee, John Lee Lee
T-24	include folk Bert Jansch Danny Thompson Pentecostal Pentangle Fairport Convention

BM-7	The album includes songs by musicians such as John Jakes, John Prine, John Cale, John Prine, and Bob Welch.
Trans. - BART	His influences include folk singers Bert Jansch and Danny Thompson , Pentecostal minister Pentangle and Fairport Convention .
Source IDs	P279 Q11399 Q380971 Q23757
Source Labels	subclass of rock music metamorphism limestone
Target	Marble is a type of rock resulting from the metamorphism of limestone .
bl-3	A type of rock is a type of acoustic limestone.
B-12	Limestone is a type of rock formed by the metamorphism of limestone .
bi-3	A type of rock, the white-finned fish is found in a limestone.
T-24	is a type of rock metamorphosed Limestone
BM-7	The Lusitania is a type of rock that is formed when the lithosphere reacts with the lithosphere to form a solid rock.
Trans. - BART	The Kermadec Formation is a type of rock metamorphosed from the Kermadi Limestone .
Source IDs	Q757297 P2308 Q174736 Q172771
Source Labels	Her Majesty's Ship class destroyer Royal Navy
Target	HMS Locust was a B - class torpedo boat destroyer of the British Royal Navy .
bl-3	HMCS Navy (DD -) was a destroyer of the Royal Navy.
B-12	HMS Porthole was a Porthos - class destroyer of the Royal Navy .
bi-3	HMS HMS Great Britain was a HMS Great Britain - class destroyer of the Royal Navy.
T-24	HMS class destroyer Royal Navy
BM-7	HMS Lusitania was a ship of the Royal Navy, a sailing frigate.
Trans. - BART	HMS Dorset was a Dorset - class destroyer of the Royal Navy .
Source IDs	P19 Q43475 Q288728 Q43475
Source Labels	place of birth Derby house painter Derby
Target	Bradley was born in Derby and became a house painter living in Derby .
bl-3	William H. P. P. P. P. was born in Derby, the house of Derby.
B-12	John Houghton (painter) was born in Derby, England and worked as a house painter in Derby .
bi-3	Henry H. Smith was born in Derby, Derby, England, and was educated at Derby Grammar School.

T-24	born in Derby, England Holidays Derby
BM-7	John Houghton was born in London, England and attended the King Edward VI School, London.
Trans. - BART	Houghton was born in Derby, England and attended Holidays in Derby .
Source IDs	XMLSchema#dateTime Q57278 P31 Q10962 Q2736 Q3374818 Q1140630
Source Labels	XMLSchema#dateTime Newcastle instance of rugby league association football Penrith Panthers National Rugby League
Target	Chris Armit (born 7 October 1983 in Newcastle, New South Wales) is an Australian professional rugby league footballer for the Penrith Panthers of the National Rugby League .
bl-3	"The "" O""" Newcastle (born 1 January 1990 in Newcastle, New South Wales) is a professional rugby league footballer for the Penrith Panthers in the National Rugby League."
B-12	Nathan Ryan (born 24 September 1990 in Newcastle, New South Wales) is an Australian professional rugby league footballer for the Penrith Panthers of the National Rugby League .
bi-3	David Thomas (born 6 May 1974 in Newcastle, New South Wales) is an Australian professional rugby league footballer who played in the National Rugby League.
T-24	XMLSchema#dateTime Newcastle, New South Wales is an rugby league football Penrith Panthers
BM-7	"Benjamin "" Ben "" Stagg (born 24 February 1990 in Sydney, New South Wales) is an Australian rugby league footballer for the Penrith Panthers in the National Rugby League (NRL
Trans. - BART	Nathan Pezzullo (born 1 January 1991 in Newcastle, New South Wales) is an Australian professional rugby league football centre for the Penrith Panthers of the National Rugby League (NRL).
Source IDs	P31 XMLSchema#dateTime Q1054574 P176 Q267282 P161 Q56016
Source Labels	Alice Adams instance of XMLSchema#dateTime romance film manufacturer RKO Pictures cast member Katharine Hepburn
Target	Alice Adams is a 1935 romantic film made by RKO , starring Katharine Hepburn .
bl-3	Alice Adams (born Alice Adams on May 8, 1978) is a Japanese romantic drama film starring RKO, directed by RKO Radio Pictures, starring Katharine Hepburn, and Hepburne.
B-12	Alice Adams is a 1949 date night romantic drama film made by RKO Radio Pictures and starring Katharine Hepburn .

bi-3	The Last Man is a 1938 romantic drama film made by RKO Radio Pictures starring Hepburn, Hepburn and Hepburn.
T-24	is a XMLSchema#dateTime romantic drama film made by RKO Katharine Hepburn Kath
BM-7	The Man Who Couldn't Say No is a 1963 British crime drama film made by R. E. Coyote Productions, with a screenplay by John G. McGowan, a screenplay by John
Trans. - BART	The Man from Another Country is a 1937 romantic drama film made by RKO , featuring Katharine Hepburn as Kath .
Source IDs	Q10460 P31 Q2389789 Q1166 P1480 Q952097
Source Labels	Big Dipper instance of steel roller coaster Michigan sourcing circumstances Muskegon
Target	Big Dipper is a steel roller coaster at Michigan ' s Adventure near Muskegon, Michigan .
bl-3	Big Big Brother is a steel roller coaster located near Muskegon, Michigan.
B-12	Big Dipper is a steel roller coaster located at the Michigan State Fairgrounds near Muskegon, Michigan .
bi-3	LZZ. Lax is a steel roller coaster located near Muskegon, Michigan.
T-24	Big Dipper is a steel family Michigan near Muskegon, Michigan
BM-7	The Big Lake is a lake in Michigan, located in the Upper Peninsula of Michigan, near the towns of Lapeer and Lapeer Lake.
Trans. - BART	The Big Dipper is a steel family structure located on the Michigan River near Muskegon, Michigan .
Source IDs	P607 P812 Q1117074 Q39614 P571 Q214504
Source Labels	conflict academic major Commonwealth War Graves Commission cemetery inception Labuan
Target	Following the war , a major Commonwealth War Graves Commission cemetery was established on Labuan .
bl-3	The war was a major Commonwealth War Graves Commission cemetery established in 1894.
B-12	After the war , a major Commonwealth War Graves Commission cemetery was established at Labuan .
bi-3	The battle was a major war cemetery, established in the Malay community.
T-24	war major Commonwealth war grave cemetery created Labuan

BM-7	The battle was the last major battle of the First World War and the first battle of the World War I. The battle was the first battle in the newly created Battle of the Bulge.
Trans. - BART	After the war , a major Commonwealth war grave cemetery was created in Labuan .
Source IDs	P20 Q118958 Q31 XMLSchema#dateTime
Source Labels	place of death Leuven Belgium XMLSchema#dateTime
Target	Roque died in Leuven , Belgium on July 27, 2006 .
bl-3	John H. L. L. L. L. A.C.
B-12	Jean-Baptiste-Louis-Marie-Joseph-Joseph de Vos died in Leuven , Belgium on April 1, 1869 .
bi-3	Paul van der Leu died in Leuven, Belgium in 1891.
T-24	died in Leuven Belgium XMLSchema#dateTime
BM-7	J. J. B. died in Leuven, Belgium on 30 July 2014.
Trans. - BART	De Vries died in Leuven , Belgium on March 1, 2015 .
Source IDs	Q1533406 P31 XMLSchema#dateTime Q506240 P57 Q122020
Source Labels	Go Down Moses instance of XMLSchema#dateTime television film director Hilary Duff
Target	The Bear is a 1998 short animated television film directed by Hilary Audus .
bl-3	""" Go "" is a 1989 Japanese film written and directed by Jessica K. S. He also written and featuring Mario as the film director of Hilary."
B-12	Go Down Moses is a 1995 television movie directed by Hilary Duff .
bi-3	The Damned, The Red River, is a 2005 television movie directed by Hilary.
T-24	The Bear is a XMLSchema#dateTime made-for-TV movie directed by Hilary
BM-7	The Last of the Mohicans is a 2007 British made-for-television film directed by James P. Hogan and written and directed by James P. Hogan.
Trans. - BART	The Bear is a 1994 made-for-TV movie directed by Hilary B . Smith .
Source IDs	Q985257 P31 Q1577187 Q5602826
Source Labels	Rio Vista instance of San Diego Trolley Green Line Coaches
Target	Rio Vista is a station on the San Diego Trolley ' s Green Line .
bl-3	Rio Vista is a station on the San Diego Vista Line.
B-12	Rio Vista is a San Diego Trolley station served by the Green Line .
bi-3	The Phoenix is a station on the San Diego Green Line of the Green Line.

T-24	Rio Vista is a San Diego Trolley Green Line
BM-7	The San Diego Zoo is a zoo in San Diego, California, located at the corner of Tilden Avenue and Mission Boulevard in the Tilden neighborhood of San Diego, California.
Trans. - BART	Rio Vista is a station on the San Diego Trolley Green Line .
Source IDs	Q1215892 Q203008 Q194116
Source Labels	Kevin Schamehorn National Hockey League Los Angeles Kings Detroit Red Wings
Target	Kevin Schamehorn played in 10 NHL games with the Los Angeles Kings and Detroit Red Wings .
bl-3	Kevin Schameckley played in the National Hockey League with the Los Angeles Kings and Detroit Red Wings.
B-12	Schamehorn played in the National Hockey League (NHL) with the Los Angeles Kings and Detroit Red Wings .
bi-3	John A. K. K. K. K. K.C. played in the National Hockey League with the Los Angeles Kings, Detroit Red Wings and Detroit Red Wings.
T-24	National Hockey League (NHL) Los Angeles Kings Detroit Red Wings
BM-7	The 1998 National Hockey League season was the 49th National Hockey League season, and the 49th season for the Los Angeles Kings and the second season for the Anaheim Ducks.
Trans. - BART	Prior to that , he played in the National Hockey League (NHL) with the Los Angeles Kings and Detroit Red Wings .
Source IDs	Q38935 P31 Q482994 Q315092 Q38848 P175 Q432319 Q83270 P175
Source Labels	SpaceShipOne instance of album Paul Gilbert heavy metal performer Speed Racer hard rock performer
Target	Space Ship One is a studio album by Paul Gilbert formerly of the heavy metal band Racer X and the hard rock band Mr .
bl-3	Space : Space is a studio album by the heavy metal band Paul McCartney, formed in 2005 in hard rock band's hard rock band, formed in 2005.
B-12	Space Ship One is a studio album by Paul Gilbert , the guitarist of the heavy metal band Racer X and the hard rock band The Damned .
bi-3	WG is a studio album by the heavy metal band X - X - X - X - X - LP, and hard rock band X.

T-24	Space Ship One is a studio album Paul Gilbert heavy metal band Racer X band Racer Mach
BM-7	The War of the Worlds is a studio album by the British heavy metal band, The Stone Roses.
Trans. - BART	Space Ship One is a studio album by Paul Gilbert , formerly of the heavy metal band Racer X and the band Racer Mach .
Source IDs	Q1349704 Q43627 Q1349704 Q1616095 Q75687 Q248
Source Labels	Parallels Desktop for Mac Mac OS Parallels Desktop for Mac hardware virtualization Apple Macintosh Intel
Target	Parallels Desktop for Mac , by Parallels , is software providing hardware virtualization for Macintosh computers with Intel processors .
bl-3	Parvair-KX-KX-KX-KX-KX-KX-KX-KX-2 is the Mac OS X-2 software for the Apple
B-12	Parallels Desktop for Mac is the Parallel Desktop for Macintosh hardware virtualization software for Apple Macintosh computers with Intel processors .
bi-3	The Madurai & Mac OS X Network, and the Intel Web server for the Apple Macintosh.
T-24	Parallels Mac OS Parallels hardware virtualization Apple Macintosh Intel
BM-7	The two competing operating systems are the Microsoft Windows-based operating systems, Windows NT and Windows CE, and the proprietary operating system, Windows CE.
Trans. - BART	Parallels (formerly Mac OS ParalleLS) is hardware virtualization software for Apple Macintosh computers running on Intel processors .
Source IDs	Q27686 Q1424786 P2825 Q188966
Source Labels	hotel Four Seasons Hotels and Resorts via contract bridge
Target	The hotel is linked to Four Seasons Hotel via a bridge .
bl-3	The hotel is located on the Four Seasons bridge via the bridge.
B-12	The hotel is connected to the Four Seasons Hotel Montreal via a bridge .
bi-3	The hotel is located on the Four Seasons Hotel, via the bridge.
T-24	hotel Four Seasons via bridge
BM-7	The hotel is located at 595 West 5th Street, Suite 101, in the West End of Manhattan.
Trans. - BART	The hotel is connected to Four Seasons via a bridge .
Source IDs	Q1143416 P31 Q308439 Q61077 P47 P131 Q1143416 P36

Source Labels	Hot Springs National Park instance of National Park Service Garland County shares border with located in the administrative territorial entity Hot Springs National Park capital
Target	Hot Springs National Park is a United States National Park in central Garland County, Arkansas , adjacent to the city of Hot Springs , the county seat .
bl-3	Hot Springs National Park is a National Park Service, located on the border of the city of Marion Springs and the county seat of Hot Springs.
B-12	Hot Springs National Park is a United States National Park in Garland County, Arkansas , adjacent to the city of Hot Springs , the county seat .
bi-3	Hot Springs (formerly Hot Springs and Hot Springs) is a National Park Service park located on the northwest shore of the Arkansas River, on the town of Hot Springs, on the capital.
T-24	Hot Springs is a National Park Service Garland County next to Lake Monmouth County
BM-7	The National Wildlife Refuge is a National Wildlife Refuge in the United States National Park System in the western portion of the U. S., located adjacent to the city of Carson in the San Joaquin Valley
Trans. - BART	Hot Springs State Park is a National Park Service park in Garland County , next to Lake Monmouth County .
Source IDs	P31 P2670 Q33260 P1412 Q1239
Source Labels	instance of has parts of the class French-based creole languages languages spoken, written or signed Indian Ocean
Target	Bourbonnais Creole is a group of French-based creole languages spoken in the western Indian Ocean .
bl-3	BÃ©lÃ© is a group of language spoken in the French language spoken in the Indian Ocean.
B-12	The Reunionan languages is a group of French-based creole languages spoken in Reunion , in the Indian Ocean .
bi-3	The Indian Sea (NAN) is a group of low-based languages spoken in the Indian Ocean.
T-24	is a consists of French creole language spoken Indian Ocean
BM-7	The Kumbh Mela language is a group of languages that are related to the Indo-Aryan language of the Indian sub - Saharan Indian sub - region.

Trans. - BART	The Reunionan Creole is a small community that consists of speakers of a French creole language spoken along the Indian Ocean coast .
Source IDs	Q219640 P31 P51 Q11399 P175 Q101505
Source Labels	Vincent Price instance of audio rock music performer Deep Purple
Target	””” Vincent Price ”” is a song by British rock band Deep Purple .”
bl-3	””” Vincent ”” is a song by the rock band Deep Purple.”
B-12	””” Vincent Price ”” is a song by British rock band Deep Purple .”
bi-3	”Philip Vincent ”” is a song by the rock band Deep Purple.”
T-24	Vincent Price is a song rock band Deep Purple
BM-7	””” The Last of the Mohicans ”” is a song by the British rock band The Who.”
Trans. - BART	””” Vincent Price ”” is a song by British rock band Deep Purple .”
Source IDs	Q18656 XMLSchema#dateTime Q18656 Q6465 Q23077 Q163 P834 Q176084
Source Labels	Manchester United F.C. XMLSchema#dateTime Manchester United F.C. department of France Lancashire Yorkshire train depot Newton Heath
Target	Manchester United Football Club was formed in 1878 as Newton Heath LYR Football Club by the Carriage and Wagon department of the Lancashire and Yorkshire Railway depot at Newton Heath .
bl-3	Manchester United, also referred to as Manchester United, as an independent, is the first baseman for the Manchester United States, Manchester United States, Lancashire, and Lancashire.
B-12	Manchester United Football Club (XML Schema# Date = 1892) was formed in 1892 as Manchester United and was originally based in the department store in Lancashire , later moving to Yorkshire and building a depot in Newton Heath .
bi-3	Manchester United was formed in 1894 as Manchester United's department centre for Lancashire depot, and was an early depot for the Newton Heath.
T-24	Manchester United XMLSchema#dateTime Newton Heath department Lancashire Yorkshire depot Newton Heath Newton Heath
BM-7	The 2010 ' s 11 English football season is the 90th season of English football in England, and the 90th season in the region of the Football League in England.

Trans. - BART	Manchester United ' s 1923 Newton Heath department store was a Lancashire and Yorkshire depot at Newton Heath , with the club ' s headquarters also being in Newton Heath .

Table A.3: Generated sentences from best model of each model category

Input	Germany is a country part of Western Europe capital Berlin inception XMLSchema#dateTime located in or next to body of water Spree
Output	Germany is a country in the northern part of Western Europe , with its capital city Berlin , which was founded on the shore of the Spree .
Input	Germany is a country part of Western Europe located in or next to body of water Spree capital Berlin inception XMLSchema#dateTime
Output	Germany is a country in the northern part of Western Europe , on the shore of the Spree River , and its capital is Berlin , which was founded on the same day as the German calendar (1801).
Input	Germany part of Western Europe is a country capital Berlin inception XMLSchema#dateTime located in or next to body of water Spree
Output	Germany , part of Western Europe , is a federal state whose capital city , Berlin , was founded on the shore of the Spree .
Input	Germany capital Berlin located in or next to body of water Spree inception XMLSchema#dateTime part of Western Europe
Output	Germany ' s capital , Berlin , on the shore of the Spree , was founded on May 1, 1801 , and is the oldest continuously inhabited part of Western Europe .
Input	Germany capital Berlin located in or next to body of water Spree inception XMLSchema#dateTime
Output	Germany ' s capital , Berlin , on the shore of the Spree , was founded on the same date .
Input	Germany capital Berlin located in or next to body of water Spree
Output	Germany ' s capital , Berlin , is located on the shore of the Spree .
Input	Germany capital Berlin part of Western Europe inception XMLSchema#dateTime
Output	Germany ' s capital city , Berlin , is part of Western Europe and was the first European city to have established a system of time zones .
Input	Germany is a country capital Berlin inception XMLSchema#dateTime
Output	Germany is a federal state with its capital in Berlin , which was established on 1 January 1949 (UTC - 5).
Input	Berlin capital Germany inception XMLSchema#dateTime
Output	Berlin , the capital city of Germany , was established on 1 January 1838 (UTC + 01 : 00).

Input	Berlin capital of Germany inception XMLSchema#dateTime
Output	Berlin , the capital of Germany , was founded on April 1, 1619 (UTC + 01 : 00).
Input	Berlin located in or next to body of water Spree capital of Germany inception XMLSchema#dateTime
Output	Berlin , on the shore of the Spree , was the capital of Germany from 1801 to 1945 , when it was incorporated into the Federal Republic .
Input	Berlin located in or next to body of water Spree capital Germany inception XMLSchema#dateTime
Output	Berlin , on the shore of the Spree , is the capital city of Germany and was founded on the same date as the world .
Input	Berlin capital of Germany inception XMLSchema#dateTime located in or next to body of water Spree
Output	Berlin , the capital of Germany , was founded on April 1, 1771 , on the shore of the Spree .
Input	Berlin capital Germany inception XMLSchema#dateTime located in or next to body of water Spree
Output	Berlin , the capital city of Germany , was founded on April 1, 1793 , by the German Emperor on the shore of the Spree .
Input	Western Europe part of Germany capital Berlin
Output	The region of Western Europe is part of Germany and its capital city is Berlin .
Input	Berlin inception XMLSchema#dateTime capital Germany
Output	Berlin was established on 1 January 1701 as the capital city of Germany .
Input	Berlin inception XMLSchema#dateTime capital Germany is a country
Output	Berlin was established on 1 January 1949 (UTC - 5) as the capital city of Germany , the federal state .
Input	Berlin inception XMLSchema#dateTime capital of Germany is a country
Output	Berlin was established on 1 January 1949 as the capital of Germany , and is the country ’ s largest city .
Input	is a country part of Western Europe capital Berlin
Output	The country is part of Western Europe and its capital city is Berlin .

Input	inception XMLSchema#dateTime Berlin capital of Germany
Output	The company was established in 2006 and is headquartered in Berlin , the capital of Germany .
Input	XMLSchema#dateTime inception Berlin
Output	The DST was established in Berlin in 1892 .
Input	Berlin capital of Germany
Output	Berlin is the capital of Germany .
Input	Berlin capital Germany
Output	Berlin is the capital city of Germany .
Input	Germany capital Berlin
Output	Germany ' s capital city is Berlin .
Input	Germany capital of Berlin
Output	Germany ' s capital of Berlin is located here .
Input	Berlin located in or next to body of water Spree
Output	Berlin is located on the shore of the Spree .
Input	Spree located in or next to body of water Berlin
Output	The Spree is located on the shore of the Berlin River .
Input	Germany part of Western Europe
Output	Germany is considered part of Western Europe .
Input	Western Europe part of Germany
Output	In Western Europe , it is part of Germany .

Table A.4: Sentence variations based on the same entities and predicates

List of Figures

List of Tables