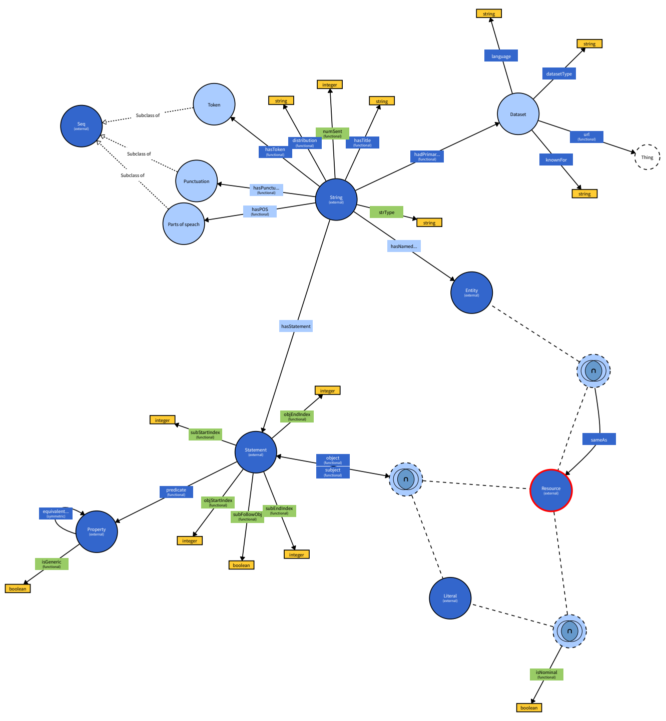
Practical Machine Learning and Deep Learning

Fall Semester 2024-2025



Entity Linking and Named Entity Recognition

Project Deliverable D1.3

Anton Kudryavtsev **Sofya Tkachenko** **Anatoly Soldatov**

27 october 2024

Table of contents

1 - What has been done so far 3
2 - Research 3
    2.1 - Relation Extraction 3
3 - Experiments 5
    3.1 - Bidirectional LSTM 5
    3.2 - BERT 7
4 - Distribution of work 7
5 - Plans 7
Bibliography 8

1 - What has been done so far

For this week we aimed to implement and compare:
1. A better PyTorch version of a bidirectional LSTM;
2. A transformer model with classification head.

Apart from that, we have also heavily focused on the research. But more on that in the following sections.

2 - Research

On our weekly meeting, we have analyzed and discussed the state-of-the-art approaches, published on Papers with code [1]. The initial list of architectures that we found useful for us was the following:
1. Simple bidirectional LSTM for NER classification;
2. BERT variations for NER classificaton;
    1. RuBERT;
    2. BERT fine-tune with zero-shot learning;
    3. BERT fine-tune with LORA;
3. NER entity linking with Graph Neural Networks (GNN);
4. Entity linking and relation extraction based on Transformer attention.

As you can see, the list is quite broad, as well as very on-the-surface. For that reason, after the meeting, we conducted even more individual research, delving into original papers and comparing our findings.

2.1 - Relation Extraction

Relation Extraction is a task to predict a relation r between two entities e_i and e_j from some known set $r \in R$. Many AI problems involve solving such task, including Information Retrieval, Knowledge Graph Construction, Knowledge Discovery, Automatic Text Reasoning, Semantic Parsing and more.

During these two weeks, we have been studying the literature on Relation Extraction, an important part of our project. We have paid special attention to the following sources:
1. EntQA: Entity Linking as Question Answering [2]
2. REBEL: Relation Extraction By End-to-end Language generation [3]
3. ReLiK: Retrieve and LinK, Fast and Accurate Entity Linking and Relation Extraction on an Academic Budget [4]

Traditional methods for relation extraction include two steps: named entity recognition and relation extraction. In the first step, entities are extracted from the text and combined in pairs. Then, relations are fetched from a knowledge base. However, this approach has some problems, such as the need for a prepared data source and the difficulty of retrieving relations without knowing the entities due to entity ambiguity.

The first paper, “EntQA: Entity Linking as Question Answering” [2], addresses these issues by proposing a two-step algorithm with a bi-encoder model. In the first step, the first encoder performs inference on the text to generate questions related to the entities. Then, the second encoder processes the same text to retrieve answers (if available) from the text. This method has several advantages. First, it does not require a knowledge base, and if pretrained, it can work with just the prompt text provided. Secondly, it solves the “answer-in-the-future” problem, which occurs when a reference between entities appears at a distance from the original entity mention.

The second paper, “REBEL: Relation Extraction by End-to-End Language Generation” [3] proposes another solution. The authors use a sequence-to-sequence model that generates triplets (e_i, e_j, r), identifying relations in the text. By using this technique, the authors managed to achieve state-of-the-art results in relation extraction tasks while extracting over 200 different relation types. However, this model is based on BART-Large, which has a significant impact on the computational requirements for training and evaluation.

The third paper, "Relik: Retrieve and Link, Fast and Accurate Entity Linking and Relation Extraction on an Academic Budget", [4], is the most similar to our project and addresses the same goal. The core idea of the proposed solution is to improve the methodology proposed in [2]. The authors implemented a model that solves both Entity Linking and Relational Extraction tasks in a single pass. This model uses a bi-encoder architecture with Reader and Retrieval encoders, respectively. The model retrieves candidate entities and relations, reads and contexts the text and candidates, then links and extracts the entities and triples, providing a final set of extracted relationships and entities.

We experimented with "ReLiK" [2] model and found it to be a landmark for this project. Example of running Relik model of proposed earlier example is the following:

Input text:

> *Catherine II, also known as Catherine the Great, was the Empress of Russia from 1762 until her death in 1796. She is considered one of the most renowned and longest-ruling female leaders in Russian history. Catherine was a proponent of Enlightened Absolutism, which advocated for the use of absolute power to bring about reforms inspired by the Enlightenment. She corresponded with notable figures like Voltaire and Diderot. Leonhard Euler was invited to the Imperial Russian Academy of Sciences by Catherine II. Leonhard Euler significantly contributed to mathematics and graph theory during his time in Russia. His achivements include Euler's formula for polyhedra and solution of Königsberg Bridge Problem.*

Model used: `relik-ie/relik-cie-xl`

Output with extracted entities:

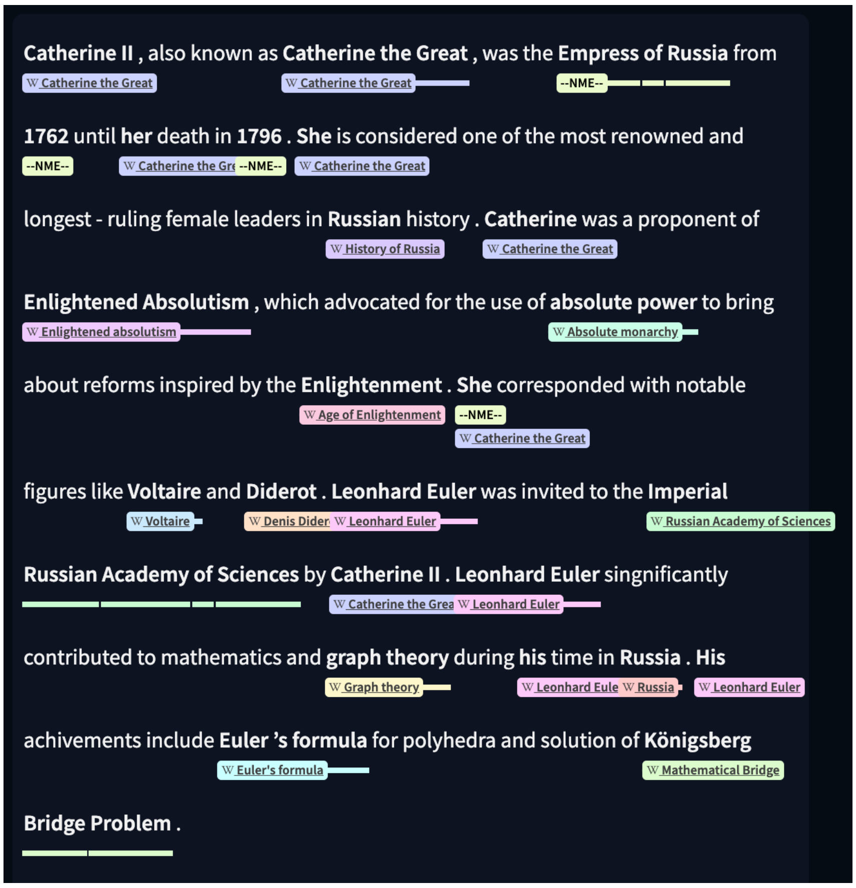

Catherine II , also known as **Catherine the Great** , was the **Empress of Russia** from **1762** until **her** death in **1796** . **She** is considered one of the most renowned and longest - ruling female leaders in **Russian** history . **Catherine** was a proponent of **Enlightened Absolutism** , which advocated for the use of **absolute power** to bring about reforms inspired by the **Enlightenment** . **She** corresponded with notable figures like **Voltaire** and **Diderot** . **Leonhard Euler** was invited to the **Imperial Russian Academy of Sciences** by **Catherine II** . **Leonhard Euler** significantly contributed to mathematics and **graph theory** during **his** time in **Russia** . **His** achivements include **Euler 's formula** for polyhedra and solution of **Königsberg Bridge Problem** .

Figure 1 - Entity Extraction by ReLiK model

Output graph with extracted relations:

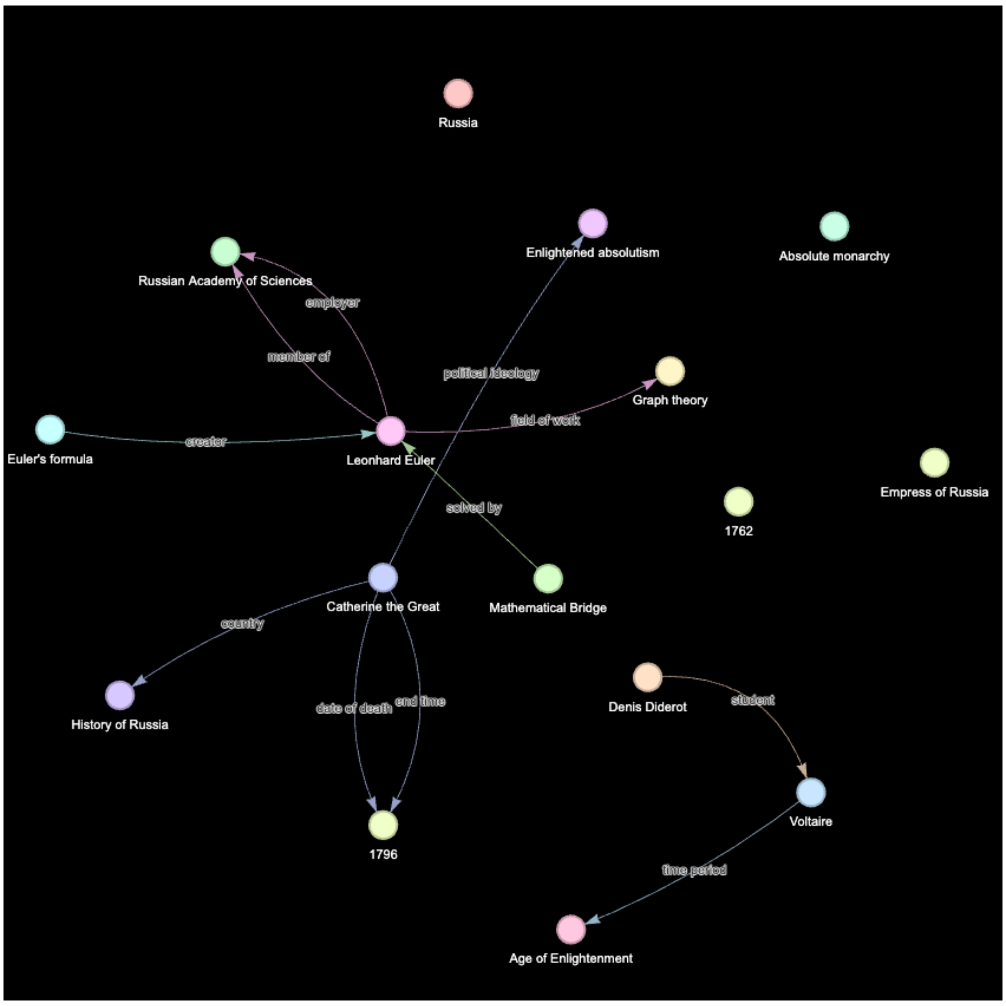


Figure 2 - Relation Extraction by ReLiK model

3 - Experiments

3.1 - Bidirectional LSTM

During the previous week, we have implemented an LSTM-based solution on Tensorflow. It's accuracy seemed sufficient (91.12%), yet the test results themselves seemed off. Moreover, as we later found out, if we update the version of Tensorflow on Kaggle, accuracy drops by another 20%! That being said, we decided to rewrite our previous implementation using PyTorch. This allowed us to prevent the situation with version dependencies, as well as gain more control over the training process.

After some experimentation, we have also decided to simplify the overall architecture, going from this in Tensorflow version:

```
vector_size = 128

i = Input(shape=(max_length,))
x = Embedding(input_dim=V+1, output_dim=vector_size, mask_zero=True)(i)
x = Dropout(0.2)(x)
x = Bidirectional(LSTM(256, return_sequences=True, recurrent_dropout=0.2))(x)
x = Bidirectional(LSTM(128, return_sequences=True, recurrent_dropout=0.2))(x)
x = TimeDistributed(Dense(K, activation='softmax'))(x)
```

Listing 1 - Tensorflow Implementation of Bi-LSTM

To the following architecture in the PyTorch version:

```
embed_dim = 128

embedding = nn.Embedding(input_dim, embed_dim, padding_idx=pad_idx)
dropout = nn.Dropout(dropout)
lstm    =    nn.LSTM(embed_dim,    hidden_dim,    num_layers=1,    bidirectional=True,
dropout=dropout)
linear = nn.Linear(hidden_dim * 2, output_dim)
```

Listing 2 - PyTorch Implementation of Bi-LSTM

As you can see, the number of trainable parameters has decreased significantly. Also, redundant second LSTM has been removed. Also, instead of training on a set number of epochs, we used early stopping with patience 10.

Here are the results of the PyTorch implementation:

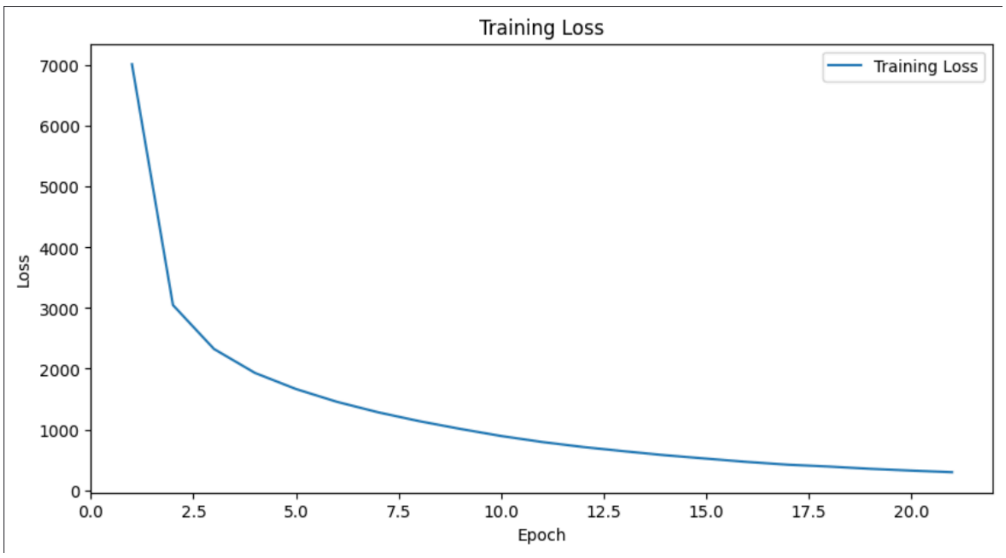


Figure 3 - LSTM train loss

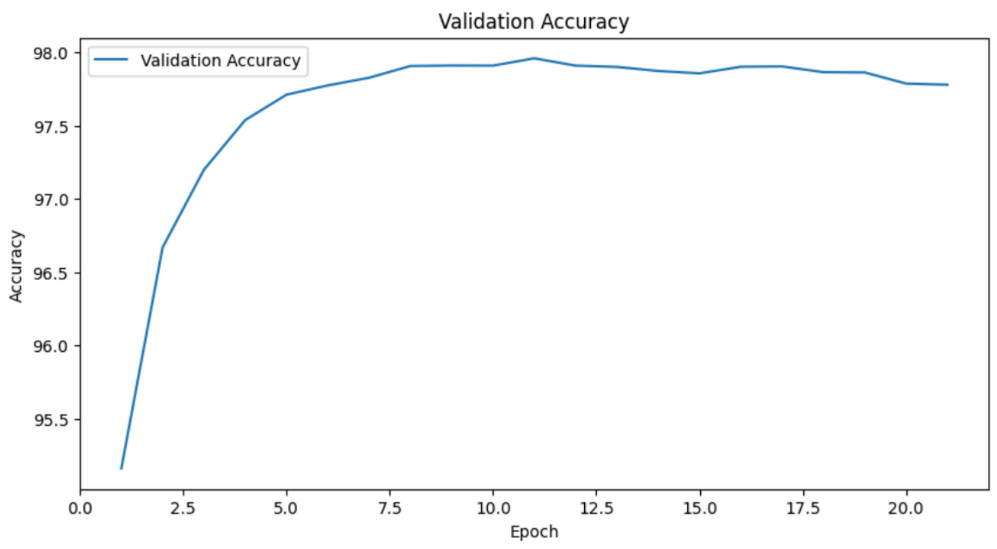


Figure 4 - LSTM validation accuracy

The final accuracy was 97.8%.

3.2 - BERT

During the experiments, the 2 approaches were used. First of all, we tried vanilla `distillbert` model on `wnut_17` dataset consisting of NER tasks. The results were quite well. After that we tried to fine-tune via LoRA on the train part of mentioned data and we gain improvement for accuracy in several percents.

4 - Distribution of work

- Anton - Entity Linking, Relation Extraction models research, report
- Sofya - Bi-LSTM based models research, report
- Anatoly - BERT based models research

5 - Plans

- Conduct more experiments with other approaches such as Zero-shot BERT, RuBERT, etc.
- Combine NER and Entity Linking tasks
- Try gain info from tranformer's attention blocks to form relations between entities
- Dive deeper into relation extraction technics and implementations

Bibliography

[1] “Papers with code.” [Online]. Available: https://paperswithcode.com/task/named-entity-recognition-ner

[2] W. Zhang, W. Hua, and K. Stratos, “EntQA: Entity Linking as Question Answering.” [Online]. Available: https://arxiv.org/abs/2110.02369

[3] P.-L. Huguet Cabot and R. Navigli, “REBEL: Relation Extraction By End-to-end Language generation,” in *Findings of the Association for Computational Linguistics: EMNLP 2021*, M.-F. Moens, X. Huang, L. Specia, and S. W.-t. Yih, Eds., Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 2370–2381. doi: 10.18653/v1/2021.findings-emnlp.204.

[4] R. Orlando, P.-L. H. Cabot, E. Barba, and R. Navigli, “ReLiK: Retrieve and LinK, Fast and Accurate Entity Linking and Relation Extraction on an Academic Budget.” [Online]. Available: https://arxiv.org/abs/2408.00103