Assignment 3: Report

Nagim Isyanbaev
n.isyanbaev@innopolis.university

Links and information

CodaLab Nickname: **Nagim123**
Github repository: https://github.com/Nagim123/NLP_Assignment3

1 First solution: Rule-based

1.1 Solution justification

I looked up solutions for the Russian NER problem and discovered that the winning team for the RuNNE competition was Pullenti. They developed a rule-based solution and got the first place. Upon further research, I found that they are working on the Pullenti SDK [1], a library for various Russian NLP tasks. I decided to utilize some analysis tools from this library to develop my own solution based on their work.

1.2 How it works

The foundation of my solution is the Pullenti SDK, which employs various analyzers to extract different types of NERs. The rules they use are basic patterns of the Russian language and some sensible assumptions. Although they do not explicitly mention them, the source code can be checked for further details.

My part of the solution involves collecting NERs such as *PERSON, PERSONPROPERTY, GEO, ORGANIZATION, MEASURE, DATE, MONEY, DECREE*, and *NAMEDENTITY* from the SDK, and then processing them. I developed custom parsers that analyze the text and evaluate the properties of the identified NERs to determine their corresponding RuNNE class. For instance, a *PERSONPROPERTY* NER can be categorized as a *PROFESSION* or *NATIONALITY* based on the context.

To identify number tokens, I utilized the *Token* class from the Pullenti SDK and checked each token to confirm if it is a number. It's important to note that not all numbers are labeled as number NER in the dataset. Therefore, I

specifically detect those number tokens which consist of alphabet characters, not just digits such as ‘one‘.

1.3 Limitations and challenges

One challenge with the Pullenti SDK is that it sometimes includes extra words in the span of NERs that it identifies, which should not be included. For instance, it may select ‘брат Иван Иванович‘ for a *PERSON* NER instead of just ‘Иван Иванович‘. As a result, I have to incorporate additional cleaning steps to ensure the accuracy of the selected character spans. Additionally, our dataset does not contain nested dates NERs, but Pullenti SDK creates them. Therefore, I also need to consider this aspect in my processing.

One significant limitation of my approach is that the Pullenti SDK does not support all types of NER. As a result, I am unable to derive classes such as *EVENT, FAMILY, IDEOLOGY, LANGUAGE, ORDINAL, PENALTY* and *RELIGION*. This limitation has a considerable impact on the final F1-score, as it decreases the overall effectiveness of the solution.

2 Second solution: BERT

2.1 Solution justification

In addition to addressing nested NER in Russian, I explored the concept of nested NER in various languages and came across a solution with good metrics on benchmarks known as ArabicNER [2]. After discovering a paper on this topic and locating the corresponding GitHub repository with the source code, I made the decision to adapt it for the Russian language.

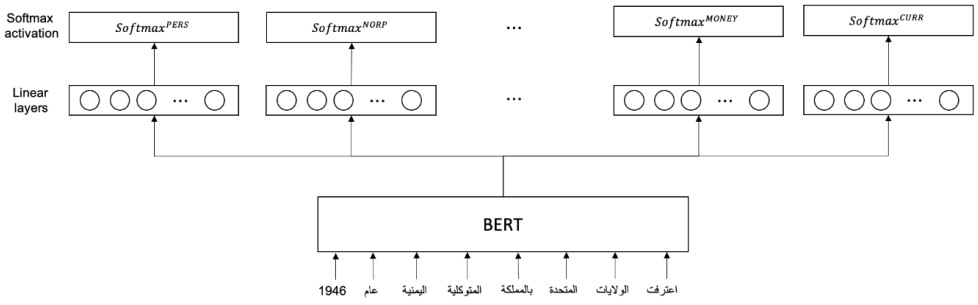
2.2 How it works

The model comprises two main parts. The first part involves a BERT model that generates embeddings for words, while the second part consists of 29 linear layers with softmax at the end. These layers are responsible for predicting each type of NER that spans the current token or predicting the absence of an NER spanning the token. This architecture forms the foundation of the entire solution.

For the word embeddings, the authors of ArabicNER utilized the AraBERT model. However, for Russian text, I used the rubert-base-cased model from DeepPavlov, which is recognized as one of the best models available for this purpose.

2.3 Limitations and challenges

The transformation of the dataset format to suit the requirements of training the ArabicNER model posed a significant challenge. Due to the differing annotation



Pic. 1: Architecture of ArabicNER

structures, there were some quality losses during the conversion between different formats. This transformation process required careful handling to minimize the impact on data quality.

An important limitation of this approach is that the model is unable to predict nested NER of the same type.

Solutions comparison

Model's metrics on development set:

Model	Mention F1	Mention Recall	Mention Precision	Macro F1	Macro F1 few-shot
Rule-based	45.42%	34.98%	64.75%	21.09%	0.00%
BERT	72.04%	68.51%	75.94%	61.45%	0.00%

Model's metrics on test set:

Model	Mention F1	Mention Recall	Mention Precision	Macro F1	Macro F1 few-shot
Rule-based	46.26%	36.01%	64.66%	24.40%	0.00%
BERT	75.97%	75.97%	75.97%	71.68%	0.00%

Conclusion: Based on metric's scores the second solution is the best one.

References

[1] Кузнецов К.И. *SDK Pullenti*. 2022. URL: `https://pullenti.ru/`.

[2] Sana Ghanem Mustafa Jarrar Mohammed Khalilia. "Wojood: Nested Arabic Named Entity Corpus and Recognition using BERT". B: (2022). DOI: `http://www.jarrar.info/publications/JKG22.pdf`.