Named Entity Recognition: intermediate results

Sofya Tkachenko
s.tkachenko

Anatoly Soldatov
a.soldatov

Anton Kudryavstev
a.kudryavtsev

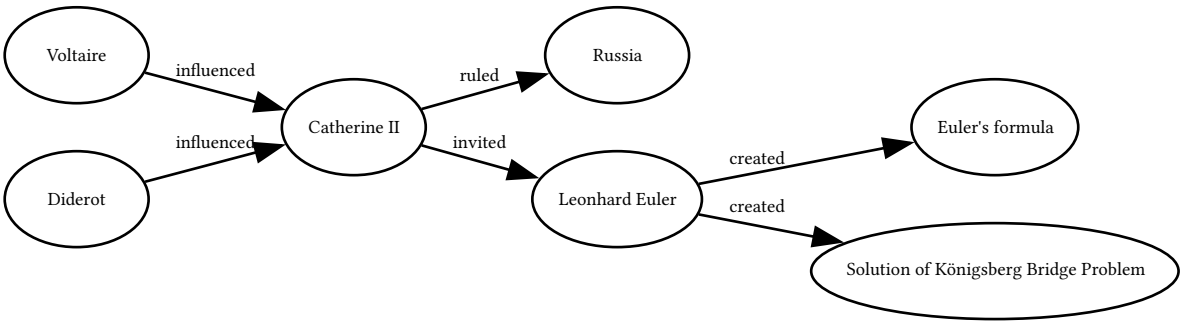
21 October 2024

1 Project Topic

We plan to implement a service that will create a knowledge graph based on the text submitted for input and present this information as a graph or a mindmap to the end user. Key technologies that we plan to utilize include named entity recognition, extraction, and linking.

Example of usage of such service would be:

> *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 singnificantly 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.*



However, processing such detailed texts and identifying complex connections can quickly become a very difficult problem, which may require us to have comprehensive knowledge and huge computing power, which we unfortunately do not have, so the final result may be less qualitative, than in the provided example.

2 Source code

The project's source code, including all notebooks and experiments, can be found in the following GitHub repository: PodYapolskiy/entity-extraction

3 Intermediate Results

We studied the topic, learned different ways of application, studied some technologies such as text embeddings, Long short-term memory unit, and more.

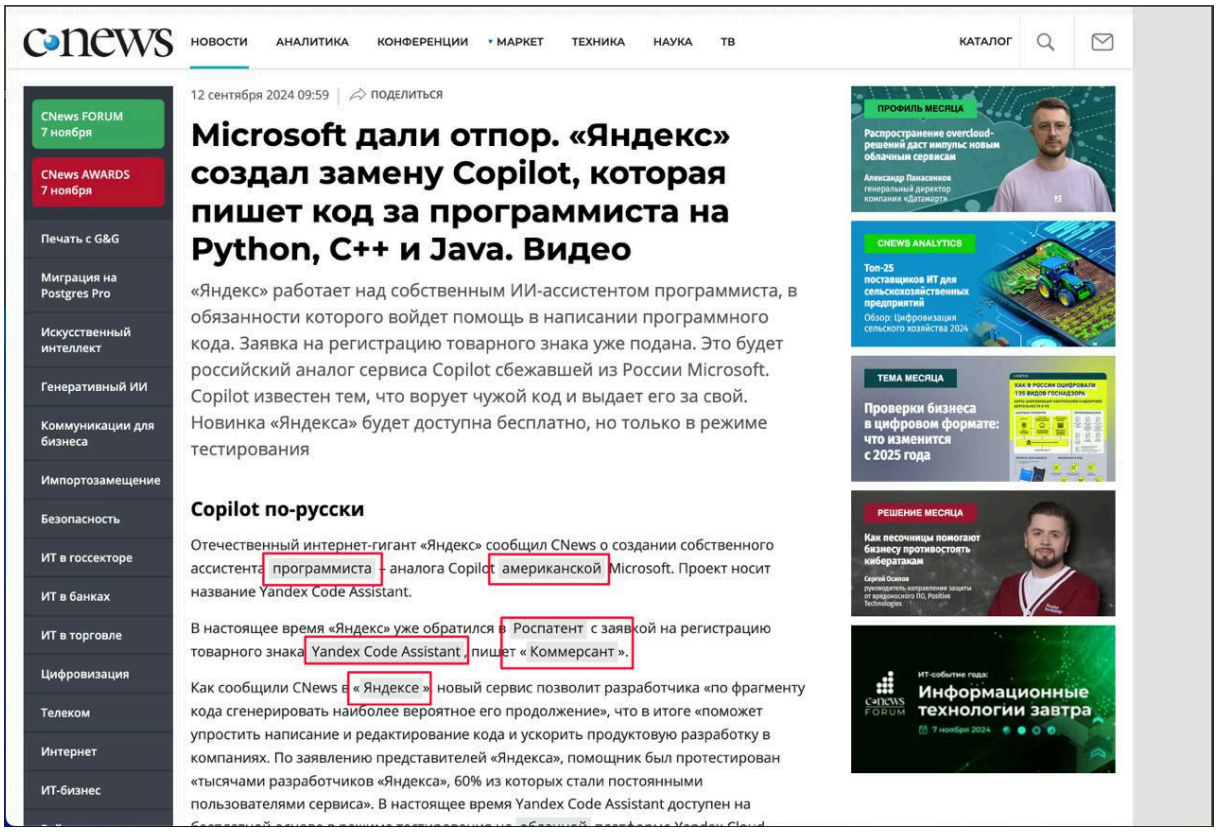
3.1 Approaches

We have found that NER could be done using one of the approaches:

- **The Rule-Based Approach**. In this approach, sets of rules are created that determine which sequences of words in the text can be named entities. These rules can be based on regular expressions, patterns, or linguistic features. Examples of libraries: spaCy (with support for custom rules), NLTK. Example of such rules could be „select only nouns“ or „find sequence (adj, noun, noun)“.
- **A Machine Learning-Based Approach**. This approach uses machine learning algorithms such as CRF (Conditional Random Fields), LSTM (Long Short-Term Memory) and BERT (Bidirectional Encoder Representations from Transformers) to train the model to recognize named entities. The model is trained on marked-up data, where entities are marked in the text. Examples of libraries: spaCy (with trainable models), Stanford NER, Flair, and all the machine learning frameworks (e.g. PyTorch, Tensorflow)
- **Hybrid Approach**. This approach combines rules and machine learning to improve NER accuracy. You can first apply rules to highlight entities, and then run the text through a machine learning model to refine the results.

3.2 Application

We have found an example of how Named Entity Recognition can be applied in news. A news website highlights relevant keywords, and users can click on them to see all related articles. For instance, by clicking on „Yandex“, a user can access all news about that company.



3.3 Datasets

We also found a source with a large number of labeled named entity texts in the Groningen Meaning Bank and a multilingual version, the Parallel Meaning Bank.

3.4 SOTA

We explored how to evaluate such models, and which solutions are considered the best in the category using Papers With Code: Named Entity Recognition. We have checked and played with several models from the leaderboard, however a more detailed research is still to be done.

We have also tested and created an application using a popular pre-trained model, Bert Base NER.

3.5 Our work

3.5.1 Data Preprocessing

The data preprocessing stage involves several steps to clean and prepare the text data for the Named Entity Recognition (NER) model. Here's a detailed breakdown:

1. **Lowercasing:** The text is converted to lowercase to ensure uniformity and reduce the vocabulary size.
2. **Removing Punctuation:** Punctuation marks are removed from the text to focus on the words themselves.
3. **Removing Multiple Spaces:** Multiple spaces are reduced to a single space to standardize the text format.
4. **Stemming:** Words are stemmed to their base or root form to further reduce the vocabulary size and capture the core meaning of words.
5. **Tokenization and Padding:** The text is tokenized into individual words. This is done by training a Tensorflow tokenizer.

The `clean_text_inplace` function applies this preprocessing pipeline.

3.5.2 Model Architecture

The model architecture is designed to handle the NER task using a combination of embedding, dropout, and LSTM layers. Here's a breakdown of the architecture:

1. **Input Layer:** The input layer expects sequences of length `max_length`.
2. **Embedding Layer:** The embedding layer maps each word in the input sequence to a dense vector of size `vector_size`. The `mask_zero=True` parameter allows the model to ignore padding tokens.
3. **Dropout Layer:** A dropout layer with a rate of 0.2 is added to prevent overfitting.
4. **Bidirectional LSTM Layers:** Two bidirectional LSTM layers are used to capture both forward and backward dependencies in the sequence. Each LSTM layer has a recurrent dropout rate of 0.2.
5. **TimeDistributed Dense Layer:** A time-distributed dense layer with `K` units and a softmax activation function is used to predict the probability distribution over the possible entity labels for each word in the input sequence.

```
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)
```

The model is defined inside a `strategy.scope()` to enable multi-GPU support using TensorFlow's `MirroredStrategy`.

3.5.3 Training

The training process involves several steps to prepare the environment, create data pipelines, define callbacks, compile the model, and fit the model to the data. Here's a detailed breakdown:

1. **Setting Visible GPUs:** The visible GPUs are set using the `CUDA_VISIBLE_DEVICES` environment variable.
2. **Limiting GPU Memory Growth:** GPU memory growth is limited to avoid out-of-memory errors, since we are using Kaggle and Colab for training. This allows us to not get banned. :)
3. **Creating Data Pipelines:** Data pipelines are created for the training and test datasets using TensorFlow's `tf.data.Dataset` API.
4. **Defining Callbacks:** Early stopping and learning rate scheduling callbacks are defined to monitor the validation loss and adjust the learning rate during training.
5. **Compiling the Model:** The model is compiled inside the `strategy.scope()` using the Adam optimizer and sparse categorical cross-entropy loss function.
6. **Fitting the Model:** The model is trained using the `model.fit` function with a batch size of 32. The training data is shuffled and split into batches using the `batch` method.
7. **Saving the Model:** The trained model is saved to a file named ‚ner_model.h5'.

This comprehensive approach ensures that the text data is properly preprocessed, the model architecture is designed to capture sequential dependencies, and the training process is optimized for performance and efficiency.

But so far the results are far from SOTA, so we intend to apply different architectures in the upcoming weeks.

4 Future

In the future, we plan to find ways of named entity linking and begin to represent them as a graph. The ideas we plan to explore are to use graph neural networks together with tokenized transformers in a duo to predict named entity linking. One of the sources we would explore is Papers With Code: Entity Linking.

We would also like to enhance the process of identifying named entities, so that it can capture more context. We will also explore additional models that are tailored to our specific needs:

- huawei-noah/TinyBERT_General_4L_312D
- sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2
- cointegrated/rubert-tiny2

5 Team

Name	Email	Participation
Anatoly Soldatov	a.soldatov@innopolis.university	Conducted research, participated in brainstorming, researched rule-based approach, tested several models
Sofya Tkachenko	s.tkachenko@innopolis.university	Found datasets, participated in brainstorming, created application

Name	Email	Participation
		using Bert-Base-NER, trained baseline using bidirectional LSTMs
Anton Kudryavtsev	a.kudryavtsev@innopolis.university	Conducted research, participated in brainstorming, researched ML-based approach, provide ideas for future research, wrote this report