

Named Entity Recognition. ruBERT

. . ИУ5-22М

```
In [1]: from datasets import load_dataset
        from transformers import DataCollatorForTokenClassification
        from transformers import AutoTokenizer, AutoModelForTokenClassification
        from transformers import TrainingArguments, Trainer

        import numpy as np
        import evaluate
```

Загрузка набора данных

Для обучения будем использовать "русскую" часть [WikiNEuRal](#)

```
In [2]: # Загрузка датасета
        dataset = load_dataset("Babelscape/wikineural")
```

```
In [3]: # Пример строки из датасета
        dataset["train_ru"][0]
```

```
Out[3]: {'tokens': ['Детство',
                    'провёл',
                    'в',
                    'Надьсомбате',
                    ',',
                    'с',
                    '1860',
                    'г',
                    '.'],
         'ner_tags': [0, 0, 0, 5, 0, 0, 0, 0, 0],
         'lang': 'ru'}
```

Предобработка данных

```
In [4]: # Загрузка токенизатора
        tokenizer = AutoTokenizer.from_pretrained("ai-forever/ruBert-base")
```

```
fake:\ММО\ДЗ\HW\venv\Lib\site-packages\huggingface_hub\file_download.py:1132:
FutureWarning: `resume_download` is deprecated and will be removed in version
1.0.0. Downloads always resume when possible. If you want to force a new download,
use `force_download=True`.
  warnings.warn(
```

```
In [5]: # Пример работы токенизатора
        example = dataset["train_ru"][0]
        tokenized_input = tokenizer(example["tokens"], is_split_into_words=True)
```

```
tokens = tokenizer.convert_ids_to_tokens(tokenized_input["input_ids"])
tokens
```

```
Out[5]: ['[CLS]',
         'детство',
         'провел',
         'в',
         'над',
         '##ь',
         '##сом',
         '##бат',
         '##е',
         ',',
         'с',
         '1860',
         'г',
         '.',
         '[SEP]']
```

```
In [6]: def tokenize_and_align_labels(examples):
        """Корректировка токенизации
        Parameters
        -----
        examples
            Входное предложение
        Returns
        -----
            tokenized_inputs
            Токенизированный вход
        """
        tokenized_inputs = tokenizer(
            examples["tokens"], truncation=True, is_split_into_words=True
        )

        labels = []
        for i, label in enumerate(examples[f"ner_tags"]):
            word_ids = tokenized_inputs.word_ids(batch_index=i)  # Токенизация
            previous_word_idx = None
            label_ids = []
            for word_idx in word_ids:  # Установка значения спец. токенов -100
                if word_idx is None:
                    label_ids.append(-100)
                elif (
                    word_idx != previous_word_idx
                ):  # Применяем метку только к первому слову в предложении при нескольк
                    label_ids.append(label[word_idx])
                else:
                    label_ids.append(-100)
                previous_word_idx = word_idx
            labels.append(label_ids)

        tokenized_inputs["labels"] = labels
        return tokenized_inputs
```

```
In [7]: # Применение токенизатора к датасету
```

```
tokenized_dataset = dataset.map(tokenize_and_align_labels, batched=True)
```

```
Map: 0%|          | 0/12372 [00:00<?, ? examples/s]
```

Asking to truncate to max_length but no maximum length is provided and the model has no predefined maximum length. Default to no truncation.

```
Map: 0%|          | 0/11597 [00:00<?, ? examples/s]
Map: 0%|          | 0/9618 [00:00<?, ? examples/s]
Map: 0%|          | 0/12678 [00:00<?, ? examples/s]
Map: 0%|          | 0/11069 [00:00<?, ? examples/s]
Map: 0%|          | 0/10547 [00:00<?, ? examples/s]
Map: 0%|          | 0/13585 [00:00<?, ? examples/s]
Map: 0%|          | 0/10160 [00:00<?, ? examples/s]
Map: 0%|          | 0/11580 [00:00<?, ? examples/s]
Map: 0%|          | 0/98640 [00:00<?, ? examples/s]
Map: 0%|          | 0/92720 [00:00<?, ? examples/s]
Map: 0%|          | 0/76320 [00:00<?, ? examples/s]
Map: 0%|          | 0/100800 [00:00<?, ? examples/s]
Map: 0%|          | 0/88400 [00:00<?, ? examples/s]
Map: 0%|          | 0/83680 [00:00<?, ? examples/s]
Map: 0%|          | 0/108160 [00:00<?, ? examples/s]
Map: 0%|          | 0/80560 [00:00<?, ? examples/s]
Map: 0%|          | 0/92320 [00:00<?, ? examples/s]
Map: 0%|          | 0/12330 [00:00<?, ? examples/s]
Map: 0%|          | 0/11590 [00:00<?, ? examples/s]
Map: 0%|          | 0/9540 [00:00<?, ? examples/s]
Map: 0%|          | 0/12600 [00:00<?, ? examples/s]
Map: 0%|          | 0/11050 [00:00<?, ? examples/s]
Map: 0%|          | 0/10460 [00:00<?, ? examples/s]
Map: 0%|          | 0/13520 [00:00<?, ? examples/s]
Map: 0%|          | 0/10070 [00:00<?, ? examples/s]
Map: 0%|          | 0/11540 [00:00<?, ? examples/s]
```

```
In [8]: # Загрузка DataCollator
data_collator = DataCollatorForTokenClassification(tokenizer=tokenizer)
```

Обучение модели

Метрики качества

```
In [9]: seqeval = evaluate.load("seqeval")
```

```
In [10]: def compute_metrics(p):
    """Функция для расчёта метрик
    Parameters
    -----
    p
        Предсказание
    Returns
    -----
    metrics
        Метрики качества
    """
    predictions, labels = p
```

```

predictions = np.argmax(predictions, axis=2)
label_list = [
    "O",
    "B-PER",
    "I-PER",
    "B-ORG",
    "I-ORG",
    "B-LOC",
    "I-LOC",
    "B-MISC",
    "I-MISC",
]

true_predictions = [
    [label_list[p] for (p, l) in zip(prediction, label) if l != -100]
    for prediction, label in zip(predictions, labels)
]
true_labels = [
    [label_list[l] for (p, l) in zip(prediction, label) if l != -100]
    for prediction, label in zip(predictions, labels)
]

results = seqeval.compute(predictions=true_predictions, references=true_labels)
return {
    "precision": results["overall_precision"],
    "recall": results["overall_recall"],
    "f1": results["overall_f1"],
    "accuracy": results["overall_accuracy"],
}

```

Загрузка и обучение модели

Для обучения будем использовать [данную версию](#) ruBERT

```

In [11]: id2label = {
    0: "O",
    1: "B-PER",
    2: "I-PER",
    3: "B-ORG",
    4: "I-ORG",
    5: "B-LOC",
    6: "I-LOC",
    7: "B-MISC",
    8: "I-MISC",
}
label2id = {
    "O": 0,
    "B-PER": 1,
    "I-PER": 2,
    "B-ORG": 3,
    "I-ORG": 4,
    "B-LOC": 5,
    "I-LOC": 6,
    "B-MISC": 7,
}

```

```
"I-MISC": 8,  
}
```

```
In [12]: # Загрузка модели  
model = AutoModelForTokenClassification.from_pretrained(  
    "ai-forever/ruBert-base", num_labels=9, id2label=id2label, label2id=label2id  
)
```

Some weights of BertForTokenClassification were not initialized from the model check point at ai-forever/ruBert-base and are newly initialized: ['classifier.bias', 'classifier.weight']

You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.

```
In [13]: # Аргументы для обучения  
training_args = TrainingArguments(  
    output_dir="ruBERT",  
    learning_rate=2e-5,  
    per_device_train_batch_size=16,  
    per_device_eval_batch_size=16,  
    num_train_epochs=3,  
    weight_decay=0.01,  
    eval_strategy="epoch",  
    save_strategy="epoch",  
    load_best_model_at_end=True,  
    push_to_hub=False,  
)
```

```
In [14]: # Описание тренера  
trainer = Trainer(  
    model=model,  
    args=training_args,  
    train_dataset=tokenized_dataset["train_ru"],  
    eval_dataset=tokenized_dataset["val_ru"],  
    tokenizer=tokenizer,  
    data_collator=data_collator,  
    compute_metrics=compute_metrics,  
)
```

```
In [15]: # Обучение модели  
trainer.train()
```

```
fake:\2 семестр\ММО\ДЗ\HW\venv\Lib\site-packages\transformers\models  
\bert\modeling_bert.py:435: UserWarning: 1Torch was not compiled with flash attentio  
n. (Triggered internally at ..\aten\src\ATen\native\transformers\cuda\sdp_utils.cpp:  
455.)
```

```
    attn_output = torch.nn.functional.scaled_dot_product_attention(  

```

[17310/17310 50:24, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Precision	Recall	F1	Accuracy
1	0.066000	0.060656	0.866045	0.876936	0.871456	0.980347
2	0.040200	0.058756	0.878942	0.884734	0.881828	0.981749
3	0.025400	0.065435	0.877973	0.887809	0.882864	0.982127

```
Out[15]: TrainOutput(global_step=17310, training_loss=0.04991684290523959, metrics={'train_runtime': 3025.2406, 'train_samples_per_second': 91.55, 'train_steps_per_second': 5.722, 'total_flos': 8860619315663712.0, 'train_loss': 0.04991684290523959, 'epoch': 3.0})
```

```
In [16]: # Качество лучшей модели
trainer.evaluate()
```

[722/722 00:24]

```
Out[16]: {'eval_loss': 0.058756403625011444,
'eval_precision': 0.8789416257501363,
'eval_recall': 0.8847336628226249,
'eval_f1': 0.8818281335522714,
'eval_accuracy': 0.9817492493531292,
'eval_runtime': 26.2233,
'eval_samples_per_second': 440.067,
'eval_steps_per_second': 27.533,
'epoch': 3.0}
```

```
In [17]: # Сохранение модели
trainer.save_model("ruBERT/ruBERT_best_model/")
```