МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ «МИСиС»

Институт информационных технологий и компьютерных наук

Кафедра инфокоммуникационных технологий

Отчёт по НИР
на тему «NER (Выделение именованных сущностей»

Выполнил студент:
Миронов Алексей Александрович
Группа: МИВТ-22-5

Москва 2024

Введение

Работа посвящена разработке и реализации глубокой нейронной сети для решения задачи именованного сущностного распознавания (NER) с использованием фреймворка PyTorch и веб-фреймворка FastAPI. Задача NER заключается в выделении и классификации именованных сущностей, таких как имена людей, организаций, местоположения и даты, из текстового контента.

Перед началом разработки были проведены предварительные исследования с использованием предобученных моделей из репозитория Hugging Face.

В рамках предварительного исследования была протестирована модель dslim/bert-base-NER. Эта модель основана на архитектуре BERT (Bidirectional Encoder Representations from Transformers) и предназначена для решения задачи NER.

Для улучшения качества решения и адаптации под конкретную задачу было решено дообучить модель dslim/bert-base-NER на собственных данных и проверить ее производительность на тестовом наборе. После этого была проведена оценка результатов и выявлены области, требующие дополнительной оптимизации.

Далее в рамках исследования была обучена модель distilbert/distilbert-base-uncased из репозитория Hugging Face. DistilBERT - это уменьшенная версия архитектуры BERT с уменьшенным размером и количеством параметров, что делает его более эффективным с точки зрения использования ресурсов.

В работе будет разработана модель глубокого обучения, основанная на архитектуре рекуррентной нейронной сети (RNN) с использованием слоев LSTM (Long Short-Term Memory), способной автоматически распознавать именованные сущности в тексте. Для дообучения и инференса модели будет использоваться веб-фреймворк FastAPI, что позволит предоставить удобный интерфейс для взаимодействия с моделью через HTTP-запросы.

Этапы разработки

1. Подготовка данных: предобработка и разделение текстовых данных на обучающий и тестовый наборы.
2. Предварительное исследование с использованием моделей из hagging face: dslim/bert-base-NER, distilbert/distilbert-base-uncased

2. Разработка модели: определение архитектуры и реализация глубокой нейронной сети на основе RNN с использованием библиотеки PyTorch.

3. Обучение модели: дообучение модели на обучающем наборе данных для распознавания именованных сущностей.

4. Разработка API: создание HTTP-методов с использованием веб-фреймворка FastAPI для дообучения и инференса модели.

5. Тестирование и оценка: оценка производительности модели на тестовом наборе данных и анализ результатов.

Предобработка данных

В работе использовались открытые данные из Kaggle. В датасете 47955 строк.
https://www.kaggle.com/datasets/rohitr4307/ner-dataset

	tokens	ner_tags
0	Thousands of demonstrators have marched throug...	[O, O, O, O, O, O, B-GEO, O, O, O, O, O, B-GEO...
1	Families of soldiers killed in the conflict jo...	[O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, ...
2	They marched from the Houses of Parliament to ...	[O, O, O, O, O, O, O, O, O, O, O, B-GEO, I-GEO...
3	Police put the number of marchers at 10,000 wh...	[O, O, O, O, O, O, O, O, O, O, O, O, O, O, O]
4	The protest comes on the eve of the annual con...	[O, O, O, O, O, O, O, O, O, O, O, B-GEO, O, O,...

Рис.1 Данные для обучения

В данных содержатся следующие метки

Код в датасете	Описание	Код в предобученной модели
ORG	Organization	ORG
PER	Person	PER
GEO	Geographical Entity	LOC
GPE	Geopolitical Entity	MISC - Miscellaneous entity
ART	Artifact	O
EVE	Event	O
NAT	Natural Phenomenon	O
TIM	Time indicator	O
O	Outside of a named entity	O

Рис.2 Метки в датасете и в предобученной модели dslim/bert-base-NER

Колонку tokens необходимо токенизировать. Из датасета удалить строчки, где количество меток не совпадает с количеством токенов.

В колонке ner_tags необходимо каждую метку определить как число, для этого созданы словари id2label, label2id

	tokens	ner_tags	labels
0	[Thousands, of, demonstrators, have, marched, ...	[O, O, O, O, O, O, B-GEO, O, O, O, O, O, B-GEO...	[16, 16, 16, 16, 16, 16, 4, 16, 16, 16, 16, 16...
1	[Families, of, soldiers, killed, in, the, conf...	[O, O, O, O, O, O, O, O, O, O, O, O, O, O, O, ...	[16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 1...
2	[They, marched, from, the, Houses, of, Parliam...	[O, O, O, O, O, O, O, O, O, O, O, B-GEO, I-GEO...	[16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 4...
3	[Police, put, the, number, of, marchers, at, 1...	[O, O, O, O, O, O, O, O, O, O, O, O, O, O, O]	[16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 1...
4	[The, protest, comes, on, the, eve, of, the, a...	[O, O, O, O, O, O, O, O, O, O, O, B-GEO, O, O,...	[16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 4...

Рис.3 Результат предобработки

Проверка моделей dslim/bert-base-NER и distilbert/distilbert-base-uncased

Модели взяты отсюда https://huggingface.co/dslim/bert-base-NER, https://huggingface.co/distilbert/distilbert-base-uncased

При проверке всех данных без дообучения на модели bert-base-NER точность составила 36%.

Разделяем данные на тестовые и обучающие. Для векторизации токенов используем собственные токенизаторы моделей.

```
DatasetDict({
    train: Dataset({
        features: ['tokens', 'ner_tags', 'labels', 'input_ids', 'token_type_ids', 'attention_mask'],
        num_rows: 33568
    })
    validation: Dataset({
        features: ['tokens', 'ner_tags', 'labels', 'input_ids', 'token_type_ids', 'attention_mask'],
        num_rows: 7193
    })
})
```

```
tokenized_ds['train'].features
```

```
{'tokens': Sequence(feature=Value(dtype='string', id=None), length=-1, id=None),
 'ner_tags': Sequence(feature=Value(dtype='string', id=None), length=-1, id=None),
 'labels': Sequence(feature=Value(dtype='int64', id=None), length=-1, id=None),
 'input_ids': Sequence(feature=Value(dtype='int32', id=None), length=-1, id=None),
 'token_type_ids': Sequence(feature=Value(dtype='int8', id=None), length=-1, id=None),
 'attention_mask': Sequence(feature=Value(dtype='int8', id=None), length=-1, id=None)}
```

Рис.4 Пример данных после векторизации

Для обучения используется класс Trainer библиотеки transformer.

Параметры обучения:

- learning_rate=1e-4
- num_train_epochs=3
- weight_decay=0
- optimizer = AdamW

Epoch	Training Loss	Validation Loss
1	0.189800	0.150947
2	0.132100	0.102621
3	0.081900	0.073359

Рис.5 Дообучение dslim/bert-base-NER

В результате дообучение модели точность прогноза на тестовых данных составила 8 %.

Epoch	Training Loss	Validation Loss
1	0.200700	0.169679
2	0.138100	0.137951
3	0.090700	0.125691

Рис.6 distilbert/distilbert-base-uncased

В результате дообучение модели точность прогноза на тестовых данных составила 46%.

Разработка нейросети

Для векторизации токенов необходимо создать словарь, для этого используется функция build_vocab_from_iterator из библиотеки torchtext.

Для того чтобы размерность в датасете была одинаковая каждая строка дополняется символом "<unk>" с соответствующей меткой.

Параметры модели:

- lr=0.005
- num_epochs = 10
- Слои:
  - Embedding
  - LSTM
  - LSTM
  - LSTM
  - LSTM
  - Linear

Embedding используется для преобразования индексов слов в векторные представления.

LSTM - это рекуррентный слой, который способен учитывать контекстную зависимость между словами в предложении. Каждый слой lstm имеет параметр bidirectional=True . Это означает, что LSTM будет обрабатывать входные данные как в прямом, так и в обратном направлении.

Полносвязный слой используется для преобразования выходных данных LSTM в прогнозы для каждой метки сущности.

```
100%|██████████| 300/300 [01:03<00:00,  4.76it/s]
Epoch 1/10, Loss: 3770.9921572208405
100%|██████████| 300/300 [01:03<00:00,  4.72it/s]
Epoch 2/10, Loss: 1109.61266207695
100%|██████████| 300/300 [01:03<00:00,  4.69it/s]
Epoch 3/10, Loss: 769.2748847007751
100%|██████████| 300/300 [01:04<00:00,  4.67it/s]
Epoch 4/10, Loss: 587.3615789413452
100%|██████████| 300/300 [01:04<00:00,  4.65it/s]
Epoch 5/10, Loss: 469.96023654937744
100%|██████████| 300/300 [01:04<00:00,  4.64it/s]
Epoch 6/10, Loss: 395.6709251999855
100%|██████████| 300/300 [01:04<00:00,  4.64it/s]
Epoch 7/10, Loss: 331.23561456799507
100%|██████████| 300/300 [01:04<00:00,  4.63it/s]
Epoch 8/10, Loss: 287.1751722097397
100%|██████████| 300/300 [01:04<00:00,  4.63it/s]
Epoch 9/10, Loss: 244.68596184253693
100%|██████████| 300/300 [01:04<00:00,  4.62it/s]
Epoch 10/10, Loss: 223.98884803056717
```

Рис.7 Обучение нейросети

В результате обучения точность на тестовых данных составила 58%.

Разработка API

Для API использовалась библиотека FastApi. Приложение обернуто в контейнер для изолирования и упрощенного развертывания.

Для использования модели были сохранены модель и словарь с помощью метода torch.save.

Описание методов:

Метод predict принимает на вход текст, для которого необходимо провести анализ именованных сущностей и возвращает список меток для каждого токена.

Метод train принимает на вход текст и список правильных меток для этого текста.

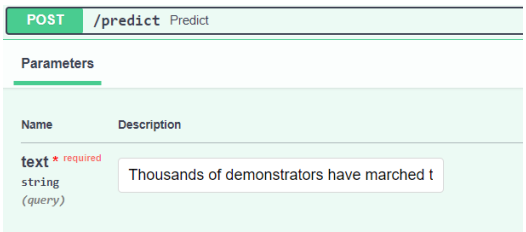

Рис.8 Метод predict: запрос

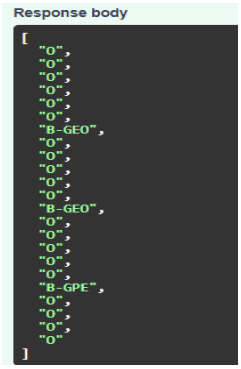

Рис.9 Метод predict: ответ

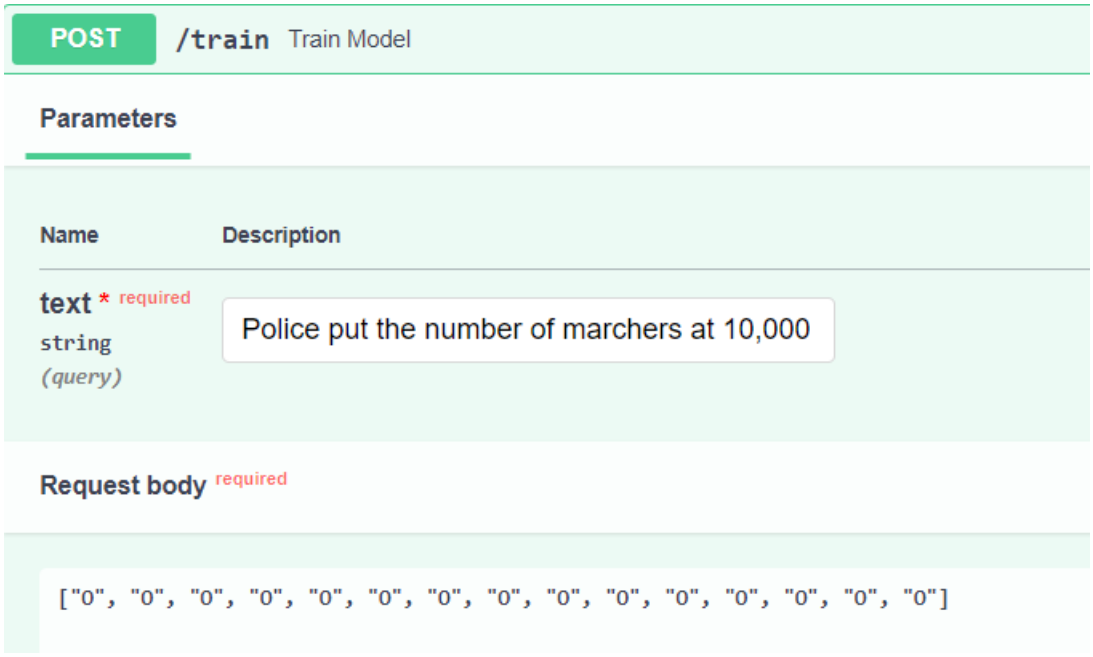

Рис.10 Метод train: запрос

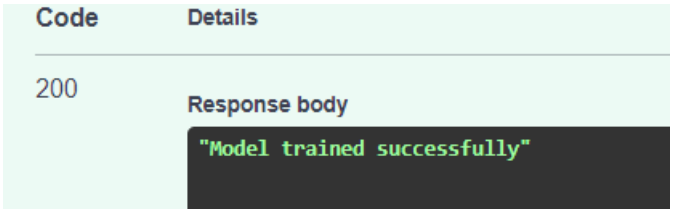

Рис.11 Метод train: ответ

Заключение

В ходе данной работы была разработана и реализована глубокая нейронная сеть для решения задачи именованного сущностного распознавания (NER) с использованием фреймворка PyTorch и веб-фреймворка FastAPI. В процессе исследования были проведены следующие основные этапы:

В начале работы были проведены тесты с использованием предобученных моделей из репозитория Hugging Face, включая модель dslim/bert-base-NER и distilbert/distilbert-base-uncased.

Дообучение модели на собственных данных: было принято решение дообучить модель dslim/bert-base-NER на собственных данных.

Тестирование модели distilbert/distilbert-base-uncased: было проведено обучение модели distilbert/distilbert-base-uncased на собственных данных.

Разработка модели глубокого обучения, основанная на архитектуре рекуррентной нейронной сети (RNN) с использованием слоев LSTM

Разработка API с использованием FastAPI: после выбора оптимальной модели было разработано веб-приложение с использованием веб-фреймворка FastAPI. Это веб-приложение предоставляет удобный интерфейс для взаимодействия с моделью через HTTP-запросы.

Тестирование и оценка производительности: В завершение работы было проведено тестирование и оценка производительности модели и веб-приложения на тестовом наборе данных. Это позволило подтвердить работоспособность и эффективность разработанного решения.

В целом, разработанное решение позволяет эффективно решать задачу именованного сущностного распознавания и предоставляет удобный интерфейс для его использования. Дальнейшие шаги могут включать в себя оптимизацию модели, расширение функциональности веб-приложения и проведение дополнительных тестов и исследований для улучшения производительности и качества решения.