Распознавание именованных сущностей (Named Entity Recognition, NER)

Елена Кантонистова

План занятия

1. Задача NER
2. Форматы разметки (IOB1 и IOB2)
3. Возвращаемся к соревнованию по NER (формулировка и описание)
4. Решение задач POS-tagging, NER и RE при помощи библиотек Natasha и SpaCy
5. Рекуррентные модели для задачи NER
6. Трансформеры для задачи NER – архитектура
7. Трансформеры для задачи NER - практика

NAMED ENTITY RECOGNITION (NER)

NER, или извлечение именованных сущностей (Named Entity Recognition), представляет собой задачу обработки естественного языка, направленную на выделение и классификацию именованных сущностей в тексте.

Именованные сущности могут включать в себя различные категории, такие как имена людей, названия организаций, местоположения, даты, числа и другие типы ключевых элементов текста.

NAMED ENTITY RECOGNITION (NER)

Сложности NER

Неоднозначность детекции сущности

Пример: ‘New York’.

Можно выявить две сущности – *New* и *York*

Можно одну – *New York*

Это зависит от контекста.

Сложности NER

Неоднозначность отнесения сущности к определенному классу

Name	Possible Categories
Washington	Person, Location, Political Entity, Organization, Vehicle
Downing St.	Location, Organization
IRA	Person, Organization, Monetary Instrument
Louis Vuitton	Person, Organization, Commercial Product

Тег зависит от контекста.

[$_{\text{PER}}$ Washington] was born into slavery on the farm of James Burroughs.
[$_{\text{ORG}}$ Washington] went up 2 games to 1 in the four-game series.
Blair arrived in [$_{\text{LOC}}$ Washington] for what may well be his last state visit.
In June, [$_{\text{GPE}}$ Washington] passed a primary seatbelt law.
The [$_{\text{VEH}}$ Washington] had proved to be a leaky ship, every passage I made...

Решение задачи NER

Несмотря на то что сущности часто бывают многословными, ***обычно задача NER сводится к задаче классификации на уровне токенов***, т. е. каждый токен относится к одному из нескольких возможных классов.

NER как Sequence Labeling

NER, рассматриваемое как задача sequence labeling, означает, что каждому токену в последовательности текста присваивается метка, указывающая на принадлежность токена к какой-то именованной сущности. Это позволяет модели определить границы именованных сущностей и их тип.

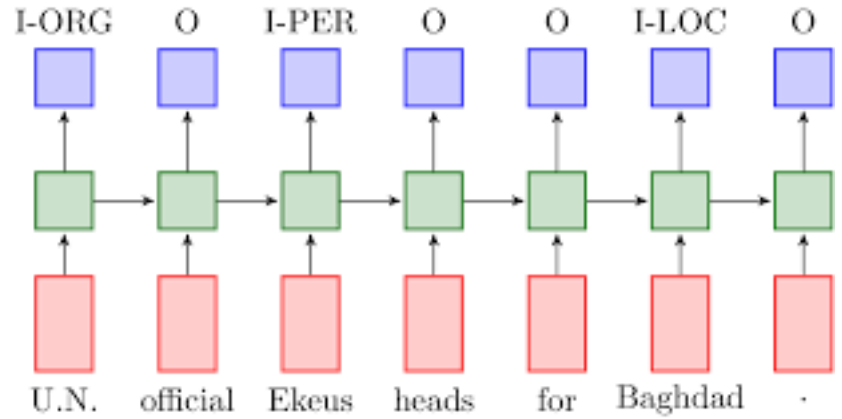
Пример текста с разметкой:

```
[Steve] [Jobs] [was] [the] [co-founder] [of] [Apple] [Inc.] [,] [born] [on] [February] [24] [,] [1955].
B-PER   I-PER  O     O     O           O    B-ORG    I-ORG  O   O      B-DATE    I-DATE O
```

NER как Sequence Labeling

Для решения задачи NER как Sequence Labeling используются различные архитектуры:

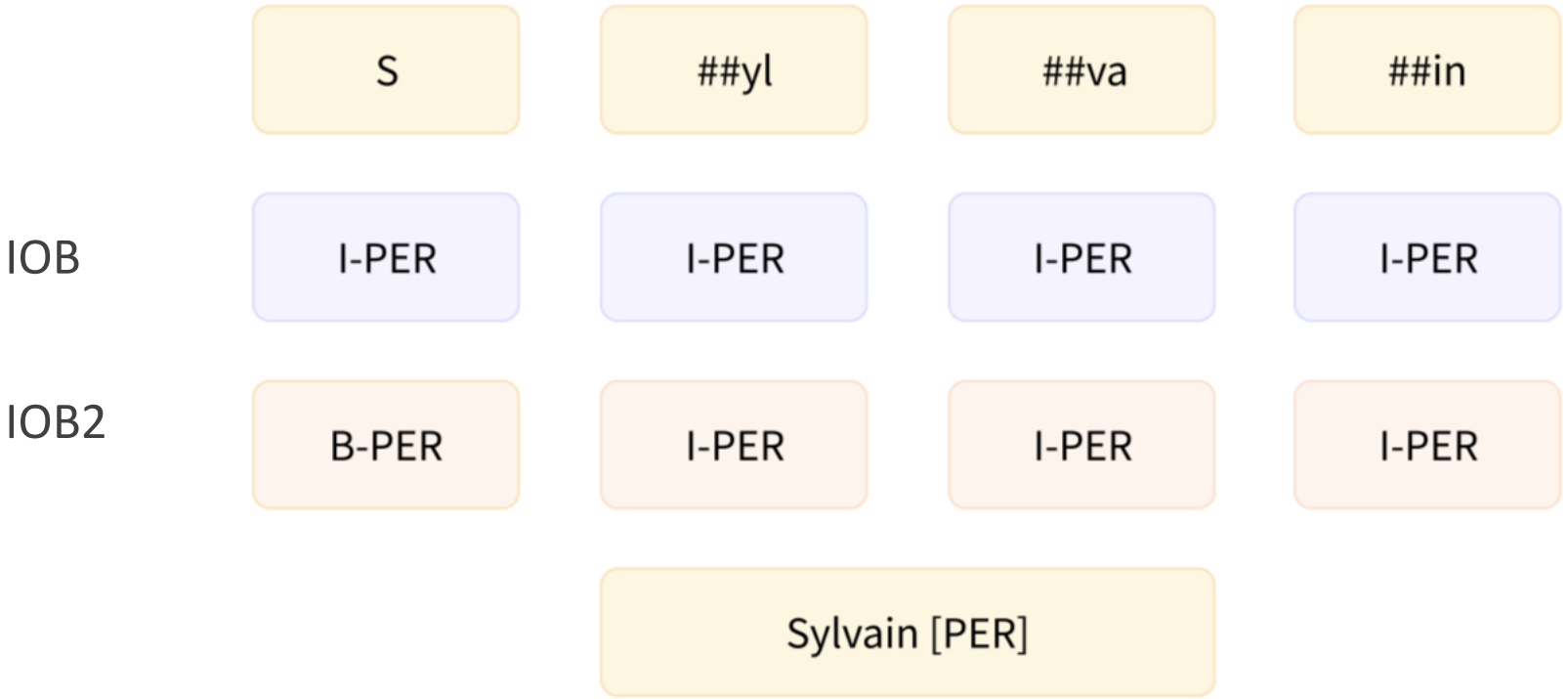
- рекуррентные нейронные сети (RNN)
- сверточные нейронные сети (CNN)
- трансформеры



Они обрабатывают входной текст токен за токеном и предсказывают соответствующую метку для каждого токена.

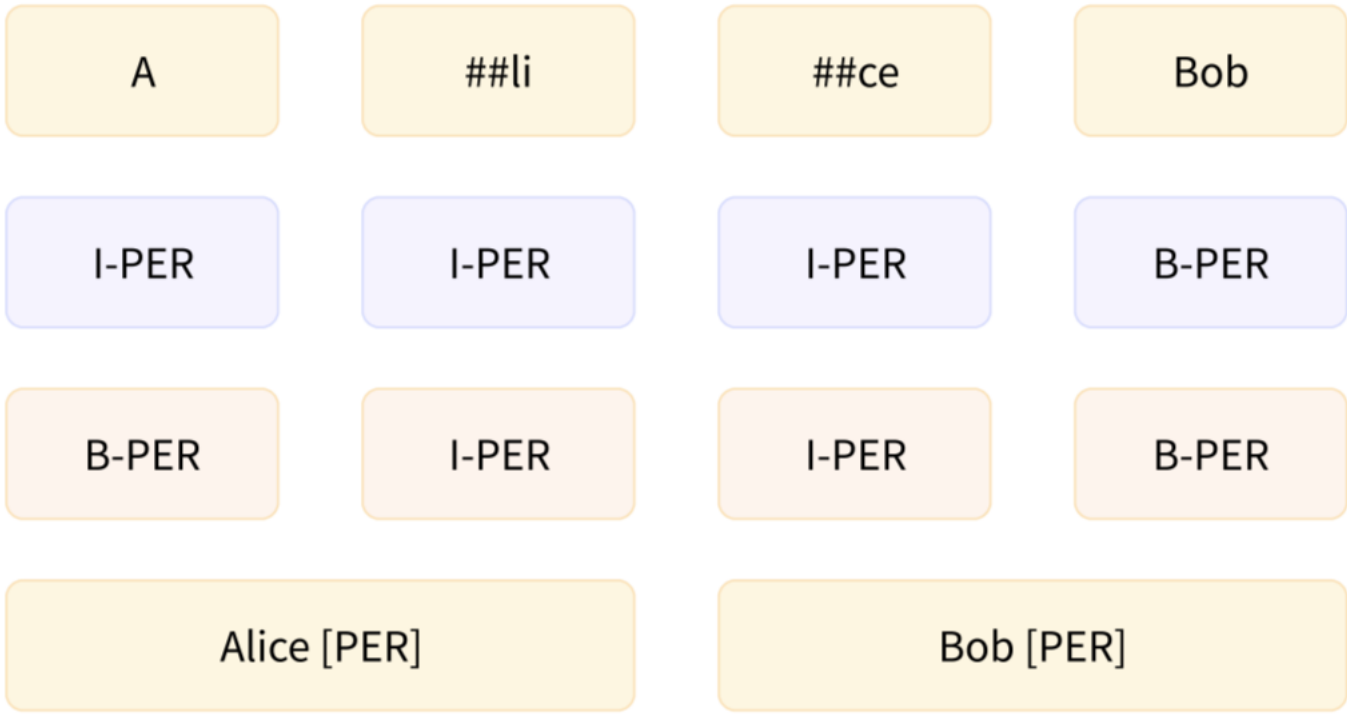
Форматы разметки: IOB (IOB1) и IOB2

Рассмотрим пример, когда токены – это части слов:



Форматы разметки: IOB (IOB1) и IOB2

Рассмотрим пример, когда токены – это части слов:



Форматы разметки: IOB (IOB1) и IOB2

IOB:

- Токены именованной сущности начинаются с префикса “I-”
- Если две именованные сущности одного типа идут друг за другом, то вторая начинается с префикса “B-”

IOB2:

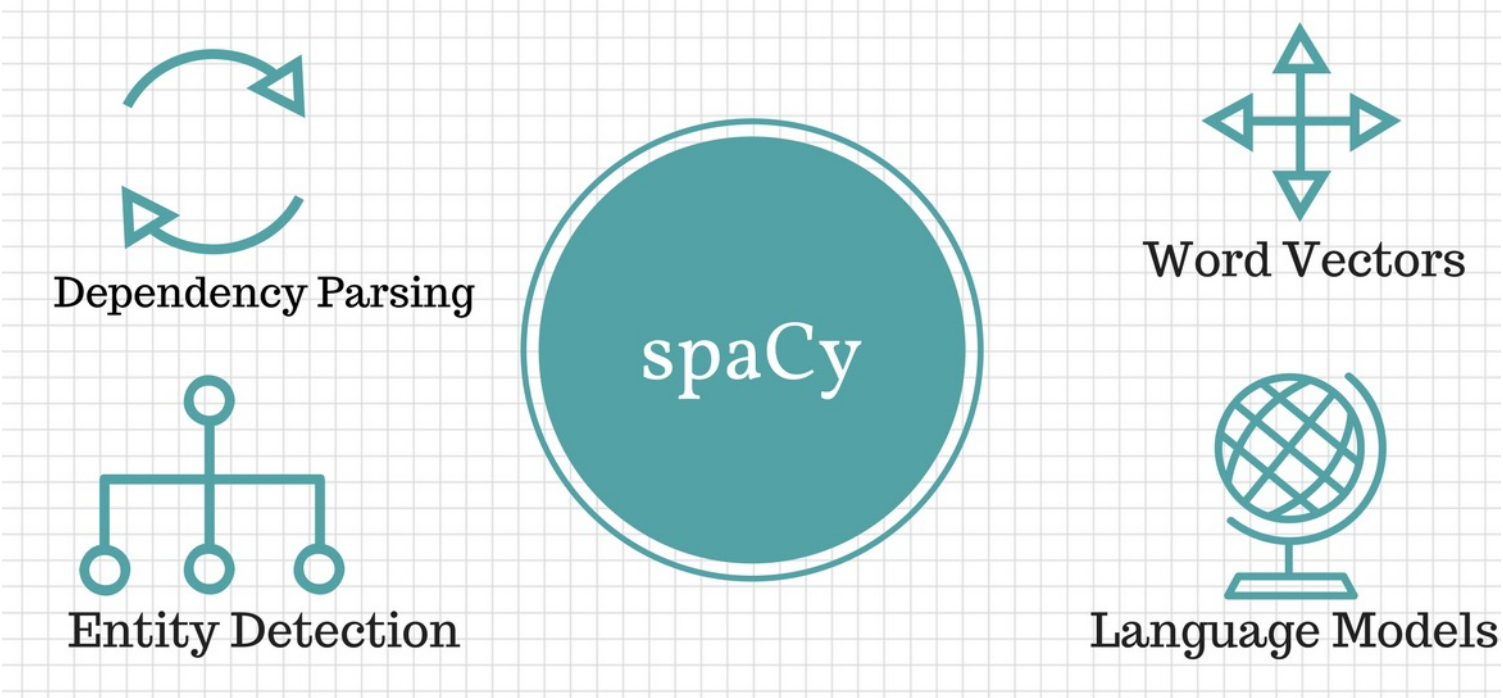
- Первый токен именованной сущности начинается с “B-”
- Все остальные токены именованной сущности начинаются с “I-”

Домашнее задание (соревнование)

https://www.kaggle.com/competitions/litbank-ner-2024

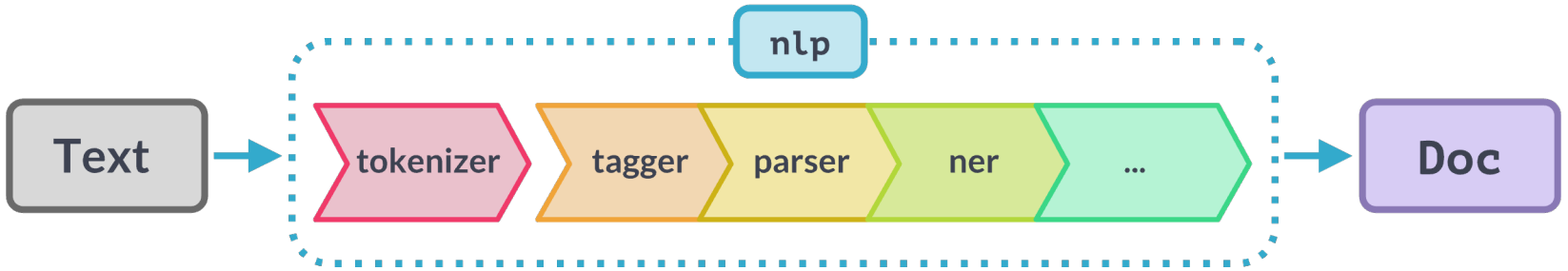
Spacy

SpaCy - это библиотека для обработки естественного языка (NLP), написанная на языке Python. Она предоставляет высокопроизводительные инструменты для выполнения различных задач в области обработки текста, включая токенизацию, лемматизацию, извлечение частей речи, анализ зависимостей и извлечение именованных сущностей (NER).



Spacy

- SpaCy — в каком-то смысле противоположность NLTK. Она значительно быстрее, так как она написана на Cython. SpaCy предоставляет эффективные инструменты для решения конкретной задачи. Она — numpy из мира NLP.
- В целом, SpaCy с её предобученными моделями, скоростью, удобным API и абстракцией гораздо лучше подходит для разработчиков, создающих готовые решения, а NLTK с огромным числом инструментов и возможностью городить любые огороды — для исследователей и студентов.
- В любом случае для создания собственных моделей ни та, ни другая библиотека не подходит. Для этого всего существуют Tensorflow, PyTorch и прочие фреймворки глубинного обучения.



NLTK versus SpaCy

Характеристика	NLTK	spaCy
Область применения	Обширный набор инструментов для NLP и обработки текста (базовые и классические методы).	Основная фокусировка на высокопроизводительных задачах NLP. Готовые модели для NER, анализа зависимостей и других задач.
Производительность	Может быть менее производительным, особенно для больших объемов данных.	Известна своей высокой производительностью и эффективностью обработки текста в реальном времени.
Интерфейс и удобство использования	Гибкий, но более подробный и требующий больше кода.	Простой и удобный интерфейс, часто требует меньше кода для распространенных задач.
Модели и предварительное обучение	Предоставляет базовые инструменты, но большинство моделей не обязательно предобучены.	Предоставляет готовые предобученные модели для различных задач, что упрощает работу.
Совместимость с глубоким обучением	Ориентирована на традиционные методы и менее интегрирована с современными методами глубокого обучения.	Лучше интегрирована с глубоким обучением, работает с TensorFlow и PyTorch.
Сообщество и поддержка	Активное сообщество, множество ресурсов и материалов для обучения.	Активное сообщество, поддержка и регулярные обновления.
Языки	Поддерживает множество языков, но предоставляет ограниченные модели для каждого языка.	Поддерживает множество языков, с предоставлением готовых моделей для некоторых из них.

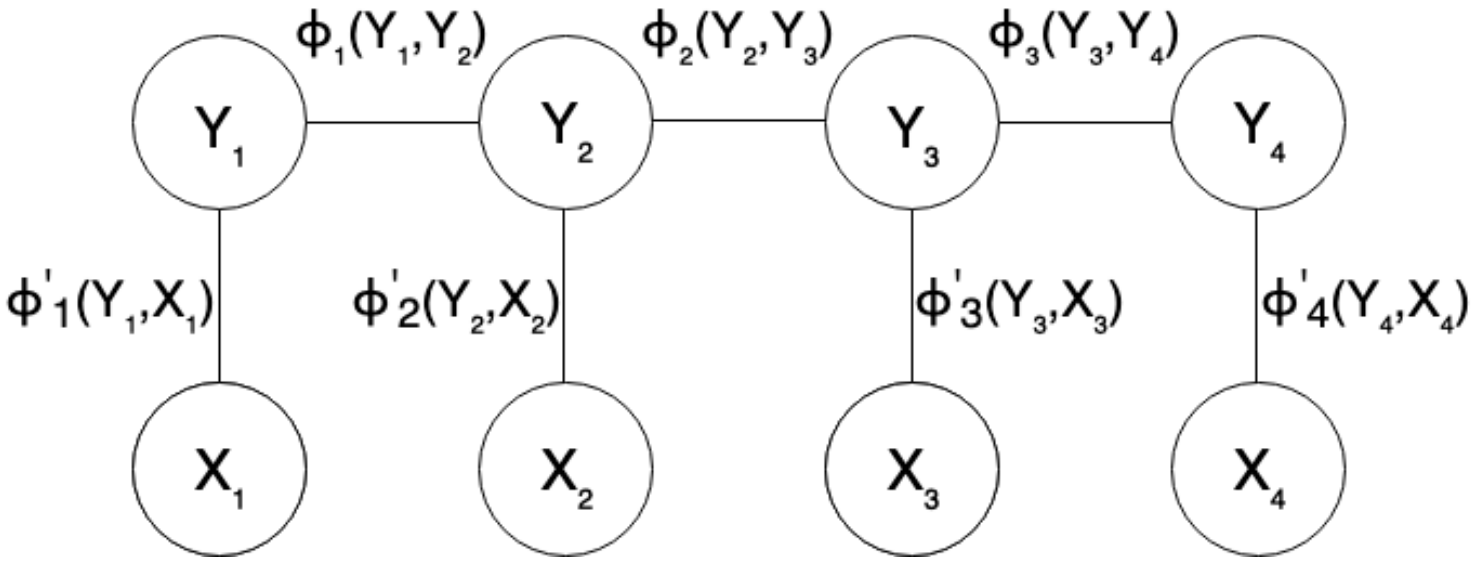
Spacy

Курс от разработчиков библиотеки:

https://course.spacy.io/en

Подходы для задачи NER: CRF

Модель не использует никаких нейронных сетей и основана на подходе CRF (Conditional Random Fields).



Модель CRF (Conditional Random Fields) используется для моделирования условных вероятностей в задачах, связанных с последовательностями данных. Она относится к классу графических моделей и используется для моделирования зависимостей между случайными переменными, особенно в контексте последовательностей данных.

Основной идеей CRF является *учет контекстуальных зависимостей между метками последовательности данных, таких как последовательности слов в предложении*.

Baseline решение соревнования (CRF)

Первый подход (без нейронных сетей) – Conditional Random Fields.

Baseline решения соревнования основан на подходе CRF при помощи библиотеки Spacy:
https://colab.research.google.com/drive/1IVsKsHZyfjSMDqr6whyC6nln_NAmAywp?usp=sharing

Практика! Примеры решения задач NLP при помощи Natasha и SpaCy

https://colab.research.google.com/drive/1EgRKRolhiosyDeQALMMBBzj-5c5fhMtw?usp=sharing

Подходы для задачи NER

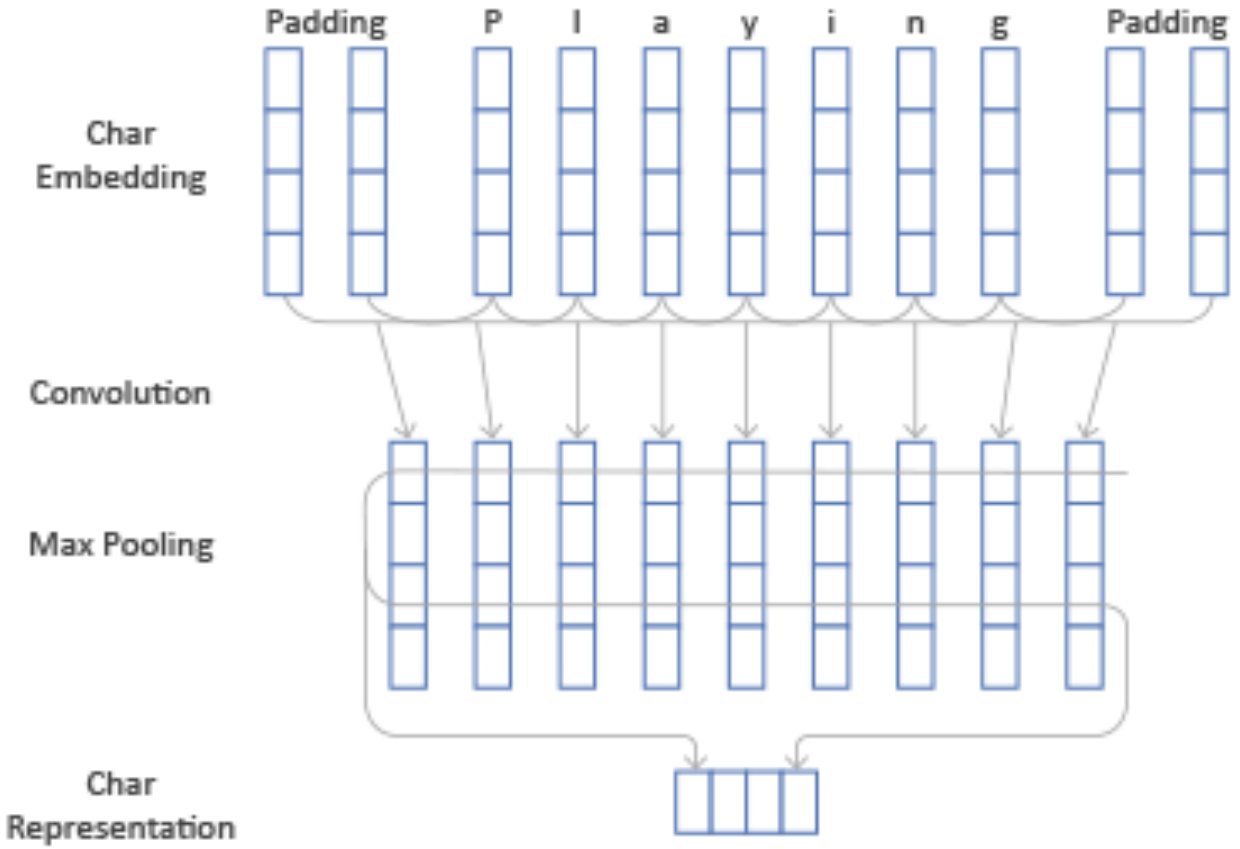
Задачу NER можно решать при помощи нейронных сетей. Рассмотрим подход с использованием:

- Сверточных нейронных сетей
- Рекуррентных нейронных сетей

Их комбинация – это SOTA в задаче NER до 2018 года, рассмотрим ее подробнее.

Извлечение признаков из токена (CNN)

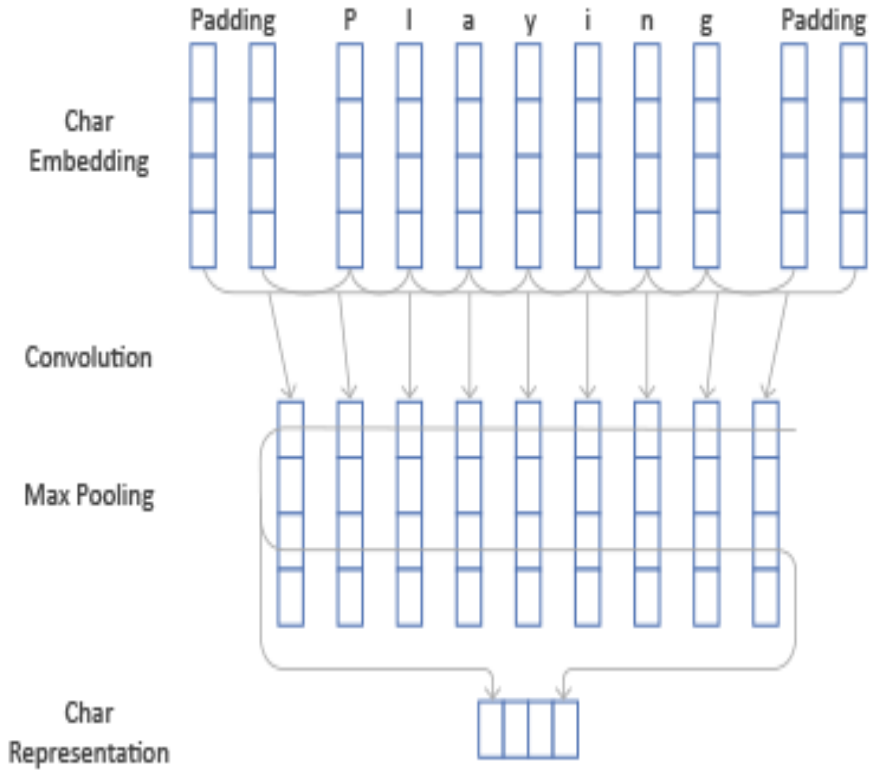
Вспомогательным шагом будет извлечение векторов признаков из токенов.



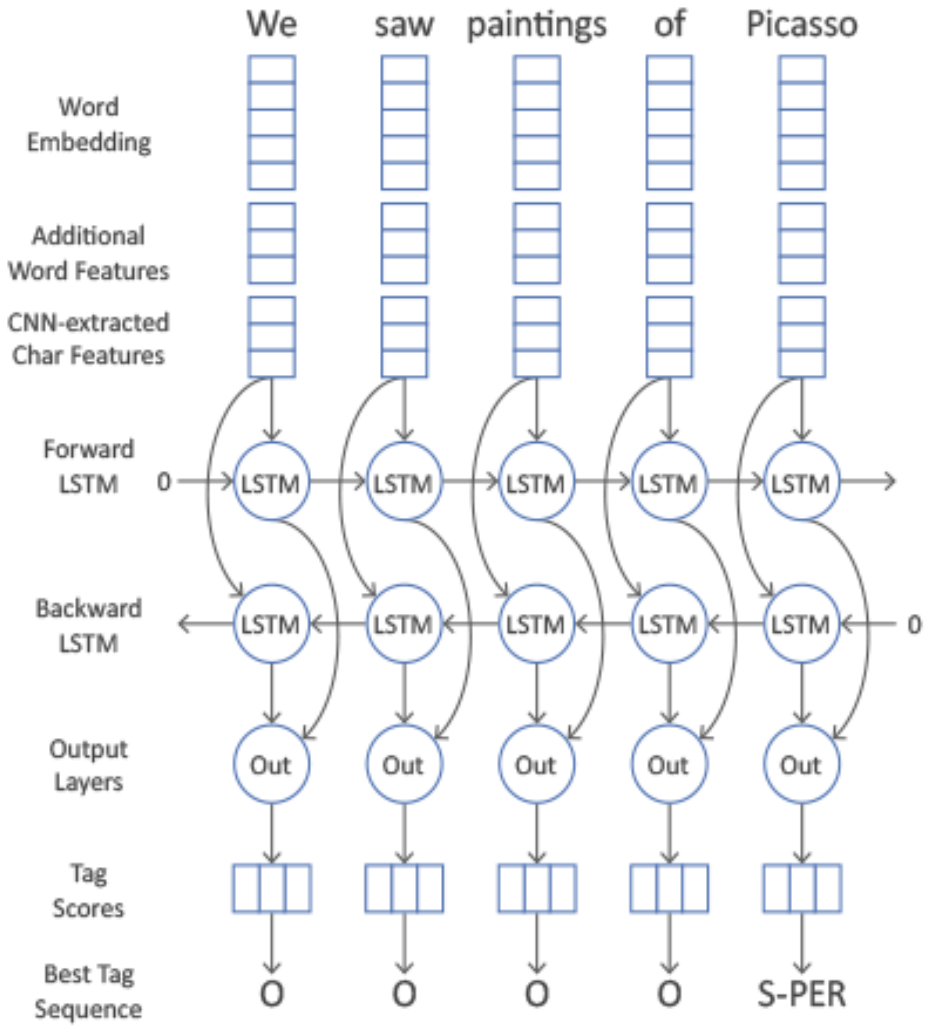
Извлечение признаков из токена (CNN)

Вспомогательным шагом будет извлечение векторов признаков из токенов.

- Нам дан токен, который состоит из каких-то символов.
- На каждый символ мы будем выдавать вектор - не очень большой символьный эмбеддинг. Символьные эмбеддинги можно предобучать, однако чаще всего они учатся с нуля — символов даже в не очень большом корпусе много, и символьные эмбеддинги должны адекватно обучиться.
- Пропускаем эмбеддинги всех символов через свертку с фильтрами не очень больших размерностей и получаем вектора размерности количества фильтров.
- Над этими векторами производим max pooling, получаем 1 вектор размерности количества фильтров. Он содержит в себе информацию о символах слова и их взаимодействии и будет являться вектором символьных признаков токена.



Архитектура CharCNN-BLSTM-CRF



BLSTM structure for tagging named entities.

Архитектура CharCNN-BLSTM-CRF

В *i*-й момент времени слой выдает вектор, являющийся конкатенацией соответствующих выходов прямого и обратного RNN. Этот вектор содержит в себе информацию как о предыдущих токенах в предложении (она есть в прямом RNN), так и о следующих (она есть в обратном RNN). Поэтому этот вектор является контекстно-зависимым признаком токена.

Получив контекстно-зависимые признаки всех токенов, мы хотим по каждому токену получить правильную метку для него.

Решение 1 (плохое):

- Простой и очевидный способ – использовать в качестве последнего слоя полносвязный с softmax размерности d, где d — количество возможных меток токена.
- Этот способ работает, однако обладает существенным недостатком — метка токена вычисляется независимо от меток других токенов. Сами соседние токены мы учитываем за счет BiRNN, но метка токена зависит не только от соседних токенов, но и от их меток. Например, вне зависимости от токенов метка I-PER встречается только после B-PER или I-PER.

Архитектура CharCNN-BLSTM-CRF

В *i*-й момент времени слой выдает вектор, являющийся конкатенацией соответствующих выходов прямого и обратного RNN. Этот вектор содержит в себе информацию как о предыдущих токенах в предложении (она есть в прямом RNN), так и о следующих (она есть в обратном RNN). Поэтому этот вектор является контекстно-зависимым признаком токена.

Получив контекстно-зависимые признаки всех токенов, мы хотим по каждому токену получить правильную метку для него.

Решение 2 (хорошее):

Стандартный способ учесть взаимодействие между типами меток — использовать CRF (conditional random fields). CRF оптимизирует всю цепочку меток целиком, а не каждый элемент в этой цепочке.

Практика! CharCNN-BLSTM-CRF для задачи NER

Пример решения задачи NER при помощи LSTM:

https://colab.research.google.com/drive/1OQW8vZ8iNhc64PgHESrJXfRTSkRgZCxn?usp=sharing

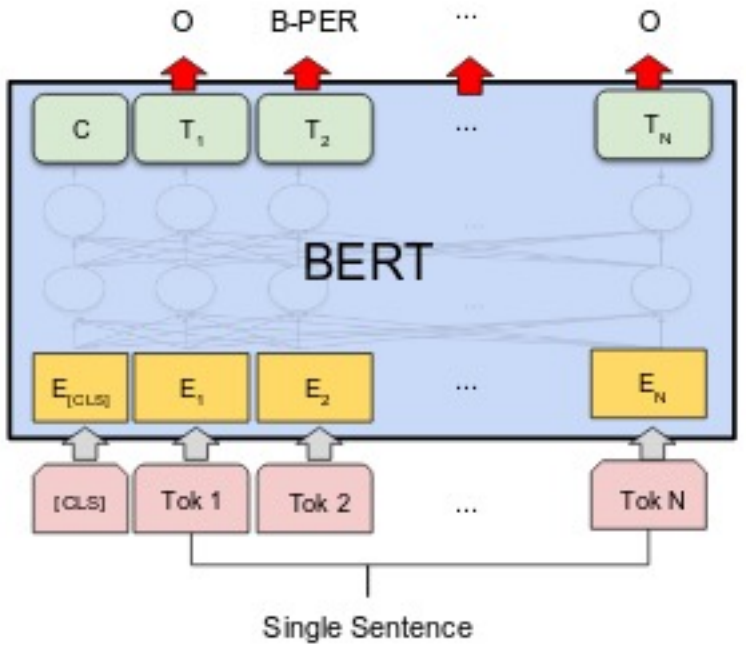
или (аналогичный ноутбук)

https://github.com/TheAnig/NER-LSTM-CNN-Pytorch/blob/master/Named_Entity_Recognition-LSTM-CNN-CRF-Tutorial.ipynb

Трансформеры для задачи NER: BERT

BERT (Bidirectional Encoder Representations from Transformers) представляет собой революционную модель в области обработки естественного языка (NLP).

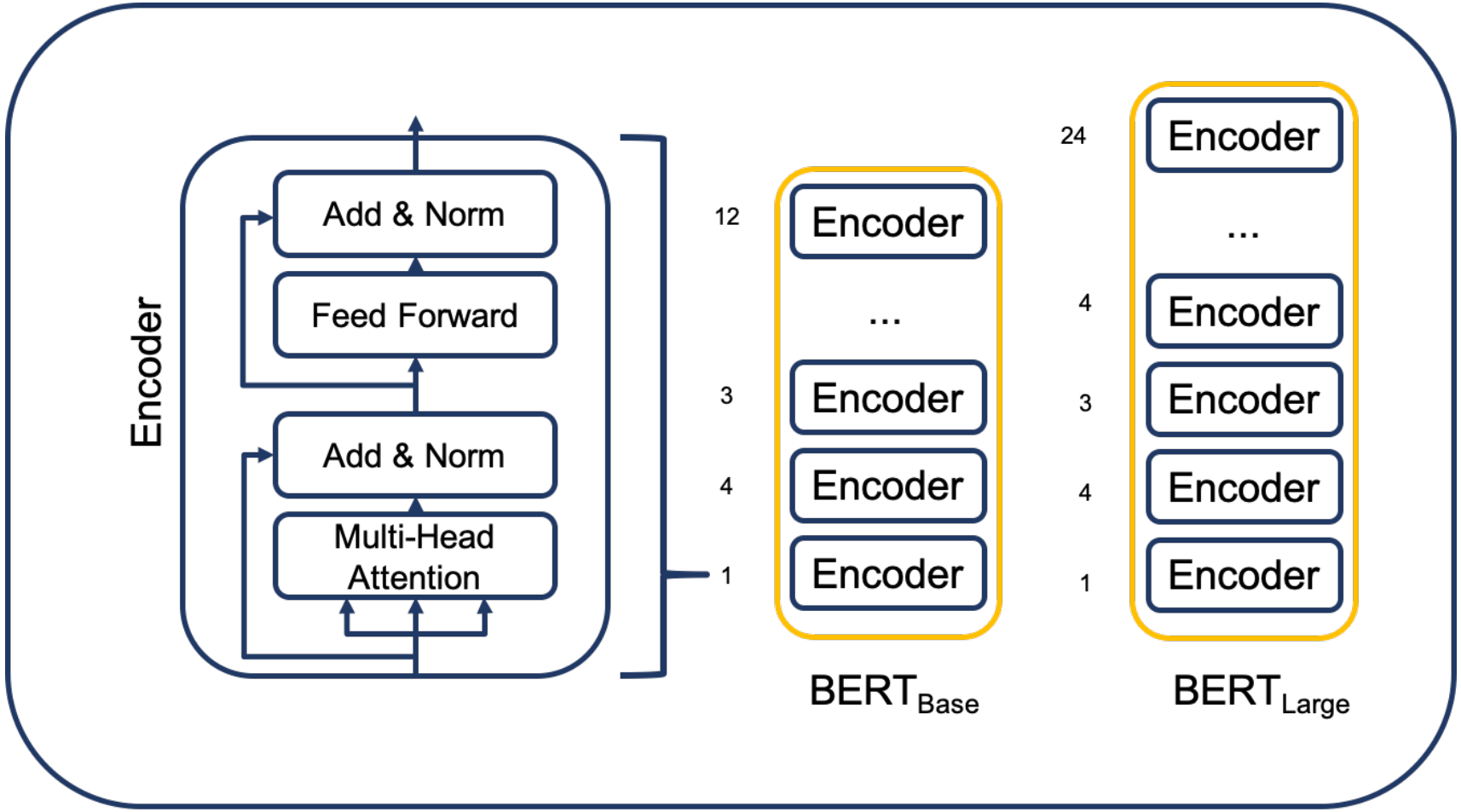
- BERT разработан для понимания и обработки языка, читая текст с начала и с конца одновременно.



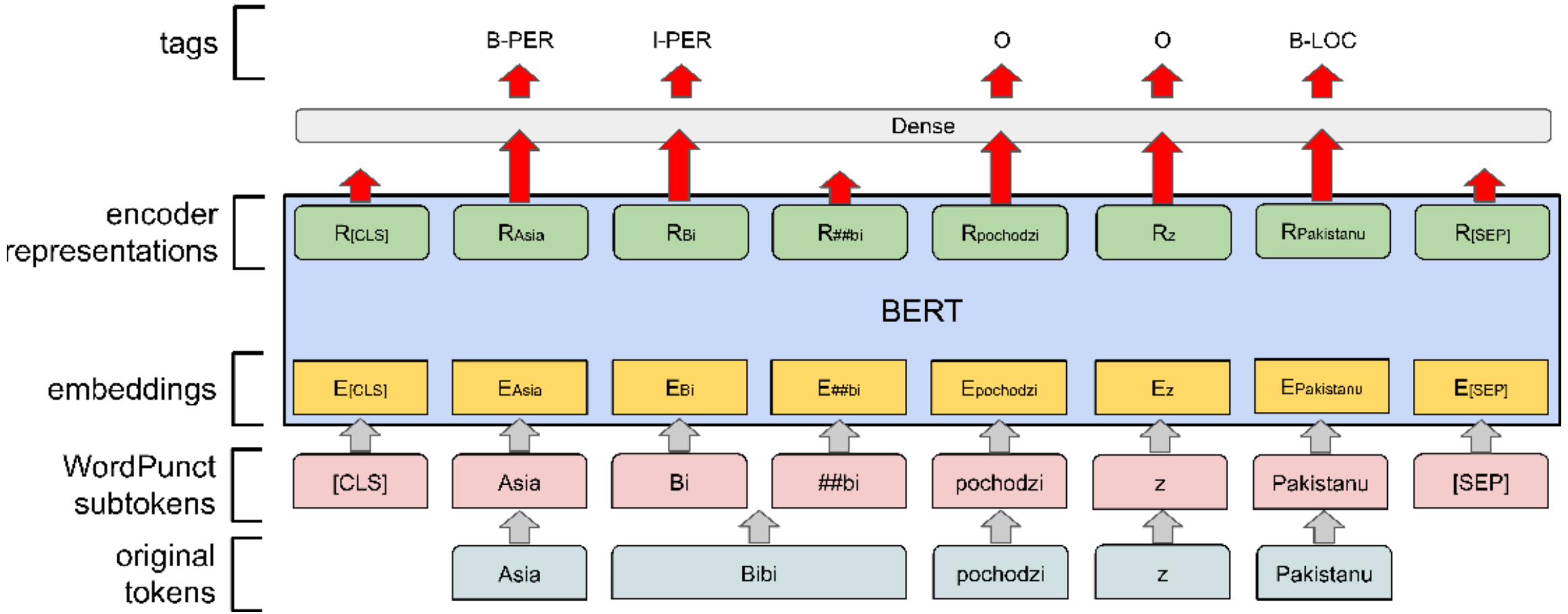
Зачем нужен BERT в задаче NER?

- Традиционные методы распознавания именованных сущностей (NER) испытывают трудности в улавливании сложного контекста и нюансов, присутствующих в естественном языке.
- Эти методы часто требуют ручного создания признаков и/или не обладают возможностью учитывать двунаправленные отношения между словами в предложении
- В результате они не справляются со сложностью языка, особенно в сценариях, где значение сущности глубоко связано с ее окружающим контекстом.

BERT (напоминание архитектуры)



BERT для задачи NER



Практика! Предобученный BERT для задачи NER

https://colab.research.google.com/drive/1H5hVo1WTEYeqJw9IBFtutdt-utLmCSBv?usp=sharing

Fine-tuning

Fine-tuning (дообучение) используется для адаптации предварительно обученной модели на конкретную задачу или набор данных. Предварительно обученные модели, такие как BERT, разрабатываются на больших объемах разнообразных данных с целью изучения общих языковых структур и представлений. Однако, эти общие представления не всегда достаточно точны или релевантны для конкретных задач.

Практика! Дообучение BERT для задачи NER

Данные: https://www.kaggle.com/datasets/abhinavwalia95/entity-annotated-corpus

Ноутбук: https://github.com/abhimishra91/transformers-tutorials/blob/master/transformers_ner.ipynb

Модификации архитектур для задачи NER

Можно не просто обучать или дообучать BERT на своих данных, но и создавать более хитрые архитектуры, использующие накопленные ранее знания и подходы к решению задачи: