https://huggingface.co/learn/nlp-course/ru/chapter1/2?fw=pt

Обработка естественного языка

Что такое обработка естественного языка (NLP), и почему мы заинтересованы в этой сфере.

Что такое NLP?

NLP - область лингвистики и машинного обучения, которая изучает все, что связано с естественными языками. Главная цель NLP не просто понимать отдельные слова, но и иметь возможность **понимать контекст**, в котором эти слова находятся.

Список типичных NLP-задач с некоторыми примерами:

- **Классификация предложений**: определить эмоциональную окраску отзыва, детектировать среди входящих писем спам, определить грамматическую правильность предложения или даже проверить, являются ли два предложения связанными между собой логически.

- **Классификация каждого слова** в предложении: вычленить грамматические составляющие предложения (существительное, глагол, прилагательное) или определить именованные сущности (персона, локация, организация).

- **Генерация текста**: закончить предложение на основе некоторого запроса, заполнить пропуски в тексте, содержащем замаскированные слова.

- **Сформулировать ответ на вопрос**: получить ответ на заданный по тексту вопрос.

- **Сгенерировать новое предложение исходя из предложенного**: перевести текст с одного языка на другой, выполнить автоматическое реферирование текста NLP не ограничивается только письменным текстом. Есть множество сложных задач, связанных с распознаванием речи и компьютерным зрением, таких как транскрибирование аудио или описание изображений.

Почему это сложно?

Компьютеры не обрабатывают информацию так же, как люди. Например, когда мы читаем предложение «**Я голоден**», мы можем легко понять его значение.

Точно так же, имея два предложения, такие как «**Я голоден**» и «**Мне грустно**», мы можем легко определить, насколько они похожи.

Для моделей машинного обучения (ML) такие задачи сложнее.

Текст должен быть обработан так, чтобы модель могла учиться на нем.

https://huggingface.co/learn/nlp-course/ru/chapter1/3?fw=pt

Трансформеры: на что они способны?

В этом разделе мы посмотрим, на что способны трансформеры и первый раз применим функцию из библиотеки Transformers: pipeline().

Трансформеры повсюду!

Трансформеры используются для решения всех видов задач NLP:

Библиотека Transformers предоставляет различную функциональность для создания и использования этих моделей.

Model Hub содержит тысячи предобученных моделей, которые может скачать и использовать любой. Вы также можете загружать свои модели на Model Hub!

pipeline()

Самый базовый объект библиотеки Transformers - pipeline(). Он соединяет в себе необходимые шаги по предобработке и постобработке данных и позволяет передавать модели на вход текст, а на выходе получать читаемый ответ.

Классификация. Положительный или отрицательный контекст

sentiment-analysis

```
# classifier POSITIVE or NEGATIVE
from transformers import pipeline

classifier = pipeline("sentiment-analysis")

classifier("I've been waiting for a HuggingFace course
my whole life.")
```

```
[{'label': 'POSITIVE', 'score': 0.9598049521446228}]
```

```
# Test 1
classifier(
    ["I love hating you."]
)
```
```
[{'label': 'NEGATIVE', 'score': 0.9939486980438232}]
```

```
# Test 2
classifier(
    ["To live a life is not to cross a field."]

)
```
```
[{'label': 'NEGATIVE', 'score': 0.9954882264137268}]
```

```
# Test 3
classifier(
    ["Don't say hop until you jump over."]
)
```
```
[{'label': 'POSITIVE', 'score': 0.7365450263023376}]
```


По умолчанию этот пайплайн выбирает специальную модель, которая была предобучена для оценки тональности предложений на английском языке.

Модель загружается и кэшируется когда вы создадаете объект classifier. Если вы перезапустите команду, будет использована кэшированная модель, т.е. загрузки новой модели не произойдет.

В процессе обработки пайплайном текста, который вы ему передали, есть **три главных шага**:

1. Текст предобрабатывается в формат, который понятен модели.
2. Предобработанный текст передается в модель.
3. Результаты модели проходят через постобработку и вы получаете читаемый ответ.

Некоторые из доступных пайплайнов:

- feature-extraction (превращение текста в векторы, подготовка)
- fill-mask (заполнение пропусков)
- ner (распознавание именованных сущностей)
- question-answering (вопрос-ответ)
- sentiment-analysis (классификация с обучением)
- summarization (главная мысль)
- text-generation (генерация текста)
- translation (перевод)
- zero-shot-classification (классификация без обучения)

Zero-shot classification

Мы начнем с более сложной задачи, где нам нужно классифицировать тексты, которые не были размечены (для них нет ответов - позитивный или негативный, агрессивный или нейтральный и тд).

Это распространенный сценарий в реальных проектах, потому что разметка текста обычно занимает много времени и требует знаний в предметной области.

Для этого варианта использования очень мощным является пайплайн zero-shot-classification: он позволяет указать, какие метки использовать для классификации, поэтому вам не нужно полагаться на метки предварительно обученной модели.

```
from transformers import pipeline

classifier = pipeline("zero-shot-classification")
classifier(
    "This is a course about the Transformers library",
    candidate_labels=["education", "politics",
"business"],
)
```

```
Using a pipeline without specifying a model name and
revision in production is not recommended.
```

```
'labels': ['education', 'business', 'politics'],

 'scores': [0.844,       0.111,      0.043]}
```

Этот пайплайн называется zero-shot потому, что вам нет необходимости дообучать модель для использования.

Она может вернуть вам оценки вероятности для любого списка меток, который вы хотите!

```
from transformers import pipeline

classifier = pipeline("zero-shot-classification")

classifier(
    "This is a course about the Transformers library",
    candidate_labels=["education",
                      "training",
                      "snowboarding"],
)
```

```
 ['education', 'training', 'snowboarding'],

 [0.682,         0.2797,      0.0382 ]}
```


Генерация текста

Теперь давайте взглянем на пайплайн генерации текста (англ. text generation).

Главная идея заключается в следующем: вы передаете на вход модели небольшой фрагмент текста, а модель будет продолжать его.

Генерация текста содержит в себе элемент случайности, поэтому ваш результат может отличаться от того, который приведен ниже в примере.

```
from transformers import pipeline

generator = pipeline("text-generation")

generator("In this course, we will teach you how to")
```

```
In this course, we will teach you how to create
beautiful and functional software that we believe is
truly available to the community.
```

Использование модели из Hub в пайплайне: **model="distilgpt2"**

```
from transformers import pipeline

generator = pipeline("text-generation",
model="distilgpt2")

generator(
    "In this course, we will teach you how to",
    max_length=35,
    num_return_sequences=2,
)
```

```
In this course, we will teach you how to get the most
out of the basics of photography and how to get
comfortable with your shots.

We will explain how to take on a camera.
```


Вы можете уточнить, для какого языка вам нужна модель, щелкнув на языковые теги, и выбрать ту, которая будет генерировать текст на другом языке.

На Model Hub даже есть предобученные модели для многоязычных задач.

Как только вы выберете модель, вы увидите, что есть виджет, позволяющий вам попробовать ее прямо на сайте. Таким образом, вы можете быстро протестировать возможности модели перед ее загрузкой.

Заполнение пропусков

Аргумент top_k указывает, сколько вариантов для пропущенного слова будет отображено. Обратите внимание, что модель заполнит пропуск на месте слова , которое часто интерпретируют как mask token (токен-маска). Другие модели могут использовать другие токены для обозначения пропуска, всегда лучше проверять это. Один из способов сделать это - обратить внимание на виджет для соответствующей модели.

```
from transformers import pipeline

unmasker = pipeline("fill-mask")

unmasker("This course will teach you all about <mask>
models.", top_k=3)
```



```
[{'score': 0.19198468327522278,
  'token': 30412,
  'token_str': ' mathematical',
  'sequence': 'This course will teach you all about
mathematical models.'},
 {'score': 0.04209203273057937 6,
  'token': 38163,
  'token_str': ' computational',
  'sequence': 'This course will teach you all about
computational models.'},
 {'score': 0.03602461889386177,
  'token': 27930,
  'token_str': ' predictive',
  'sequence': 'This course will teach you all about
predictive models.'}]
```

Распознавание именованных сущностей (NER)

Распознавание именованных сущностей (англ. named entity recognition) - это задача, в которой модели необходимо найти части текста, соответствующие некоторым сущностям, например: **персонам, местам, организациям**.

```
from transformers import pipeline

ner = pipeline("ner", grouped_entities=True)

ner("My name is Sylvain and I work at Hugging Face in Brooklyn.")
```

```
[{'entity_group': 'PER', 'score': 0.9981694, 'word': 'Sylvain', 'start': 11, 'end': 18},

{'entity_group': 'ORG', 'score': 0.9796019, 'word': 'Hugging Face', 'start': 33, 'end': 45},

{'entity_group': 'LOC', 'score': 0.9932106, 'word': 'Brooklyn', 'start': 49, 'end': 57}]
```


В этом примере модель корректно обозначила Sylvain как персону (PER), Hugging Face как организацию (ORG) и Brooklyn как локацию (LOC).

Мы передали в пайплайн аргумент grouped_entities=True для того, чтобы модель сгруппировала части предложения, соответствующие одной сущности: в данном случае модель объединила “Hugging” и “Face” несмотря на то, что название организации состоит из двух слов.

Вопросно-ответные системы

Пайплайн question-answering позволяет сгенерировать ответ на вопрос по данному контексту:

```
from transformers import pipeline

question_answerer = pipeline("question-answering")

question_answerer(
    question="Where do I work?",

    context="My name is Sylvain and I work at Hugging Face in Brooklyn",
)
```

```
'answer': 'Hugging Face'
```


Обратите внимание, что пайплайн извлекает информацию для ответа из переданного ему контекста.

Автоматическое реферирование (саммаризация)

Автоматическое реферирование (англ. summarization) - задача, в которой необходимо сократить объем текста, но при этом сохранить все важные аспекты (или большинство из них) изначального текста.

Так же, как и в задаче генерации текста, вы можете указать максимальную длину **max_length** или минимальную длину **min_length** результата.

```
from transformers import pipeline

summarizer = pipeline("summarization")
summarizer(
    """
    America has changed dramatically during recent years. Not only has the number of
    graduates in traditional engineering disciplines such as mechanical, civil,
    electrical, chemical, and aeronautical engineering declined, but in most of
    the premier American universities engineering curricula now concentrate on
    and encourage largely the study of engineering science. As a result, there
    are declining offerings in engineering subjects dealing with infrastructure,
    the environment, and related issues, and greater concentration on high
    technology subjects, largely supporting increasingly complex scientific
    developments. While the latter is important, it should not be at the expense
    of more traditional engineering.

    Rapidly developing economies such as China and India, as well as other
    industrial countries in Europe and Asia, continue to encourage and advance
    the teaching of engineering. Both China and India, respectively, graduate
    six and eight times as many traditional engineers as does the United States.
    Other industrial countries at minimum maintain their output, while America
    suffers an increasingly serious decline in the number of engineering graduates
    and a lack of well-educated engineers.
""",
    max_length=20
)
```

```
[{'summary_text': ' America has changed dramatically during recent years. The number of engineering graduates in the U.'}]
```

Перевод

Для перевода вы можете использовать модель по умолчанию просто указав пару языков, между которыми будет осуществляться перевод (например, "translation_en_to_fr"). Однако самый простой способ - попробовать использовать Model Hub.

```
# Перевод с английского языка на немецкий язык
from transformers import pipeline

translator = pipeline("translation",
                      model="Helsinki-NLP/opus-mt-en-de")

translator("This course is produced by Hugging Face.")
```

```
[{'translation_text':

'Dieser Kurs wird von Hugging Face produziert.'}]
```