Нижегородский государственный университет им. Н.И. Лобачевского
Институт информационных технологий, математики и механики
Кафедра Математического обеспечения и суперкомпьютерных технологий

Летняя школа Intel – ННГУ по компьютерному зрению

Практическая работа №5
Использование OpenVINO в обработке естественного языка

Васильев Е.П.

Нижний Новгород
2021

1 Цели и задачи работы

Цель работы состоит в изучении задач, входящих в область обработки естественного языка (Natural Language Processing, NLP) и знакомстве с возможностями OpenVINO в NLP задачах.

Для достижения поставленной цели необходимо решить следующие задачи:

- Изучить приложение BERT Question Answering Python Demo
- Реализовать обертку над BERT Question Answering Python Demo для получения ответов на вопросы про OpenVINO И OpenCV

Отметим, что процедура настройки рабочего окружения подробно описана в предыдущей практике, поэтому данный шаг в настоящем описании опущен.

2 Задачи обработки естественного языка

Анализ текстов. Анализ текстов реализуется в трёх основных форматах: классификации, отражении содержания и анализе тональности.

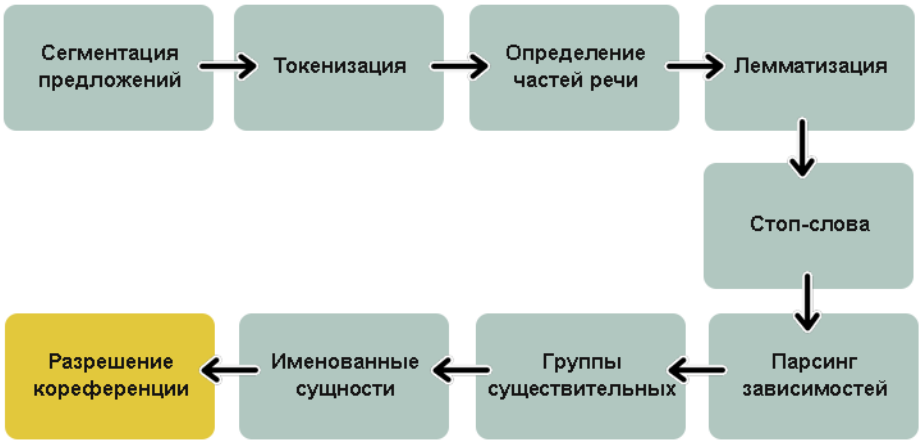
Распознавание и синтез речи. Распознавание речи представляет собой процесс преобразования речевого сигнала в цифровую информацию, например, в текст. Синтез речи работает в обратном направлении, формируя речевой сигнал по печатному тексту.

Разработка диалоговых систем. Диалоговыми системами можно считать умные помощники (Яндекс.Алиса, Siri, Alexa), чат-боты — текстовые системы, следующие сценариям диалога, QA-системы.

Выделение сущностей и фактов. Ещё одна популярная задача NLP - извлечение именованных сущностей (Named-entity recognition, NER) из текста. Задача NER - понять, что участок текста «1999 года» является датой, «Иван Петров» - персоной, а «пакет акций» - активом.

2.1 Общая схема конвейера NLP

Реализация любой сложной задачи обычно означает построение пайплайна (конвейера).



Суть этого подхода в том, чтобы разбить задачу на ряд последовательных подзадач и решать каждую из них отдельно. В построении пайплайна можно условно выделить две

части: предобработку входных данных (обычно занимает больше всего времени) и построение модели. Основных этапов — семь.

1. Первые два шага пайплайна, которые выполняются для решения практически любых задач NLP, — это сегментация (деление текста на предложения) и токенизация (деление предложений на токены, то есть отдельные слова).

2. Вычисление признаков каждого токена. Вычисляются контекстно-независимые признаки токена. Это набор признаков, не зависящих от соседних с токеном слов.

- Один из самых часто использующихся признаков — часть речи.

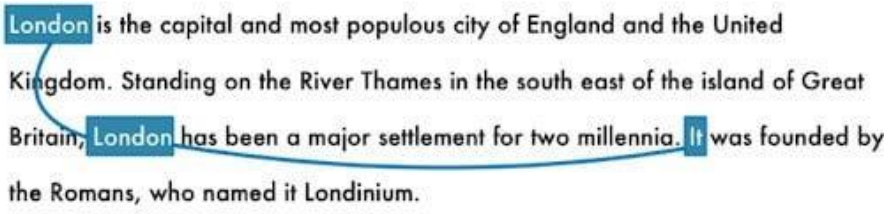
- Для языков со сложной морфологией (русский язык) также важны морфологические признаки: например, в каком падеже стоит существительное, какой род у прилагательного. Из этого можно сделать выводы о структуре предложения.
- Морфология также нужна для приведения слов к начальной форме, с помощью которой мы можем уменьшить объём признакового пространства.

Например: I had a pony. I had two ponies.

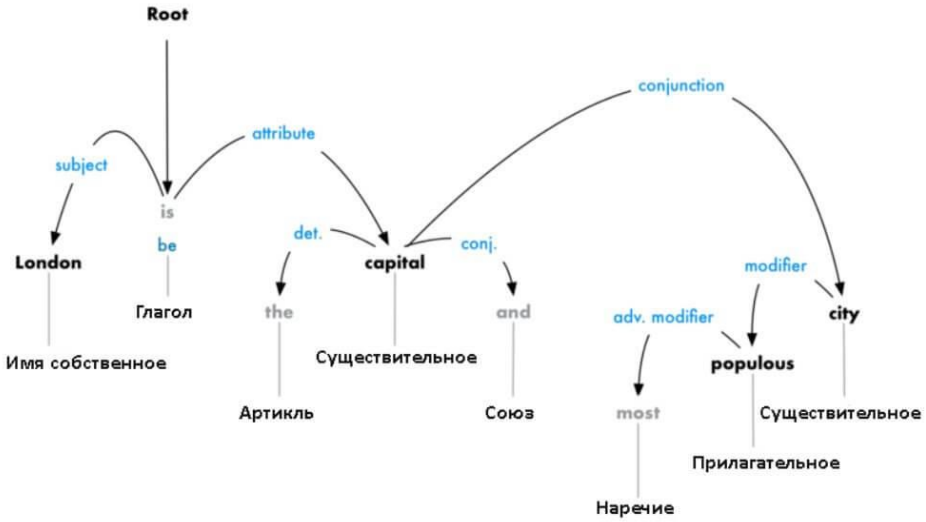
Оба предложения содержат существительное «pony», но с разными окончаниями. Если тексты обрабатывает компьютер, он должен знать начальную форму каждого слова, чтобы понимать, что речь идёт об одной и той же концепции пони. Иначе токены «pony» и «ponies» будут восприняты как совершенно разные. В NLP этот процесс называется лемматизацией.

3. Определение значимости и фильтрация стоп-слов. В русском и английском языках очень много вспомогательных слов, например «and», «the», «a». При статистическом анализе текста эти токены создают много шума, так как появляются чаще, чем остальные. Поэтому их отмечают как стоп-слова и отсеивают.

4. Разрешение кореференции. В русском и английском языках очень много местоимений вроде he, she, it или ты, я, он и т. д. Это сокращения, которыми мы заменяем на письме настоящие имена и названия. Человек может проследить взаимосвязь этих слов от предложения к предложению, основываясь на контексте. Но NLP-модель не знает, что означают местоимения, ведь она рассматривает всего одно предложение за раз.



5. Парсинг зависимостей. Конечная цель этого шага — построение дерева, в котором каждый токен имеет единственного родителя. Корнем может быть главный глагол. Также нужно установить тип связи между двумя словами:



Это дерево парсинга демонстрирует, что главный субъект предложения — это существительное «London». Между ним и «capital» есть связь «be». Вот так мы узнали, что Лондон — это столица. Если бы мы проследовали дальше по веткам дерева (уже за границами схемы), то могли бы узнать, что «London is the capital of Great Britain».

6. Перевод обработанного текста в векторную форму. Данный шаг позволяет сформировать векторные представления слов. Таким образом, у слов, используемых в одном и том же контексте, похожие векторы.

7. Построение модели в зависимости от поставленной цели. Например, модель для классификации или генерации новых текстов.

Приведённый пример пайплайна не является единственно верным. Для решения конкретной задачи некоторые шаги можно исключить или добавить новые. Однако этот пайплайн содержит все наиболее типичные этапы и подходы, позволяющие извлекать практическую пользу из NLP.

3 Запуск BERT Question Answering Python Demo

Для запуска BERT Question Answering Python Demo требуется скачать не только модель, но и файл словаря, который будет использоваться для токенизации. Файл словаря уже лежит в папке с практикой, скачать уже конвертированную модель можно командой ниже.

```
python downloader.py --name bert-small-uncased-whole-word-masking-squad-0001
```

После этого можно запустить демо.

```
python bert_question_answering_demo.py -v vocab.txt -m bert-small-uncased-whole-word-masking-squad-0001.xml --
input="https://en.wikipedia.org/wiki/Intel" -c
```

Если запустить демо без параметра **--questions**, то оно будет работать в интерактивном режиме. Будут выдаваться вопросы на ответы на английском языке.

4 Автоматизация запуска демо с различными параметрами и парсинг вывода

В некоторых ситуациях требуется не писать программу «с нуля», а использовать готовое приложение (возможно многократно) из своей программы, для запуска нового приложения внутри своей Python программы используется модуль **subprocess**. Представленная в файле **automate_it.py** программа формирует командную строку для запуска демо с параметрами, взятыми из переменных.

```
import argparse
import subprocess


def parse_cmd_output(text):
    # Cut technical information from cmd output

    return text


def build_argparser():
    parser = argparse.ArgumentParser(description="Simple object tracker demo")
    parser.add_argument("-m", required=True, help="Path to the model")
    parser.add_argument("-q", required=False, help="Path to the questions file")
    parser.add_argument("-i", required=False, help="Path to the input sites file")
    return parser


def main():
    args = build_argparser().parse_args()

    # Prepare input parameters for script
```

```
    path_to_demo = "C:/Program Files
(x86)/Intel/openvino_2021.3.394/deployment_tools/open_model_zoo/demos/bert
_question_answering_demo/python/bert_question_answering_demo.py"
    path_to_model = args.m #"bert-small-uncased-whole-word-masking-squad-
0001/FP32/bert-small-uncased-whole-word-masking-squad-0001.xml"
    question = "What operating system is required?"
    site = "https://en.wikipedia.org/wiki/OpenVINO"

    # Prepare text command line
    cmd = f'python "{path_to_demo}" -v vocab.txt -m {path_to_model} --
input="{site}" --questions "{question}"'

    # Run subprocess using prepared command line
    returned_output = subprocess.check_output(cmd)

    # Process output
    answer = parse_cmd_output(returned_output)

    # Write output to file
    with open('answer.txt', 'w') as the_file:
        the_file.write(f'{answer}\n')


if __name__ == "__main__":
    main()
```

В текущем варианте программы в файл записывается не только ответ на заданный вами вопрос, но и множество служебной информации.

Реализуйте функцию **`parse_cmd_output`**(), чтобы в выходной файл записывались только ответы, данные моделью Bert на вопрос.

5 Дополнительные задания

- Реализуйте чтение файла с вопросами и сбор ответов для каждого вопроса в файл.
- Реализуйте чтение файла с сайтами и сбор ответов на все вопросы для каждого сайта.

6 Литература

1. Шолле Ф. Глубокое обучение на Python. – СПб.: Питер. – 2018. – 400с.
2. Girshick R., Donahue J., Darrell T., Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. – 2014.
3. Dai J., Li Y., He K., Sun J. R-FCN: Object detection via region-based fully convolutional networks. – 2016.
4. Рамальо Л. Python. К вершинам мастерства / Пер. с англ. Слинкин А.А. – М.: ДМК Пресс. – 2016. – 768 с.
5. Страница репозитория Open Model Zoo [https://github.com/openvinotoolkit/open_model_zoo].
6. Документация Intel Distribution of OpenVINO Toolkit [https://docs.openvinotoolkit.org/latest/index.html].
7. BERT Question Answering Python Demo [https://docs.openvinotoolkit.org/latest/omz_demos_bert_question_answering_demo_python.html]
8. Правильный NLP: как работают и что умеют системы обработки естественного языка [https://tproger.ru/articles/natural-language-processing/]