УДК 004.891.2

ИЗВЛЕЧЕНИЕ ФРАГМЕНТОВ ТЕКСТА

Ермолович Д.С. [1], студент гр.053504

Белорусский государственный университет информатики и радиоэлектроники[1]
г. Минск, Республика Беларусь

Боброва Н.Л. – доцент кафедры информатики

Аннотация. Этот проект — решение проблемы извлечения необходимого фрагмента текста, соответствующего запросу. Модель машинного обучения работает с русскими текстами. Цель исследования — понять, какой подход к решению этой проблемы лучше, чем QA или NER. Весь код имеет открытый исходный код, вот ссылка на GitHub: перейдите по следующему URL: https://github.com/YermalovichDzmitry/Text_Extraction_NLP_2023 .

Ключевые слова. machine learning, artificial intelligence, transformers, deep learning, natural language processing.

ВВЕДЕНИЕ

Часто при работе с документами приходиться приходиться извлекать: арбитражные иски, госзакупки, исполнительные производства. Это нужно для того, чтобы сформировать анкету заявки. Но человек справляется с этой задачей не быстро: ему нужно прочитать текст, проанализировать его, выделить нужный фрагмент текста. Эти действия занимают время. Поэтому была разработана модель машинного обучения, которая поможет отделу госзакупок извлечь из документа необходимый фрагмент текста для формирования формы заявки.

1 ОПИСАНИЕ МОДЕЛИ

Решаемая проблема была сведена к двум: *QA* и *NER*. В качестве модели был выбран *sbert_large_nlu_ru*, поскольку он подходит для русских текстов, а модели *nlu* универсальны, то есть их можно использовать для *QuestionAnswering*, *NER*, *classification*.

1.1 *QA*

Идея: модели на вход даются вопрос и текст, и она предсказывает начало и конец ответа.
На рис. 1 показана архитектура модели Bert, которая решает QA проблему.

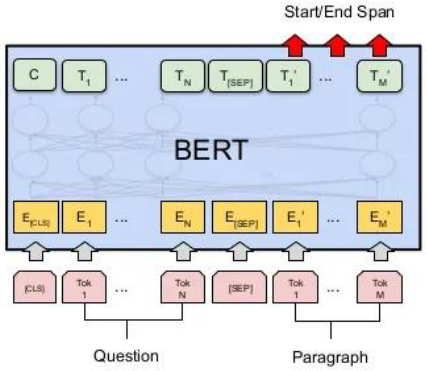

Рисунок 1 – *Bert* для *QA*

Данные обучения были разделены на *train/val/test* с соотношением 0,8/0,1/0,1.
Данные были приведены к структуре:
1 *input_ids* — индексы токенов в словаре.
2 *attention_mask* — устанавливает 1, если токены присутствуют, и 0, где их нет.
3 *start_token* — индекс стартового токена, уже смещен относительно вопроса.
4 *end_token* — индекс конечного токена, уже смещен относительно вопроса.
5 *segment_ids*— 0 — вопрос, 1 — текст.
6 *shift*. *Shift* показывает, на сколько токенов был сдвинут ответ относительно вопроса.

1.2 NER

Идея: в качестве входных данных модели передается текст, и она выбирает объекты, являющиеся метками.

На рисунке 2 показана архитектура модели Берта, которая решает проблему *NER*.

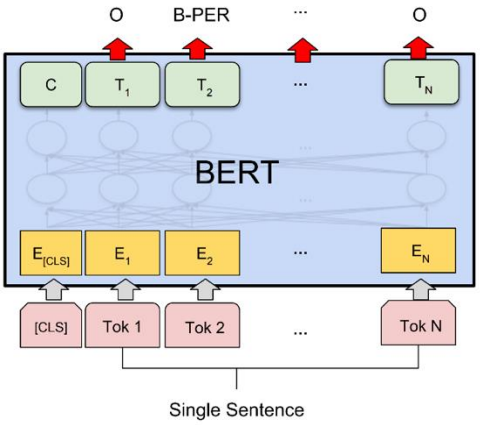

Рисунок 2 – *Bert* для *NER*

Словари сущностей, которые были определены.

label2id = {'[PAD]': 0, 'O': 1, 'B-EXE': 2, 'I-EXE': 3, 'B-GUA': 4, 'I-GUA': 5}

id2label = {0:'[PAD]', 1:'O', 2:'B-EXE', 3:'I-EXE', 4:'B-GUA',5:'I-GUA'}

BIO концепция была использована. *EXE* – “обеспечение исполнения контракта”, *GUA* – “обеспечение гарантийных обязательств”

В этой задаче, '*input_ids*', '*attention_mask*', 'labels' были выбраны. Также я решил проблему с длинными последовательностями.

Предварительная обработка для *NER*

Были выделены индексы тех текстов, на которые запрос не нашел ответа. Такие тексты были исключены из обучающей выборки, так как основная цель *NER* — найти сущности "обеспечение исполнения контракта" или "обеспечение гарантийных обязательств", и если оно не исключено, делается запрос на "обеспечение исполнения контракта" и ничего не сделает *NER* не нашел, а ведь есть "обеспечение гарантийных обязательств", то это запутает модель.

Данные обучения были разделены на *train/val/test* с соотношением 0,8/0,1/0,1 относительно уже исключенных пропусков. Но эти пробелы были добавлены позже в тест.

2 DATASET

Набор данных был предоставлен *Kontur*. Был дан текст, запрос на него, текст ответа, а также «answer_start» и «answer_end» ответа в тексте. В таблице 1 приведены типы данных и количество примеров, которые им соответствуют

Таблица 1 - Тип данных и количество примеров.

Тип данных	Количество примеров
Train dataset	1799
Test dataset	318

Тестовый набор данных был закрыт на этапе решения.

Кроме того, до конца соревнования для тестовых данных не было предоставлено « answer_start» и «answer_end», поэтому набор *test* данных был разделен на train/val/test. В данной работе тестовые данные, на которых не были представлены «answer_start» и «answer_end», были названы закрытыми, а тестовые данные, сформированные для теста — открытыми.

Есть два типа запросов: “обеспечение исполнения контракта” и “обеспечение гарантийных обязательств”. В таблице 2 приведены *labels* и их количество

Таблица 2 - *Labels* и их количество.

Label name	Количество *labels*
Обеспечение исполнения контракта	988
Обеспечение гарантийных обязательств	811

Данные сбалансированы по меткам.

В части наблюдений отсутствует фрагмент текста, который нужно извлечь (пустая строка внутри поля «extracted_part», где «answer_start» и «answer_end» равны нулю). Это значит, что в тексте документа нет необходимого фрагмента текста, соответствующего пункту анкеты.

Количество пустых фрагментов = 307. В таблице 3 приведены количество пустых фрагментов на тип запроса.

Таблица 3 - Количество пустых фрагментов на тип запроса.

Label name	Number of passes
Обеспечение исполнения контракта	4
Обеспечение гарантийных обязательств	303

Большая часть пропусков принадлежит “Обеспечение гарантийных обязательств”.

3 ЭКСПЕРИМЕНТЫ

3.1 Метрики

Для оценки модели использовалась метрика «*Accuracy*»: доля наблюдений, в которых извлеченный моделью фрагмент текста полностью соответствует истине.

3.2 Настройка эксперимента

3.2.1 Установка эксперимента для *QA Bert*

Базовая *BertConfig* для *sbert_large_nlu_ru*,
learning_rate = 5e-5
Epochs = 5, но после второй эпохи модель начала переобучаться, модель была взята со второй эпохи.
*clip_grad_norm_* с *max_norm=max_grad_norm* = 1.0
optimizer = AdamW

3.2.1 Установка эксперимента для NER Bert

Базовая *BertConfig* для *sbert_large_nlu_ru*,
learning_rate = 5e-5
Epochs = 7, но после 5 эпохи модель начала переобучаться, модель взята из пятой эпохи.
*clip_grad_norm_* с *max_norm=max_grad_norm* = 1.0
optimizer = AdamW

3.3 Результаты моделей и анализ ошибок

3.3.1 QA

После обучения модели «*Accuracy*» открытого тестового набора данных составила 0.77, а на закрытом тестовом наборе данных — 0,79. В таблице 4 приведены QA результаты.

Таблица 4 – *QA results*.

Тип тестовых данных	Название модели	Токенизатор	Точность
Открытый	sbert_ large_nlu_ru	WordPiece	0.77
Закрытый	sbert_ large_nlu_ru	WordPiece	0.79

После анализа ошибок выяснилось, что в большинстве случаев, когда модель допустила ошибку, она правильно идентифицирует начало, но неправильно — конец. Иногда она выделяет чуть больше информации, иногда чуть меньше. Примеры, когда модель допустила ошибку, можно увидеть в *notebook* на GitHub.

3.3.2 *NER*

Точность на всех тестовых данных = 0.88.

Данные на тесте несбалансированные, то есть модель обучалась без данных с start = 0 и end = 0, поэтому в тест передавались данные, на которых она не обучалась. И алгоритм определяет хорошо, start = 0 и end = 0, поэтому точность очень высокая.

Результат очень хороший, но в этих тестовых данных 307 пустых, то есть start=0 и end=0. Непонятно, насколько точно алгоритм выделяет текст там, где он на самом деле находится. Поэтому в следующем тесте пустые примеры удалялись и точность оценивалась по тем примерам, которые обязательно содержат текст.

Точность тестовых данных без данных с start = 0 и end = 0 = 0,75.

Точность данных закрытых испытаний = 0.78.

В таблице 5 приведены *NER* результаты.

Таблица 5 – NER результаты.

Тип тестовых данных	Название модели	Токенизатор	Точность
Открытый без пропусков	sbert_ large_nlu_ru	WordPiece	0.75
Открытый с пропусками	sbert_ large_nlu_ru	WordPiece	0.88
Закрытый	sbert_ large_nlu_ru	WordPiece	0.78

Изучение неправильно выделенных предложений.

После анализа ошибок выяснилось, что в большинстве случаев, когда модель допустила ошибку, она правильно определяет начало, но неверно конец: то ли чуть больше информации, то ли чуть меньше. Примеры, где модель допустила ошибку, можно увидеть в *notebook* на GitHub.

6 Полученные результаты

В таблице 6 приведены QA и NER результаты

Таблица 6 – QA и NER результаты.

Тип тестовых данных	Название модели	Токенизатор	Точность
Открытый без пропусков NER	sbert_ large_nlu_ru	WordPiece	0.75
Открытый с пропусками NER	sbert_ large_nlu_ru	WordPiece	0.88
Закрытый NER	sbert_ large_nlu_ru	WordPiece	0.78
Открытый QA	sbert_ large_nlu_ru	WordPiece	0.77
Закрытый QA	sbert_ large_nlu_ru	WordPiece	0.79

Изучение результатов двух моделей

После обучения двух моделей их результаты были проанализированы на закрытом тестовом наборе данных. В выяснилось, что 76% выбранных текстов совпали. Это говорит о том, что модели работают хорошо. Оказалось, что многие примеры, в которых модели дают разные результаты, очень

похожи друг на друга, то есть имеют одинаковое начало, но разное окончание выделенного текста. Также в этих примерах очень сложно выделить окончание, поскольку в большинстве случаев оно заканчивается либо «___», либо «%», либо транскрипцией цифр. То есть такие окончания, что даже человеку не всегда очевидно, где заканчивается выделенный текст.

ВЫВОД ПО НАУЧНОЙ РАБОТЕ

В результате работы были реализованы и обучены две модели: BertForQuestionAnswering и BertForTokenClassification. В результате изучения результатов было установлено, что процент совпадения между ними составляет 76%. Примеры, где у них разные ответы, эти ответы очень похожи, то есть у многих из них одинаковое начало, но немного разный конец. Результат с BertForQuestionAnswering был выбран в качестве прогнозов, поскольку результаты на открытом наборе тестовых данных выше, точность на открытом наборе тестовых данных BertForQuestionAnswering = 0,77 и BertForTokenClassification = 0,75. Кроме того, результаты на наборе данных закрытого тестирования оказались выше для BertForQuestionAnswering, точность BertForQuestionAnswering = 0,79, точность BertForTokenClassification = 0,78.

Список источников

1 Искусство распознавания: как мы разрабатывали прототип AutoML для задачи Named Entity Recognition [Electronic resource]. – Access mode : https://habr.com/ru/companies/vtb/articles/651525/.–Дата доступа: 22.02.2024.

UDC 004.891.2

EXTRACTING TEXT SNIPPETS

Yermalovich D.S. [1]

Belarusian State University of Informatics and Radioelectronics[1]*, Minsk, Republic of Belarus*

Bobrova N.L. – Associate Professor at the Department of Infotmatics

Annotation. This project is a solution to the problem of extracting the necessary piece of text that matches a query. The machine learning model works with Russian texts. The goal of the study is to understand which approach is better than QA or NER to solve this problem. All code is open source, here is the link to GitHub: Go to the following URL: https://github.com/YermalovichDzmitry/Text_Extraction_NLP_2023.

Keywords. machine learning, artificial intelligence, transformers, deep learning, natural language processing.