Постановка задачи

Требуется разработать retrieval-based чат-бота, который отвечает на вопросы реплика определенного персонажа сериала.

Github проекта: https://github.com/YurezSml/MIPT_NLPGener_HW1

Обученная модель: https://disk.yandex.ru/d/kt6NxfKMJADhdg

Colab с запуском чат-бота: https://colab.research.google.com/drive/15yGyk3K_r-KSFm2ZilEsja0z9fnq8rVl

Описание и анализ данных данных

В качестве рассматриваемого сериала был выбран сериал «Друзья»

Источник данных: https://www.kaggle.com/datasets/gopinath15/friends-netflix-script-data

Датасет состоит из 5 столбцов

Text – реплика персонажа и технический текст

Speaker – персонаж, говорящий реплику

Episode – номер и название эпизода

Season – номер сезона

Show – название шоу

Всего датасет содержит 69974 записи. Как будет видно далее, из анализа, не все они являются репликами персонажей, часть можно отнести к техническим записям, связанным с рекламными вставками, окончанием серии и т.д. Несколько записей исходного датасета приведены на рисунке 1.

Ввод [123]:
```
df = pd.read_csv('Friends.csv')
df
```

Out[123]:

	Text	Speaker	Episode	Season	Show
0	Originally written by Marta Kauffman and David...	NaN	Episode-01-The One Where Monica Gets a New Roo...	Season-01	Friends
1	Transcribed by guineapig.	NaN	Episode-01-The One Where Monica Gets a New Roo...	Season-01	Friends
2	CENTRAL PERK. (ALL PRESENT EXCEPT RACHEL AND ...	SCENE 1	Episode-01-The One Where Monica Gets a New Roo...	Season-01	Friends
3	There's nothing to tell! He's just some guy I...	MONICA	Episode-01-The One Where Monica Gets a New Roo...	Season-01	Friends
4	C'mon, you're going out with the guy! There's...	JOEY	Episode-01-The One Where Monica Gets a New Roo...	Season-01	Friends
...	...	...	...	...	...
69969	Then I'm happy too. (They're still hugging - ...	Ross	Episode-15-The One Where Estelle Dies	Season-10	Friends
69970	COMMERCIAL BREAK	NaN	Episode-15-The One Where Estelle Dies	Season-10	Friends
69971	Estelle's memorial service. Joey is giving a ...	[Scene	Episode-15-The One Where Estelle Dies	Season-10	Friends
69972	Thank you all for coming. We're here today to...	Joey	Episode-15-The One Where Estelle Dies	Season-10	Friends
69973	THE END	NaN	Episode-15-The One Where Estelle Dies	Season-10	Friends

69974 rows × 5 columns

Рисунок 1 – Исходный датасет, состояние до обработки

В ходе анализа и первичной обработки датасета все имена авторов реплик были приведены к одному виду. В датасете обнаружены пропущенные значения для поля с авторами реплик (6284), в ходе анализа было принято решение использовать часть из них в качестве возможных разделителей при разбиении датасета на несвязанные диалоги. Остальные записи с пропущенными авторами реплик были удалены, как разрывающие контекст внутри диалога. Кроме того, в качестве разделителей между диалогами были приняты специальные отметки в поле ‘Text’.

Результаты анализа и первичной обработки представлены в файле https://github.com/YurezSml/MIPT_NLPGener_HW1/blob/main/Analysys.ipynb

Подготовка данных

В результате первичной обработки был подготовлен датасет, состоящий из двух полей:

Text – реплика персонажа и технический текст

Speaker – персонаж, говорящий реплику

Независимые или слабо зависимые по контексту диалоги отделены с помощью разделителя 'break' в поле ‘Speaker’ (Рисунок 2). Нельзя говорить о полной независимости диалогов друг от друга т.к. возможно продолжение общего контекста как через несколько диалогов, так и в разных сериях. Чтобы устранить это и сделать максимально связный контекстом датасет с репликами требуется значительное количество времени, ручного труда и погружение в сюжет.

	Text	Speaker
0	central perk.	break
1	there's nothing to tell! he's just some guy i ...	monica
2	c'mon, you're going out with the guy! there's ...	joey
3	so does he have a hump? a hump and a hairpiece?	chandler
4	wait, does he eat chalk?	phoebe
...	...	...
63484	yeah, yeah, oh!	ross
63485	oh! oh, i'm so happy.	rachel
63486	then i'm happy too.	ross
63487	estelle's memorial service. joey is giving a s...	break
63488	thank you all for coming. we're here today to ...	joey

63489 rows × 2 columns

Рисунок 2 – Датасет после первичной обработки

Полученный датасет позволяет выбрать реплики любого из персонажей в качества объекта для обучения. В своей работе я выбрал Чендлера (chandler), поскольку его реплики и шутки являются одними из самых ярких в сериале и выделяются на фоне остальных. Реплики персонажей преобразовывались в датасет для обучения следующим образом: в качестве ответа записывается реплика выбранного персонажа. В поле с вопросом записываются предшествующие реплики других персонажей, но не более заданного количества реплик REPL_DEPTH, определяющего глубину контекста для реплики. Если между выбранным персонажем и другими происходит диалог, то предыдущие пары вопрос-ответ записываются в поле с контекстом, но не более заданного количества пар CONT_DEPTH, определяющего глубину контекста диалога. На рисунке 3 представлено распределение количества реплик в зависимости от глубины контекста диалога. Дополнительное исследование показало, что максимальный из диалогов имеет 84 предшествующих пары вопрос-ответ.

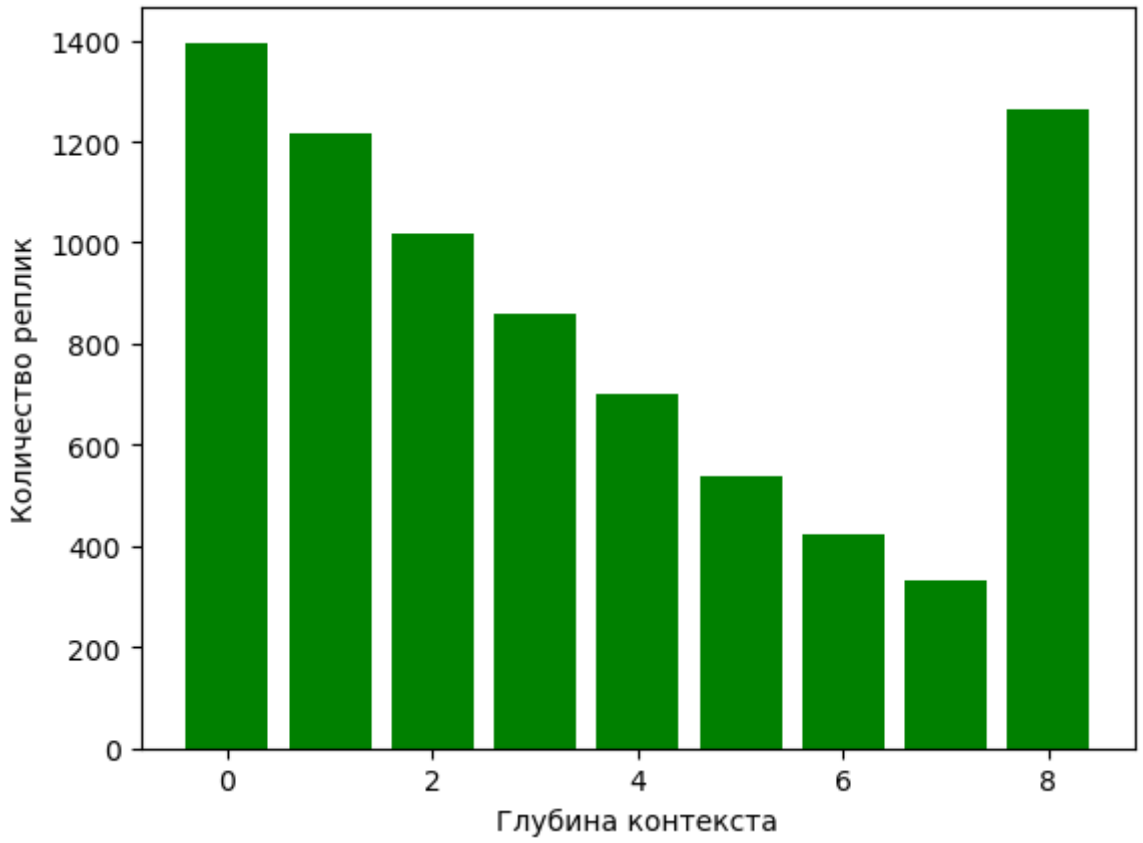


Рисунок 3 – Количество реплик от глубины контекста диалогов в итоговом датасете

Архитектура и обучение модели

В качестве модели для дообучения я выбрал distilbert. Также пробовал дообучать bert, но скорость обучения на доступном мне железе сильно ниже. Модель дополнялась линейным слоем, в токенизатор добавлялся спецтокен [U_token]. Архитектура обучения модели представлена на рисунке 4.

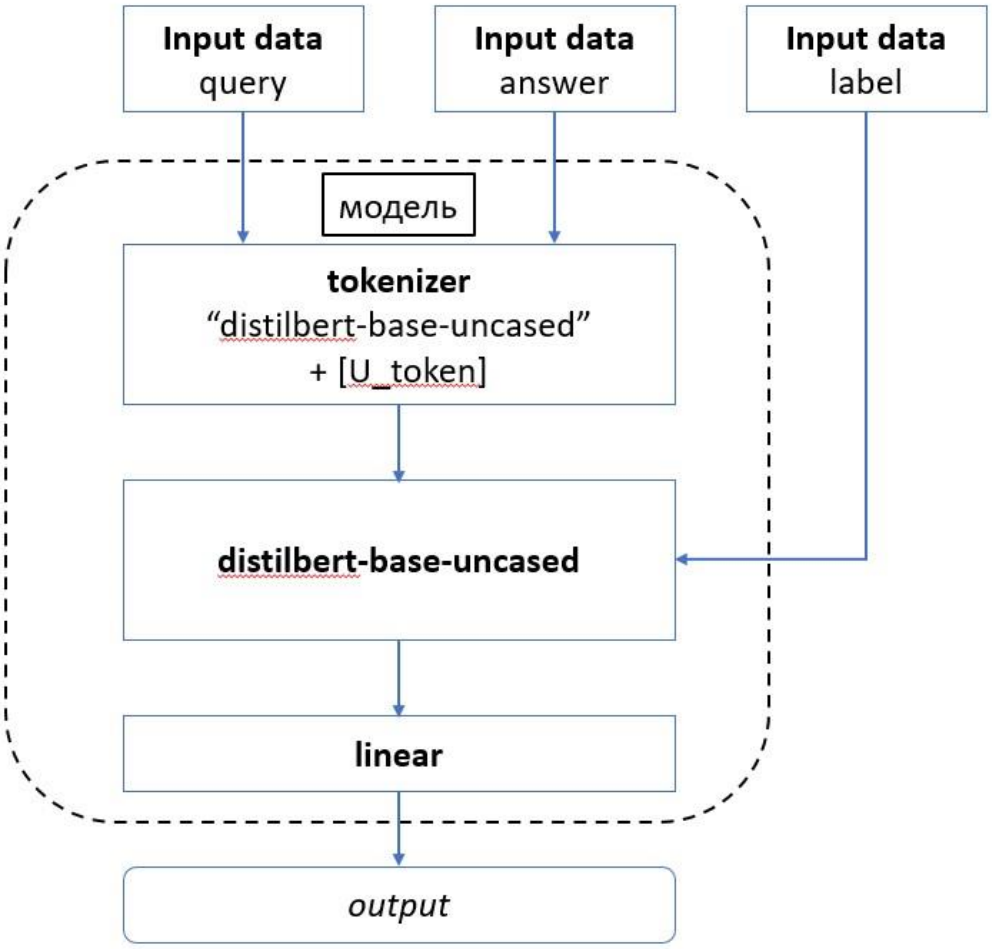


Рисунок 4 – Архитектура обучения модели

При подготовке датасета для обучения с помощью [U_token] отделялись реплики, соответствующие вопросам и ответам в контексте. Для определения максимальной длины последовательности в токенайзере анализируем распределение вопросов и ответов в зависимости от их длины (Рисунок 5). По результатам анализа максимальная длина была установлена равной 512.

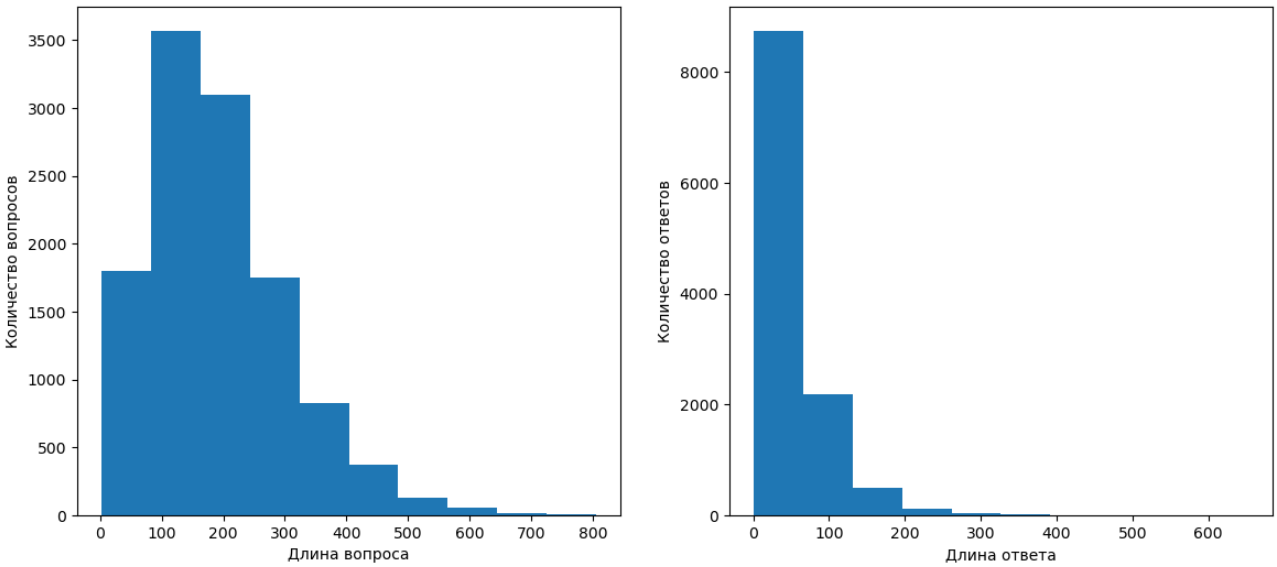
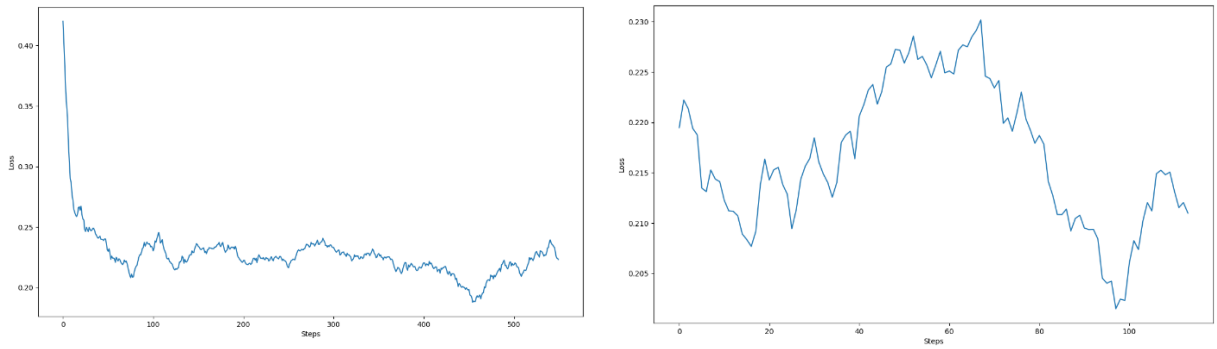


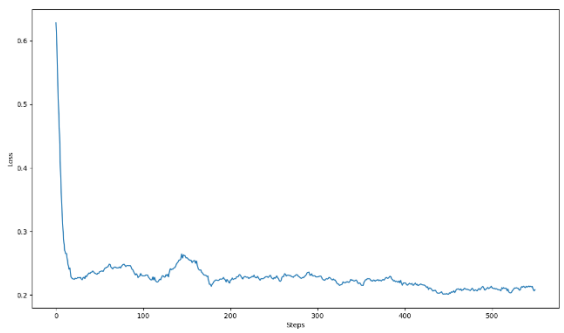
Рисунок 5 – Количество вопросов/ответов от их длины

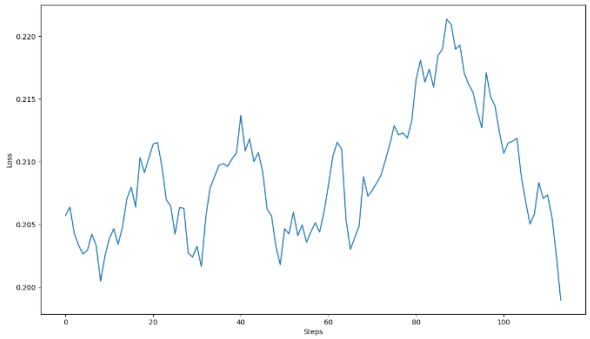
Модель обучалась одну эпоху т.к. более длительное обучение (анализировалось вплоть до 10 эпох) не оказало существенного влияния на результат. В ходе обучения модели возникло предположение, что она возможно попадает в локальный минимум и можно попытаться преодолеть его, увеличив шаг обучения. Было проанализировано влияние шага и шедулера на результаты обучения, результаты представлены на рисунках 6 – 9. Можно сделать вывод, что наиболее стабильный результат получается при обучении с линейным шедулером и шагом, равным 1E-4. Для всех вариантов обучение проводилось несколько раз, для анализа воспроизводимости результатов, для косинусного шедулера было получено, что результат обучения может довольно сильно изменяться от одного старта к другому. Примеры этого представлены на рисунке 9.



a b

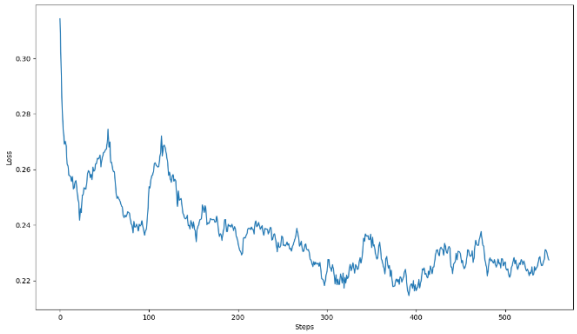
Рисунок 6 – График обучения модели на train (a) и val (b) датасетах, линейный шедулер, learning rate = 3E-5

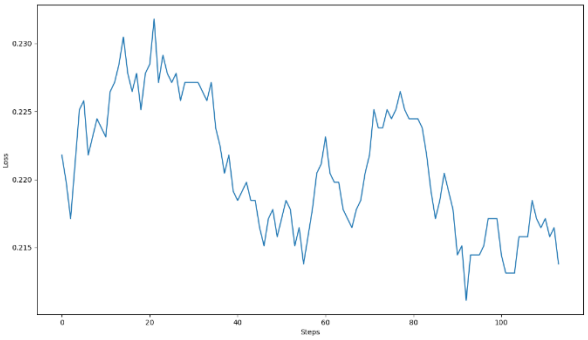




a b

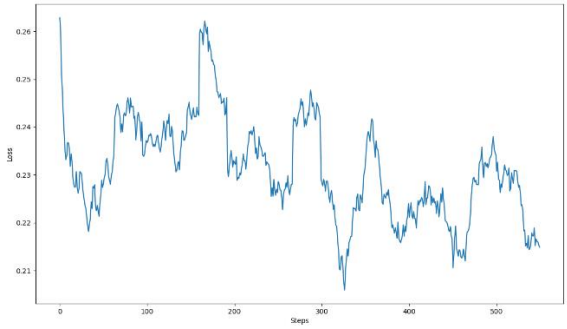
Рисунок 7 – График обучения модели на train (a) и val (b) датасетах, линейный шедулер, learning rate = 1E-4

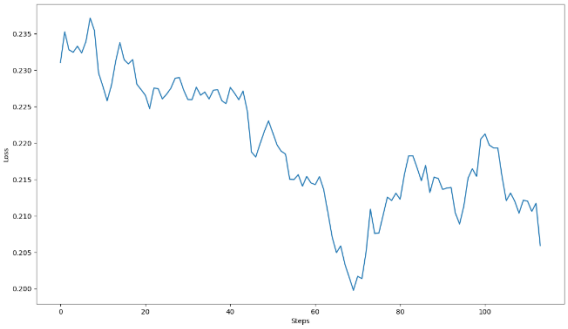




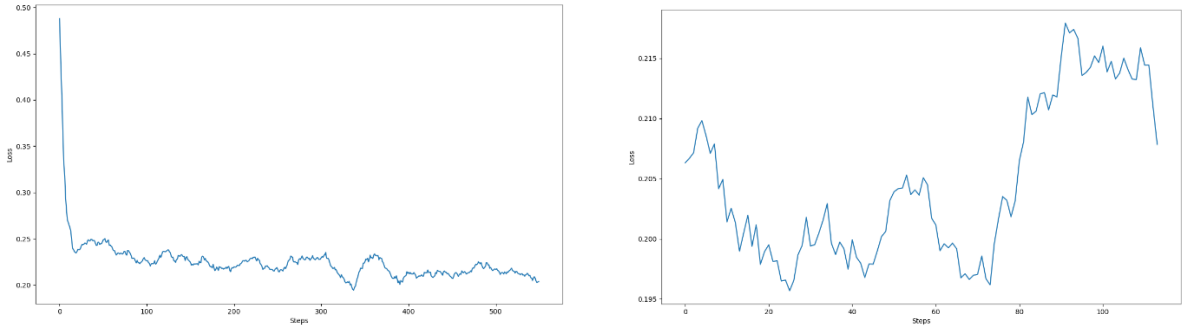
a b

Рисунок 8 – График обучения модели на train (a) и val (b) датасетах, линейный шедулер, learning rate = 5E-4





a b



a b

Рисунок 9 – Графики обучения модели на train (a) и val (b) датасетах, косинусный шедулер, learning rate = 1E-4

Инференс и валидация модели

Инференс модели реализован в виде вебсервиса с четырьмя методами:

1. ping – проверка работоспособности сервера
2. request – возвращает 1 наиболее релевантный ответ
3. retrieve – возвращает 5 наиболее релевантных ответов
4. dialog – разработан для демонстрации возможности бота вести диалог, принимает 3 вопроса и последовательно выдает на них ответы, с учетом контекста предыдущих пар вопрос-ответ

Кроме того, реализованы два возможных способа сбора корпуса реплик для выбора наиболее релевантных.

1. В качестве корпуса подаются все реплики персонажа (медленная реализация)
2. В качестве корпуса выбирается и подается установленное число реплик (по умолчанию равное 500), наиболее близких к вопросу. Близость к вопросу определяется с помощью KNN. (быстрая реализация)

Общая архитектура чат-бота представлена на рисунке 10 для медленной реализации и на рисунке 11 для быстрой.

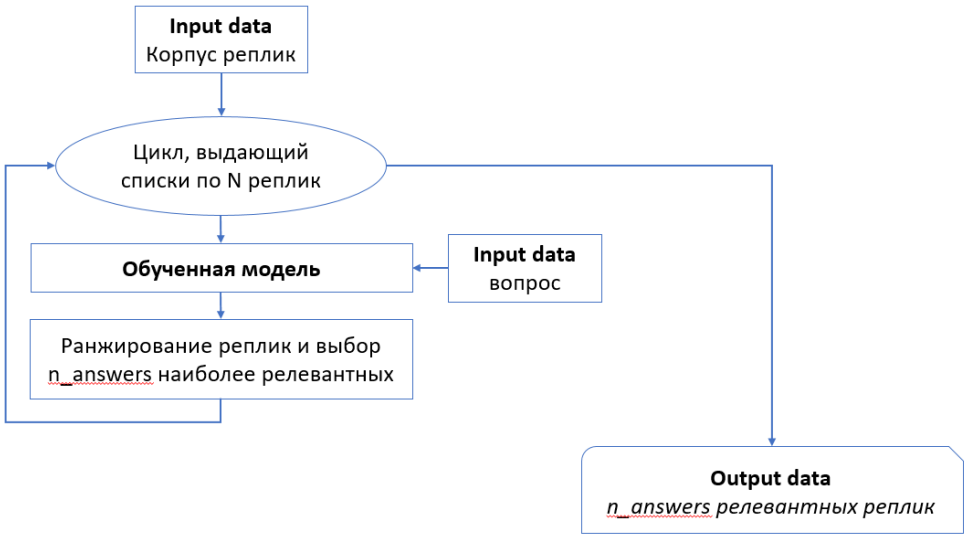


Рисунок 10 – Архитектура чат-бота для медленной реализации

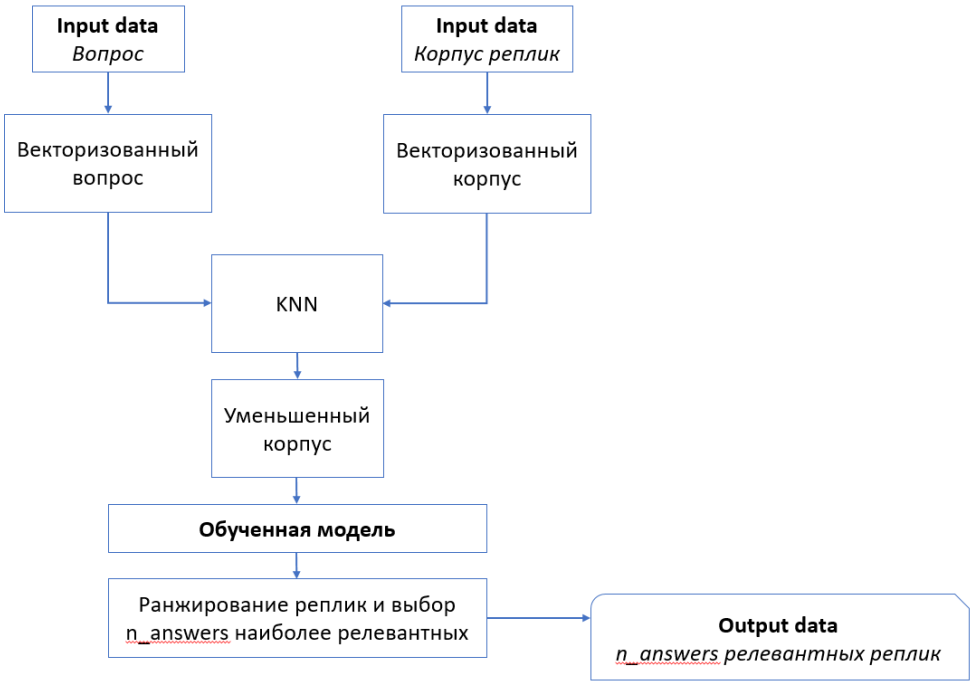


Рисунок 11 – Архитектура чат-бота для быстрой реализации

Для валидации результатов выбраны следующие тесты:

1. Простое приветствие (‘hey!’), медленная реализация
   Проверка ответа бота на простой вопрос и оценка времени работы медленного алгоритма. Результат представлен на рисунке 12. Результат выглядит как продолжение какого-то другого диалога.

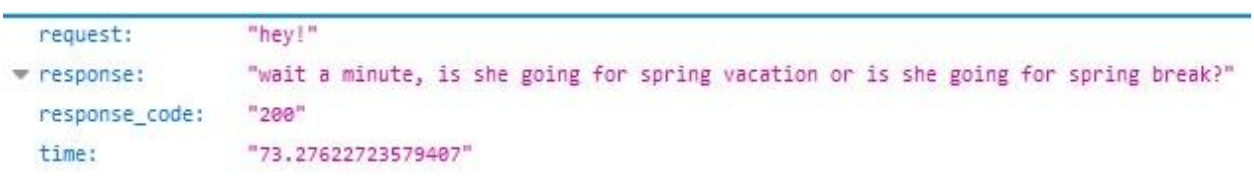


Рисунок 12 – Результат работы чат-бота на тестовой задаче № 1

2. Простое приветствие (‘hey!’), быстрая реализация
   Проверка ответа бота на простой вопрос и оценка времени работы быстрого алгоритма. Результат представлен на рисунке 13. Результат в чем-то соответствует сути персонажа, на приветствие следует ответ в виде шутки и нелепой ситуации. Видно, что скорость выполнения увеличилась в несколько раз.

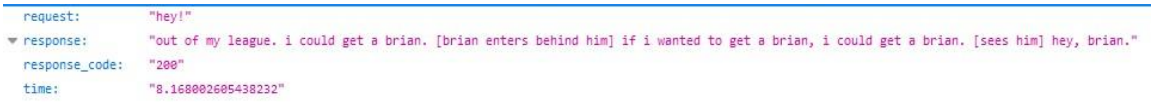


Рисунок 13 – Результат работы чат-бота на тестовой задаче № 2

3. Частый вопрос другого персонажа из диалогов в сериале (‘oh my god!’, типичное приветствие Дженис), медленная реализация
   Проверка ответа бота на встречающийся в исходном датасете вопрос. Результат представлен на рисунке 14. По смыслу похоже на возможный ответ на такое восклицание.

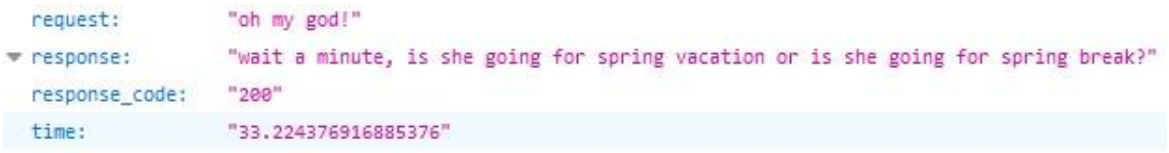


Рисунок 14 – Результат работы чат-бота на тестовой задаче № 3

4. Частый вопрос других персонажей из диалогов в сериале (‘what are you doing?’) , быстрая реализация
   Проверка ответа бота на встречающийся в исходном датасете вопрос. Результат представлен на рисунке 15. В целом большинство реплик подходят по смыслу в качестве ответов на вопрос.

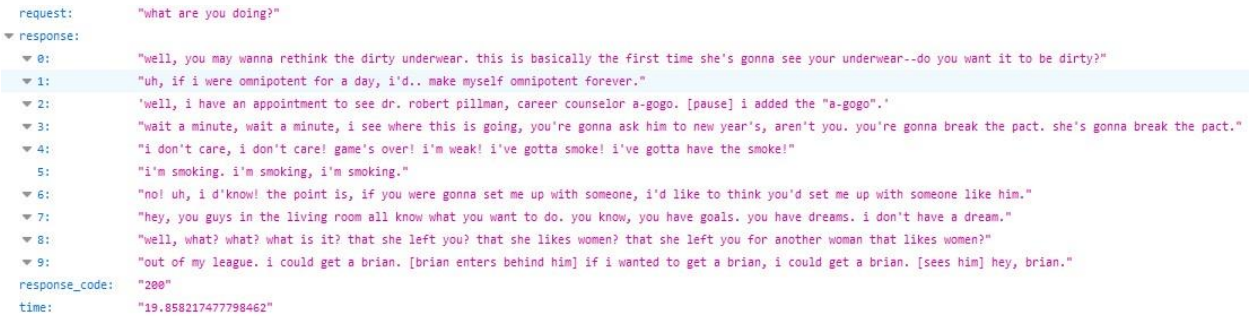


Рисунок 15 – Результат работы чат-бота на тестовой задаче № 4

5. Диалог из трех реплик, медленная реализация
   Проверка работы бота на имитации реального диалога. Результат представлен на рисунке 16. Диалог получился относительно связным.

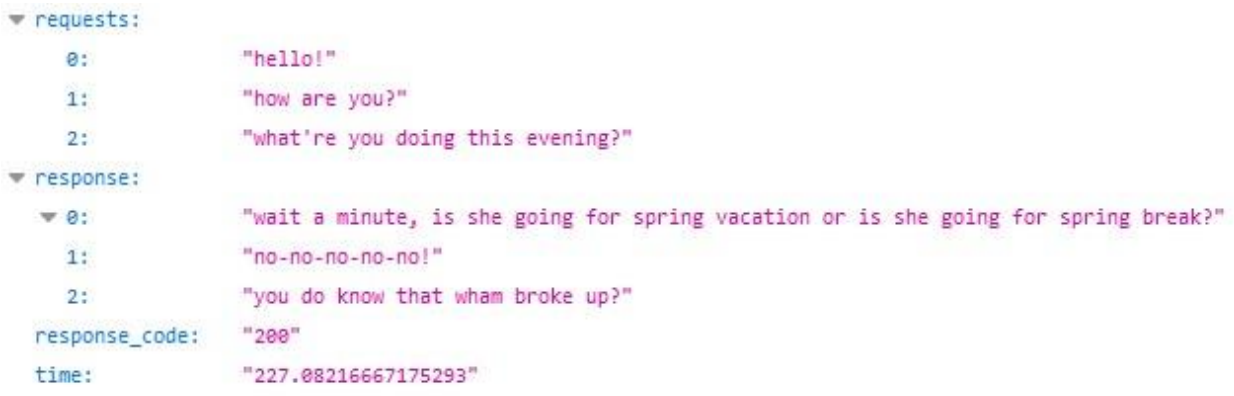


Рисунок 16 – Результат работы чат-бота на тестовой задаче № 5

Выводы

На основе retrieval подхода разработан чат-бот. Дообученная на датасете из реплик сериала «Друзья» модель distilbert оценивает релевантность реплик, далее ранжирующий механизм отбирает из них наиболее релевантные. Реализовано решение на базе метода к-ближайших соседей (KNN), позволяющее ускорить работу чат-бота за счет предварительного отбора реплик персонажа, наиболее близких к задаваемому вопросу. Чат-бот реализован в виде веб-интерфейса с четырьмя методами, позволяющими как задавать боту отдельные вопросы, так и направлять сразу несколько вопросов для поддержания диалога. Работоспособность бота продемонстрирована на тестовых примерах. Качество работы оценивалось исходя из адекватности ответов заданному вопросу.

Инструкция

Запуск чат-бота осуществляется посредством запуска файла inference.py на локальной машине или в google colab с помощью скрипта по ссылке ниже. При запуске inference.py на локальной машине в одной директории с ним должна присутствовать директория model, содержащая следующие файлы и папки:

1. charact_corpus.pkl - корпус реплик персонажа (клонируется с github)
2. tokenizer - дообученный токенайзер модели (клонируется с github)
3. friends_model.pkl - Дообученная модель (скачивается с яндекс.диск)

Гитхаб с директорией model:

https://github.com/YurezSml/MIPT_NLPGener_HW1/tree/main/model

Яндекс.диск с обученной моделью:

https://disk.yandex.ru/d/kt6NxfKMJADhdg

Colab с запуском бота:

https://colab.research.google.com/drive/15yGyk3K_r-KSFm2ZilEsja0z9fnq8rVl

После запуска автоматически открывается страница index.html с ссылками, по которым проверяются тестовые примеры.

Инференс модели реализован в виде вебсервиса с четырьмя методами:

1. ping
   не получает на вход параметры. возвращает "status": 'working!' в случае успешного подключения

2. request
   получает на вход 2 параметра через GET: query и type. Параметр type может принимать значения ‘fast’ или ‘slow’. В случае успешного срабатывания возвращает
   "response_code": "200"
   "request": query
   "response": ответ_модели
   "time": время выполнения запроса

3. retrieve – возвращает заданное количество релевантных ответов
   получает на вход 3 параметра через GET: query, type и num_answers. Параметр type может принимать значения ‘fast’ или ‘slow’. В случае успешного срабатывания возвращает
   "response_code": "200"
   "request": query
   "response": num_answers_релевантных_ответов_модели
   "time": время выполнения запроса

4. dialog – разработан для демонстрации возможности бота вести диалог, принимает 3 вопроса и последовательно выдает на них ответы, с учетом контекста предыдущих пар вопрос-ответ
   получает на вход 4 параметра через GET: query_1, query_2, query_3 и type. Параметр type может принимать значения ‘fast’ или ‘slow’. В случае успешного срабатывания возвращает
   "response_code": "200"
   "request": список_вопросов
   "response": список_ответов
   "time": время выполнения запроса