Восстановление полных реплик в диалоге

с помощью генеративных языковых моделей семейства ruGPT

Козиев Илья, 2022

Revision 18.06.2022

Где появляются неполные реплики?

- Неформальный диалог
- Читчат

Базовый пример - знакомство:

```
- Эй, тебя как зовут?
- Джульетта Мао  ⇒ Меня зовут Джульетта Мао
```

```
- Как тебя зовут сын мой.
- Леонардо Ди Каприо.  ⇒ Меня зовут Леонардо Ди Каприо
```

```
- Привет, а зовут тебя как, кстати?
- Стас, а тебя?  ⇒ Меня зовут Стас. Как тебя зовут?
```

Где появляются неполные реплики?

Достаточно длинные диалоги могут вестись в таком же стиле:

```
- Где жил?
- В Шанхае          ⇒ Я живу в Шанхае
- Давно?            ⇒ Ты давно живешь в Шанхае?
- Два года уже      ⇒ Я уже два года живу в Шанхае
- Как там погода?   ⇒ Какая погода была в Шанхае?
```

```
- Тебе когда-нибудь ломали что-нибудь?
- Да.      ⇒ однажды мне сломали кое-что
- Что?     ⇒ что тебе ломали?
- Жизнь.   ⇒ мне ломали жизнь
```

```
- Что в жизни главное?
- Я думаю, что иметь цель ⇒ главное в жизни – иметь цель
- Она есть у всех людей?  ⇒ цель в жизни есть у всех людей?
- Думаю, что нет          ⇒ цель в жизни есть не у всех людей
```

Где появляются неполные реплики?

Эллипсисы ☞:

```
- Как же тебя зовут, а?
- Меня – Стас, а тебя?  ⇒ Меня зовут Стас. Как тебя зовут?
```

Анафора ☞:

```
- Ты собак любишь?
- Не люблю я их  ⇒ я не люблю собак
```

Гэппинг ☞:

```
- Ты кошек любишь?
- Их – нет  ⇒  я не люблю кошек
```

Сложный гэппинг:

```
- В 25 лет вы получаете пенсию?
- Не я - отец.  ⇒  Я не получаю пенсию. Отец получает пенсию
```

Где появляются неполные реплики?

Иногда от реплики остается только наречие:

```
- Девушка, а Вы животных любите?
- Очень!  ⇒ Я очень люблю животных
```

Или даже предлог:

```
- Ты чай предпочитаешь с лимоном?
- Без  ⇒  Я предпочитаю чай без лимона
```

Где появляются неполные реплики?

В редких случаях и главное слово в словосочетании может опускаться:

```
- Мама, купи мне собаку.
- А ты будешь за ней ухаживать?
- А ты мне здоровую купи.  ⇒ купи мне здоровую собаку
```

Иногда требуется привлечение здравого смысла:

```
- Мне на голову упала коробка.
- А что в ней было?  ⇒ что было в коробке|голове?
```

```
- Доктор, мой ребенок не ест мясо. Чем его заменить?
- Собакой. Собака всегда ест мясо. ⇒ :)
```

Типичные паттерны: например, восстановление подлежащего (см. pro drop)

```
- Согласна?
- Да  ⇒ я согласна
```

Отрицания в диалоге:

```
- Я не прав?
- Нет. (Да.) ⇒ ты не прав
```

Где появляются неполные реплики?

Раскрытие не сводится к копированию слов из контекста:

```
- Как прошли выходные?
- В Простоквашино ездила...  ⇒  я на выходных ездила в Простоквашино
```

Все вышесказанное может быть в разных сочетаниях одновременно:

```
- Где твой кот?
- Жена к ветеринару повезла.  ⇒ жена повезла моего кота к ветеринару
- Заболел?  ⇒  твой кот заболел?
```

Сложные предложения:

```
- Я сварила суп, иди ешь.
- Из чего?  ⇒  из чего ты сварила суп?
```

Где появляются неполные реплики?

Импликатура - смысл “между строк”:

- Ты разве ещё не ел?
- Тебя ждал ⇒ я еще не ел. я ждал тебя.

- Экзамен сдала?
- Увы :(⇒ я не сдала экзамен

Голофразы:

- Деньжат одолжишь?
- Щас! ⇒ я не одолжу деньжат

Где появляются неполные реплики?

Отказ от использования коротких реплик возможен:

1) в стилистических целях – подчеркнуть свою мысль

2) если есть подозрения, что собеседник (чатбот, иностранец) не владеет полным набором навыков работы в таком коммуникационном регистре

3) чтобы устранить возможность неправильного толкования

```
- Ребенок не ест мясо. Его можно чем-то заменить?
- Конечно, собакой – она всегда ест мясо
```

Чем мешают неполные реплики?

Отсутствие части слов и “неправильный” синтаксис затрудняет работу многих алгоритмов NLP:

- Регулярные выражения
- Классификаторы интентов
- Детекторы оскорблений, токсичности
- Частеречная разметка
- Синтаксический анализ
- Semantic Role Labeling
- Выделение фактов

Постановка задачи и ограничения

- Работаем только с диалогом, читчатом
- Короткие реплики (до ~20 слов)
- Контекст – последние 2 или 3 реплики, при необходимости применяем модель **рекурсивно**
- Не обрабатываем случаи катафоры, только левый контекст
- Не требуем от модели помечать антецедент анафоры, референт для кореференции, источник заполнения эллипсиса
- Удаляем вводные слова и фразы, междомения
- По возможности интерпретируем импликатуры и голофразы
- Восстанавливаем местоименные подлежащие, даже если форма сказуемого содержит информацию о лице/числе
- Нормализуем порядок слов, вопросительные слова в начало

Близкие задачи в NLP

- Раскрытие анафоры
- Заполнение эллипсиса
- Кореференция

Генеративная модель как решение

- RuGPT – семейство больших генеративных языковых моделей с архитектурой GPT
- Ключевая особенность – претрейн на большом корпусе
- Ряд моделей свободно доступен на https://huggingface.co/sberbank-ai
- Адаптация модели на задачу – файнтюн на небольшом корпусе

Генеративная модель как решение

Плюсы:

- Простота инференса: никаких парсеров, тэггеров, словарей
- Простота расширения датасета: разметки как таковой нет
- Инструментарий для GPT: анализ, дистилляция и т.д.
- Можно использовать для генерации сырой разметки

Обучающий датасет

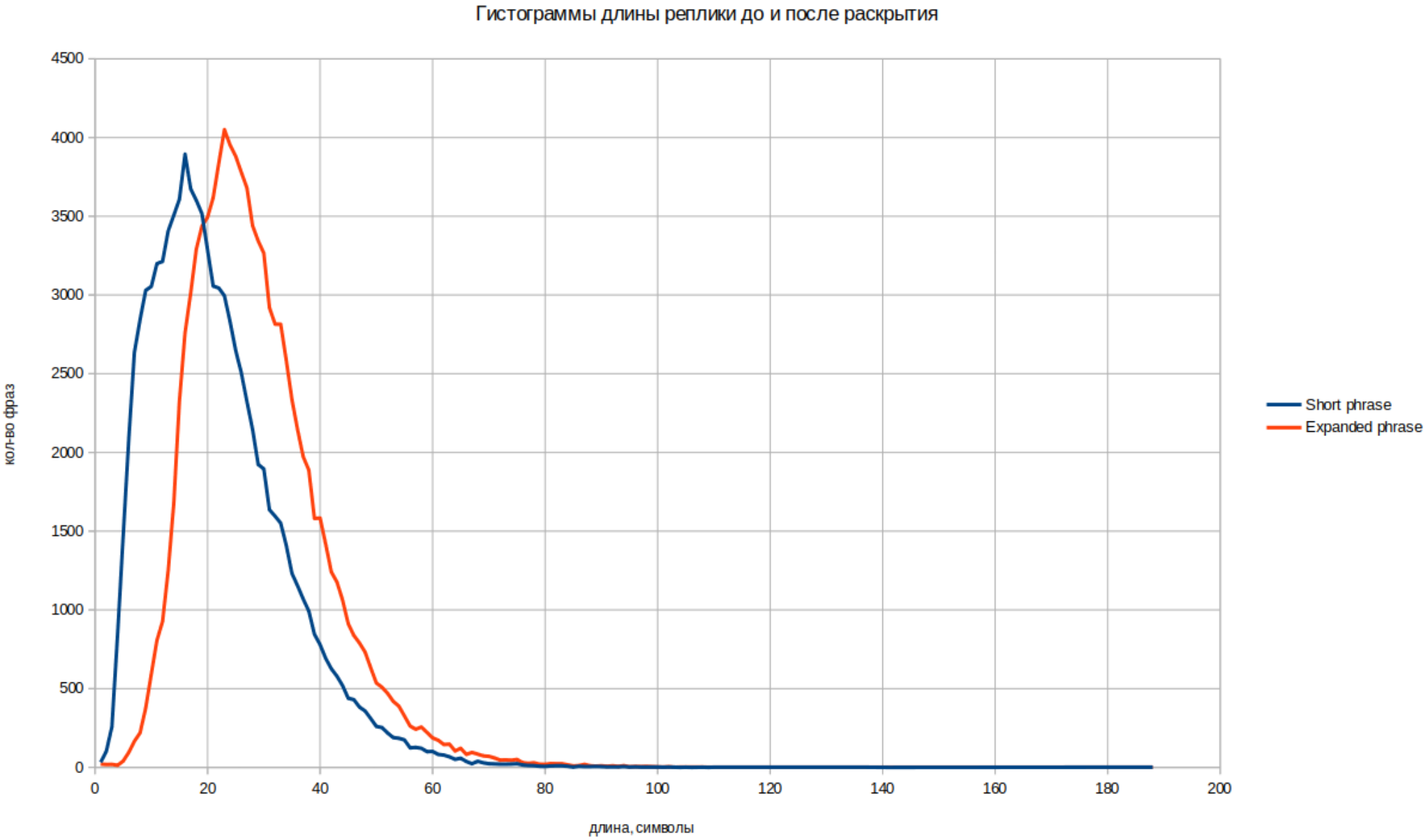
Ручная разметка, примерно 110,000 фрагментов:

```
Как вы догадались, что задержанный — вор?
По шапке.
На нем она горела? | шапка горела на задержанном?
```

Для файнтюна GPT конвертируются в сэмплы:

```
<s>- Как вы догадались, что задержанный — вор?
- По шапке.
- На нем она горела? # шапка горела на задержанном?</s>
```

Обучающий датасет



Синтетические данные

- Пока было мало ручных данных, использовались **синтетические**
- Генерация синтетики – dependency parser (UDPipe + СинТагРус) и морфологический словарь (ruword2tags)
- С помощью правил генерируются вопросы, короткие и полные ответы к утвердительным предлождениям:

```
T: Голодная кошка отчаянно преследует жирную мышку.

1) Кто ловит мышку? Голодная кошка ⇒ Голодная кошка ловит мышку
2) Кого преследует кошка? Жирную мышку ⇒ Кошка преследует жирную мышку
3) Какая кошка преследует мышку? Голодная ⇒ Голодная кошка преследует мышку
```

- Оценки качества синтетики: сильно **хуже** ручных данных
- Актуальная модель обучена **только** на ручных данных

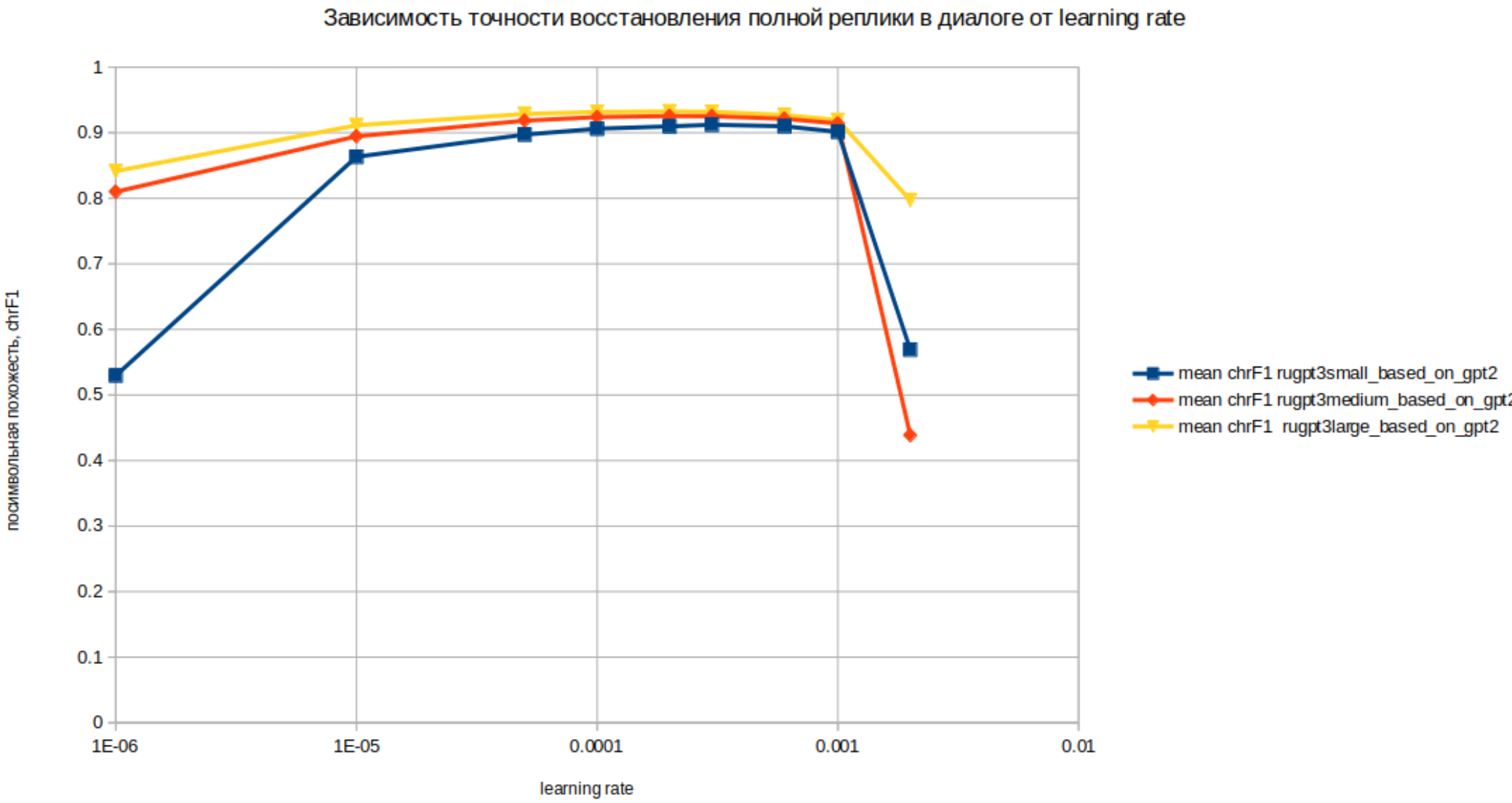
Оценка

- Файнтюн моделей small, medium и large семейства sberbank-ai/rugpt3*_based_on_gpt2
- Тренируются с помощью ru-gpts/pretrain_transformers.py
- **1 эпоха**
- grid search для подбора learning rate
- 3-fold кросс-валидация
- Метрики: а) перплексия, б) посимвольная похожесть на 3-символьных шинглах
- Коэффициент Жаккара и chrF1 дают оценку, насколько точно модель выдает эталонный полный текст реплики

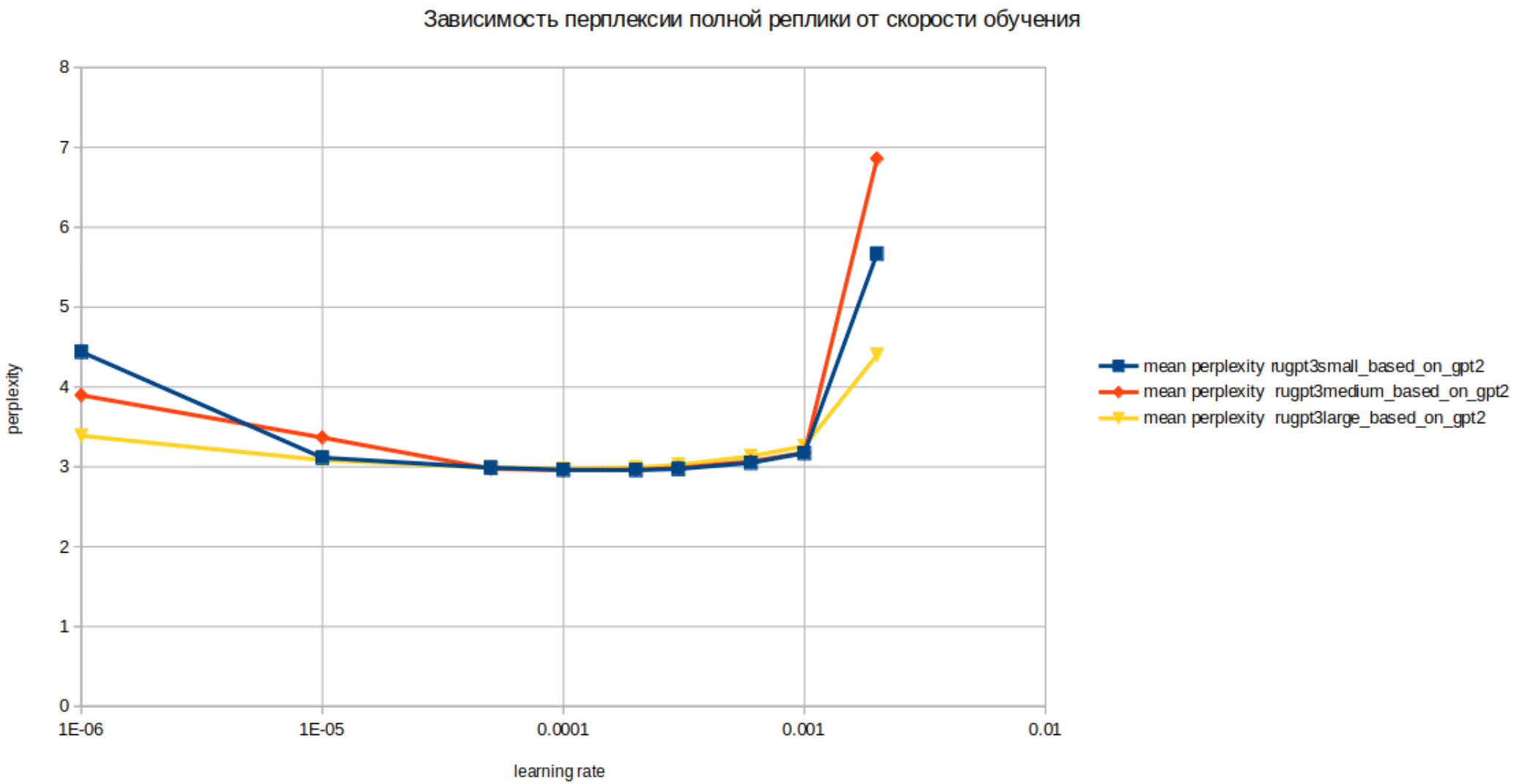
Оценка

learning rate	mean chrF1 rugpt3small_based_on_gpt2	mean chrF1 rugpt3medium_based_on_gpt2	mean chrF1 rugpt3large_based_on_gpt2
1E-06	0.529523628173215	0.809920604205701	0.841463599851881
1E-05	0.863327725266234	0.894392367286638	0.911348739820123
5E-05	0.897085498922204	0.918327352355936	0.928580141773602
0.0001	0.906100945040981	0.923846332641589	0.931729441105742
0.0002	0.909682085410882	**0.92543308713616**	**0.932408785365981**
0.0003	**0.912161969934597**	0.9251295765727	0.93195696519913
0.0006	0.909859537783758	0.921306048848982	0.927012377425078
0.001	0.901312727758346	0.914072612706664	0.919323844591289
0.002	0.568983553490307	0.438466434394724	0.797015715084427

Оценка


Зависимость точности восстановления полной реплики в диалоге от learning rate
посимвольная похожесть, chrF1
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1E-06
1E-05
0.0001
0.001
0.01
learning rate
mean chrF1 rugpt3small_based_on_gpt2
mean chrF1 rugpt3medium_based_on_gpt2
mean chrF1 rugpt3large_based_on_gpt2

Оценка


Зависимость перплексии полной реплики от скорости обучения
perplexity
learning rate
0
1
2
3
4
5
6
7
8
1E-06
1E-05
0.0001
0.001
0.01
mean perplexity rugpt3small_based_on_gpt2
mean perplexity rugpt3medium_based_on_gpt2
mean perplexity rugpt3large_based_on_gpt2

Модель на huggingface

Модель доступна для свободного использования

На базе sberbank-ai/rugpt3large_based_on_gpt2

Карточка: https://huggingface.co/inkoziev/rugpt_interpreter

Пример использования с transformers:

```
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM


device = "cuda" if torch.cuda.is_available() else "cpu"
model_name = "inkoziev/rugpt_interpreter"
tokenizer = AutoTokenizer.from_pretrained(model_name)
tokenizer.add_special_tokens({'bos_token': '<s>', 'eos_token': '</s>', 'pad_token': '<pad>'})
model = AutoModelForCausalLM.from_pretrained(model_name)
model.to(device)
model.eval()

# На вход модели подаем последние 2-3 реплики диалога. Каждая реплика на отдельной строке, начинается с символа "-"
# В конце добавляем символ "#"
input_text = """<s>- Как тебя зовут?
- Джульетта Мао #"""

encoded_prompt = tokenizer.encode(input_text, add_special_tokens=False, return_tensors="pt").to(device)

output_sequences = model.generate(input_ids=encoded_prompt, max_length=100, num_return_sequences=1,
pad_token_id=tokenizer.pad_token_id)

text = tokenizer.decode(output_sequences[0].tolist(), clean_up_tokenization_spaces=True)[len(input_text)+1:]
text = text[: text.find('</s>')]
print(text)
```

Ссылки

1) Разрешение анафоры (Dialogue evaluation 2014) https://www.dialog-21.ru/evaluation/2014/anaphora/
2) Automatic Gapping Resolution for Russian (Dialogue shared task) https://github.com/dialogue-evaluation/AGRR-2019
3) Anaphora and Coreference Resolution for Russian (Dialogue shared task) http://www.dialog-21.ru/en/evaluation/2019/disambiguation/anaphora/
4) SARG: A Novel Semi Autoregressive Generator for Multi-turn Incomplete Utterance Restoration https://arxiv.org/pdf/2008.01474v3.pdf
5) CHRF: character n-gram F-score for automatic MT evaluation https://aclanthology.org/W15-3049.pdf