ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет Санкт-Петербургская школа физико-математических и компьютерных наук

Вихорев Дмитрий Александрович

Развитие методов семплирования текста для расширения возможностей направленной генерации

Выпускная квалификационная работа
по направлению подготовки 01.04.02 Прикладная математика и информатика
образовательная программа «Машинное обучение и анализ данных»

Рецензент: Кротова Ирина Витальевна

Научный руководитель: Мухин Михаил Сергеевич

Консультант: Ямщиков Иван Павлович

Санкт-Петербург 2023

Содержание

Abstract

Texts with given attributes can be generated using a language model trained on texts with the corresponding attributes. Modern language models are very large, so it can be very difficult from a computational point of view to train them for each separate task of directed generation. The paper explores an approach that allows to get rid of this drawback, using the example of generating funny texts. To generate funny texts, the following approach is used: first, for a user-specified context, several continuations are generated using a large language model, and then the ranking model selects the best of them. To do this, a large dataset of jokes was collected and preprocessed from various sources; on it, a model is trained that selects the most appropriate continuation for the context specified by the user.

Генерация текстов с заданными атрибутами может быть произведена с помощью языковой модели, обученной на имеющих соответствующие атрибуты текстах. Современные языковые модели очень большие, поэтому обучать их под каждую отдельную задачу направленной генерации бывает очень тяжело с вычислительной точки зрения. В работе исследован подход, позволяющий избавиться от этого недостатка, на примере генерации смешных текстов. Для генерации смешных текстов используется следующий подход: сначала для заданного пользователем контекста с помощью большой языковой модели генерируется несколько продолжений, а затем ранжирующая модель выбирает самое лучшее из них. Для этого из различных источников собран и предобработан большой датасет шуток; на нём обучена модель, выбирающая наиболее подходящее продолжение для заданного пользователем контекста.

Ключевые слова: языковая модель, декодирование, генерация, юмор.

Введение

Объектом исследований данной работы является направленная генерация текстов. В качестве предмета исследований работы выступает улучшение направленной генерации текстов языковой моделью путём изменения способа декодировки из неё. В работе сформулирована гипотеза о том, что дообучение ранжирующей модели на датасете текстов с заданными атрибутами, позволит генерировать языковой моделью тексты с этими атрибутами. Данная гипотеза экспериментально подверждена путём подсчёта соответствующих метрик в случае, когда в качестве текстов, имеющих определённую направленность, выступает датасет шуток. Актуальность и значимость работы можно сформулировать в виде двух тезисов:

- Современные языковые модели большие, переобучать их для каждой задачи направленной генерации слишком затратно.
- Языковые модели при генерации текстов всё ещё сильно опираются на локальный контекст. От этого можно частично избавиться методом, рассмотренным в работе.

Цель работы: Обучить нейросеть по заданной пользователем текстовой строке генерировать смешное продолжение, не переобучая под это языковую модель. Для достижения поставленной цели нужно решить следующие задачи:

- Сбор и предобработка подходящего датасета шуток
- Обучение различных ранжирующих моделей на собранном датасете
- Выбор лучшей модели после проведения соответствующих экспериментов

В работе получен следующий результат: при различных типах отрицательных примеров произведено обучение модели, ранжирующей панчлайны шутки по уровню юмора и помогающей генерировать более смешные тексты. Структура работы выглядит так:

1. В главе «Обзор способов декодировки из языковой модели» рассмотрены основные методы декодировки из языковой модели, а также приведена мотивация использования архитектуры RankGen авторами статьи для улучшения генерации текстов.

2. В главе «Направленная генерация текстов» рассмотрены различные пути для решения задачи направленной генерации текстов, а также приведены подробные рассуждения насчёт метода, применяемого в данной работе, и его плюсов.

3. Глава «Датасет» содержит описание сбора и предобработки большого датасета шуток, нужного для решения поставленной цели.

4. В последней главе с названием «Эксперименты» указаны подробности проводимых экспериментов по обучению ранжирующей юмор модели.

1 Обзор способов декодировки из языковой модели

1.1 Генерация текстов языковой моделью

Языковая модель по началу текста предсказывает распределение вероятностей следующего слова в нём. Разыгрывая соответствующую дискретную случайную величину с конечным числом принимаемых значений (количество значений равно размеру заранее заданного словаря токенов), можно сгенерировать следующее слово текста. Продолжая данный процесс, можно генерировать тексты любой длины.

1.2 Простые способы декодировки из языковой модели и их недостатки

Существует множество способов, как из предсказанного языковой моделью распределения получить слово. Самые очевидные из них:

1. Выбор слова с наибольшей вероятностью (greedy decoding),
2. Генерация слова в соответствии с конечнозначным дискретным распределением, которое предсказано языковой моделью (ancestral sampling).

Первый подход плох, потому что обычно слова в текстах распределены по степенному закону (небольшой процент уникальных слов встречается очень часто): сгенерированные таким способом тексты будут содержать в себе очень мало уникальных слов, т.е. будут скучными, однотипными и повторяющимися. Второй подход плох тем, что современные языковые модели сильно опираются на локальный контекст, и имеется шанс сгенерировать слово с маленькой вероятностью на каком-либо шаге ге-

нерации: после генерации такого слова возможен, например, резкий уход от изначальной тематики текста, о чём сказано в [1].

1.3 Продвинутые способы декодировки из языковой модели

Существуют более продвинутые методы декодировки из языковой модели:

- nucleus sampling [1],
- top-k [2],
- contrastive search [3],
- contrastive decoding [4],
- typical sampling [5].

Эти методы призваны решить некоторые из недостатков простых подходов из предыдущего параграфа:

- Подходы top-k и nucleus sampling решают проблему метода ancestral sampling, описанную в прошлом параграфе, путём запрета генерации слов с маленькой вероятностью. Метод top-k запрещает генерировать все слова, кроме k самых вероятных. Подход nucleus sampling запрещает генерировать все слова кроме самых вероятных, вероятность которых в сумме не должна превышать заранее заданного порога p.
- Метод contrastive search при генерации слова пытается выбрать слово с наибольшей вероятностью, максимально не похожее по смыслу на все слова префикса. Похожесть смысла двух слов определяется

в данном случае через скалярное произведение векторных представлений этих слов.

- Метод contrastive decoding позволяет генерировать правдоподобные с точки зрения модели-эксперта и неправдоподобные с точки зрения модели-любителя тексты. Таким образом авторы подхода пытаются избавиться от таких нежелательных вещей, как повторение текста, смещение темы и различные противоречия, которые чаще встречаются при генерации текстов с помощью маленьких языковых моделей.

- Метод typical sampling позволяет найти компромисс между правдоподобием текста и его информационным содержанием. На каждом шаге генерации из всех возможных слов выбирается такое, которое содержит среднее количество информации.

1.4 Улучшение декодировки с помощью RankGen

Однако, как уже было сказано, современные языковые модели при генерации слова слишком опираются на локальный контекст, из-за чего генерируемые тексты хоть и получаются правдоподобными с точки зрения модели, являются не очень связными. Для решения этой проблемы в [6] был предложен RankGen[1]- большая ранжирующая модель-кодировщик для текстов, помогающая генерировать более похожие на написанные человеком тексты. RankGen обучают по двум строкам определять является ли одна строка непосредственным продолжением другой строки в тексте.

Генерация текстов с использованием RankGen выглядит следующим образом:

[1]Используемая в данной работе модель находится по ссылке: https://huggingface.co/kalpeshk2011/rankgen-t5-base-all

1. Для текстового префикса генерируется N суффиксов с помощью большой языковой модели и простого метода декодировки (например, nucleus sampling),

2. С помощью RankGen по заданному префиксу выбирается лучший из N сгенерированных суффиксов.

Также при генерации текстов можно использовать beam search, что дополнительно улучшает результат.

Подход с использованием RankGen помогает генерировать более похожие на написанные человеком тексты, потому что RankGen при ранжировании сгенерированных суффиксов полагается на префикс и суффиксы целиком, а не только на локальный контекст, как это делает языковая модель.

В данной главе рассмотрены как простые, так и продвинутые способы декодирования из языковой модели, включая метод с использованием RankGen.

2 Направленная генерация текстов

2.1 Генерация текстов в заданном стиле

Иногда хочется генерировать тексты с заданным стилем (например, тексты в стиле определённого автора) или атрибутами (тексты, содержащие определённые слова, или тексты с заданным эмоциональным состоянием). Для этого можно обучить языковую модель на текстах, содержащих данный стиль или атрибут. Однако такой подход имеет несколько недостатков:

- Современные языковые модели очень большие, для их обучения требуется много вычислительных ресурсов. Обучать отдельную языковую модель под каждый стиль или атрибут слишком затратно.
- Современные языковые модели при генерации текстов всё ещё сильно опираются на локальный контекст.

Также примерно в одно время с [6] вышла статья про направленную генерацию текстов [7], в которой, используя динамику Ланжевена, генерируются тексты с заданным стилем или атрибутами:

1. Генерируется текст из случайного набора токенов
2. Этот текст итеративно улучшается в соответствии с динамикой Ланжевена: происходит попытка повысить правдоподобность текста с точки зрения языковой модели и показатель некоторого классификатора атрибута, делая шаг в направлении соответствующего градиента, впрыскивая на каждом шаге гауссовский шум для добавления стохастичности к процедуре.

Такой подход к генерации позволяет не дообучать языковую модель для направленной генерации, он очень трудоёмкий: требуется многократно вычислять градиенты для получения хороших результатов.

2.2 Использование RankGen для направленной генерации текстов

Для устранения недостатков, описанных в предыдущем параграфе, можно применить RankGen, различающий, следует ли одна строка за другой в тексте. Если дообучить его не на нейтральных текстах, а на текстах в заданном стиле, то он будет выбирать не только более «человечные» тексты, но и более подходящие под стиль, на котором его дообучили.

Конечно, дообучение языковой модели под заданный стиль в связке с дообучением RankGen на соответствующих текстах, поможет дополнительно улучшить направленную генерацию текстов. Однако в данной работе ограничимся лишь дообучением ранжирующей модели, а в качестве языковой модели будем использовать стандартные обученные на большом количестве текстов языковые модели.

2.3 Направленная генерация текстов с использованием RankGen на примере генеративного юмора

Для того, чтобы продемонстрировать направленную генерацию текстов с использованием RankGen, возьмём в качестве текстов в некотором стиле датасет шуток и дообучим модель оценивать то, насколько панчлайн соответствует сетапу шутки.

Таким образом:

1. При генерации языковой моделью текстов по заданному сетапу будет генерироваться несколько панчлайнов,
2. Ранжирующая модель будет оценивать каждый из них и выбирать

лучший.

В статье [8] описано то, как обычно выглядит шутка, состоящая из сетапа и панчлайна:

1. Сетап должен быть максимально неоднозначным, содержащим несколько различных интерпретаций, то есть иметь высокое значение энтропии распределения панчлайнов при условии этого сетапа.

2. Панчлайн должен быть максимально неожиданным, нарушающим все ожидания читателя, то есть иметь маленькую вероятность при условии заданного сетапа.

Второй пункт свидетельствует о том, что подходы к декодировке, опирающиеся на выбор самых правдоподобных с точки зрения модели (а это все рассмотренные подходы кроме [5, 6]) панчлайнов по заданному сетапу, непригодны для генерации юмора из обычной языковой модели. Это ещё один аргумент в пользу актуальности предложенного решения задачи.

В данной главе рассмотрен подход к адаптации модели RankGen под направленную генерацию (на примере генерации юмора) языковой моделью, а также плюсы этого подхода.

3 Датасет

3.1 Поиск подходящих данных

В результате продолжительного поиска было найдено несколько статей [9, 10, 11, 12], где используется несколько различных датасетов шуток:

- В [9] решается задача выявления оскорбительных шуток. Для этого используется 92153 различных шуток с сайта reddit, каждая из которых относится к одной из трёх категорий: clean, dark, dirty. Плюс этого датасета в том, что каждая шутка уже разбита на сетап и панчлайн, так как посты на этом сайте имеют тему и основной текст, где в качестве сетапа шутки выступает тема поста, а в качестве панчлайна шутки- основной текст поста. Ещё один плюс в том, что каждая шутка имеет рейтинг юмора, полученный путём подсчёта голосов пользователей. Из минусов можно выделить то, что подавляющее большинство примеров в данном датасете очень токсичные.

- В статье [10], где исследуется влияние роли пола и возраста на восприятие и оценку юмора, содержится датасет из 231657 различных однострочных шуток. Из плюсов этого датасета можно выделить то, что он содержит очень много примеров. Из минусов- он не содержит разбиения на сетап и панчлайн, нужного для тренировки ранжирующей модели, а также в нём нет рейтинга шуток.

- В статье [11] решается задача выявления юмора в текстах. Для обучения моделей был использован датасет шуток из разных источников, каждый из которых покрывает определённый тип юмора. Датасеты включают шутки из нескольких предложений (датасет коротких шуток), шутки из одного предложения (датасет из каламбуров, игры слов) и более смешанные шутки (датасет с сайта reddit, в нём

16 тысяч разных шуток).

- В статье [12] решается задача генерации текстов различного стиля одной языковой моделью на основе LSTM. Так как в качестве одного из стиля используются шутки, для обучения нейросети требуется большой набор шуток. Помимо 231657 шуток, использующихся также в [10], в статье также есть набор из 97 тысяч других шуток.

3.2 Обработка данных

Для того, чтобы получить один большой датасет шуток, все найденные датасеты были собраны в один. После этого полученный набор данных прошёл несколько этапов предобработки:

1. Чистка от дубликатов. Так как шутки в исходных данных довольно часто повторяются, нужно оставить лишь уникальные из них. После данного этапа размер датасета составил примерно 350 тысяч примеров.

2. Удаление примеров, содержащих символы, отличные от латинских букв и знаков препинания, и слишком длинных и коротких примеров. Хотелось бы, чтобы ранжирующая модель выбирала шутки, основанные на игре слов, а всякие символьные шутки в датасете могут затруднить обучение такой нейросети, ведь это может быть слишком сложным для неё. Среди слишком больших и слишком маленьких по длине примеров содержится очень много плохих данных, по этой причине они и были удалены из датасета. После этого этапа размер датасета уменьшился до 250 тысяч примеров.

3. Деление каждого примера на сетап и панчлайн. Так как только примерно треть из оставшихся на предыдущем этапе 250 тысяч шуток имеют деление на сетап и панчлайн (они с сайта reddit, значит для

них есть ещё и рейтинг юмора, выставленный пользователями), основная часть датасета нуждается в таком делении. Деление шутки на сетап и панчлайн хорошо было бы производить с помощью разметки людей. Однако большой размер датасета не позволяет этого сделать, поэтому разметка была произведена исходя из некоторых правил, описанных в следующем параграфе главы. После данного этапа получен датасет размера 195 тысяч шуток, каждая из которых имеет деление на сетап и панчлайн.

4. Удаление слишком токсичных примеров из датасета. Так как юмор неразрывно связан с токсичностью и агрессией, а мы не хотели бы, чтобы генерируемые тексты получались такими, было решено убрать самые токсичные примеры с помощью соответствующего классификатора[1]. После этого этапа получили датасет из 90 тысяч примеров.

5. Удаление наименее смешных примеров датасета. К концу предыдущего этапа был получен нетоксичный размеченный (делённый на сетап и панчлайн) датасет, который не содержит символов, отличных от латинских букв и знаков препинания. Выглядит хорошо, однако нам нужно оставить самые смешные примеры. Хорошо бы это сделать с помощью разметки людей, но примеров всё ещё много. По этой причине применим классификатор юмора[2], чтобы отсеять несмешные примеры с точки зрения этого классификатора. После этого этапа осталось 65 тысяч шуток.

[1]Классификатор токсичных сообщений можно найти по ссылке: https://huggingface.co/s-nlp/roberta_toxicity_classifier

[2]Классификатор юмора расположен по ссылке: https://huggingface.co/thanawan/bert-base-uncased-finetuned-humordetection

3.3 Процедура деления неразмеченной шутки на сетап и панчлайн

Алгоритм разметки неразмеченной части датасета размера 250 тысяч шуток следующий:

1. Если шутка не содержит знаков препинания, то убираем её из датасета. Мотивация этого шага алгоритма заключается в том, что для тренировки нашей модели нам нужны шутки с делением на сетап и панчлайн, а если знаков препинания в примере нет, то нужное нам деление найти сложно либо его совсем нет. Этот шаг алгоритма отбрасывает довольно много примеров из нашего набора данных: после него размер датасета- 207 тысяч шуток.

2. Иначе если шутка содержит лишь один знак "?"не в конце примера, то делим шутку на сетап и панчлайн по этому знаку. Мотивация: довольно распространённый тип шуток имеет вид <сетап>?<панчлайн>, где в качестве сетапа используется вопрос, ответ на который может быть максимально неопределённым, а в качестве панчлайна- максимально неожиданный ответ [8]. Этот шаг алгоритма позволяет разметить примерно 70 тысяч неразмеченных примеров.

3. Иначе если шутка содержит лишь один знак "!"не в конце примера, то делим шутку на сетап и панчлайн по этому знаку. Мотивация аналогична предыдущему пункту, но после просмотра датасета глазами было решено выбрать именно такой приоритет знаков "?"и "!". Этим шагом разметили ещё 4 тысячи примеров.

4. Иначе если шутка содержит лишь один знак "."не в конце примера, то делим шутку на сетап и панчлайн по нему. Мотивация аналогична предыдущим двум пунктам. Данный шаг позволил разметить примерно 26 тысяч примеров.

5. Иначе если пример содержит лишь одну последовательность знаков из множества '?', '!', '.', то делим по нему. Этот шаг разметил 16 тысяч примеров.

6. Иначе делим по знакам ';', ':', ',' аналогично предыдущим пунктам, что позволяет разметить ещё 20 тысяч примеров.

После применения данного алгоритма разметки в датасете осталось примерно 12 тысяч неразмеченных примеров. Так как эти примеры остались неразмеченными, каждый из них содержит несколько одинаковых знаков препинания. Это значит, что делить такие шутки на сетап и панчлайн проблематично. Кроме того, такого деления может и вовсе не быть. Например, может быть ситуация где нет чёткого деления на сетап и панчлайн (не подходит для обучения ранжирующей модели), а может быть такое, что панчлайнов вообще несколько (шутка многоуровневая, что для обучения нашей модели также не очень подходит). Из-за малого количества подобного типа шуток, можем их отбросить из дальнейшего рассмотрения. После чего и получается датасет из 195 тысяч примеров.

3.4 Описание итогового датасета

После предобработки датасета получили набор из 65 тысяч шуток, состоящих лишь из латинских букв и знаков препинания. При этом каждая шутка данного датасета нетоксичная и смешная с точки зрения соответствующих классификаторов, а также имеет разбиение на сетап и панчлайн. Кроме того, часть из шуток имеет рейтинг, так как взята с сайта reddit.

4 Эксперименты

4.1 Общие сведения об устройстве и обучении модели

- RankGen переводит префикс и суффикс текста в общее векторное пространство и смотрит на то, какое скалярное произведение получается в итоге. Если скалярное произведение маленькое, то это значит, что переданный в модель претендент на суффикс не является истинным суффиксом заданного префикса. Если же это скалярное произведение большое, то претендент на суффикс является истинным суффиксом для префикса текста.

- Для проведения экспериментов была взята самая маленькая модель RankGen размера 110M параметров. Данный выбор объясняется тем, что вычислительных мощностей на момент обучения хватило только на неё.

- Модель взята с model hub платформы huggingface, для её обучения использовался API этой платформы, позволяющий значительно упростить написание кода, нужного для тренировки нейросетей, обрабатывающих тексты.

- Полученный в предыдущей главе датасет был переразмечен следующим образом: шутки, имеющие рейтинг, были разбиты на 5 квантилей с номерами от 1 до 5. Если шутка входит в квантиль с номером 5, то она считается наиболее смешной, если в квантиль с номером 1- наименее смешной. Оставшаяся часть шуток осталась без рейтинга, поэтому назначим каждой из них среднюю оценку 3 (по сути при тренировке будем использовать квантили только шуток с известным рейтингом, для остальных шуток возьмём усреднённое значение).

- Далее этот датасет был поделён на тренировочную, валидационную и тестовую части. На тренировочной части происходило дообучение модели, на валидационной части происходил подсчёт функции потерь во время обучения, чтобы решить, когда остановить процесс тренировки, на тестовой части была произведена оценка итоговых результатов.

4.2 Отрицательные примеры при обучении и мотивация их выбора

Аналогично оригинальной статье про RankGen было произведено обучение модели следующими 4 способами в зависимости от использования типа отрицательных примеров:

1. Книжные и генеративные отрицательные примеры
2. Книжные отрицательные примеры
3. Генеративные отрицательные примеры
4. Без отрицательных примеров

Книжные отрицательные примеры позволяют уменьшить значение скалярного произведения между векторными представлениями сетапа шутки и панчлайна другой шутки тренировочного датасета. Такой тип отрицательных примеров должен помочь в выборе более связанных с сетапом панчлайнов.

Генеративные отрицательные примеры позволяют уменьшить значение скалярного произведения между сетапом шутки из датасета и сгенерированным языковой моделью панчлайном. Такой тип отрицательных примеров должен помочь в выборе более «человечных» текстов.

4.3 Параметры обучения

Для каждого типа отрицательных примеров были выбраны свои параметры обучения:

Тип отриц. примеров	Макс. к-во эпох	Фактич. к-во эпох	LR
Книж. + генератив.	10	2.67	2e-5
Книжные	10	2	2e-5
Генеративные	10	1	2e-6
-	1	1	2e-7

Значение функции потерь на валидационном датасете для всех типов обучения начинает довольно быстро уменьшаться. Это приводит к тому, что модели с лучшей обобщающей способностью мы получаем уже после первых эпох, что отображено в таблице.

Отдельное внимание стоит обратить на выбор более низких значений learning rate при обучении модели без использования книжных отрицательных примеров. Такой выбор обусловлен тем, что при этом типе тренировки векторные представления сетапа и панчлайна каждой шутки могут удлиняться до бесконечности, тем самым увеличивая скалярное произведение между соответствующим сетапом и панчлайном. Это связано с особенностями оптимизируемых в данном случае функций потерь.

Ниже приведены графики, демонстрирующие процесс обучения.

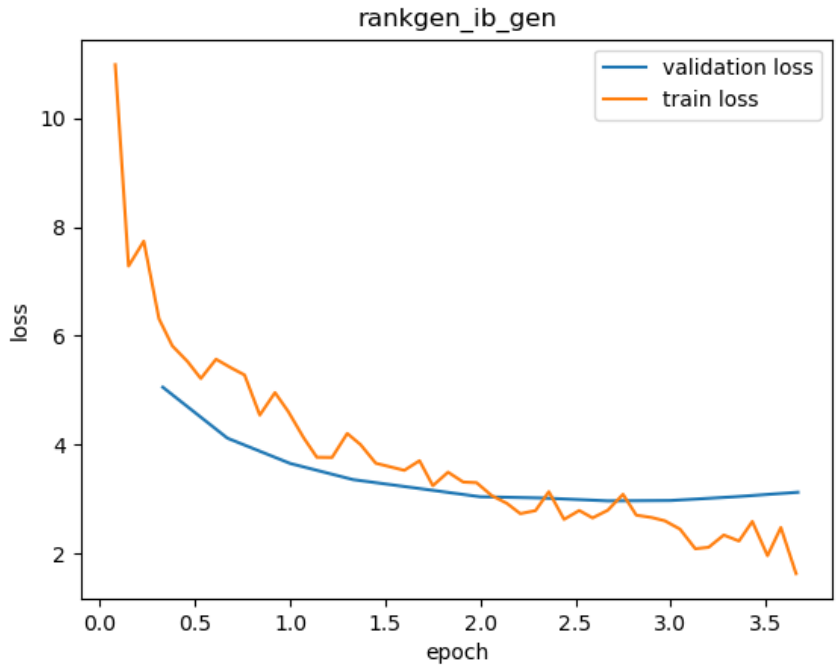

Рис. 1: Обучение модели с использованием двух типов отрицательных примеров

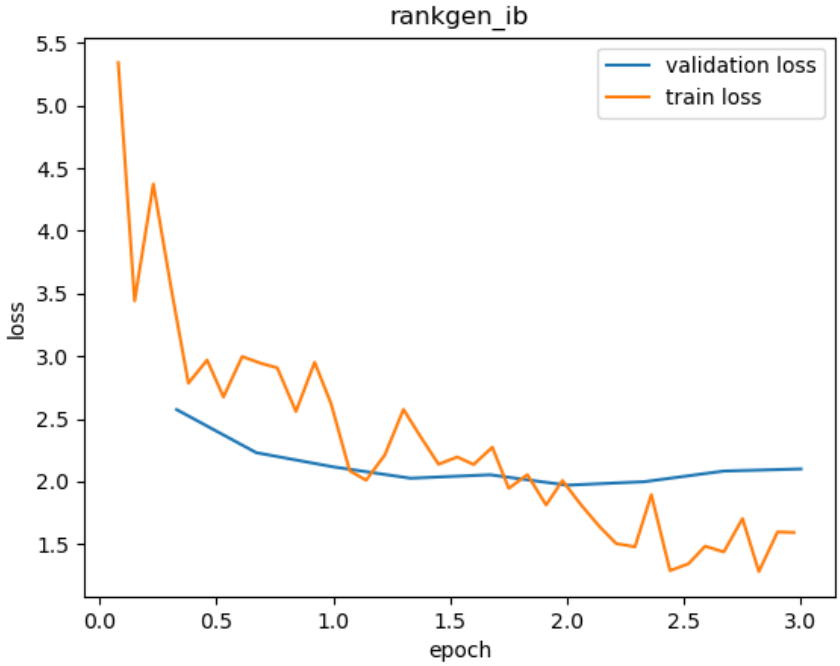

Рис. 2: Обучение модели с использованием книжных отрицательных примеров

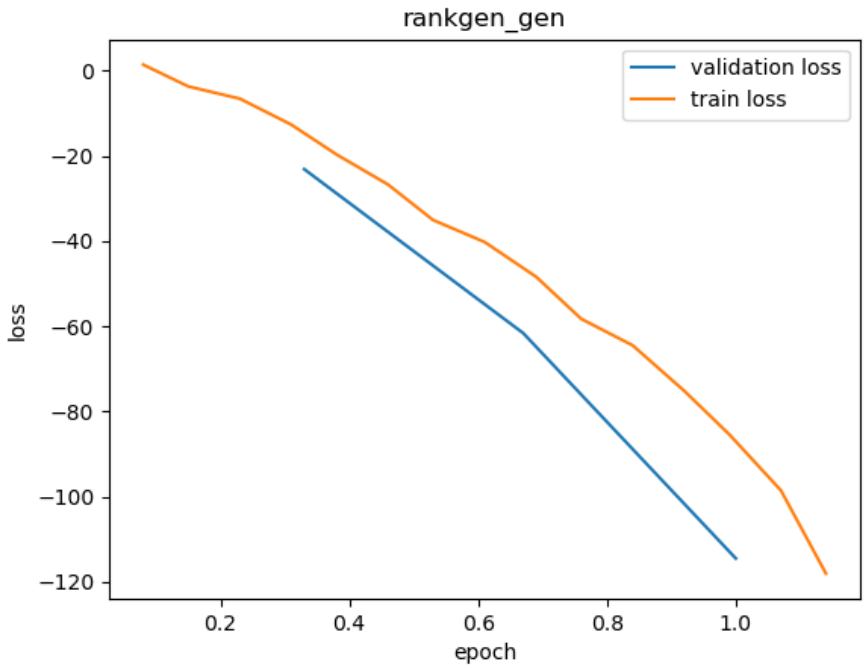

Рис. 3: Обучение модели с использованием генеративных отрицательных примеров

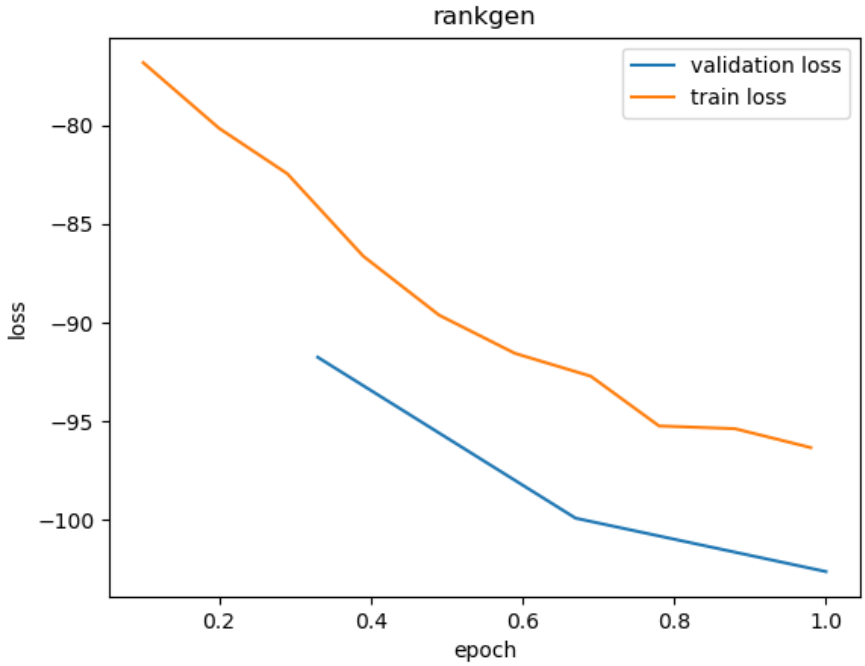

Рис. 4: Обучение модели без использования отрицательных примеров

4.4 Оценка результатов обучения

Для оценки качества направленной генерации был взят классификатор юмора[1] и с его помощью произведена классификация юмора на 1000 тестовых примерах:

Тип панчлайна	% смешных примеров с исп. панчлайна
ground truth	100
gpt2-large	89.2
gpt2-large + rankgen-ib-gen	94.1
gpt2-large + rankgen-ib	91.8
gpt2-large + rankgen-gen	92.8
gpt2-large + rankgen	93.6

Как можно заметить из результатов, классификатор юмора действительно определяет сгенерированные с использованием rankgen панчлайны как более смешные. Как и ожидалось, подход с использованием обоих типов отрицательных примеров, работает лучше чем без них. Однако использование лишь одного типа отрицательных примеров по какой-то неизвестной причине лишь ухудшает направленность генераций. Также неожиданностью оказалось и то, что использование генеративных отрицательных примеров помогает генерировать более смешные тексты, чем использование книжных отрицательных примеров.

[1]Можно найти по ссылке: https://huggingface.co/thanawan/bert-base-uncased-finetuned-humordetection

Заключение

В результате работы:

- Был собран большой датасет шуток с делением на сетап и панчлайн.
- При различных типах отрицательных примеров произведено обучение модели, ранжирующей панчлайны шутки по уровню юмора и помогающей генерировать более смешные тексты.
- Полученный метод незначительно улучшает качество направленной генерации за счёт более медленного процесса генерации.

В дальнейшем планируется:

- Улучшить датасет, что можно сделать путём дополнительного отсеивания несмешных шуток различными классификаторами и выбором наилучших примеров людьми.
- Провести аналогичные эксперименты на модели RankGen самого большого размера (1.2B параметров).
- Попробовать использовать итеративную процедуру дообучения модели RankGen: очень смешные с точки зрения модели генеративные панчлайны считать положительными примерами.
- Попробовать использовать дообученый на шутках RankGen в качестве классификатора в [7] (опционально, т.к. многократный подсчёт градиентов при генерации текста трудоёмок). Если использовать этот подход в совокупности с предыдущим пунктом, то получим процедуру, аналогичную обучению GAN [13], где RankGen используется в качестве дискриминатора (определяет смешно или нет).

Список источников

[1] Ari Holtzman и др. “The Curious Case of Neural Text Degeneration”. В: *arXiv* (апр. 2019). DOI: 10.48550/arXiv.1904.09751. eprint: 1904.09751.

[2] Angela Fan, Mike Lewis и Yann Dauphin. “Hierarchical Neural Story Generation”. В: *arXiv* (май 2018). DOI: 10.48550/arXiv.1805.04833. eprint: 1805.04833.

[3] Yixuan Su и Nigel Collier. “Contrastive Search Is What You Need For Neural Text Generation”. В: *arXiv* (окт. 2022). DOI: 10.48550/arXiv.2210.14140. eprint: 2210.14140.

[4] Xiang Lisa Li и др. “Contrastive Decoding: Open-ended Text Generation as Optimization”. В: *arXiv* (окт. 2022). DOI: 10.48550/arXiv.2210.15097. eprint: 2210.15097.

[5] Clara Meister и др. “Locally Typical Sampling”. В: *arXiv* (февр. 2022). DOI: 10.48550/arXiv.2202.00666. eprint: 2202.00666.

[6] Kalpesh Krishna и др. “RankGen: Improving Text Generation with Large Ranking Models”. В: *arXiv* (май 2022). DOI: 10.48550/arXiv.2205.09726. eprint: 2205.09726.

[7] Sachin Kumar, Biswajit Paria и Yulia Tsvetkov. “Gradient-Based Constrained Sampling from Language Models”. В: *arXiv* (май 2022). DOI: 10.48550/arXiv.2205.12558. eprint: 2205.12558.

[8] Yubo Xie, Junze Li и Pearl Pu. “Uncertainty and Surprisal Jointly Deliver the Punchline: Exploiting Incongruity-Based Features for Humor Recognition”. В: *arXiv* (дек. 2020). DOI: 10.48550/arXiv.2012.12007. eprint: 2012.12007.

[9] Leonard Tang и др. “The Naughtyformer: A Transformer Understands Offensive Humor”. В: *arXiv* (нояб. 2022). DOI: 10.48550/arXiv.2211.14369. eprint: 2211.14369.

[10] J. A. Meaney и др. “Don’t Take it Personally: Analyzing Gender and Age Differences in Ratings of Online Humor”. В: *arXiv* (авг. 2022). DOI: 10.48550/arXiv.2208.10898. eprint: 2208.10898.

[11] Orion Weller и Kevin Seppi. “Humor Detection: A Transformer Gets the Last Laugh”. В: *arXiv* (авг. 2019). DOI: 10.48550/arXiv.1909.00252. eprint: 1909.00252.

[12] Bhargav Chippada и Shubajit Saha. “Knowledge Amalgam: Generating Jokes and Quotes Together”. В: *arXiv* (июнь 2018). DOI: 10.48550/arXiv.1806.04387. eprint: 1806.04387.

[13] Ian J. Goodfellow и др. “Generative Adversarial Networks”. В: *arXiv* (июнь 2014). DOI: 10.48550/arXiv.1406.2661. eprint: 1406.2661.