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

**Факультет Санкт-Петербургская школа**
**физико-математических и компьютерных наук**

Мосин Владислав Дмитриевич

**Алгоритмы сэмплирования текстовых данных для ускорения обучения языковых моделей при помощи обучения по плану**

Выпускная квалификационная работа - БАКАЛАВРСКАЯ РАБОТА
по направлению подготовки *01.03.02 Прикладная математика и информатика*
образовательная программа «Прикладная математика и информатика»

Рецензент
Ю.М. Куратов

Руководитель
канд. физ.-мат. наук
И.Е. Кураленок

Консультант
Dr. rer. nat
И. П. Ямщиков

Санкт-Петербург 2021

# Оглавление

**Аннотация** **3**

**Введение** **5**

**1. Обзор Литературы** **8**

1.1. Обработка естественного языка . . . . . 8
1.2. Предобработка данных . . . . . 8
1.3. Модели для обработки естественного языка . . . . . 9
1.4. Обучение по плану . . . . . 11
1.5. Обучение по плану в различных областях машинного обучения . . . . . 12
1.6. Негативные результаты обучения по плану . . . . . 13
1.7. Обучение по плану в обработке естественного языка . . . . . 14
1.8. Выводы . . . . . 15

**2. Стратегии сэмплирования для обучения по плану** **16**

2.1. Сэмплеры с повторением примеров . . . . . 16
2.2. Сэмплеры без повторения примеров . . . . . 18
2.3. Реализация сэмплирующих алгоритмов . . . . . 19
2.4. Выводы . . . . . 19

**3. Эксперименты** **21**

3.1. Конфигурация экспериментов . . . . . 21
3.2. Метрики для оценки сложности . . . . . 21
3.3. Эксперименты на задаче предобучения . . . . . 23
3.4. Эксперименты на задаче классификации . . . . . 25
3.5. Эксперименты на задаче машинного перевода . . . . . 26
3.6. Эксперименты с различной токенизацией на задаче машинного перевода 27
3.7. Выводы . . . . . 28

**Заключение** **29**

**Список литературы** **30**

# Аннотация

Современные методы обработки естественного языка основаны на глубоком обучении, а используемые нейронные сети имеют от нескольких миллионов до сотен миллиардов параметров, что приводит к тому, что время их тренировки и количество требуемых вычислительных ресурсов очень высоко. На данный момент существует несколько способов для решения данной проблемы, одним из которых является обучение по плану. Обучение по плану состоит из двух основных частей: алгоритма сэмплирования данных и способа оценки их сложности. Целью данной работы является исследование сэмплирующих алгоритмов, а также их влияния на время, необходимое для обучения нейронных сетей. В процессе исследований было разработано несколько различных сэмплирующих алгоритмов с различными особенностями, а также проведено большое количество экспериментов на задачах классификации, машинного перевода и восстановление маскированных частей. В результате экспериментов было установлено, что обучение по плану не даёт прироста скорости обучения, а на некоторых задачах даже замедляет его. Также были проведены эксперименты с различной токенизацией текстов и было показано, что она не влияет на результаты.

Ключевые слова: обработка естественного языка, обучение по плану, сэмплирование.

Modern natural language processing methods based on deep learning. Neural networks, used in these methods, have from millions to hundreds of billion parameters, which results in high time and hardware resource consumption. Nowadays, there are many ways to deal with this problem and one of them is curriculum learning. Curriculum learning consists of two main parts: data sampling algorithm and measuring difficulty of data. The main purpose of this work is to investigate sampling algorithms and their influence on time, which requires training a neural network. During the research, some different sampling strategies with different characteristics were developed. Also, many experiments on masked language modeling, machine translation, and classification were performed. After these experiments, it was determined that curriculum learning did not give learning speed increase and for some tasks even decreases it. Also, some experiments with different tokenization strategies were performed. It was shown that tokenization does not influence performance.

Keywords: natural language processing, curriculum learning, sampling

# Введение

## Актуальность работы

На сегодняшний день обработка естественного языка используется во многих сферах нашей повседневной жизни: голосовые ассистенты, переводчики, браузеры, мессенджеры. Задачи, решаемые в рамках обработки естественного языка довольно разнообразны: классификация текстов, например, на эмоциональную окраску или спам - не спам, предсказание слова по контексту или следующего предложения по предыдущему, различного рода генерация одного текста по другому, например, перевод или суммаризация. Все эти задачи востребованы в современном мире и их надо решать как можно быстрее, задействуя как можно меньше вычислительных ресурсов, при этом не теряя в качестве. На данный момент лучшие по качеству решения основаны на механизме внимания [1], предложенном в 2017 году. К таким относятся, например, BERT [2] и GPT [16]. Однако, в данном случае за качество приходится платить временем и количеством вычислительных ресурсов. Архитектура вышеперечисленных сетей использует от нескольких миллионов до сотен миллиардов параметров. Для того чтобы обучить такие сети и не столкнуться с переобучением нужны большие корпуса данных, размер которых обычно составляет несколько миллионов записей. Все эти причины приводят к тому, что время тренировки становится слишком большим, и с этим хочется как-то побороться. Основной способ уменьшить время тренировки - это использование моделей, которые уже что-то знают про естественный язык, а именно обучались на большом корпусе неразмеченных данных предсказывать пропущенное слово по его контексту, такие модели называются предобученными, и они позволяют значительно сократить время тренировки уже на конкретную задачу, этот процесс называется дообучением [18, 19, 8]. Однако, даже при использовании предобучения и дообучения время, необходимое для получения качественного результата необходимо довольно много времени. Одним из способов уменьшить время ещё сильнее является обучение по плану [5]. Этот способ показал отличные результаты в различных областях машинного обучения, например, в глубоком обучении [11, 24], компьютерном зрении [21, 25], обучении с подкреплением [7, 17]. В обработке естественного языка также есть результаты, показывающие, что обучение по плану работает [4, 6, 14], однако, их не очень много, и алгоритмы сэмплирования не очень разнообразны. Это позволяет поставить довольно широкий исследовательский вопрос о том, какие сэмплирующие алгоритмы показывают лучшие результаты на задачах обработки естественного языка. Также в последнее время стали появляться негативные результаты, показывающие, что обучение по плану не всегда работает [23], и область применения данного метода может быть довольна узка, например, требовать какого-то особого формата данных. Отсюда вытекает ещё один исследовательский вопрос, который бу-

дет рассмотрен в данной работе, а работает ли обучение по плану в задачах обработки естественного языка и влияют ли на это используемые гиперпараметры.

## Цели и задачи

Целью данной работы является изучение влияния различных алгоритмов сэмплирования в обучении по плану на скорость и качество обучения языковых моделей на различных задачах обработки естественного языка.

- Реализовать возможность использовать различные алгоритмы сэмплирования без изменения остального кода.
- Предложить и реализовать различные сэмплирующие алгоритмы
- Сравнить полученные алгоритмы на различных задачах обработки естественного языка
- Исследовать влияние гиперпараметров на обучение по плану, в частности влияние токенизации.

## Достигнутые результаты

- Реализовано дополнение к библиотеке HuggingFace [13], позволяющее изменять процедуру сэмплирования из датасета, реализуя класс "Сэмплер" с единственным методом и подставляя его в класс "Тренер".
- Придумана и реализовано пять алгоритмов сэмплирования, обладающих различными свойствами
- Проведено большое количество экспериментов на различных задачах обработки естественного языка: предобучение, классификация и машинный перевод, показавших, что обучение по плану не помогает улучшить результаты.
- Проверена гипотеза о влиянии таких гиперпараметров, как параметризация токенизации на обучение по плану. Эксперименты показали, что влияние отсутствует.

## Структура работы

В секции "Обзор литературы" будут рассмотрены существующие методы для решения задач обработки естественного языка, а также область обучения по плану на различных задачах машинного обучения. Также в данной секции будет рассказано про существующие подходы к токенизации текстов.

В секции "Стратегии сэмплирования" будут рассмотрены предложенные стратегии сэмплирования для обучения по плану, а также объяснено, в чем их различия, и почему были выбраны именно такие алгоритмы. Также будет рассказано о способе реализации выбора алгоритма сэмплирования в библиотеке HuggingFace [13].

В секции "Эксперименты" будет подробно рассказано об экспериментах на трех различных задачах обработки естественного языка: предобучение, классификация и машинный перевод. Также будут описаны эксперименты с токенизацией.

# 1. Обзор Литературы

## 1.1. Обработка естественного языка

Глобально задачи обработки естественного языка можно поделить на несколько больших групп:

- Обучение на неразмеченных данных. Самые известные задачи в этой категории - это восстановление замаскированного слова по контексту, а также предсказание того, являются ли предложения последовательными в тексте.
- Классификация текстов.
- Генерация выходной текстовой последовательности по входной. Самые известные задачи в этой категории - машинный перевод и ответы на вопросы.

Несмотря на существенное смысловое различие, все задачи обработки естественного языка решаются при помощи одного глобального пайплайна, состоящего из двух частей: предобработка данных и обучение модели на этих данных.

## 1.2. Предобработка данных

Естественный язык - это последовательность слов, однако, их количество очень велико, к тому же с помощью приставок и суффиксов слова можно генерировать практически до бесконечности, следовательно, получается словарь неограниченного размера, а с таким модели работать не умеют. Также язык можно рассматривать как последовательность символов - букв, пробелов, знаков препинания. Однако, у этого подхода также есть недостатки, сам по себе символ не несёт достаточно информации. Для того чтобы не было проблемы бесконечного словаря, а также чтобы части содержали в себе достаточно информации используют различные методы токенизации. На данный момент выделяются три основных метода токенизации, всем три метода являются обучаемыми и нуждаются в корпусе данных.

- BPE [10]. Алгоритм состоит из инициализации словаря всеми символами из алфавита и рекурсивного увеличения до фиксированного размера на самую частую пару токенов уже добавленных в словарь.
- WordPiece [20]. Аналогично предыдущему алгоритму инициализация происходит всеми символами их алфавита, однако, рекурсивное добавление происходит уже на основании функции потерь некоторой языковой модели.
- Unigram LM [15]. Этот алгоритм в отличии от предыдущих удаляет токены из словаря, а не добавляет. Инициализация происходит всеми сочетаниями символами меньших определенной длины, а также встречающихся в корпусе текстов

хотя бы несколько раз, конкретное количество является гиперпараметром и может изменяться.

Несмотря на то что методы токенизации выглядят довольно похожими, однако, показывают различные результаты, например, работа Kaj Bostrom [3] показывает, что Unigram LM лучше коррелирует с разделением слова на морфемы, чем BPE.

| | |
|---|---|
| **Original:** | `Completely preposterous suggestions` |
| **BPE:** | `_Comple t ely _prep ost erous _suggest ions` |
| **Unigram LM:** | `_Complete ly _pre post er ous _suggestion s` |

Рис. 1: Сравнение BPE и Unigram LM. Текст.

Также в данной работе показано, что распределение токенов словарях для BPE и Unigram LM не совпадает.

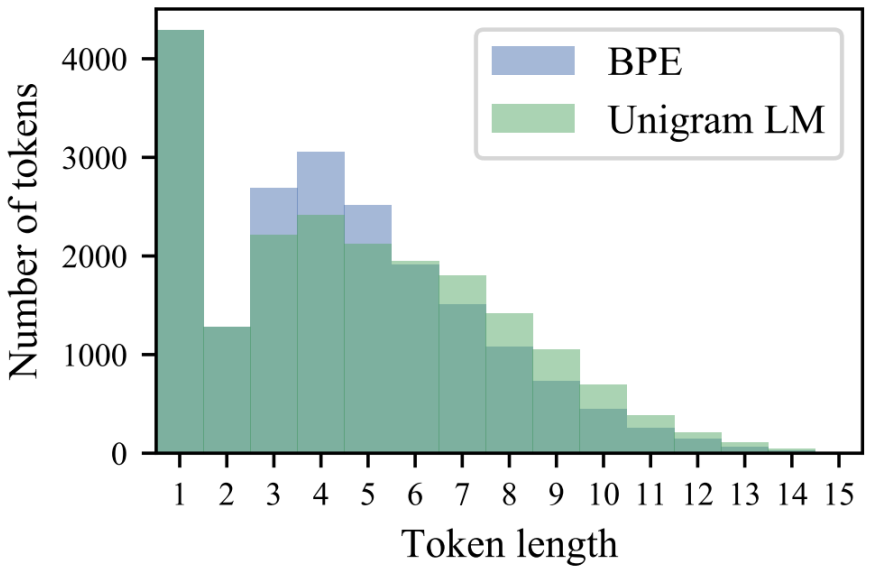


Рис. 2: Сравнение BPE и Unigram LM. Распределение.

Таким образом, на данный момент существует несколько различных способов предобрабатывать текстовые данные, однако, какой из них оптимальнее остается нерешенной задачей.

## 1.3. Модели для обработки естественного языка

Тексты довольно сложные данные, общий смысл текста зачастую зависит от последовательно слов, возможно, с этим и связано то, что первые успехи нейронных сетей

связаны с появлением LSTM [12], например, Shuohang Wang [22] показал, что LSTM достигает приличных результатов на задаче NLI.

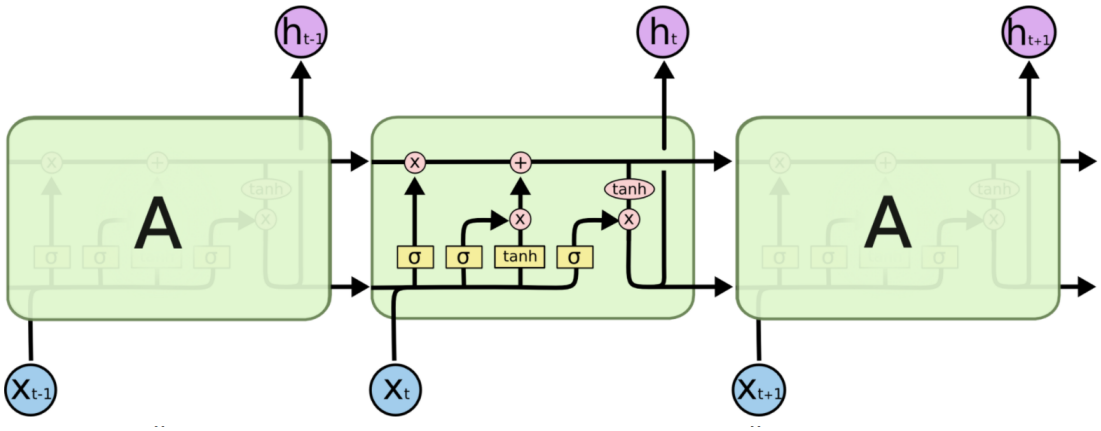


Рис. 3: Архитектура LSTM.

LSTM - разновидность рекуррентной нейронной сети, которая позволяет не затухать градиентам в алгоритме обратного распространения ошибки. Это достигается при помощи того, что скрытое состояние не домножается на матрицу весов напрямую, а всего лишь изменяется при помощи поэлементного умножения и сложения с другим вектором.

Однако, прорыв в обработке естественного языка связан с появлением механизма внимания [1]. Идея механизма внимания в том, что каждое следующее состояние нейронной сети зависит от предыдущих в большей или меньшей степени, и вот учитывание этой зависимости и называется механизмом внимания.

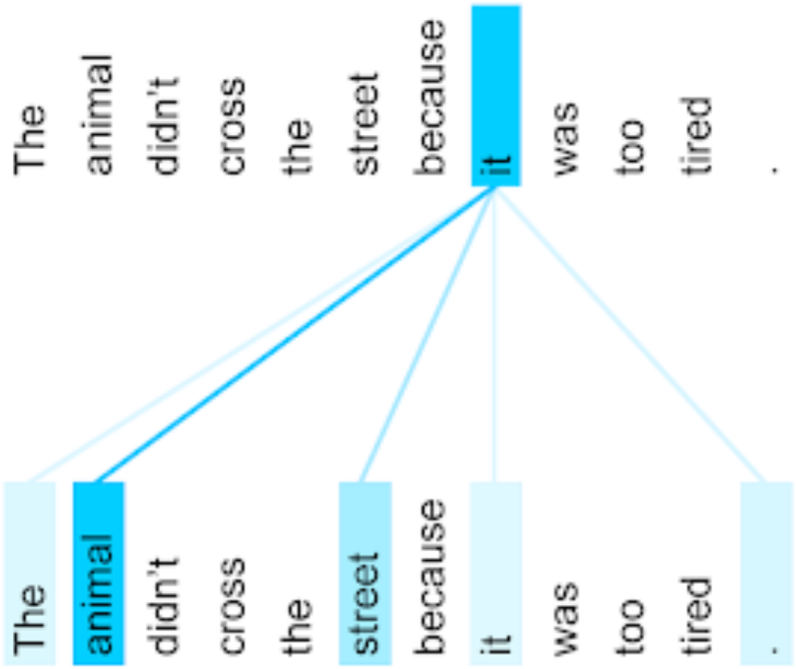


Рис. 4: Механизм внимания. Распределение связанности слова it с другими

На этом механизме основан целый класс архитектур - трансформеры, состоящие из слоев энкодера и декодера и использующие механизм multi-head внимания. Такие архитектуры обычно являются довольно большими, однако, в отличие от обыкновенных рекуррентных сетей позволяют параллелить процесс вычисления, что дает им огромное преимущество.

На базе архитектуры трансформер было создано несколько широкоизвестных сетей, которые на данный момент показывают лучшие результаты в области. Наверное, самые известные из них - это BERT [2] и GPT [16]. Основное отличие этих двух архитектур друг от друга в том, что первая использует только стек энкодеров, а вторая только стек декодеров.

## 1.4. Обучение по плану

Идея обучения по плану идет из нашей повседневной жизни, например, в школе сначала проходятся простые темы, а затем всё более и более сложные. Первым упоминанием обучения по плану считается статья Bengio из 2009 года [5]. Автор показал, что если модель научить сначала различать простые геометрические фигуры, то сложные ей будет различать существенно проще. Глобально обучение по плану состоит из двух частей: сортировки датасета на основании метрики сложности и алгоритма сэмплирования, который определяет в какие элементы попадут в батч, на котором

модель и будет учиться - производить градиентный спуск.

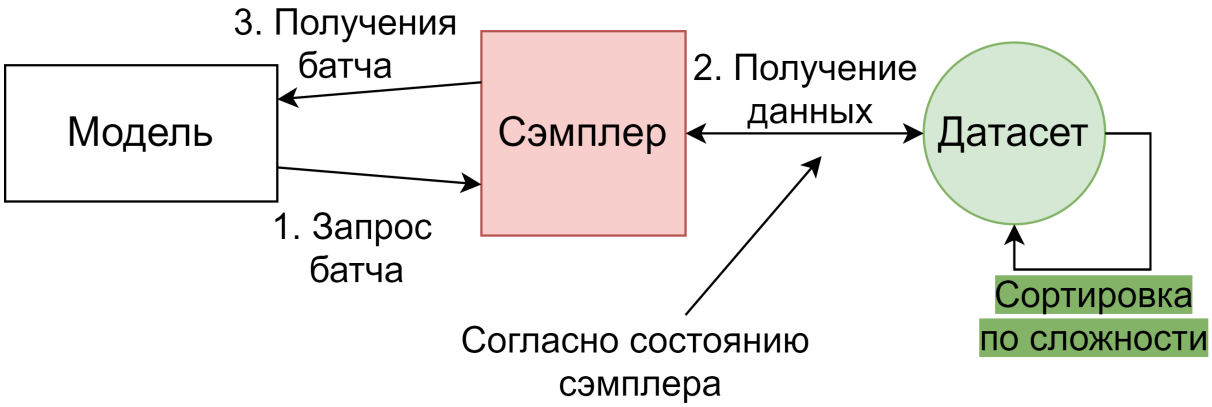


Рис. 5: Обучение по плану. Модель запрашивает батч для градиентного спуска, сэмплер согласно своему состоянию выбирает нужные элементы из датасета и отдаёт их модели

## 1.5. Обучение по плану в различных областях машинного обучения

Обучение по плану находит своё применение в различных областях машинного обучения, например, в компьютерном зрении Hacohen и Weinshall в 2019 показали при помощи данного подхода высокие результаты. Основная суть их работы в использовании модельной метрики сложности и функции сэмплирования с префикса. Сэмплирование происходит следующим образом, авторы вводят функцию скорости, которая отвечает за рост префикса, с которого собственно и происходит сэмплирование.

**Algorithm 1** Curriculum learning method

**Input:** *pacing function* $g_\vartheta$, *scoring function* $f$, data $\mathbb{X}$.
**Output:** sequence of mini-batches $\left[\mathbb{B}'_1, ..., \mathbb{B}'_M\right]$.
sort $\mathbb{X}$ according to $f$, in ascending order
$result \leftarrow []$
**for all** $i = 1, ..., M$ **do**
  $size \leftarrow g_\vartheta(i)$
  $\mathbb{X}'_i \leftarrow \mathbb{X}\,[1, ..., size]$
  uniformly sample $\mathbb{B}'_i$ from $\mathbb{X}'$
  append $\mathbb{B}'_i$ to $result$
**end for**
**return** $result$

Рис. 6: Обучение по плану. Компьютерное зрение. Общий алгоритм

Авторы данной статьи попробовали несколько различных стратегий увеличения префикса, однако, все они являются дискретными и количество увеличений не так велико. Основная идея состоит в том, что после нескольких шагов обновления модели префикс нужно немного увеличить на некоторый шаг. Авторы попробовали один раз увеличивать префикс, увеличивать префикс на экспоненциально растущее значение, а также аналогично предыдущему, но с нефиксированным количеством обновлений модели между последовательными увеличениями префикса.

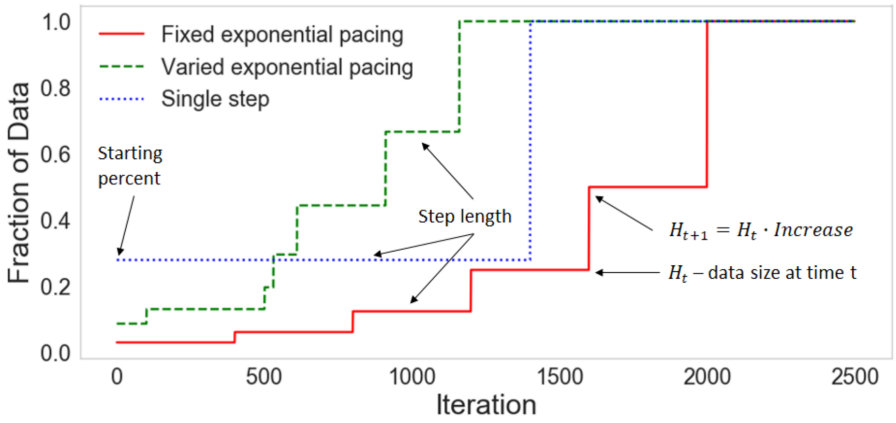


Рис. 7: Обучение по плану. Компьютерное зрение. Различные функции увеличения префикса

В других областях машинного обучения также есть работы показывающие хорошие результаты, например, в обучении с подкреплением Narvekar в 2020 году [7] использовал очень простой алгоритм сэмплирования, который заключается в разбиении отсортированной последовательности примеров из датасета на батчи, без какого-либо сэмплирования.

## 1.6. Негативные результаты обучения по плану

Несмотря на большое количество статей, показывающих хорошие результаты, не так давно появилось большое исследование в области компьютерного зрения [23], в котором авторы провели большое количество исследований с различными разновидностями обучения по плану. Их результаты показывают, что обучение по плану не дает прироста качества и скорости на стандартных задачах, однако, у них получилось создать пару искусственных задач, на которых обучение по плану и правда дает прирост. Такими задачами являются обучение на шумных данных (к каждому элементу из датасета добавлено случайное количество шума) и обучение на малом количестве данных.

## 1.7. Обучение по плану в обработке естественного языка

В области обработки естественного языка также существует несколько работ связанных с обучением по плану. Однако, их количество не очень велико, например, по сравнению с количеством работ в компьютерном зрении. Скорее всего это связано с тем, что тексты, в отличие от картинок, не умеют строгой, фиксированной структуры. Тексты состоят из предложений, предложения из слов, слова из букв, причем количество элементов не фиксировано ни в одной из вложенностей. Причем не любая последовательность букв становится словом и не любой набор слов является предложением. Однако, несмотря на большое количество трудностей с оценкой сложности текстов, работы, показывающие, что обучение по плану работает для задач обработки естественного языка. Например, Platanios в 2019 году [4] провел несколько экспериментов с обучением по плану на задаче машинного перевода и показал увеличение скорости обучения на 70% и метрики BLEU (количество совпадений n-грамм) на 2.2 процентных пункта. Автор использовал алгоритм сэмплирования с постепенно увеличивающегося префикса.

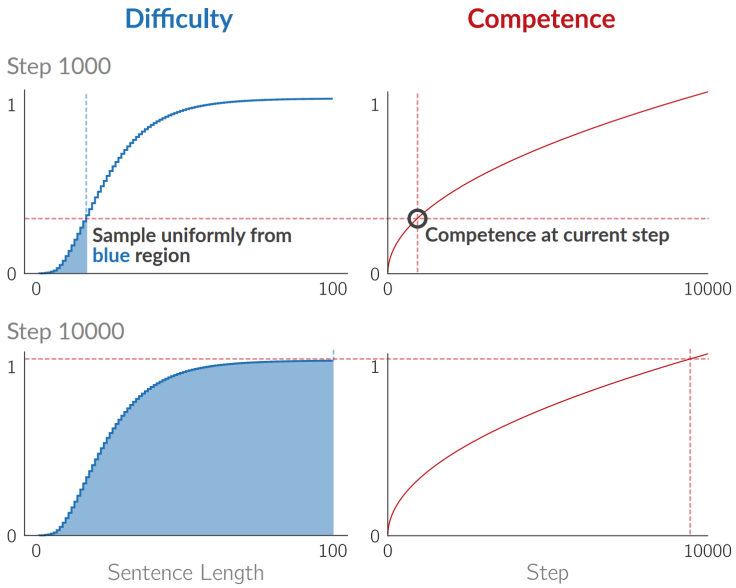


Рис. 8: Сэмплирование с постепенно увеличивающегося префикса.

В данной работе было использовано несколько различных функций сэмплирования, однако, все они из одного семейства. Семейство задается следующим образом

$$prefix\_len = \min\left(1, \left(t\frac{1 - c_0^{1/p}}{T} + c_0^p\right)^p\right)$$

где $t, T$ - номер текущего шага и суммарное количество шагов соответственно, $c_0$ - константа, в работе была зафиксирована значением 0.01, $p$ - гиперпараметр, отвечающий

за скорость роста префикса в начале.

Также интересные результаты были получены в работе Kocmi в 2017 году [14], они получили увеличение метрики BLEU на 1.2 процентных пункта. Их процедура сэмплирования довольна интересна. Во-первых, они замечают, что классический подход к алгоритмам сэмплирования - случайный выбор элементов с постепенно увеличивающегося префикса датасета имеет один не очень желательный внешний эффект - чем более простым является элемент, тем чаще он попадает модели. Для того чтобы убрать такой внешний эффект автор предлагает следующий алгоритм. Для начала датасет разделяется на несколько корзин по увеличению сложности, причем чем сложнее корзина, тем меньше там элементов, что, кстати, объяснимо, так как в датасете обычно простых примеров существенно больше, чем сложных. Изначально сэмплирование происходит из первой корзины, суммарное количество элементов равно разности размеров первой и второй корзины, затем элементы сэмплируются уже из первой и второй корзины равновероятно, количество раз равно разности между размерами второй и третьей корзин и так далее.

## 1.8. Выводы

- Существует большое количество работ в области обучения по плану, однако, существующие алгоритмы сэмплирования не очень разнообразны, особенно в области обработки естественного языка.

- В обработке естественного языка нет работ, анализирующих обучение по плану на задаче классификации текстов и задачах, не требующих размеченного датасета (unsupervised).

- Существует работа, показывающая, что обучение по плану не работает в простом случае, а требует каких-либо искуственных условий. Также в большинстве работ результаты довольно сильно зависят от постановки эксперимента: предобработки данных и гиперпараметров.

# 2. Стратегии сэмплирования для обучения по плану

В этой главе будут предложены сэмплирующие стратегии для обучения по плану, а также описан способ их реализации.

## 2.1. Сэмплеры с повторением примеров

На данный момент в области естественного языка предложено довольно малое количество разнообразных сэмплирующих стратегий, однако, несмотря на малое разнообразие хорошие результаты были показаны, например, в работе Platanious [4]. Как было описано выше, данная стратегия занимается сэмплированием с постепенно растущего префикса и имеет интересную особенность, связанную с тем, что распределение примеров, показанных модели, смещается в сторону более простых примеров. В данной работе мы придумали и протестировали два сэмплера с повторениями примеров с различными свойствами распределений данных по сложности, показываемых модели. Первый из них - это сэмплер, который оставляет распределение неизменным, однако, сложность примеров всё равно возрастает со временем. Алгоритм сэмплирования заключается в выборе элементов из равномерного распределения, дисперсия которого остается зафиксированной, а медиана увеличивается со временем. Такой механизм похож на сэмплирование с некоторого участка со смещением в сторону центра, однако, в подходе с распределением модель будет получать элементы в том числе и из других частей датасета, что в теории не позволит ей переобучиться на конкретный участок данных. Вторая сэмплирующая стратегия смещает распределение примеров в сторону увеличенной сложности. В отличие от классической стратегии, сэмплирующей с префикса, который увеличивается, эта стратегия наоборот уменьшает размер множества, из которого происходит сэмплирование, а именно сэмплирует с линейно уменьшающегося со временем суффикса. Важной характеристикой всех сэмплеров данной группы является то, что все они не фиксируют количество раз, которое элемент попадает в обучающую выборку, что приводит к тому, что фактически выборка, на которой происходит тренировка меньше всего датасета.

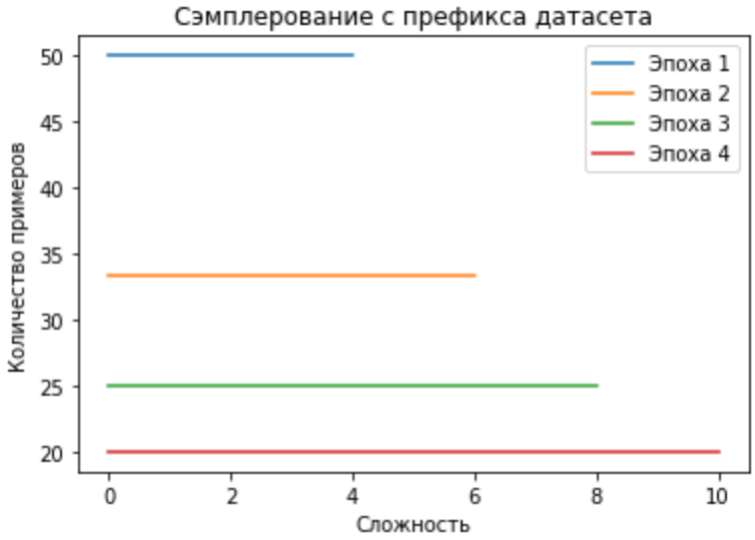


Рис. 9: Сэмплирование с постепенно увеличивающегося префикса. Распределение сложностей по эпохам

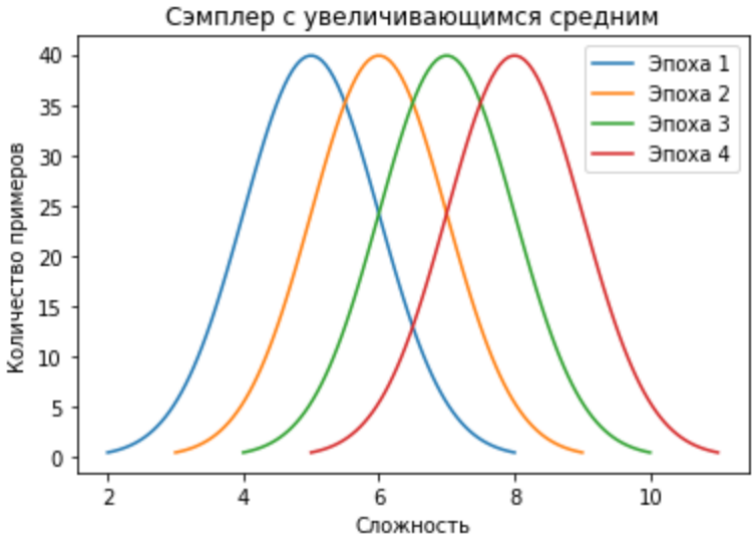


Рис. 10: Сэмплирование из нормального распределения с увеличивающимся средним. Распределение сложностей по эпохам

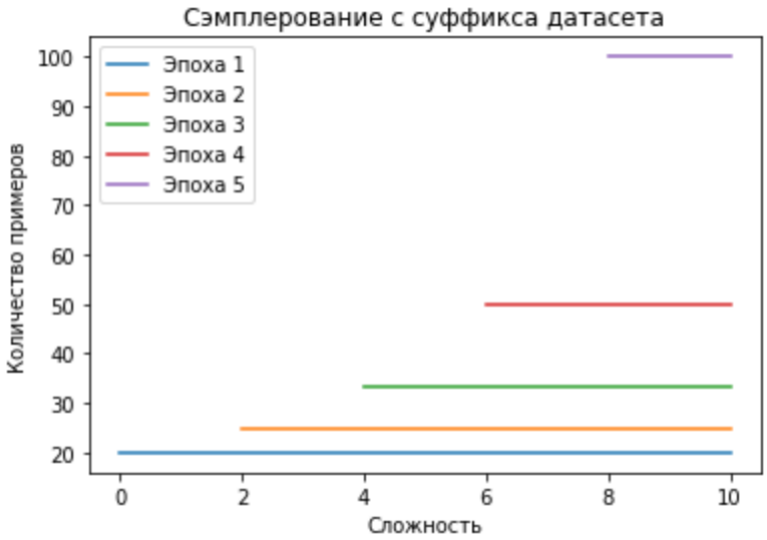


Рис. 11: Сэмплирование с постепенно уменьшающегося суффикса. Распределение сложностей по эпохам

## 2.2. Сэмплеры без повторения примеров

В предыдущей части были рассмотрены алгоритмы сэмплирования, которые не гарантируют, что все примеры из датасета будут использованы одинаковое количество раз, таким образом содержательная часть датасета, та на которой происходило обучение, уменьшается. Для того чтобы избавиться от такого внешнего эффекта в данной секции будут рассмотрены алгоритмы, позволяющие избавиться от данного внешнего эффекта. Первая сэмплирующая стратегия состоит из случайного разделения датасета на батчи и дальнейшей их сортировки по средней сложности элементов. Особенностью данного подхода является то, что батчи формируются случайным образом, а изменятся только порядок, в котором происходит обучение. Вторая сэмплирующая стратегия из данный группы появилась из-за того, что мы заметили, что большинство метрик оценки сложности текстов имеют положительную корреляцию с их длиной. Для того чтобы убрать такую корреляцию мы применили следующий алгоритм:

- Сортируем датасет по длине текста
- Разделяем последовательно на $N$ корзин по сложности (в $i$-ой корзине все элементы имеют меньшую сложность, чем в $i+1$-ой), где $N$ - размер батча, который будет использован при сэмплировании
- Каждую из корзин сортируем по выбранной метрике сложности, отличной от длины.
- При обучении формируем в батчи по одному из каждой корзины. В $i$-ый батч попадают $i$-ые элементы из каждой корзины.

Такой алгоритм позволяет тренировать модель практически на стабильном по длине распределении, однако, на изменяющемся по другой, целевой метрике.

## 2.3. Реализация сэмплирующих алгоритмов

Для проведения экспериментов мы воспользовались готовой библиотекой HuggingFace [13] для тренировки моделей глубокого обучения, так как такой подход позволяет минимизировать вероятность ошибки при проведении экспериментов, а также возможность использовать готовые рецепты для тренировки моделей с уже подобранными гиперпараметрами, что позволяет значительно уменьшить количество экспериментов и более разумно использовать ограниченные вычислительные мощности. Однако, данная библиотека не предполагает возможности для пользователя изменять алгоритм сэмплирования. Для решения данной проблемы мы создали свой класс для тренировки модели, отнаследовавшись от стандартного и переопределив только методы, отвечающие за сэмплирование.

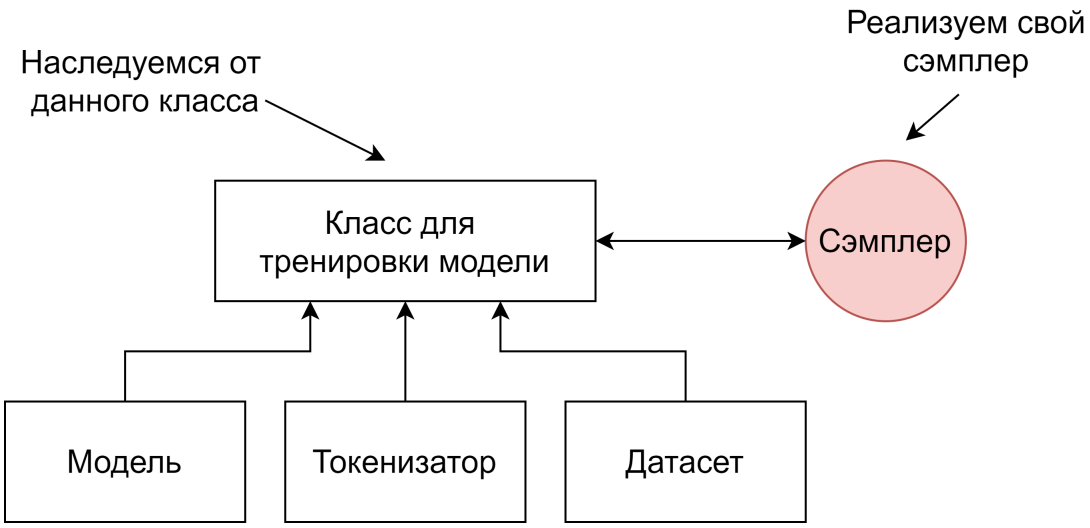


Рис. 12: Реализация различных алгоритмов сэмплирования при помощи библиотеки HuggingFace

Таким образом, для проведения экспериментов на нужно просто реализовать различные сэмплеры - классы с одним методом, который собственно и производит сэмплирование из датасета, остальная часть остается без изменений.

## 2.4. Выводы

В данной секции были приведены различные алгоритмы сэмплирования, обладающие широким спектром различных свойств, а также был описан метод, позволяющий

легко реализовывать различные сэмплирующие стратегии и встраивать их в общий пайплайн для тренировки моделей.

# 3. Эксперименты

В данной главе будут описаны эксперименты, проведенные на различных задачах обработки естественного языка, также будет приведен анализ полученных результатов и рассмотрено влияние предобработки данных на обучение по плану.

## 3.1. Конфигурация экспериментов

В рамках данной работы были проведены эксперименты на трех различных группах задач обработки естественного языка: предобучение, классификация, генерация выходной последовательности по входной.

1. Предобучение. Особенность задач данной задачи состоит в том, что они не требуют размеченного датасета, а следовательно, могут быть натренированы на огромном количестве данных. Из данной группы мы выбрали задачу восстановления замаскированного слова по его контексту [2]. Обучение проводилось на датасете BooksCorpus, содержащем 74 миллиона отрывков текстов из различных книг. Для данной задачи использовалась архитектура BERT [2] в версии base.

2. Классификация. Данная класс задач состоит из отнесения текста к одной из нескольких групп. Для данной задачи было выбран датасет Hyperpartisan News Detection, состоящий из двух миллионов коротких статей. В данной задаче нужно определять, есть ли в статьях политическая окраска - предвзятость в отношении той или иной политической силы. Для данной задачи использовалась архитектура BERT [2] в версии base.

3. Генерация выходной последовательности по входной. Из данного класса мы выбрали самую известную и часто встречающуюся задачу - машинный перевод. Был выбран WMT16 датасет, состоящий из 4.5 миллионов пар фраз на английском и немецком. Для данной задачи была выбрана архитектура T5 [9].

Для всех задач использовались гиперпараметры по умолчанию из библиотеки HuggingFace [13], а также токенизатор WordPiece [20] в версии, игнорирующей регистр символов и соответствующей обучаемой модели.

## 3.2. Метрики для оценки сложности

В данной работе были использованы следующие метрики для оценки сложности текстов.

1. Длина. Сложность текста равна количеству токенов в представлении после токенизации. Довольно популярная метрика, которая, например, была использована Platanious [4] в экспериментах на задаче машинного обучения.

2. Максимальный ранг. Рассматривается некоторый датасет текстов, по которому строится частотный словарь - каждому слову сопоставляется его частота - количество его вхождений в датасет. Затем слова сортируются по частоте. Ранг слова - это его порядковый номер в таком словаре. Посчитав ранг каждого слова, достаточно легко посчитать саму метрику по тексту. Для этого нужно просто найти ранг всех слов из данного текста, а затем взять максимальный.

3. TSE и excess entropy. Эти две метрики изначально были созданы для оценки сложности конечных последовательностей случайных величин [26] и основаны на теории информации. Основная мотивация использовать такие метрики заключается в том, что после токенизации текст превращается в числовую последовательность, в которой важен не только набор токенов, но и их порядок, однако, большинство метрик оценки сложности текстов не берут это во внимание. Для использования данных метрик нужно научиться преобразовывать тексты, в последовательность случайных величин. Для этого мы используем следующий способ: заменяем токен, на бинарную случайную величину, которая принимает единицу с вероятностью равной частоте встречаемости данного токена на данной позиции, деленной на количество текстов.

4. Вероятность. Для того чтобы оценить вероятность текста, нужно для начала определить вероятность каждой его части - каждого слова. Вероятность слова - это просто его доля от всех слов в тексте. При условии, что слова в тексте независимы (понятно, что это не так), то вероятность текста можно вычислить как произведение вероятностей всех его слов. Однако, значение такого произведения будет очень близко к нулю, поэтому в качестве метрик берется $-\log$ данного произведения.

5. TF-IDF. Данная метрика широко применяется в информационном поиске для определения расстояния между документами. TF-IDF – это произведение TF на IDF, где TF – это вероятность вхождения токена в документ, а IDF – это величина, обратная доле документов, в которых встречается данный токен. Посчитав такую метрику для каждого слова в тексте, можно оценить его сложность как среднее значение TF-IDF по всем словам.

## 3.3. Эксперименты на задаче предобучения

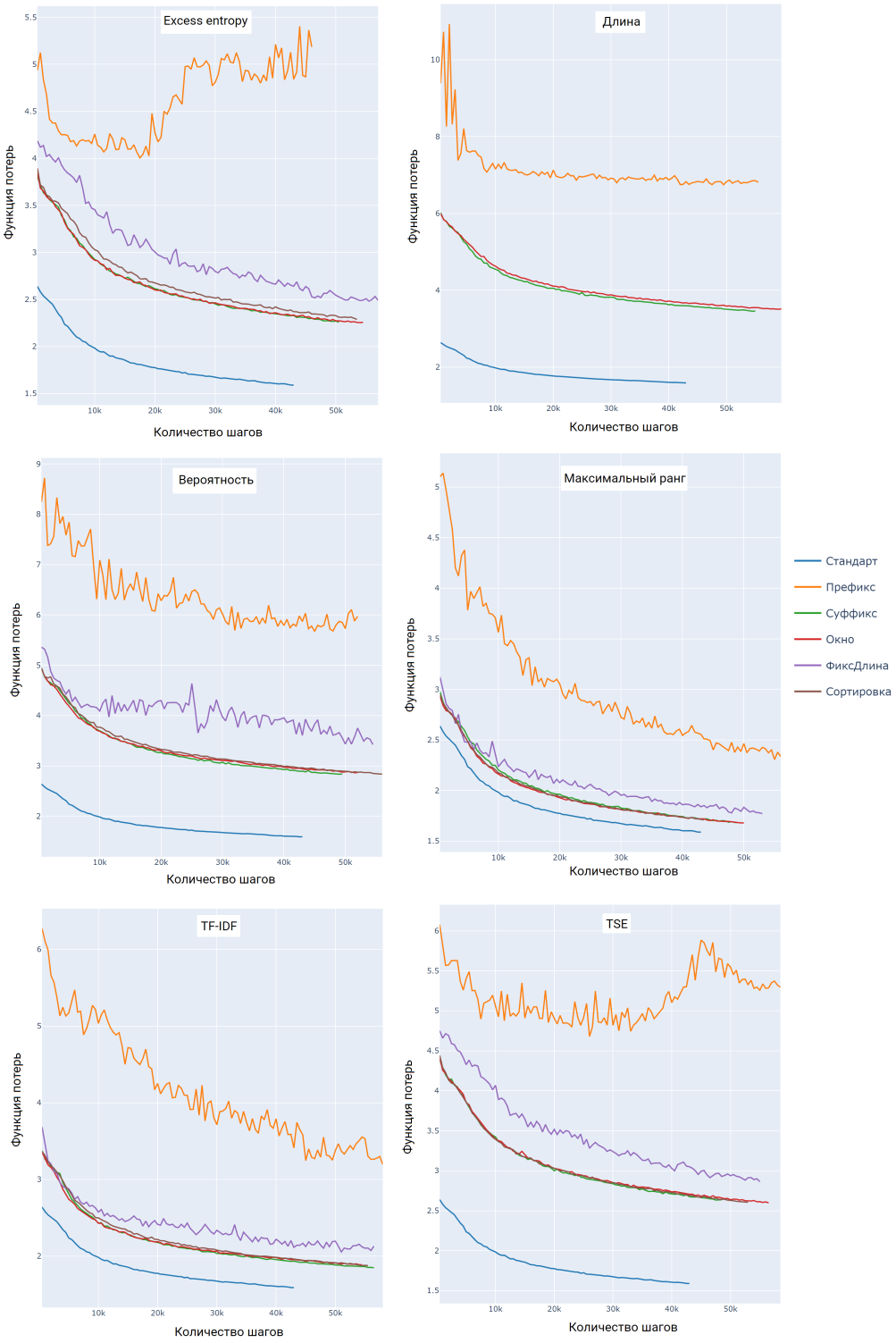


Рис. 13: Результаты на задаче восстановления пропущенного слова по контексту

Эксперименты на задаче восстановления слова по контексту показывают, что обучение по плану только ухудшает результаты. Алгоритм сэмплирования "Суффикс" показывает самые плохие результаты, что говорит о том, что для данной задачи сдвиг показываемых модели примеров в сторону простых только существенно хуже, чем в сторону сложных (сэмплер "Суффикс"). Изначально у нас была идея, что на данной задаче длина может быть не самой хорошей метрикой, так как длинный контекст может помогать определять пропущенное слово, а большинство метрик имеют положительную корреляцию с длиной. Для проверки данной гипотезы и был создан сэмплер ФиксДлина, однако, данная гипотезы не подтвердилась, и сэмплер ФиксДлина показывает довольно плохие результаты. Важным является то, что поведение всех сэмплеров с различными метриками одинаково, Суффикс, Окно и Сортировка ведут себя одинаково, ФиксДлина немного хуже, а Префикс показывает самые плохие результаты.

## 3.4. Эксперименты на задаче классификации

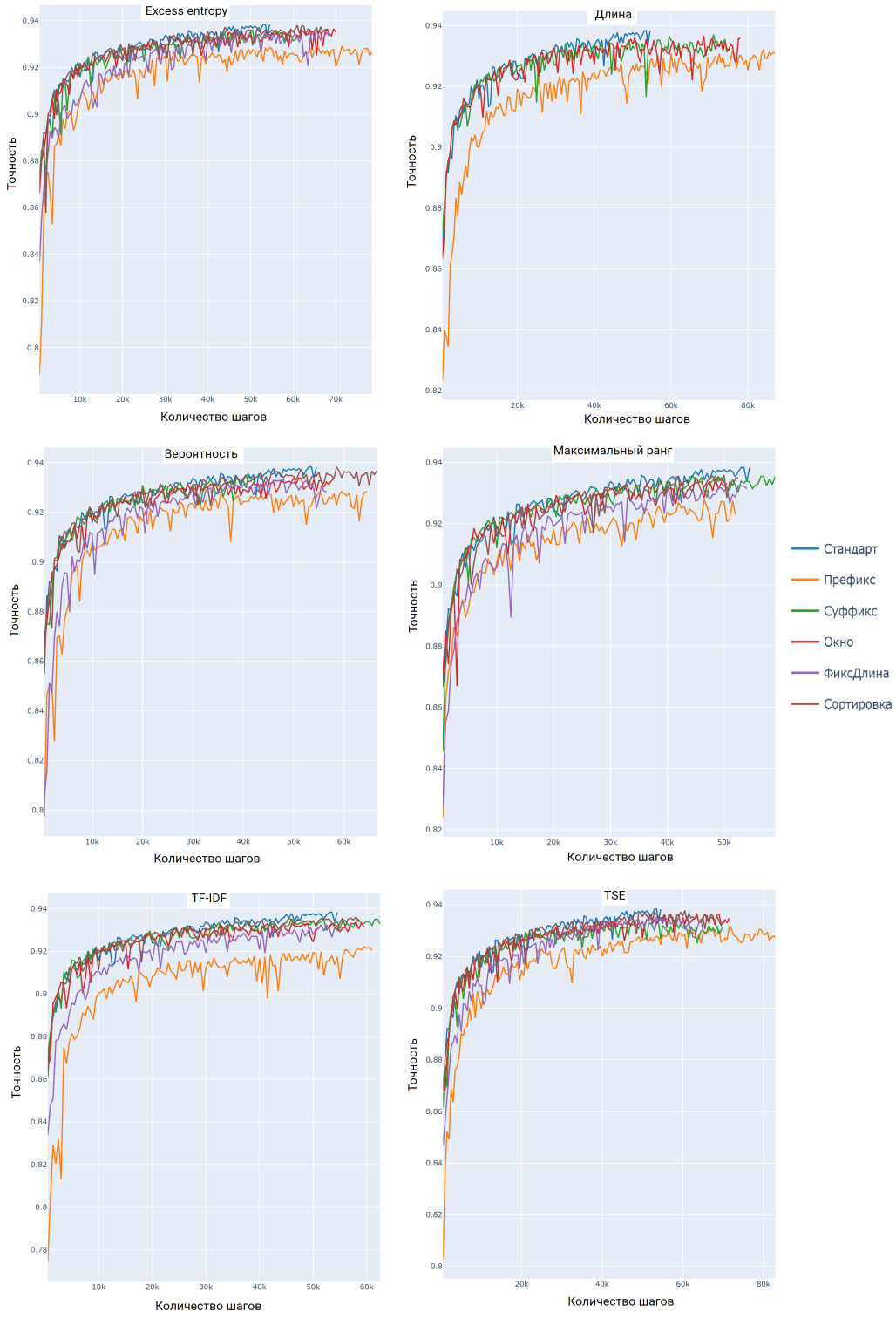


Рис. 14: Результаты на задаче классификации твитов по политической окраске

На задаче классификации результаты немного отличаются от результатов на предобучении. Всё ещё Префикс выдает очень плохие результаты, хуже чем все остальные алгоритмы сэмплирования. Также не очень хорошо себя показывает алгоритм ФиксДлина, остальные три алгоритма показывают результаты, сравнимые со стандартным алгоритмом обучения.

## 3.5. Эксперименты на задаче машинного перевода

В процессе проведения экспериментов на задаче машинного перевода мы столкнулись с проблемой несовершенства библиотеки HuggingFace. Существующее API для подсчета различных метрик по ходу обучения модели не позволяет проводить валидацию на большом корпусе данных, так как при осуществлении данной операции разово загружают все предсказания в память. Предсказание для одного текста представляет из себя двумерный массив размерности (размер словаря, длина последовательности), что при размере словаря в 30000 и длине последовательности в 500 получается 15M чисел, учитывая тот факт, что каждое число представляется 4 байтами, в итоге получаем 60 Мб на всего лишь одну запись. Если валидационная выборка имеет более 600 записей, что довольно мало, то такие предсказания не помещаются в видеопамять на GPU. Для решения данной проблемы был реализован следующий пайплайн: на одной видеокарте происходит обучение, причем вместо подсчета метрик на валидации происходит простое сохранение состояния модели на диск, на второй видеокарте последовательно, не загружая все данные в память единоразово, считаются метрики. В данных экспериментах мы использовали метрику BLEU - процент совпадений $n$-грамм в предсказании и правильном значении. Так как при проведении экспериментов получилось, что метрика BLEU имеет довольно большую дисперсию, сравнение алгоритмов сэмплирования будет происходить по усреднению BLEU за 11 шагов (с 50k шага, до 100k шага с интервалом в 5k).

Таблица 1: Результаты на задаче машинного перевода. Стандарт - обучения без изменения сложности батча от времени

| Сэмплер | Метрика сложности | | | Среднее |
|---|---|---|---|---|
| | Длина | EE | TSE | |
| Префикс | 10.1 | 11.3 | 11.2 | 10.7 |
| Суффикс | 16.7 | 16.8 | 17.0 | 16.8 |
| Окно | 15.9 | 17.0 | 16.8 | 16.6 |
| Сортировка | 16.3 | 16.6 | 16.6 | 16.5 |
| ФиксДлина | - | 12.8 | 13.4 | 13.1 |
| Стандарт | | | | 16.9 |

Результаты показывают, что обучение без плана выигрывает у обучения по пла-

ну для всех сэмплеров. Однако, как и в предыдущих экспериментах, существенное проигрывают только Префикс и ФиксДлина. Остальные сэмплеры показывают схожие со стандартом результаты, а на некоторых метриках даже немного превосходят, однако, отличие слишком мало, чтобы можно было сказать, что обучение по плану заработало на задаче машинного перевода.

## 3.6. Эксперименты с различной токенизацией на задаче машинного перевода

Получив негативные результаты на предыдущих задачах, мы попробовали выдвинуть гипотезы, с чем это может быть связано. Одной из наших гипотез является то, что правильно выбранный способ токенизации положительно влияет на обучение по плану. Данная гипотеза связана с тем, что Wu в своей работе [23] говорит, что обучение по плану чувствительно к гиперпараметрам, а Platanious в своей работе по обучению по плану [4] в области обработки естественного языка использует довольно сложный алгоритм токенизации, который самостоятельно тренирует, а не использует один из предобученных, находящихся в общем доступе. Первым экспериментом мы проверили, что различная токенизация (в зависимости от размера словаря) изменяет не только сам размер словаря, но и распределение по сложности в датасете.

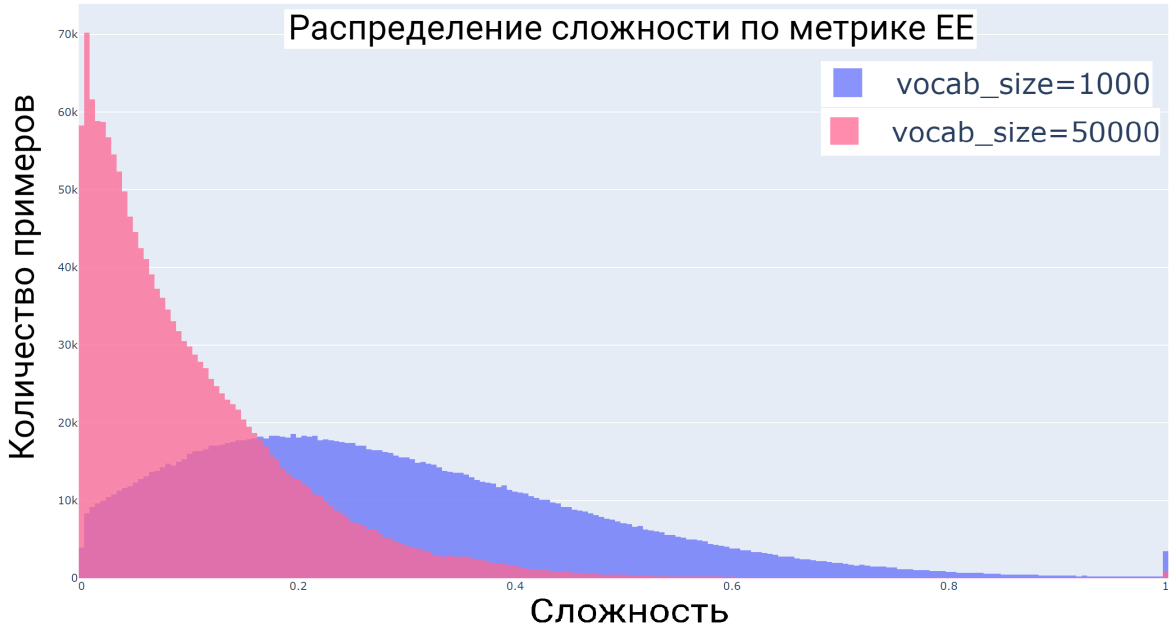


Рис. 15: Распределение сложности текстов по метрике EE при токенизации UnigramLM для размера словаря 1k и 50k

Результаты показали, что распределение и правда изменяется, что позволяет надеяться, что результаты обучения по плану также поменяются. Мы провели эксперимен-

ты на задаче машинного перевода, использую стандартную процедуру тренировки, а также обучение по плану с метрикой TSE и сэмплером "Префикс" (как показывающий самые хорошие результаты), в качестве токенизаторов использовали BPE [10], WordPiece [20], UnigramLM [15] с размерами словаря 30k (используется по умолчанию) и 5k токенов.

Таблица 2: Результаты стандартного обучения при различных токенизациях

| | 5k | 30k |
|---|---|---|
| BPE | 12.6 | 16.8 |
| UnigramLM | 12.7 | 16.7 |
| WordPiece | 12.4 | 16.9 |

Таблица 3: Результаты обучения по плану при различных токенизациях

| | 5k | 30k |
|---|---|---|
| BPE | 12.5 | 16.6 |
| UnigramLM | 12.7 | 16.6 |
| WordPiece | 12.6 | 16.8 |

Полученные результаты показывают, что при изменении алгоритма токенизации результаты практически не измеменяются, а при уменьшении размера словаря, качество падает одинаково и при стандартном обучении, и при обучении по плану. Следовательно, показать, что такой гиперпараметр как токенизация не влияет на обучение по плану.

## 3.7. Выводы

Были проведены эксперименты на трех задачах обработки естественного языка: предобучение, классификация, машинный перевод. На задаче предобучения обучение по плану ухудшает результаты для всех сэмплирующих алгоритмов, причем, если чаще сэмплировать простые примеры чаще (сэмплер "Префикс"), модель не справляется даже обучиться. На задачах классификации и машинного перевода сэмплер "Префикс" также получает результаты значительно хуже, чем стандартный алгоритм, однако, сэмплеры "Сортировка", "Суффикс" и "Окно" показывают результаты, которые сравнимы со стандартным обучением. На основании работ Wu [23] и Platanious [4], в которых говорится, что правильно подобранные гиперпараметры, в частности токенизация, очень важны, была выдвинута гипотеза, что стандартная процедура токенизации не подходит, и её нужно заменить на другую. Однако, гипотеза не подтвердилась.

# Заключение

В данной работе было исследовано влияние разнообразных алгоритмов сэмплирования на обучение по плану на различных задачах обработки естественного языка. Был придуман и реализован широкий спектр алгоритмов сэмплирования, который обладают различными свойствами и показывают отличающиеся друг от друга результаты на задачах предобучения, классификации и машинного перевода. Было показано, что все алгоритмы сэмплирования на всех задачах не могут превзойти стандартный алгоритм обучения, аналогично работе Wu [23] в компьютерном зрении. Причем поведение на различных задачах отличается, например, на предобучении обучении по плану существенно проигрывает для всех алгоритмов сэмплирования, однако, на задачах классификации и машинного перевода различий почти нет. На основании существующих статей была выдвинута гипотеза, о важности правильной токенизации для обучения по плану, однако, в результате проверки оказалось, что гипотеза ошибочна.

Изначально, обучение по плану пришло из области обучения с подкреплением, где использовалось как способ хоть как-то научиться решать задачу, а постепенное усложнение происходило только после того, как модель научилась более простую задачу. В обработке естественного языка, как и в компьютерном зрении подход к обучению по плану видоизменился, однако, результаты на обыкновенных задачах получились довольно спорные.

Главный вывод данной работы состоит в том, что на стандартных задачах обработки естественного языка обучение по плану не помогает. В качестве возможных направлений дальнейших исследований в данной области стоит выделить поиск конкретных задач обработки естественного языка, которые могут более эффективно решаться при помощи обучения по плану.

# Список литературы


[1] Attention is all you need / Ashish Vaswani, Noam Shazeer, Niki Parmar et al. // arXiv preprint arXiv:1706.03762. — 2017.

[2] Bert: Pre-training of deep bidirectional transformers for language understanding / Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova // arXiv preprint arXiv:1810.04805. — 2018.

[3] Bostrom Kaj, Durrett Greg. Byte pair encoding is suboptimal for language model pretraining // arXiv preprint arXiv:2004.03720. — 2020.

[4] Competence-based curriculum learning for neural machine translation / Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig et al. // arXiv preprint arXiv:1903.09848. — 2019.

[5] Curriculum learning / Yoshua Bengio, Jérôme Louradour, Ronan Collobert, Jason Weston // Proceedings of the 26th annual international conference on machine learning. — 2009. — P. 41–48.

[6] Curriculum learning for natural language understanding / Benfeng Xu, Licheng Zhang, Zhendong Mao et al. // Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. — 2020. — P. 6095–6104.

[7] Curriculum learning for reinforcement learning domains: A framework and survey / Sanmit Narvekar, Bei Peng, Matteo Leonetti et al. // Journal of Machine Learning Research. — 2020. — Vol. 21, no. 181. — P. 1–50.

[8] Domain-specific language model pretraining for biomedical natural language processing / Yu Gu, Robert Tinn, Hao Cheng et al. // arXiv preprint arXiv:2007.15779. — 2020.

[9] Exploring the limits of transfer learning with a unified text-to-text transformer / Colin Raffel, Noam Shazeer, Adam Roberts et al. // arXiv preprint arXiv:1910.10683. — 2019.

[10] Gage Philip. A new algorithm for data compression // C Users Journal. — 1994. — Vol. 12, no. 2. — P. 23–38.

[11] Hacohen Guy, Weinshall Daphna. On the power of curriculum learning in training deep networks // International Conference on Machine Learning / PMLR. — 2019. — P. 2535–2544.

[12] Hochreiter Sepp, Schmidhuber Jürgen. Long short-term memory // Neural computation. — 1997. — Vol. 9, no. 8. — P. 1735–1780.

[13] HuggingFace's Transformers: State-of-the-art natural language processing / Thomas Wolf, Lysandre Debut, Victor Sanh et al. // arXiv preprint arXiv:1910.03771. — 2019.

[14] Kocmi Tom, Bojar Ondrej. Curriculum learning and minibatch bucketing in neural machine translation // arXiv preprint arXiv:1707.09533. — 2017.

[15] Kudo Taku. Subword regularization: Improving neural network translation models with multiple subword candidates // arXiv preprint arXiv:1804.10959. — 2018.

[16] Language models are few-shot learners / Tom B Brown, Benjamin Mann, Nick Ryder et al. // arXiv preprint arXiv:2005.14165. — 2020.

[17] Narvekar Sanmit. Curriculum Learning in Reinforcement Learning. // IJCAI. — 2017. — P. 5195–5196.

[18] Pre-trained models for natural language processing: A survey / Xipeng Qiu, Tianxiang Sun, Yige Xu et al. // Science China Technological Sciences. — 2020. — P. 1–26.

[19] Revisiting pre-trained models for chinese natural language processing / Yiming Cui, Wanxiang Che, Ting Liu et al. // arXiv preprint arXiv:2004.13922. — 2020.

[20] Schuster Mike, Nakajima Kaisuke. Japanese and korean voice search // 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) / IEEE. — 2012. — P. 5149–5152.

[21] Soviany Petru. Curriculum Learning with Diversity for Supervised Computer Vision Tasks // arXiv preprint arXiv:2009.10625. — 2020.

[22] Wang Shuohang, Jiang Jing. Learning natural language inference with LSTM // arXiv preprint arXiv:1512.08849. — 2015.

[23] Wu Xiaoxia, Dyer Ethan, Neyshabur Behnam. When Do Curricula Work? // arXiv preprint arXiv:2012.03107. — 2020.

[24] Zaremba Wojciech, Sutskever Ilya. Learning to execute // arXiv preprint arXiv:1410.4615. — 2014.

[25] A fully progressive approach to single-image super-resolution / Yifan Wang, Federico Perazzi, Brian McWilliams et al. // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. — 2018. — P. 864–873.

[26] A unifying framework for complexity measures of finite systems / Nihat Ay, Eckehard Olbrich, Nils Bertschinger, Jürgen Jost // Proceedings of ECCS / Citeseer. — Vol. 6. — 2006.