Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

(УНИВЕРСИТЕТ ИТМО)

Научно исследовательская работа на тему

“Нормализация текста для задачи TTS”

Выполнил: Кучер Максим

Санкт-Петербург 2023

Оглавление

1. Введение
2. Анализ данных
3. Обзор и изучение решение
4. Обзор выбранного подхода (обзор выбранного подхода и результат )
5. Заключение
6. Список использованных источников
7. **Введение**

Предварительная обработка текста и нормализация являются важными методами, используемыми в обработке естественного языка (NLP) для подготовки текстовых данных, прежде чем их можно будет использовать для анализа и моделирования. В этом процессе необработанные текстовые данные преобразуются в структурированный формат, который может быть легко понят алгоритмом машинного обучения.

Основная цель предварительной обработки текста — удалить любую ненужную информацию, стандартизировать формат текста и обеспечить его готовность к дальнейшему анализу. Далее пойдет речь о самых распространенных методов, используемые при предварительной обработке текста.

Задача сегментации на предложения с учетом сокращений в составе нормализации текстов является одной из важнейших и заключается в разделении текста на отдельные предложения, учитывая наличие сокращений внутри предложений. Например, если текст содержит сокращение «Mr.» в середине предложения, то задача заключается в определении, является ли это сокращение концом предложения или частью имени.

Токенизация представляет собой разбиение текстовых данных на отдельные слова или фразы, называемые токенами. Токенизация важна, потому что она позволяет алгоритму машинного обучения анализировать каждое слово или фразу отдельно. Другими словами генерирует признаковое пространство.

Для задачи текст в речь (TTS) на уровне нормализации, регистр и предлоги имеют большое значение. Регистр может оказаться важным, потому что в некоторых языках, например, в английском, он может изменять смысл слова. Например, слова «cat» (кошка) и «Cat» (аббревиатура от «Computerized Axial Tomography») имеют разное значение, и речевой движок должен уметь различать их в контексте. Предлоги также имеют большое значение, потому что они могут изменять смысл всего предложения. Например, в предложениях «Я говорю с ним» и «Я говорю о нем» предлог «с» и «о» имеют разное значение, и речевой движок должен правильно интерпретировать их. Таким образом, при нормализации текста для задачи TTS, важно учитывать регистр и предлоги, чтобы генерированный речевой выход был точным и понятным для слушателя.

Задачу нормализации можно поделить на несколько подзадач, которые могут различаться в зависимости от конкретного используемого языка и диалекта. Некоторые из общих подзадач нормализации текста для TTS включают в себя:

* Расширение слов: в TTS важно расширять аббревиатуры и сокращения, чтобы гарантировать их правильное произношение. Например, аббревиатура «ин-т» следует расширить до «институт», а аббревиатуру «***АО***» следует расширить до «***акционерное общество***». Есть и случае по сложнее когда “40 км2” должно быть вербализовано не как “сорок километров два” и даже не как “сорок километров в квадрате” а как “сорок квадратных километров”.
* Нормализация чисел: системы TTS должны учитывать правильное согласование формы для количественных, порядковых и римских числительных а также обрабатывать числа таким образом, чтобы они звучали естественно при произнесении вслух. Например, число «123» может быть расширено до «сто двадцать три» или «один-два-три» в зависимости от контекста или желаемого стиля речи. Таким образом «Начинаем на счет 123» должно быть преобразовано в «Начинаем на счет раз два три» а текст «Я живу на проспекте Энгельса 123» в «Я живу на проспекте Энгельса сто двадцать три».
* Нормализация даты, времени, телефонных номеров: даты и время должны быть нормализованы таким образом, чтобы их можно было произнести вслух. Например, дату «15.02.2023» можно произнести как «Пятнадцатое февраля, двадцать третьего года». Сложности тут могут возникнуть с разным форматами написания дат, например в других регионах используется различное расположение дня и месяца, месяц.день.год или вовсе год.месяц.день. Также есть задача правильно вербализировать телефонные номера. Система должна определять либо по специфике числа либо по его контексту что «+79123423454» это номер телефона и произнести каждую цифру отдельно.
* Устранение неоднозначности омографов: омографы — это слова, которые пишутся одинаково, но имеют разное значение и произношение. Например, слово «замок» может означать жилую и оборонительную постройку в которой жили феодалы прежних веков или устройство фиксации для запирания дверей. Системы TTS должны использовать контекст для устранения неоднозначности омографов и выбора правильного произношения.
* Нормализация сокращений - это процесс преобразования сокращенных форм слов или выражений в их полные формы. В тексте могут использоваться различные сокращения, такие как "др." (друг), «доктор», «т.д.» (и так далее), «г.» (год), «стр.» (страница) и т.д. Однако при использовании этих сокращений в речи или при распознавании текста могут возникнуть трудности, так как они не всегда однозначно интерпретируются. Нормализация сокращений включает в себя анализ контекста, в котором они используются, и преобразование их в соответствующие полные формы. Например, сокращение «др.» может быть интерпретировано как «друг» или «доктор», в зависимости от контекста, в котором оно используется. Поэтому при нормализации сокращений необходимо учитывать контекст и выбирать наиболее подходящую полную форму.
* Нормализация специальных символов таких как например знак доллара, который обычно стоит перед числом, а должен произносится после. Например «$20» должен быть вербализован как «двадцать долларов» а не как «доллар двадцать».

Ниже приведены еще примеры сложных для распознавания фраз;

* «4е» — «четвертый(ые)», но «д. 4е» — «четыре е»;
* «длиной до 4 км» — «длиной до четырех километров», но «будем идти до 4 км» — «будем идти до четвертого километра»;
* «4 части» — четыре части, но «нет 4 части» — «нет четвертой части»;
* «равно 4/9» — «равно четыре девятых», но «д. 4/9» — «дом четыре дробь девять» или «дом четыре пять»;
* «1949 г» — «тысяча девятьсот сорок девятый год», но «г Санкт-Петербург» — «город Санкт-Петербург». Буква «г» вербализуется по разному.

Для решения задачи нормализации текста в TTS и STT был разработан ряд архитектур и моделей. Одними из первых систем была система преобразования текста в речь MITalk, разработанная еще в 1987 году, подробнее в статье [1]. Системы TTS того времени были полностью основана на правилах. Еще пример ранних систем это Многоязычная TTS-система Bell Labs, подробнее в статье [2]. В ней уже использовались веса для перехода конечное число состояний. Данный подход используется до сих пор в некоторых актуальных системах TTS. Все подходы к решению задачи можно поделить на несколько подмножеств.

1. Системы, основанные на правилах

Системы на основе правил — это простой и эффективный подход к нормализации текста, который включает использование созданных вручную правил для распознавания и нормализации различных типов входного текста. Эти правила обычно создаются экспертами в предметной области и основаны на определенных шаблонах входного текста. Например, система, основанная на правилах, может заменять легко заменять стандартные аббревиатуры их полными формами. Системы на основе правил просты в разработке и могут быть эффективны при обработке определенных типов вводимого текста, но их создание может занимать много времени, и они могут плохо обобщаться на новый или неожиданный ввод. Их основной минус в том что они не могут самостоятельно дообучаться, что является значительным недостатком в реалиях постоянно развивающегося языка. Над такого рода системами нужна постоянная работа экспертов.

1. Статистические модели

Статистические модели включают использование вероятностных подходов для изучения закономерностей во входном тексте и прогнозирования правильного результата нормализации. Эти модели обычно обучаются на больших объемах аннотированных текстовых данных и могут быть эффективны при обработке широкого диапазона входного текста. Существуют различные статистические модели, которые использовались для нормализации текста, в том числе скрытые марковские модели (HMM). Одним из распространенных подходов является использование модели маркировки последовательностей, такой как условное случайное поле (CRF), для прогнозирования нормализованного вывода для каждого слова во входном тексте. Однако статистические модели могут быть сложными и дорогостоящими в вычислительном отношении для обучения, и они не всегда могут давать точные результаты.

1. Алгоритмы машинного обучения

Алгоритмы машинного обучения, такие как деревья решений, машины опорных векторов (SVM) и нейронные сети, также использовались для нормализации текста в TTS и STT. Эти алгоритмы способны изучать сложные шаблоны во входном тексте и могут хорошо обобщать новые и невидимые данные. В частности, в последние годы методы глубокого обучения показали многообещающие результаты с использованием рекуррентных нейронных сетей (RNN), сверточных нейронных сетей (CNN) и моделей на основе преобразователя.

RNN — это тип нейронной сети, который хорошо подходит для задач моделирования последовательностей, таких как нормализация текста. Одним из распространенных подходов является использование двунаправленной сети с долговременной кратковременной памятью (BiLSTM), которая может фиксировать контекст каждого слова во входном тексте. Другой подход заключается в использовании модели «последовательность к последовательности», такой как архитектура кодер-декодер, которая может генерировать нормализованный вывод непосредственно из входного текста.

CNN — это еще один тип нейронной сети, который можно использовать для нормализации текста. Один из подходов заключается в использовании CNN на уровне символов, которая может изучать представления отдельных символов во входном тексте. Это может быть полезно для обработки входного текста, содержащего слова с ошибками или нестандартные сокращения.

Трансформеры — это тип нейронной сети, который использовался для ряда задач NLP, включая нормализацию текста. Одной из популярных моделей на основе преобразователя является модель BERT (представления двунаправленного кодировщика от преобразователей), которая может изучать контекстуализированные представления слов во входном тексте. Другой подход заключается в использовании модели последовательностей на основе преобразователя, такой как модель GPT (Generative Pre-trained Transformer), которая может генерировать нормализованные выходные данные непосредственно из входного текста.

Рассмотрим примеры актуальных архитектур которые решают основанные на вышеперечисленных подходах:

**Sequitur-G2P** описанный в статье [3] - это статистический подход к преобразованию графемы в фонему (G2P), который широко используется в системах TTS. Преобразование G2P - это процесс преобразования последовательности графем (букв и символов) в последовательность фонем (наименьших единиц звука в языке). Преобразование G2P важно для TTS, поскольку оно позволяет системе генерировать естественно звучащую речь из текста. Sequitur-G2P основан на модели преобразования последовательности, которая учится преобразовывать орфографическое представление слов в их фонематическую транскрипцию. Он был разработан Novak et al. в 2012 году и является расширением исходного алгоритма Sequitur, который представляет собой иерархическую статистическую модель для предсказания последовательности. Алгоритм Sequitur-G2P состоит из двух основных компонентов: компонента обучения и компонента декодирования. Во время обучения для изучения статистической модели используется большой набор данных сопоставлений слов и фонем. Модель обучена прогнозировать последовательность фонем по последовательности графем для каждого слова в наборе данных. Процесс декодирования включает в себя поиск наиболее вероятной последовательности фонем с учетом последовательности графем входного слова. Это делается путем вычисления условной вероятности каждой возможной последовательности фонем с учетом входной последовательности графем и выбора последовательности с наибольшей вероятностью.

Sequitur-G2P имеет несколько преимуществ по сравнению с другими подходами G2P. Он способен обрабатывать невидимые слова и редкие слова с высокой точностью благодаря своей иерархической модели, которая может обобщать новые слова на основе их подслов. Он также не зависит от языка, что означает, что его можно применять к любому языку без необходимости использования специфичных для языка функций или ресурсов. Однако Sequitur-G2P также имеет некоторые ограничения. Одним из главных ограничений это ее неспособность нормализовывать специальные символы и численные сокращения. Также система может быть дорогостоящим в вычислительном отношении, особенно во время декодирования, и может потребовать больших объемов обучающих данных для достижения высокой точности. Это также может привести к противоречивым результатам для определенных слов или языков из-за статистического характера модели.

**Phonetisaurus** — это еще один статистический подход к преобразованию графемы в фонему (G2P). Подробно о нем можно почитать в статье [4]. Phonetisaurus использует взвешенный преобразователь с конечным состоянием (WFST) для преобразования G2P. WFST — это математическая модель, которая представляет набор пар вход-выход в виде ориентированного графа. В случае с Phonetisaurus входом является последовательность графем, а выходом — последовательность фонем для данного слова. Модель обучается на большом наборе данных сопоставлений слов и фонем с использованием статистических методов для определения оптимального набора весов для WFST. В отличие от Sequitur-G2P поиск включает в себя обход WFST с учетом весов, связанных с каждым переходом, для нахождения оптимального пути по графу.

Одним из ключевых преимуществ Phonetisaurus является его гибкость и модульность. Это позволяет включать в модель G2P различные языковые функции и ресурсы, такие как словари произношения и лингвистические правила. Это делает его полезным для работы с языками со сложными фонологическими правилами или для разработки языковых моделей для систем TTS. Также как и Sequitur-G2P, Phonetisaurus не умеет работать с специальными символами и числительными сокращениями.

В **Google Kestrel TTS** (подробнее в статье [5]) нормализация текста выполняется также с использованием комбинации подходов на основе правил и данных. То что отличает ее от других это то что в этой системе реализована разбиение токенов на классы. Входные данные маркируются в соответствии с их смысловой составляющей и объединяются в токены больше чем одно слово. Например дата «13 декабря 1998» будет представляться в системе как дата { день: «13», месяц: «декабрь», год: «1998» } после чего уже данное представление вербализуется в форму «тринадцатое декабря тысяча девятьсот девяносто восьмой год». Данный вид токенизации с классификацией также помогает решить проблему с порядком слов. Например в случае с выражением «$13» Kestrel TTS токенизатор представит его в таком виде: деньги { валюта: «USD», сумма { целое число: «13» } и в процессе вербализации расположение слов будет правильным «тринадцать долларов».

В 2018 была выпущена **Tacotron 2** (подробнее в статье [6]) это современная нейросетевая система TTS, которая пока не работает для русского языка. Она состоит из двух архитектур. Одна из них использует технологию трансформных сетей с механизмом вниманием, которая предсказывает последовательность кадров спектрограммы из входной последовательности символов. Вторая часть это модифицированная версия WaveNet, которая генерирует выборки формы волны во временной области в зависимости от предсказанных кадров спектрограммы. Подробнее о ней можно почитать в статье [5]. Она включает этап предварительной обработки для нормализации текста.

В Tacotron 2 нормализация текста выполняется с использованием комбинации методов, основанных на правилах, и методов, управляемых данными. Методы на основе правил включают предварительную обработку входного текста для обработки распространенных сокращений, сокращений и других нарушений в тексте.

Методы основанные на данных в Tacotron 2 включают использование глубокой нейронной сети с несколькими слоями ячеек долговременной кратковременной памяти (LSTM). Одним из преимуществ подхода Tacotron 2 к нормализации текста является его способность обрабатывать сложный вводимый текст, такой как текст на нескольких языках. Архитектура сети показана на рисунке 3.1

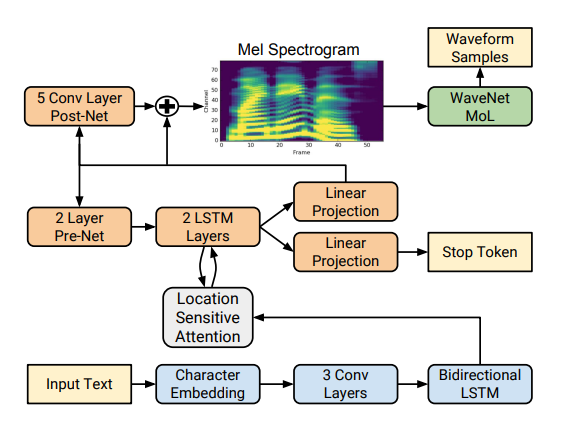


рис 3.1 Блок-схема системной архитектуры Tacotron 2.

На выходе получилась система способная отличать омонимию на основе семантики. Например слово «desert» будет восприниматься как существительное или глагол именно в зависимости от контекста, также архитектура будет по разному воспринимать смысл слова «read» даже если оно во всех формах пишется одинаково. Tacotron2 также умеет делать паузы в случае знаков препинания и акценты на словах записанных заглавными.

**2) Анализ данных**

Для работы был выбран датасет из kaggle соревнования [13] Text Normalization Challenge - Russian language. Он содержит 761424 предложений и 10574516 токенов. Датасет составлен из следующих колонок: «sentence\_id» которое является id предложения, «token\_id» это id токена относительно предложения, с помощью него можно восстановить порядок слов в предложении; «class» это столбец в котором хранятся классы токенов, некоторые из классов это: «PLAIN» - токены которые обозначают собой обычные слова, «PUNCT» - токены обозначающие знаки препинания, «LETTERS» - токены обозначающие буквы; Главные два столбца это Столбец «before» который содержит необработанный токен и столбец «after» который содержит нормализованный текст. На рисунке 2.1 пример предложений из набора данных.

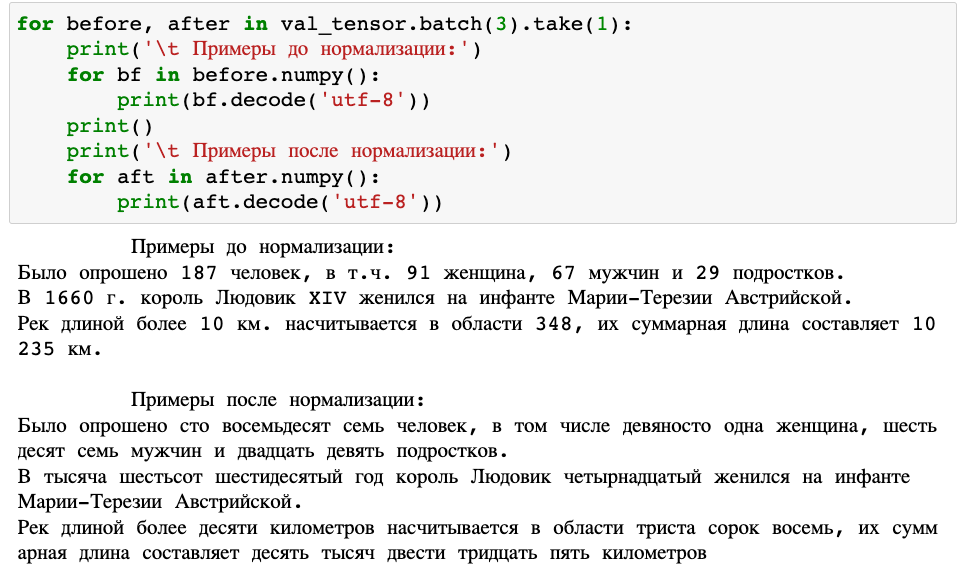


рис 2.1 пример предложений из набора данных

Для работы с моделью нужно разметить текст так, чтобы каждый элемент был представлен в виде токена или идентификатора токена (числовое представление). Токенизация — это процесс разбиения текста на «токены». В зависимости от токенизатора эти токены могут представлять части предложений, слова, подслова или символы. Был написан класс CustomTokenizer со схожим с Токенизатором tensorflow функционалом и настроен на основе нашего набора слов. Его можно использовать для того чтоб получать численные представления слов. На рисунке 2.2 пример предложений после токенизации.

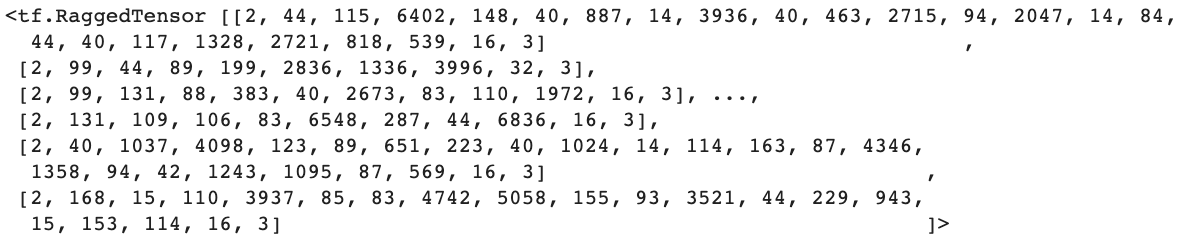


рис 2.2 пример предложений после токенизации

**3) Обзор и изучение решение**

Очевидной идеей кажется представить задачу нормализации текста для TTS и STT в виде задачи последовательность к последовательности (seq2seq), также называемой машинным переводом, так как буквально нам нужно перевести текст на входе в другой частично состоящий из других символов но сохранив семантику исходного текста.

В сфере последовательного машинного перевода (Seq2Seq) за последние годы был достигнут огромный прогресс. Модели Seq2Seq состоят из двух основных компонентов: кодера и декодера. Кодер берет ≠входную последовательность и генерирует представление фиксированной длины (часто называемое вектором контекста), которое декодер использует для создания выходной последовательности. В последние годы подход Seq2Seq претерпел революцию благодаря использованию моделей трансформеров, которые стали последним словом техники в машинном переводе.

Трансформеры — это тип архитектуры нейронной сети, который впервые был представлен в статье Attention Is All You Need 2017-г года [7]. Основная идея преобразователя заключается в использовании механизма внутреннего внимания (self-attention) для захвата глобальных зависимостей между словами в предложении. В отличие от рекуррентных нейронных сетей (RNN), которые обрабатывают входные данные последовательно, преобразователи могут обрабатывать целые последовательности параллельно. Это делает их более эффективными и позволяет им захватывать более дальние зависимости.

Чтобы применить архитектуру трансформера к машинному переводу, исследователи адаптировали платформу Seq2Seq для использования преобразователей вместо RNN в качестве кодировщика и декодера. Кодер-преобразователь принимает входную последовательность и генерирует последовательность скрытых состояний, которая фиксирует значение входных данных. Затем декодер-преобразователь использует эти скрытые состояния и внутреннее внимание для генерации выходной последовательности слово за словом. Архитектура трансформера показана на рисунке 3.1

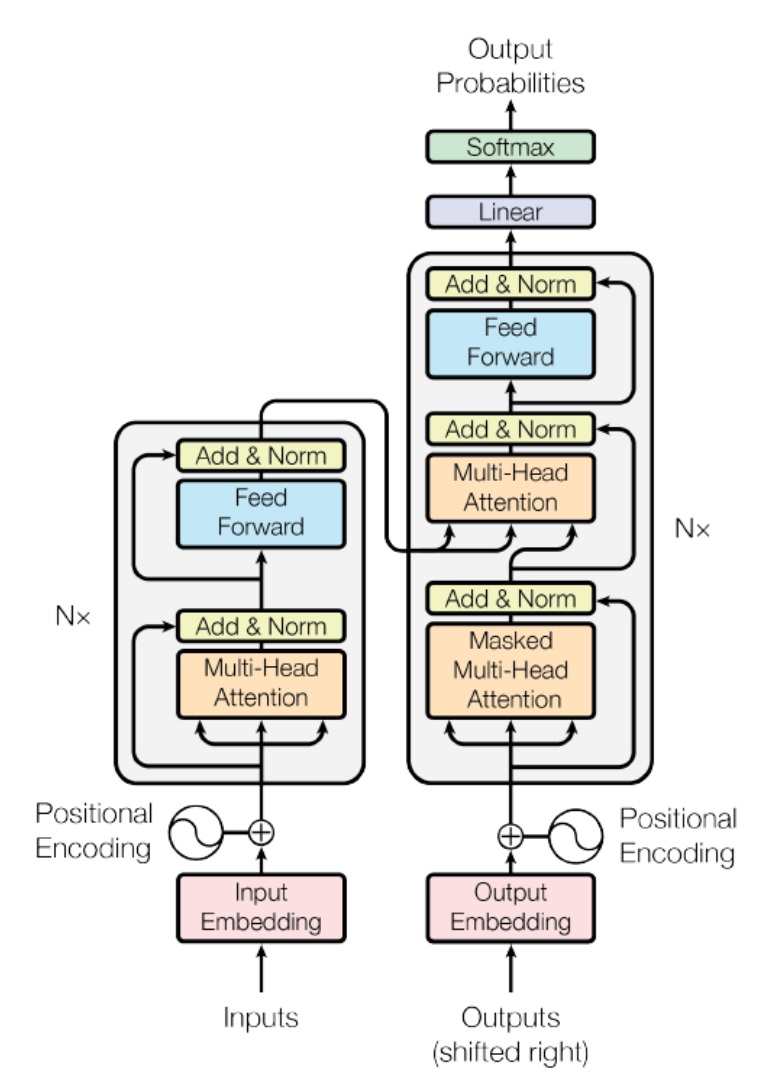


рис 3.1 Архитектура трансформера из [7] Attention Is All You Need

Подход Seq2Seq на основе трансформатора имеет несколько преимуществ по сравнению с предыдущими методами. Одним из наиболее значительных преимуществ является то, что преобразователи лучше захватывают дальние зависимости, чем RNN. Это означает, что они могут производить более точные переводы для более длинных предложений. Кроме того, трансформеры могут научиться более эффективно согласовывать исходные и целевые предложения, что может улучшить общее качество перевода. Еще одно преимущество подхода на основе преобразователя заключается в том, что он лучше распараллеливается, чем предыдущие методы, что позволяет быстрее проводить обучение и логические выводы.

Несмотря на свои преимущества, подход Seq2Seq на основе трансформатора не лишен недостатков. Одна из основных проблем заключается в том, что преобразователям требуется большое количество обучающих данных для достижения самых современных характеристик. Это связано с тем, что преобразователи имеют большое количество параметров и требуют большого количества данных, чтобы избежать переобучения. Кроме того, обучение трансформаторов может быть дорогостоящим в вычислительном отношении и требовать мощного оборудования для достижения современной производительности. Впрочем это общая проблема всех вышеперечисленных подходов.

Таким образом, подход Seq2Seq, основанный на преобразовании, стал передовым в машинном переводе благодаря его способности фиксировать долгосрочные зависимости и производить высококачественные переводы. Хотя у него есть свои проблемы, исследователи продолжают совершенствовать и совершенствовать этот подход, и, вероятно, он останется важной областью исследований NLP на долгие годы.

В контексте нашей задачи трансформерная модель отвечает всем требованиям. Во первых он умеет преобразовывать текстовые последовательности в другие текстовые последовательности, во вторых он умеет учитывать контекст и в третьих он сам подбирает правила преобразования учитывая статистику. Поэтому будем тестировать именно данную модель.

**4) Обзор выбранного подхода**

Для тестирования была выбрана модель преобразования последовательности в последовательность для нормализации текста на русском языке. Трансформатор был первоначально предложен в статье [7] Attention Is All You Need. Для работы была выбрана библиотека TensorFlow 2 - это открытая платформа для машинного обучения и глубокого обучения, созданная Google.

Наша задача похоже на задачу генерации текста, за исключением того, что здесь у вас есть дополнительный входной «контекст» (текст до нормализации), на котором модель «обусловлена».

Эта настройка называется «принуждение учителя», потому что независимо от выходных данных модели на каждом временном шаге она получает истинное значение в качестве входных данных для следующего временного шага. Это простой и эффективный способ обучения модели генерации текста. Это эффективно, потому что вам не нужно запускать модель последовательно, выходные данные в разных местах последовательности могут быть вычислены параллельно.

Таким образом можно обучить модель. Вам нужно будет написать цикл вывода и передать выходные данные модели обратно на вход. Это медленнее (временные шаги не могут выполняться параллельно) и более сложная задача для изучения (модель не может правильно воспроизвести конец предложения, пока не получит правильное начало), но может дать более стабильную модель, поскольку модель должна научиться исправлять собственные ошибки во время обучения.

Наша трансформерная сеть состоит из четырех слоев, общую структуру слоя можно наблюдать на рисунке 4.1.

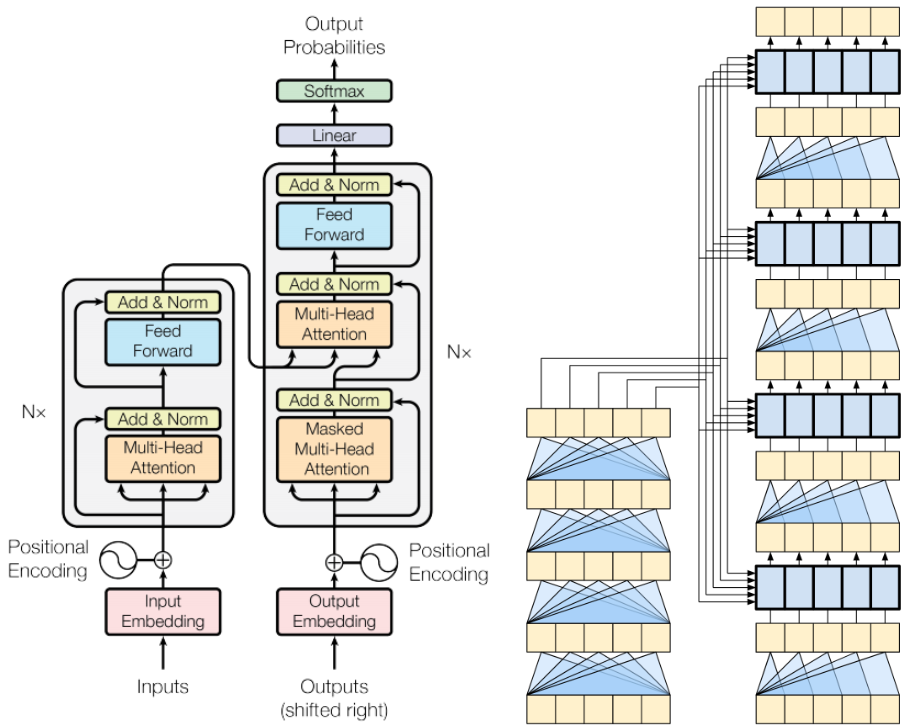


рис 4.1 Общая структура сети

Далее были написаны компоненты сети.

* Слой эмбединга и позиционного кодирования рисунок 4.2

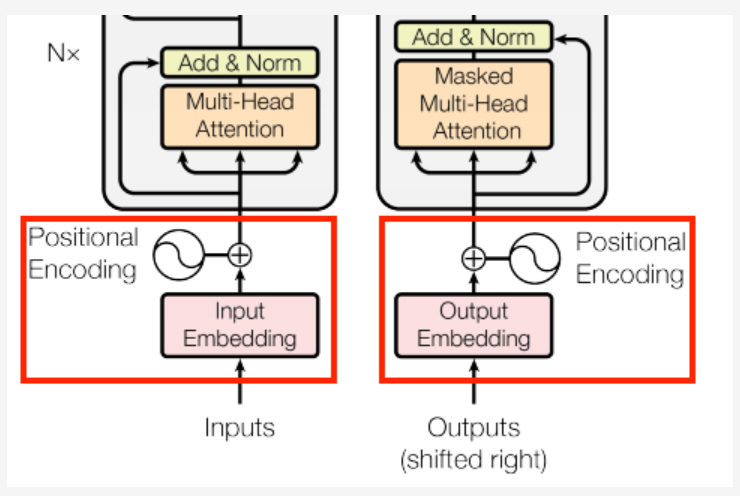


рисунок 4.2 Слой эмбединга и позиционного кодирования

Учитывая последовательность токенов, как входные токены, так и целевые токены должны быть преобразованы в векторы с использованием слоя tf.keras.layers.Embedding.

Слои внимания, используемые во всей модели, воспринимают свои входные данные как набор векторов без порядка. Поскольку модель не содержит повторяющихся или сверточных слоев. Ему нужен какой-то способ определить порядок слов, иначе он увидит входную последовательность как один экземпляр

### Нормализация (Add and Norm) рисунок 4.3

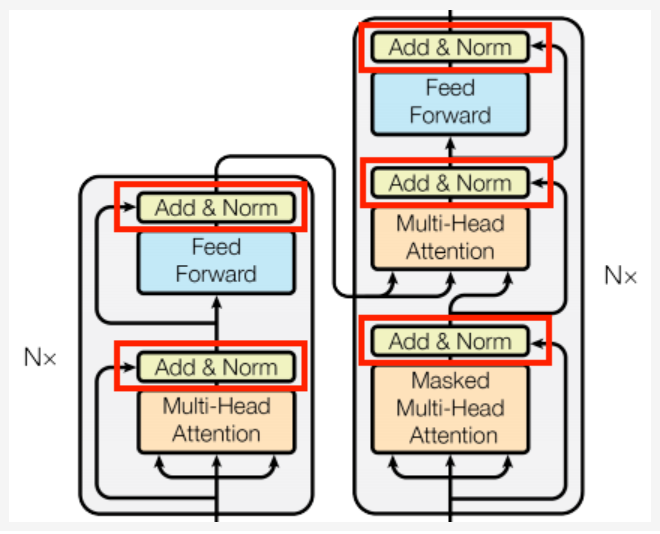


рисунок 4.3 Нормализация (Add and Norm)

Эти блоки "Add & Norm" разбросаны по всей модели. Каждый из них присоединяется к остаточному соединению и пропускает результат через слой «LayerNormalization».

Проще всего организовать код вокруг этих остаточных блоков. В следующих разделах будут определены пользовательские классы слоев для каждого из них.

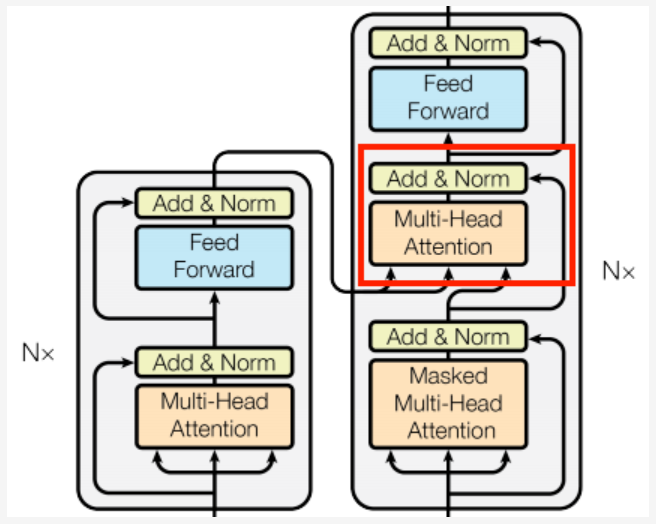
Остаточные блоки "Add & Norm" включены, чтобы обучение было эффективным. Остаточное соединение обеспечивает прямой путь для градиента (и гарантирует, что векторы **обновляются** слоями внимания, а не **заменяются**), в то время как нормализация поддерживает разумный масштаб выходных данных.

Примечание. Реализации, приведенные ниже, используют слой «Добавить», чтобы обеспечить распространение масок Keras (оператор «+» — нет).

### Базовый слой внимания рисунок 4.4

Слои внимания используются во всей модели. Все они идентичны, за исключением того, как настроено внимание. Каждый из них содержит layers.MultiHeadAttention, layers.LayerNormalization и layers.Add.

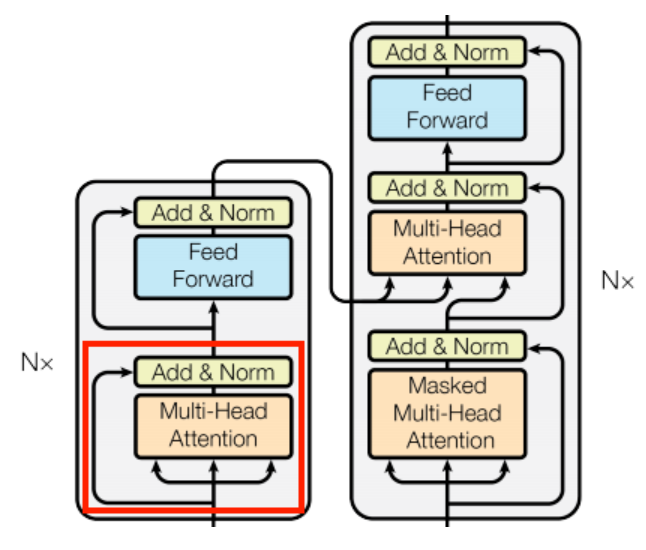
### Слой перекрестного внимания рисунок 5.6



### рис 4.4 Слой перекрестного внимания

Этот уровень соединяет кодер и декодер. Этот слой является наиболее простым способом использования внимания в модели.

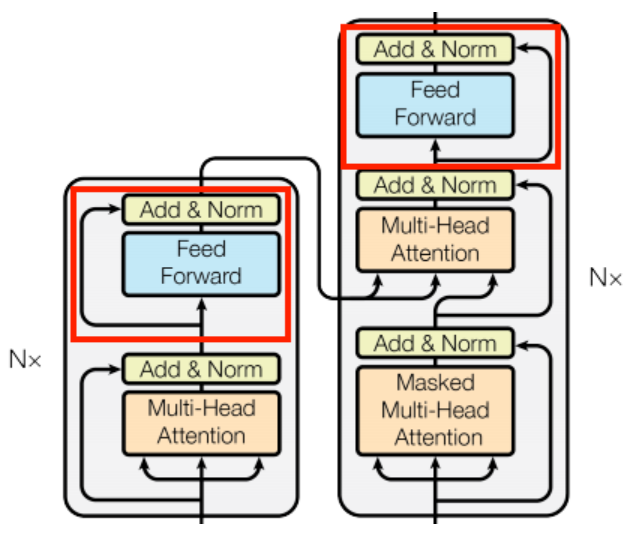
### Слой глобального self-attention рисунок 4.5



### рис 4.5 Слой глобального self-attention

Этот слой отвечает за обработку контекстной последовательности и распространение информации по ее длине.

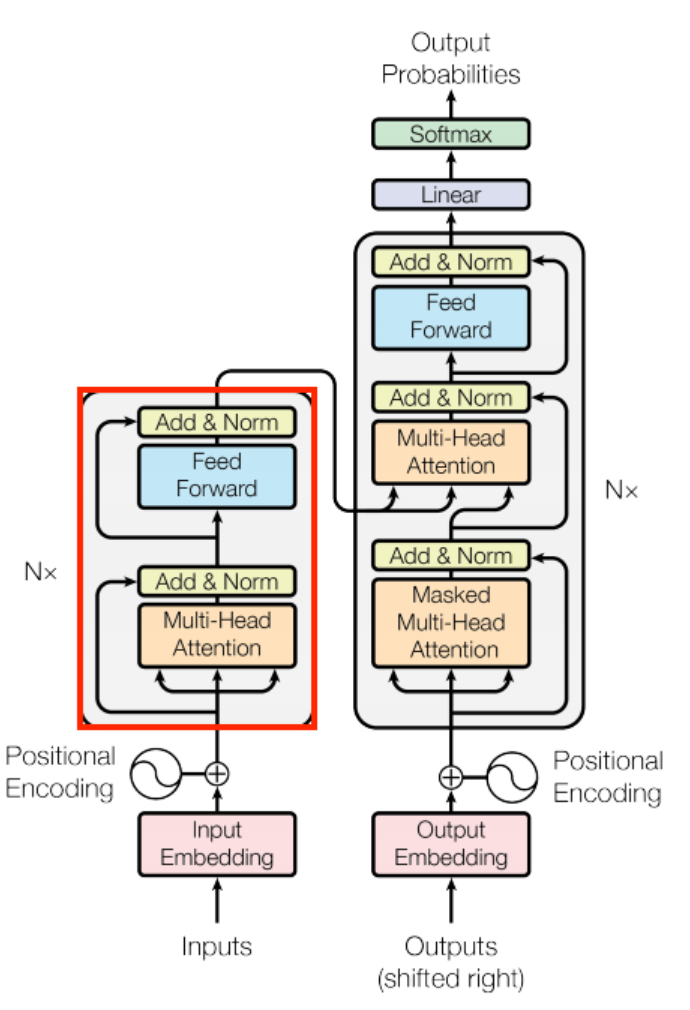
### Сеть прямого распространения рисунок 4.6



### рисунок 4.6 Сеть прямого распространения

Сеть состоит из двух линейных слоев (tf.keras.layers.Dense) с промежуточной активацией ReLU и выпадающего слоя. Как и в случае со слоями внимания, код здесь также включает остаточное соединение и нормализацию.

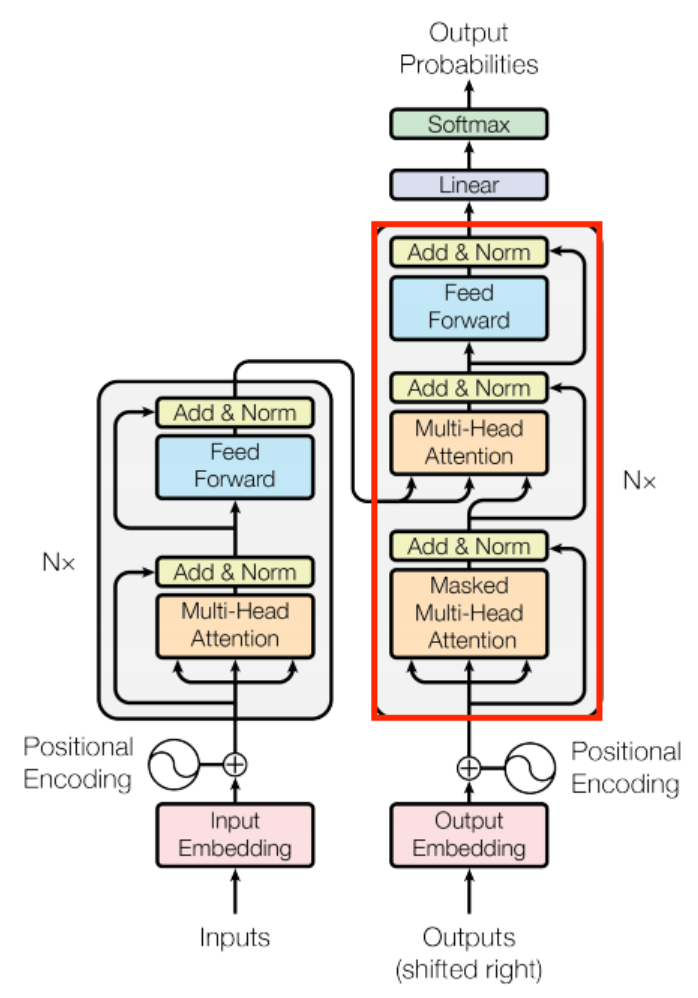
### Слой кодировщика рисунок 4.7



### рис 4.7 Слой кодировщика

Кодер содержит стек из N слоев кодировщика. Где каждый EncoderLayer содержит слой GlobalSelfAttention и FeedForward. Кодер состоит из: Слой PositionalEmbedding на входе и стек слоев EncoderLayer.

### Слой декодера рисунок 4.8



### рис 4.8 Слой декодера

Стек декодера немного сложнее: каждый DecoderLayer содержит CausalSelfAttention, CrossAttention и FeedForward.

Для тренировки сети был использован оптимизатор Adam с настраиваемым планировщиком скорости обучения в соответствии с формулой оригинального Transformer. В качестве функции потерь был использован tf.keras.losses.SparseCategoricalCrossentropy. Другие гипер параметры обучения это: размер пакета (batch\_size) = 64 и количество эпох = 20.

Для тестирования был использован датасет из 187 предложений, в каждом из которых есть числа, даты, сокращения. В роли эталона был использован нормализованный человеком датасет. Для оценки качества алгоритма была использована, классическая для seq2seq задачи, метрика BLEU.

BLEU (Bilingual Evaluation Understudy) - это метрика, которая используется для оценки качества машинного перевода. BLEU была разработана в 2002 году и стала одной из самых популярных метрик для оценки качества перевода.

Принцип работы BLEU основывается на том, что она сравнивает сгенерированный текст с эталонным переводом и определяет, насколько хорошо сгенерированный текст соответствует эталонному. Эталонный перевод обычно представляет собой человеком выполненный перевод или же какой-то другой текст, который является эталоном для сравнения.

BLEU работает следующим образом:

1. Разбивает сгенерированный текст и эталонный перевод на отдельные слова и создает список слов для каждого.
2. Сравнивает список слов сгенерированного текста с каждым списком слов эталонного перевода.
3. Вычисляет точность совпадений слов сгенерированного текста с эталонным переводом.
4. Для каждого слова в списке слов сгенерированного текста находит наиболее близкое по смыслу слово в эталонном переводе.
5. Вычисляет точность совпадений этих близких слов в списке сгенерированного текста и эталонного перевода.
6. Суммирует точности совпадений для всех слов в списке и вычисляет итоговый показатель BLEU.
7. Повторяет шаги 1-6 для каждого сгенерированного текста и сравнивает его с соответствующим эталонным переводом.
8. Среднее значение BLEU для всех сравнений считается показателем качества перевода.

BLEU может иметь значения от 0 до 1, где 1 означает, что сгенерированный текст и эталонный перевод идентичны, а 0 - что они не имеют никаких совпадений.

**5) Заключение**

Для того чтоб более наглядно понять насколько хорошо справилась с задачей наша модель, метрика BLEU была посчитана также и для не преобразованного датасета, для того чтоб использовать данную оценку как бейзлайн. В итоге для не преобразованного датасета получили BLEU = 0.06988, для преобразованного нашей моделью датасета получили BLEU = 0.07218. Стоит отметить что результат модели лучше чем бейзлайн. Также после просмотра выхода модели вручную, было замечено что модель чаще всего ошибается с вербализацией сокращений, что скорее всего можно отнести к нехватке данных для обучения, так как самые популярные аббревиатуры все же преобразуются без явных проблем.

В заключении можно отметить что архитектура трансформерной сети может быть успешно использована для нормализации текста в задаче TTS. Она хорошо справляется с нормализацией чисел но плохо работает с сокращениями, например «г.» всегда преобразовывается в «год», но иногда «г.» это сокращение от слова «город». Нужно увеличить выборку для того чтоб модель лучше справлялась с различными сокращениями. Еще одна сложность это сокращение имён, она должна понимать что «А. С. Пушкин» это «Александр Сергеевич Пушкин» а не «Алексей Сергеевич Пушкин».

**6) Список использованных источников**

1. Аллен, Джонатан, Шарон М. Ханникатт, Деннис Клатт.1987 г..*От текста к речи: система MITalk* // Издательство Кембриджского университета, Кембридж.
2. Sproat, Richard, editor. 1997. *Multilingual Text-to-Speech Synthesis: The Bell Labs Approach* // Kluwer Academic Publishers, Boston.

# Maximilian Bisani. Joint-sequence models for grapheme-to-phoneme conversio; // [Электронный ресурс]. Режим доступа: <https://doi.org/10.1016/j.specom.2008.01.002> (дата обращения 2.03.2023)

# Phonetisaurus: Exploring grapheme-to-phoneme conversion with joint n-gram models in the WFST framework. // [Электронный ресурс]. Режим доступа: <https://www.cambridge.org/core/journals/natural-language-engineering/article/abs/phonetisaurus-exploring-graphemetophoneme-conversion-with-joint-ngram-models-in-the-wfst-framework/F1160C3866842F0B707924EB30B8E753> (дата обращения 2.03.2023)

# The Kestrel TTS text normalization system. // [Электронный ресурс]. Режим доступа: <https://www.researchgate.net/publication/277932107_The_Kestrel_TTS_text_normalization_system> (дата обращения 02.03.2023)

# Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions. // [Электронный ресурс]. Режим доступа: <https://doi.org/10.48550/arXiv.1712.05884> (дата обращения 05.03.2023)

# Attention Is All You Need; Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. // [Электронный ресурс]. Режим доступа: <https://arxiv.org/abs/1706.03762> (дата обращения 02.03.2023)

1. Огромный открытый датасет русской речи версия 1.0. // [Электронный ресурс]. Режим доступа: <https://habr.com/ru/post/474462/> (дата обращения 05.03.2023)

# Neural Machine Translation by Jointly Learning to Align and Translate; Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio. // [Электронный ресурс]. Режим доступа: <https://arxiv.org/abs/1409.0473> (дата обращения 04.03.2023)

# Нормализация текста в задачах распознавания речи.

# // [Электронный ресурс]. Режим доступа: <https://habr.com/ru/post/491260/> (дата обращения 04.03.2023)

# Russian STT Text Normalization. // [Электронный ресурс]. Режим доступа: <https://github.com/snakers4/russian_stt_text_normalization> (дата обращения 02.03.2023)

# The Annotated Encoder-Decoder with Attention, A pytorch tutorial implementing Bahdanau et al. (2015). // [Электронный ресурс]. Режим доступа: <https://jasmijn.ninja/annotated_encoder_decoder/> (дата обращения 02.03.2023)

1. Text Normalization Challenge - Russian language. // [Электронный ресурс]. Режим доступа: <https://www.kaggle.com/c/text-normalization-challenge-russian-language/overview> (дата обращения 12.03.2023)
2. Нейронный машинный перевод с Transformer и Keras // [Электронный ресурс]. Режим доступа: <https://www.tensorflow.org/text/tutorials/transformer#text_tokenization_detokenization> (дата обращения 12.03.2023)
3. Yi Tay, Mostafa Dehghani, Dara Bahri, Donald Metzler 2022. Efficient Transformers: A Survey // [Электронный ресурс]. Режим доступа: <https://arxiv.org/abs/2009.06732> (дата обращения 15.03.2023)