МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Кафедра дискретной математики и алгоритмики

КЛИМУК Татьяна Алексеевна

ЗАДАЧА НОРМАЛИЗАЦИИ СОДЕРЖАЩИХ ОБЩЕПРИНЯТЫЕ АББРЕВИАТУРЫ И ВЫРАЖЕНИЯ ТЕКСТОВ

Магистерская диссертация

специальность 1-31 81 09 «Алгоритмы и системы обработки больших объемов информации»

Научный руководитель
Елена Павловна Соболевская
кандидат физико-математических наук, доцент

Допущена к защите
«____» _____ 2019 г.
Зав. кафедрой дискретной математики и алгоритмики
_____ В.М. Котов
доктор физико-математических наук, профессор

Минск, 2019

ОГЛАВЛЕНИЕ

РЕФЕРАТ

Магистерская диссертация 47с., 3 рис., 8 таблиц, 13 источников.

Ключевые слова: ЛЕКСЕМА, ВЫРАЖЕНИЕ, НОРМАЛЬНАЯ ФОРМА, НОРМАЛИЗАЦИЯ ТЕКСТА, ОБРАБОТКА ЕСТЕСТВЕННОГО ЯЗЫКА, ЗАДАЧА КЛАССИФИКАЦИИ, НЕЙРОННАЯ СЕТЬ.

Объект исследования – методы нормализации слов и выражений естественного языка.

Цель работы – изучить методы нормализации слов и выражений языка, разработать и реализовать алгоритм нормализации текстов.

Методы исследования – анализ, эксперимент, тестирование, сравнение.

Результаты исследования:

- изучены наиболее распространённые подходы к решению задачи нормализации текстов;
- выделено две основных подзадачи:
  1. классификация выражений;
  2. построение нормализованной последовательности;
- рассмотрены методы решения выделенных подзадач;
- подготовлены данные для тренировки моделей машинного обучения;
- получена модель классификации выражений;
- обучена модель нормализации слов и выражений.

Область применения – решение задач нормализации текстов естественного языка, распознавания и синтеза речи, машинного перевода текстов.

РЭФЕРАТ

Магістарская дысертацыя 47 ст., 3 мал., 8 табліц, 13 крыніц.

Ключавыя словы: ЛЕКСЕМА, ВЫРАЗ, НАРМАЛЬНАЯ ФОРМА, НАРМАЛІЗАЦЫЯ ТЭКСТУ, АПРАЦОЎКА НАТУРАЛЬНАЙ МОВЫ, ЗАДАЧА КЛАСІФІКАЦЫІ, НЕЙРОННАЯ СЕТКА.

Аб'ект даследавання – метады нармалізацыі слоў і выразаў натуральнай мовы.

Мэта працы – вывучыць метады нармалізацыі слоў і выразаў мовы, распрацаваць і рэалізаваць алгарытм нармалізацыі тэкстаў.

Метады даследавання – аналіз, эксперымент, тэсціраванне, параўнанне.

Вынікі даследавання:

- вывучаны найбольш распаўсюджаныя падыходы да вырашэння задачы нармалізацыі тэкстаў;
- выдзелена дзве асноўных падзадачы:
  1. класіфікацыя выразаў;
  2. пабудова нармалізаванай паслядоўнасці;
- разгледжаны метады рашэння выдзеленых падзадач;
- падрыхтаваны даныя для трэніроўкі мадэляў машыннага навучання;
- атрымана мадэль класіфікацыі выразаў;
- навучаная мадэль нармалізацыі слоў і выразаў.

Вобласць прымянення – рашэнне задач нармалізацыі тэкстаў натуральнай мовы, распазнання і сінтэзу маўлення, машыннага перакладу тэкстаў.

ABSTRACT

Master thesis 47 p., 3 pic., 8 tables, 13 resources.

Key words: LEXEME, EXPRESSION, NORMAL FORM, TEXT NORMALIZATION, NATURAL LANGUAGE PROCESSING, CLASSIFICATION PROBLEM, NEURAL NET.

Object of research – methods of natural language tokens normalization.

The aim of this work is to study methods of language tokens normalization, to develop and implement a text normalization algorithm.

Research methods – analysis, experiment, testing, comparing.

Study results:

- the most common approaches to solving the problem of text normalization were studied;
- two main subtasks were highlighted:
  1. tokens classification;
  2. normalized sequence construction;
- the methods of solving the selected subtasks were considered;
- data for machine learning models training was prepared;
- tokens classification model was obtained;
- tokens normalization model was trained.

Field of application – tasks of natural language texts normalization, speech recognition and synthesis, machine translation of texts.

ВВЕДЕНИЕ

В настоящее время стремительно увеличиваются объёмы потоков передаваемой информации. Человек становится не способным достаточно быстро и эффективно обрабатывать весь поступающий к нему объём данных. Естественным следствием этого процесса является то, что крупные компании вкладывают значительные средства в автоматизацию обработки информации.

Существенная доля всей производимой человечеством информации представляется с помощью естественных языков, которые являются наиболее сложными для автоматической обработки. Связано это с тем, что каждый язык обладает своими грамматическими, морфологическими и семантическими особенностями, которые плохо переносятся на другие языки, в особенности на языки из других языковых групп. Иначе говоря, алгоритм обработки одного естественного языка может быть абсолютно не применим к другому. Более того, многие естественные языки имеют тонкости, которые сложны не только для автоматической машинной обработки, но и для понимания людям, не являющимися носителями данного языка.

Среди основных современных задач обработки текстов можно выделить следующие: синтез речи, то есть генерация речевого сигнала по его письменному представлению, распознавание речи, а именно, порождение письменного представление текста по его речевому эквиваленту, а также задачу машинного перевода текста с одного естественного языка на другой естественный язык. В последние годы, с развитием теории глубинного обучения, был сделан огромный шаг на пути к решению этих задач. Однако, машинная обработка текстов до сих пор уступает человеческой.

Важной частью на пути к более качественному решению главных задач из области обработки текстов является решение задачи нормализации текстов, которая заключается в приведении условных сокращений, символов, аббревиатур и других словопозиций языка к некоторой единой форме. Необходимость в решении данной задачи возникает, например, потому, что при генерации речи по письменному представлению даты человек не будет проговаривать слово «точка», несмотря на то, что в тексте точка есть; или можно рассмотреть процесс перевода аббревиатуры: человек переведёт отдельно каждое слово и потом составит новую аббревиатуру, а не транслитерирует элементы аббревиатуры.

В рамках данной работы рассматривается применение основных современных алгоритмов анализа текстов к задаче нормализации. В первой гла-

ве изучаются основные сложности задачи и теоретическая сторона подходов к её решению: проводится анализ распространённых методов нормализации текстов, производится декомпозиция исходной задачи на несколько составляющих и задаются направления экспериментов для решения выделенных подзадач. В последующих главах описана практическая сторона исследования, а именно, реализация и применение рассмотренных методов к решению поставленной задачи нормализации текстов на английском и русском языках, описание полученных результатов, анализ точек роста и возможности оптимизации рассмотренных подходов.

1 ЗАДАЧА НОРМАЛИЗАЦИИ ТЕКСТОВ И МЕТОДЫ ЕЁ РЕШЕНИЯ

1.1 Обработка естественного языка

Под обработкой естественного языка обычно понимается автоматизированный анализ и синтез текстов на естественных языках. Решение таких задач можно связать с построением искусственного интеллекта, если расширить понятие анализа языка до его понимания, а задачу синтеза ограничить генерацией только семантически и грамматически правильных текстов. Основная цель при изучении этих проблем – это обеспечение возможности взаимодействия компьютера и пользователя с помощью естественного языка, наиболее удобной формы общения для человека.

Понимание естественного языка является очень сложной задачей, так как его распознавание часто требует глубоких познаний системы окружающего мир, а также и возможности взаимодействия с ним.

Основными современными задачами обработки текстов являются следующие.

- Извлечение информации из текста.
- Обработка изображения и генерация его подписи или описания.
- Анализ высказываний.
- Анализ тональности (эмоциональной окраски) текста или его части.
- Распознавание речи.
- Синтез речи.
- Машинный перевод текстов.
- Морфологическая разметка, т.е. определение частей речи и их аннотирование.
- Построение вопросно-ответных систем, а именно формирование ответов на вопросы.
- Генерация текста.
- Аннотирование текстов.

1.2 Задача нормализации текстов

Многие из указанных выше задач обработки естественного языка основываются на обработке печатного текста. Обработка текста, в свою очередь, подразумевает предварительное приведение текста к нормальной форме, или решение задачи текстовой нормализации. Возникновение этой задачи также обусловлено стремлением генерировать более естественные, максимально грамматически и семантически точные, тексты в задачах обработки естественного языка.

В общем случае *задача нормализации* текстов представляет собой приведение слов и выражений (*лексем*) естественного языка к единой, общепринятой форме. В качестве выражения могут выступать устойчивые числовые, словесные или число-словесные выражения, например, условные записи дат, телефонных номеров, аббревиатуры.

Описанная задача имеет множество сложностей. Приведём некоторые из них.

- Приведение символов к единому регистру может изменить смысл лексемы. Так, например, лексема «РАН» (Российская академия наук) при приведении к нижнему регистру превращается в родительный падеж слова «раны» и с большой вероятностью будет неуместна в анализируемом тексте.

- Одно и то же слово может принимать различные формы в зависимости от контекста. В английском языке форма, обычно, зависит только от числа, в то время как, например, в русском и немецком языках есть зависимость от рода, времени, падежа.

- В некоторых языках словообразование происходит посредством форматирования корня слова (например, в японском языке), что усложняет задачу машинного перевода, так как одно японское слово может соответствовать нескольким английским.

- В различных языках существуют различные формы записей для одинаковых по смысловой нагрузке лексем. Так, например, в русском языке даты обычно записываются в форме троек (день, месяц, год), а в английском языке принято формировать тройки (месяц, день, год). Таким образом, нормализация текстов сильно зависит от анализируемого языка.

- Большинство естественных языков продолжают развиваться в настоящее время. Это приводит к регулярному появлению новых слов в языке,

а значит, для дальнейшей применимости алгоритмов нормализации текстов, они должны быть способны с высокой точностью обрабатывать новые лексемы.

Важно отметить, что в результате приведения лексем к общей форме, алгоритму анализа легче распознать их сходство, благодаря чему возрастает полнота, т.е. оценка того, насколько много информации было извлечено из текста. Однако это способствует и тому, что при упрощении лексема может изменить изначально заложенный смысл, из-за чего снижается показатель точности анализа, а именно понижается точность распознавания смысла имеющейся в тексте информации.

1.2.1 Значимость задачи нормализации

Нормализация текстов на естественных языках имеет наибольшую ценность как подзадача других проблем в области анализа текстов.

Главной областью применения является задача преобразования печатного текста в речь [1]. Это связано с тем, что числа, даты, аббревиатуры и другие «не стандартные» слова должны произноситься по-разному в зависимости от контекста. Ярким примером является лексема «VI», которая может произноситься как «шесть», «шестой», «ви ай». Несложно видеть, что предварительная нормализация текста решает описанную проблему.

Другой возможностью для применения алгоритмов нормализации является приложение к задачам информационного поиска [2]. Так, например, простейшая операция удаления диакритических знаков (надстрочных, подстрочных, внутристрочных знаков, уточняющих значения других знаков) повышает полноту поиска для запросов, записанных латиницей.

Помимо прочего, нормализация текстов используется для решения задач машинного перевода [3], в частности для корректной обработки аббревиатур и специальных сокращений.

1.3 Методы нормализации текстов

1.3.1 Правиловые алгоритмы

Первыми подходом к решению задачи нормализации текстов являются правиловые алгоритмы, являющиеся прямым переносом лингвистических знаний на анализ текстовых данных. Разработка алгоритмов такого рода включает в себя поиск языковых закономерностей и последующее применение их в качестве правил.

Правиловые алгоритмы отличаются высокой скоростью работы и простотой, но также имеют несколько существенных недостатков, в числе которых

низкая покрывающая способность и сложность адаптации к изменениям языка.

Среди основных правиловых подходов можно выделить стемминг и лематизацию, которые будут рассмотрены далее.

1.3.1.1 Стемминг

Стеммингом называется процесс нахождения основы заданного слова. Важным моментом является то, что понятие основы слова в большинстве случаев не совпадает с понятием морфологического корня слова.

Алгоритм корректного поиска основы слова называется алгоритмом стемминга, а реализация называется стеммером.

Простой стеммер ищет флективную форму в поисковой таблице. Поисковые таблицы стеммеров, в основном, генерируются в полуавтоматическом режиме. Например, для слова «run» автоматически будут получены формы «running», «runs», «runned» и «runly», несмотря на то, что последние формы скорее всего не появятся в тексте, написанном носителем языка.

Главным преимуществом такого подхода является его простота, и, как следствие, скорость, а также легкость обработки исключений. Среди основных недостатков можно выделить то, что все флективные формы должны быть явно перечислены в таблице: новые слова не смогут обрабатываться, даже при условии их корректности. Помимо прочего, есть разумные ограничения на объём поисковой таблицы.

Некоторым усовершенствованием простейших стемминговых алгоритмов являются алгоритмы усечения окончаний. Алгоритмы усечения префиксов также могут быть реализованы, но не во всех языках слова обладают приставками.

Данные алгоритмы не используют таблицы, состоящие из флективных форм. Вместо этого, как правило, хранится низкоразмерный список «правил», который используется алгоритмами для поиска основы с учётом текущей формы лексемы. Например, для английского языка имеют место правила, вида:

- если слово оканчивается на «ed», удалить «ed»;
- если слово оканчивается на «ing», удалить «ing»;
- если слово оканчивается на «ly», удалить «ly».

Алгоритмы усечения работают намного эффективнее, чем алгоритмы полного перебора. Однако, они всё ещё не эффективны в некоторых ситуациях, например, для слов «ran» и «run». Решения, полученные алгоритмами усечения окончаний, ограничиваются теми частями речи, которые имеют хорошо

известные окончания и суффиксы с некоторыми исключениями. Это является серьезным ограничением, так как не все части речи подпадают под чётко определённый набор правил.

1.3.1.2 Лемматизация

Как известно, большинство слов изменяется при использовании в различных грамматических формах: например, конец слова заменяется на грамматическое окончание. *Лемматизация* совершает обратное преобразование: убирает грамматическое окончание и добавляет суффикс или окончание начальной формы. Таким образом, лемматизацией называется процесс приведения словоформы к её словарной форме, лемме.

Во время лемматизации производится определение части речи слова для последующего применения особенного набора правил нормализации для каждой отдельно взятой части речи. Определение части речи принято производить до применения алгоритма выделения основы, поскольку для некоторых языков правила стемминга зависят от части речи данного слова. Такой подход сильно зависит от корректности определения части речи. Несмотря на наличие частичного совпадения между принципами нормализации некоторых лексических категорий, указание некорректной категории или неспособность определить корректную категорию нивелирует преимущество такого подхода над алгоритмом усечения окончаний.

Современные методы лемматизации превосходят алгоритмы стемминга, т.к. при определении части речи по словоформе используется её локальный, а иногда и глобальный, контекст. Таким образом, главная идея заключается в том, что леммер получает значительно больше информации об обрабатываемом слове, и следовательно, он имеет возможность применять более точные правила нормализации словоформ.

Аналогично стеммингу, большинство известных алгоритмов лемматизации являются словарными, то есть в процессе их работы применяется электронный морфологический словарь, содержащий правила преобразований слов или частей речи. Словарь формируется одним из двух способов: вручную, посредством обнаружения лингвистических закономерностей и формирования списка правил, или автоматически, посредством обработки большого корпуса текстов. Среди недостатков такого подхода можно выделить отсутствие возможности анализировать слова, отсутствующие в словаре.

Кроме словарных алгоритмов выделяются также стохастические подходы, использующие набор правил и некоторый набор эвристик, которые позволяют обрабатывать слова, не встречающиеся в словаре.

На практике наиболее распространёнными считаются гибридные подходы, которые в первую очередь используют словари, а в сложных случаях, не

покрываемых словарём, применяют стохастический алгоритм.

1.3.2 Использование методов машинного обучения

С развитием области машинного обучения, в особенности глубинного обучения, появилась возможность разрабатывать алгоритмы нормализации текстов с высокой способностью к обнаружению и выучиванию языковых связей и закономерностей. Рассмотрим подробнее эти методы и их применение к рассматриваемой задаче.

1.3.2.1 Постановка задачи машинного обучения

Задача машинного обучения для текстовой нормализации ставится следующим образом. Имеется конечное множество текстов, для которых известны нормальные формы и, возможно, некоторые другие метки. Такое множество называют множеством прецедентов или обучающей выборкой. Соответствия для произвольных текстов не известны. На основе имеющих данных требуется построить алгоритм, способный с высокой вероятностью строить корректное соответствие для произвольного текста.

На данный момент наиболее распространёнными и эффективными являются алгоритмы обработки, в том числе и нормализации текстов, основанные на обучении и применении нейронных сетей.

1.3.2.2 Общие сведения о нейронных сетях

Нейронная сеть, или *искусственная нейронная сеть*, – это математическая модель, основанная на принципе строения и функционирования сетей нервных клеток живых организма, в частности человеческого мозга. Понятие нейронной сети впервые возникло при попытке моделирования процессов происходящих в мозге человека.

Искусственная нейронная сеть представляет собой множество нейронов, соединённых и взаимодействующих между собой. Каждый нейрон периодически получает сигналы, обрабатывает их, и некоторым образом реагирует (посылает определённый сигнал далее) или не реагирует вовсе. Несмотря на простую концепцию, нейронные сети могут решать сложные задачи, в частности задачи анализа текстов.

Рассмотрим виды нейронных сетей, применяемых в задачах анализа текстов.

1.3.2.3 Нейронные сети прямого распространения

Нейронная сеть прямого распространения является разновидностью искусственной нейронной сети, в которой никакие связи между нейронами не формируют цикл. В основном такие сети представляют собой многослойные перцептроны с применяемой промежуточной сигмоидной функцией активации.

Перцептрон – математическая модель восприятия информации мозгом, предложенная Фрэнком Розенблаттом в 1957 году, является одной из первых и простейших моделей нейронных сетей.

Перцептрон состоит из трёх типов элементов: датчиков (S-элементы), принимающих сигналы, ассоциативных элементов (A-элементы), получающих сигналы от датчиков, и реагирующих элементов (R-элементы). Таким образом, перцептроны позволяют создавать наборы «ассоциаций» между входными сигналами и необходимой реакцией на выходе.

Математически реакция одного R-элемента описывается следующим образом: $f(x) = sign(\sum_{i=1}^{n} \omega_i x_i - \Theta)$ (рисунок 1.1). Обучение R-элемента заключается в подборе весов ω_i.

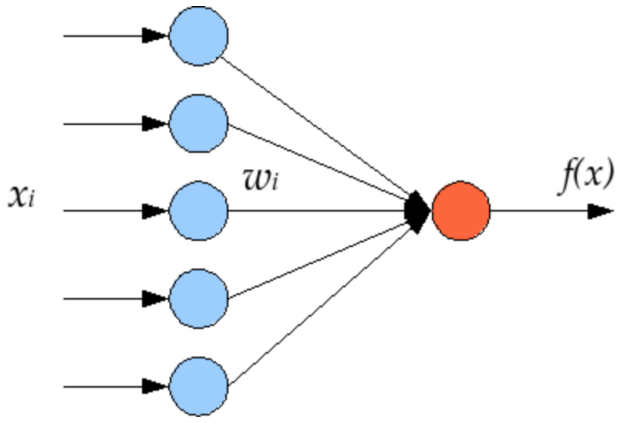

Рисунок 1.1 – Реакция перцептрона

Таким образом, нейронная сеть прямого распространения принимает на вход некоторый числовой вектор $x \in \mathbb{R}^n$, последовательно попеременно применяет линейные и нелинейные преобразования (называемые внутренними, или скрытыми, слоями преобразований), и выдаёт числовой вектор $y \in \mathbb{R}^m$.

Задача обучения нейронной сети состоит в том, чтобы минимизировать значение некоторой заранее определённой функции потерь, принимающей два аргумента: правильное соответствие для заданных входных данных и полученное соответствие.

Существует множество техник обучения нейронных сетей, и в частности многослойных перцептронов, но самым популярным подходом является обратное распространение ошибки. В этом случае обучение состоит из двух этапов.

1. Прямой проход.
   На вход нейронной сети подаются данные, для которых известно истинное соответствие. Эти данные пропускаются через скрытые слои, в процессе чего сохраняются активации нейронов. В итоге на реагирующих элементах образуется некоторое выходное соответствие.

2. Обратный проход.
   Значения реагирующих элементов сравниваются с корректным результатом и вычисляется значение функции ошибки. Затем ошибка распространяется по сети в обратном направлении, с помощью чего корректируются веса. Чаще всего применяется метод градиентного спуска, т.е. вычисляется градиент от полученного значения функции потерь по каждому аргументу (весу связи, используемой в нейронной сети), и этот градиент используется для обновления веса.

Эмпирически показано, что обученные таким образом нейронные сети обладают высокой способностью к обнаружению и выучиванию сложных зависимостей во входных данных, благодаря чему они получили широкое распространение при решении прикладных задач, связанных с анализом данных.

1.3.2.4 Свёрточные нейронные сети

Свёрточные нейронные сети — специальная архитектура нейронных сетей, предназначенная, в первую очередь, для эффективного распознавания изображений. Данная архитектура делает упор на некоторые особенности зрительной коры, для которой были открыты «простые клетки», реагирующие на прямые линии под разными углами, и «сложные клетки», реакция которых связана с активацией определённого набора простых клеток. В итоге, идея свёрточных нейронных сетей заключается в чередовании свёрточных слоёв и субдискретизирующих слоёв. Но, несмотря на адаптированность к задачам анализа изображений, свёрточные нейронные сети нашли своё применение и в задачах анализа текстов [4]. Связано это с тем, что такая архитектура способна анализировать не только конкретную словопозицию, но и её контекст, что является необходимым условием в задачах анализа текстов.

Рассмотрим подробнее основные принципы работы свёрточных сетей.

- Слой свёртки.

  Этот слой является основой свёрточной нейронной сети. Слой свёртки включает в себя множество фильтров для каждого канала. Ядро свёртки обрабатывает фрагменты объекта, суммируя результаты матричного произведения для каждого фрагмента. Весовые коэффициенты ядра

свёртки подбираются в процессе обучения. Главной особенностью свёрточного слоя является сравнительно небольшое количество параметров, и, как следствие, быстрая обучаемость.

- Слой субдискретизации.

  Пулинг представляет собой нелинейное уплотнение карты признаков, при этом группы элементов уплотняются до одного, проходя нелинейное преобразование. Наиболее употребительна при этом функция максимума. Преобразования затрагивают не пересекающиеся прямоугольники, каждый из которых агрегируется в одно максимальное значение. Пулинг интерпретируется так. Если на предыдущей операции свёртки уже были выявлены некоторые признаки, то для дальнейшей обработки имеющаяся детализация уже не нужна, и можно вычислить менее подробные, более общие признаки. Помимо этого фильтрация ненужных деталей помогает не переобучаться. Слой пулинга, как правило, вставляется после слоя свёртки перед слоем следующей свёртки.

1.3.2.5 Рекуррентные нейронные сети

Рекуррентная нейронная сеть (*RNN*) – это разновидность нейронных сетей, в которых связи между нейронами образуют направленную последовательность. Такая архитектура даёт возможность обрабатывать последовательные цепочки, например, строки – последовательности символов, или тексты – последовательности слов. В отличие от простейших многослойных перцептронов, рекуррентные сети обладают способностью использовать внутреннюю память для обработки последовательностей произвольной длины. Но главное преимущество рекуррентных нейронных сетей заключается в том, что они могут эффективно запоминать и использовать данные с предыдущих шагов, или, что нашло большое применение в задачах анализа текста, могут эффективно анализировать, запоминать и использовать контекст.

Рекуррентные нейронные сети функционируют следующим образом. На каждом шаге на вход поступает очередной элемент последовательности $x_t \in \mathbb{R}^n$. В нейроне хранится скрытый вектор состояния h_{t-1}, содержащий в себе информацию с предыдущих шагов. Пара (x_t, h_t) называется модулем (рисунок 1.2). Очередное скрытое состояние в общем виде определяется по формуле: $h_t = \sigma(W^{hh}h_{t-1} + W^{hx}x_t)$.

Ключевая идея рекуррентных нейронных сетей заключается в том, что рекуррентные матрицы весов на каждом шаге одинаковы, таким образом можно обучать фиксированное небольшое количество параметров для обработки

последовательностей произвольной длины.

Для получения выходных значений каждого модуля служит еще одна матрица весов W^s, умноженная на скрытое представление h: $\hat{y}_t = softmax(W^{(s)}h_t)$, где $softmax(x)_i = \frac{e^{x_i}}{\sum_{k=1}^{N} e^{x_k}}$.

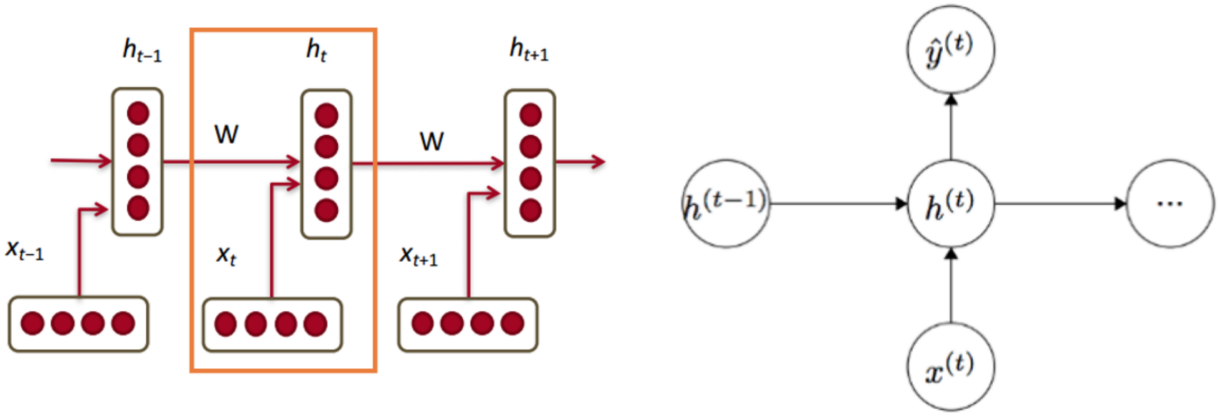

Рисунок 1.2 – Элементы рекуррентной нейронной сети

Обучение рекуррентных нейронных сетей также производится с помощью алгоритма стохастического градиентного спуска и обратного распространения ошибки.

В последнее время наибольшее распространение имеют следующие разновидности рекуррентных сетей.

- Управляемый рекуррентный блок (GRU).

  Данная архитектура была предложена после детального анализа процесса обратного распространения ошибки. Несложно заметить, что во время работы алгоритма для RNN в общем виде градиенты будут распространяться по рекуррентной сети от последнего шага к самому раннему. При достаточно малом исходном градиенте к третьему или четвертому модулю градиент почти исчезает, поскольку по правилу производной сложной функции градиенты перемножаются, и тогда скрытые состояния самых первых шагов фактически не обновляются.

  Метод GRU позволяет вычислять h_t иначе. Вычисления разбиваются на три блока: фильтр обновления, фильтр сброса состояния и новый контейнер памяти. Фильтры являются функциями от входного элемента и скрытого состояния на предыдущем шаге.

  Фильтр обновления в общем случае описывается формулой: $z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1})$.

  Фильтр сброса – $r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1})$.

Главное отличие от обычных RNN состоит в том, что для каждого фильтра используется свой вес.

Контейнер памяти описывается следующим образом: $\tilde{h_t} = tanh(Wx_t + r_t \circ Uh_{t-1})$. Легко видеть, что если множитель фильтра сброса состояния близок к нулю, то и все произведение также приблизится к нулю, и следовательно, информация из предыдущего шага не будет учтена. В этом случай нейрон вычисляет функцию только от нового элемента.

- Сеть с долговременной и кратковременной памятью ($LSTM$) [5].

  LSTM также, как и GRU состоит из последовательности фильтров и контейнера памяти:

  - входной фильтр: $i_t = \sigma(W^{(i)}x_t + U^{(i)}h_{t-1})$;
  - фильтр забывания: $f_t = \sigma(W^{(f)}x_t + U^{(f)}h_{t-1})$;
  - выходной фильтр: $o_t = \sigma(W^{(o)}x_t + U^{(o)}h_{t-1})$;
  - $\tilde{c_t} = tanh(W^{(c)}x_t + r_t \circ U^{(c)}h_{t-1})$.

  LSTM принимает на вход больше информации и поэтому считается обобщением GRU.

LSTM и GRU архитектуры нейронных сетей получили широкое распространение в задачах обработки текстов в силу того, что они способны обрабатывать тексты произвольной длины и анализировать контекст. Также распространены biLSTM и biGRU, которые учитываю не только контекст слева, но и контекст справа от рассматриваемого выражения. Такие подходы наиболее часто употребимы в задачах машинного перевода, где для точности перевода необходимо помнить, в каком контексте был введён тот или иной термин, и понимать в каком контексте используется слово на данный момент.

1.3.2.6 Нормализация текстов при помощи нейронных сетей

Одним из основных подходов, применяемых в решении задач обработки текстов на естественных языках, являются модели, принимающие на вход последовательность и генерирующие другую последовательность, (также именуются *Seq2Seq* или Sequence to Sequence моделями) [6].

Такие генеративные модели строятся из двух взаимосвязанных рекуррентных сетей (RNN), именуемых «*кодировщиком*» и «*декодировщиком*». Кодировщик строит максимально ёмкое внутреннее представление входной последовательности или её части. Полученное представление поступает на вход декодировщику, и, используя его, декодировщик восстанавливает целевую последовательность слов (рисунок 1.3).

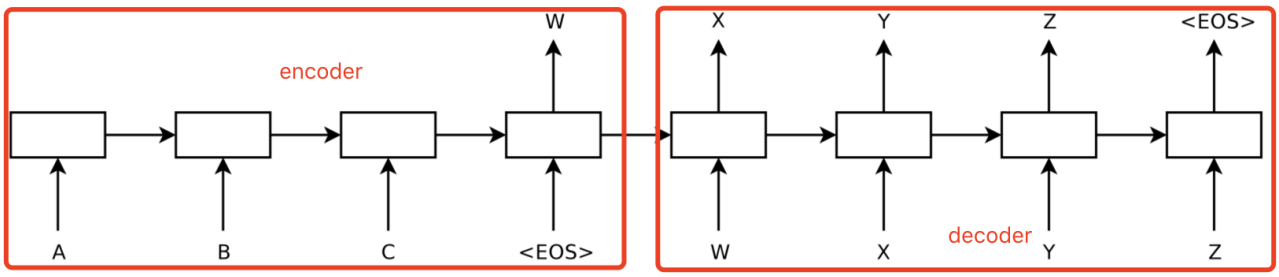

Рисунок 1.3 – Схема работы Seq2Seq модели

Наиболее употребима такая архитектура в задачах машинного перевода, где входной и выходной последовательностями являются предложения на разных языках. Также такие сети используются для реализации вопросно-ответных и диалоговых систем, тогда на вход поступает вопрос, а декодер генерирует ответ. В случае задачи нормализации текстов, Seq2Seq модель по произвольному тексту генерирует нормализованный текст.

Рассмотрим подробнее принципы работы такой сети, а именно кодировщика и декодировщика.

Существует несколько вариантов организации входных данных для кодировщика.

1. В первую очередь это последовательность байт (строки). Такая организация даёт возможность нейронной сети выучить принципы построения слов и легко обрабатывать незнакомые слова.

2. Второй вариант – это преобразование слов в числовые векторы. Для этого используется так называемая матрица представлений (embedding matrix). Количество строк этой матрицы равно размеру словаря, а число столбцов — размерности вектора. Каждая строка матрицы соответствует векторному представлению соответствующего слова. Матрица может быть получена в последствие обучения некоторой отдельной модели, например, word2vec.

На вход декодировщику на исходном такте подается зарезервированный символ «SOS» (start of string), затем, на каждом последующем такте подается скрытое состояние с предыдущего шага и сгенерированное в предыдущую итерацию (или несколько итераций) слово (или символ). Генерация ответа продолжается до тех пор, пока не будет сгенерировано специальное слово-маркер конца строки «EOS» (end of string). Во время обучения в качестве сгенерированного символа на следующий такт передается целевой символ, а распределение на предсказанных символах передается в функцию потерь.

Кроме текущего состояния и выхода кодировщика, декодировщик может использовать дополнительные сигналы, именуемые «вниманием» (англ.

attention) [7]. Чаще всего эти сигналы формируются следующим образом. Все внутренние состояния кодировщика обрабатываются некоторой нейронной архитектурой, возвращающей вектор весов для обработанных состояний. Далее взвешенная сумма состояний поступает на вход декодировщику и используется наравне с выходом кодировщика. На каждом шаге работы декодировщика сигнал обновляется с учётом предыдущего внутреннего состояния декодировщика. Такая архитектура позволяет нейронной сети выучить «важность» фрагментов поступающей на вход последовательности и точнее предсказывать результат.

Процесс предсказания моделью можно описать так: модель генерирует распределение вероятностей для каждого известного ей слова или символа. Результатом шага предполагается наиболее предпочтительное слово среди всех известных, однако для повышения качества финального результата можно применять лучевой поиск (beam search) [8]. Второй вариант подразумевает поддержания множества из фиксированного числа наиболее вероятных гипотез, что позволяет получить правильный результат, даже если наиболее вероятное начало последовательности не являлось оптимальным.

1.4 Связанные задачи

Как было описано ранее, нормализацию текста естественного языка можно провести с помощью единственной Seq2Seq модели. Однако, решая вспомогательные задачи, можно повысить качество работы конечной модели.

1.4.1 Задача классификация выражений

Одной из вспомогательных задач для задачи нормализации текстов является *задача классификации языковых выражений.*

Определим задачу следующим образом. Имеется множество выражений (в том числе и обыкновенных слов) и множество классов: аббревиатуры, сокращения, даты, символы, телефонные номера и т. п. Требуется по выражению установить его классовую принадлежность.

Решение такой задачи позволяет улучшить качество нормализации текста, поскольку для каждого отдельно взятого типа выражения существует меньше вариантов преобразования, чем в общем случае, а значит, информируя конечную модель о типе подаваемого на вход значения, будет упрощаться решение конечной задачи и, следовательно, качество нормализации будет расти.

Для решения задачи классификации необходимо получить конечное множество объектов, для которых известно, к какому классу они относятся (прецеденты). Далее будет строится алгоритм, способный извлекать информацию

из прецедентов и указывать номер (или наименование) класса, к которому относится произвольный объект из исходного множества.

1.4.1.1 Классификация с помощью градиентного бустинга

Градиентный бустинг является одной из техник машинного обучения. Такой подход применим как к задачам классификации, так и к задачам регрессии. В процессе обучения алгоритм строит модель предсказаний, которая представляет собой ансамбль (комбинацию) простейших моделей предсказания, в большинстве случаев решающих деревьев. Построение ансамбля происходит поэтапно, аналогично другим бустинговым методам. Для обучения может использоваться произвольная дифференцируемая функция потерь.

Известно несколько подходов к применению градиентного бустинга к задаче классификации выражений. Различаются они организацией входных данных.

- Преобразование выражения в числовой вектор фиксированной длины. Реализовать такой подход можно, используя готовые матрицы преобразований, или посредством обучения и применения нейронных моделей.

  Минусом такого подхода является то, что при необходимости извлечения детальной информации, возникает задача построения специфичных числовых векторов (англ. embeddings), что является сравнимой по сложности задачей относительно задачи классификации.

- Кодирование отдельных символов выражений, таким образом кодируя всю последовательность.
  Недостаток такого подхода заключается в переменной длине кода, в то время как градиентный бустинг работает с фиксированным количеством факторов. Проблема решается установлением стандартной длины последовательности, добавлением нулей к коротким последовательностям и урезанием длинных.

Преобразованные выражения являются факторами (входными данными) для градиентного бустинга. Важно отметить, что обычно для получения лучшего результата кодируется не только выражение подлежащее классификации, но и его контекст.

1.4.1.2 Классификация с помощью нейронных сетей

Нейронные сети, также как и многие другие алгоритмы машинного обучения, применимы к задачам классификации.

Входные данные формируются таким образом, чтобы итоговые классы представлялись в виде *one-hot вектора*, т.е. числового вектора размерности, равной количеству классов, содержащего нули на всех позициях, кроме одной, соответствующей предсказываемому классу, на которой должна находиться единица.

Плюсом такого подхода является то, что нейронная сеть сможет предсказывать не единственный, самый правдоподобный класс, а вероятность принадлежности объекта к каждому из известных классов.

Выражения допустимо кодировать образом, аналогичным кодированию для градиентного бустинга, но, в то же время, можно отказаться от кодирования, и обучать рекуррентную нейронную сеть, принимающую на вход текстовые строки.

1.5 Выводы

В данной главе были проведены следующие исследования:

- изучены основные подходы к решению задачи нормализации текстов;
- выделена связанная задача классификации выражений;
- рассмотрены известные методы решения задачи определения типа выражений;
- приведён анализ положительных и отрицательных характеристик рассмотренных алгоритмов.

2 ПРИМЕНЕНИЕ МЕТОДОВ КЛАССИФИКАЦИИ ВЫРАЖЕНИЙ

2.1 Состав данных для обучения и сравнения моделей

Рассматривается два набора предложений: на русском и английском языках. Для каждого выражения и слова в предложениях известен класс.

Для русского языка имеется база из 761 436 предложений, в среднем состоящих из 13 слов, максимум – 654. Общее число известных слов равно 801 232. Из них 1 058 слова могут быть отнесены к двум и более классам, что составляет 0.13% от общего количества слов.

Для английского языка рассматривается обучающая выборка из 748 066 предложений, состоящих в среднем из 13 слов, максимум – 256. Общее число известных слов составляет 486 443 единиц, из которых 1 404 (0.28%) относятся к двум и более различным классам, в зависимости от контекста.

В таблице ниже приведены известные классы лексем и количество примеров для обучения в каждой из выборок (таблица 2.1).

Таблица 2.1 — Количество размеченных выражений по классам

Тип лексемы	Краткое описание	Количество английских примеров	Количество русских примеров
«PLAIN»	обычное слово	7 353 693	7 360 439
«PUNCT»	знак препинания	1 880 507	2 288 640
«DATE»	дата	258 348	185 959
«LETTERS»	аббревиатура	152 795	189 528
«CARDINAL»	число	133 744	272 442
«VERBATIM»	спецсимвол	78 108	157 912
«MEASURE»	измерения	14 783	40 534
«ORDINAL»	порядковое число	12 703	46 738
«DECIMAL»	десятичная дробь	9 821	7 297
«MONEY»	денежная сумма	6 128	2 690
«DIGIT»	цифры	5 442	2 012
«ELECTRONIC»	веб-адрес	5 162	5 832
«TELEPHONE»	телефонный номер	4 024	10 088
«TIME»	время	1 465	1 945
«FRACTION»	дробь	1 196	2 460
«ADDRESS»	адрес	522	–

Также приведём статистики того, как часто при нормализации меняется

форма лексемы, в зависимости от её типа (таблица 2.2). Эти величины помогут определить, насколько важна точность определения каждого отдельного класса.

Таблица 2.2 — Доля выражений определённого типа, которые изменяют форму при нормализации

Тип лексемы	Доля изменений для английских примеров	Доля изменений для русских примеров
«LETTERS»	0.94	0.99
«MONEY»	0.99	0.97
«DIGIT»	1	1
«PLAIN»	0.004	0.07
«CARDINAL»	1	0.99
«ORDINAL»	1	0.99
«MEASURE»	0.99	0.98
«PUNCT»	1	1
«VERBATIM»	0.33	0.08
«DATE»	1	0.99
«TELEPHONE»	1	1
«FRACTION»	1	0.98
«ELECTRONIC»	0.96	0.97
«TIME»	1	0.97
«DECIMAL»	1	0.99
«ADDRESS»	1	–

Легко видеть, что большая часть выборки представляет собой «PLAIN» данные, которые редко изменяются, таким образом основная задача состоит в том, чтобы как можно более точно обрабатывать остальные классы.

2.2 Подготовка данных

Для обучения и сравнения моделей классификации необходимо произвести разбиение данных на два множества:

- обучающую выборку, которая будет использоваться для подбора параметров моделей;
- валидационный набор, на котором будут производиться замеры и сравниваться результаты.

В рамках данной работы используется сравнительно небольшой набор данных, поэтому было принято решение 90% имеющихся данных использовать для обучения, а оставшиеся 10% для валидации.

Остаётся две возможности для разбиения данных на указанные множества:

- разбиение множества выражений на два не пересекающихся множества;
- разбиение множества предложений.

Для экспериментов за основу был взят второй подход, в связи с тем, что разбивка данных для конечной задачи нормализации производилась именно таким образом, т.е. обучающие и валидационные данные имеют ненулевое пересечение.

2.3 Применение градиентного бустинга

2.3.1 Представление данных

Как описывалось в первой главе, есть два варианта представления данных, подаваемых на вход алгоритму градиентного бустинга:

- преобразование выражений в векторы фиксированной длины;
- кодирование символов выражений, с последующим расширением нулями или урезанием, в зависимости от длины.

В рамках экспериментов было принято решение использовать второй подход, т.к. его использование показало высокие результаты в других работах из предметной области [9]. Стандартом длины для кодирование выражения было выбрано число 20, так как более 99.5% выражений из входных наборов имеют длину, не превосходящую 20.

Для более качественной классификации помимо самого нормализуемого выражения или слова на вход алгоритму обучения подавался и его контекст, а именно предшествующее и следующее выражение (если таковые имелись). К примеру, в случае английского языка входные данные представляли собой матрицу из 8 925 814 примеров, по 60 факторов в каждом, а валидационные данные имели размерность $992\ 627 \times 60$. Классификация производилась на 16 классов для английского языка, и 15 – для русского.

2.3.2 Обучение модели

Для экспериментов была выбрана библиотека для обучения моделей градиентного бустинга «XGBoost» [10], использовался класс XGBClassifier, позволяющий обучать как модели бинарной классификации, так и мультиклассовой.

В качестве метрики качества алгоритма была выбрана метрика «merror». Её значение вычисляется по формуле:

$$merror(y_{predicted}, y_{true}) = \frac{\sum_{i=0}^{|y_{true}|}[y^i_{true} \neq y^i_{predicted}]}{|y_{true}|}.$$

Таким образом, алгоритм стремится минимизировать количество ошибочных классификаций.

Обучение производилось в течение 18 часов на CPU 2.7 GHz Intel Core i5. За это время прошло по 90 итераций обучения моделей. Итоговая точность классификации выражений на английском языке для такого подхода оказалась равной 99.01%, на русском – 99.27%.

2.3.3 Полученные результаты

В результате проведённого эксперимента был получен классификатор, который с 1% ошибок может указывать метку класса словопозиции с учётом её контекста.

2.4 Применение нейронных сетей

2.4.1 Представление данных

В отличие от распространённых подходов по обучению нейронных сетей, в рамках которых на вход подаётся строка, которая обрабатывается рекуррентной нейронной сетью, предлагается использовать хеширование слов (Word Hashing) [11]. Суть подхода заключается в представлении коротких текстов, или, в нашем случае, выражений в виде набора n-грамм, в частности, в оригинальной публикации использовались триграммы. Главными преимуществами такого подхода является краткость (т.к. количество n-грамм конечно и мало при небольших n), а также способность обрабатывать незнакомые слова.

В процессе экспериментов было опробовано три подхода:

1. представление выражения и его контекста в виде набора триграмм;
2. представление текущего, предшествующего и следующего выражения в виде отдельных наборов триграмм;
3. представление текущего, предшествующего и следующего выражения в виде отдельных наборов одиночных символов и символьных биграмм.

Всего было выделено 190 различных символов, 7 352 символьных биграмм и 74 799 символьных триграмм, что означает то, что в первом случае на

вход нейронной сети подавались векторы размерности 74 799, во втором – $74\ 799 \times 3 = 224\ 379$, а в третьем – $(190 + 7\ 352) \times 3 = 22\ 626$.

Поскольку нейронные сети в первую очередь решают задачу регрессии, подавать в качестве целевого результата метку класса не представляется возможным. В связи с этим целевыми значениями стал шестнадцатиразрядные векторы, состоящие из нулей, за исключением единиц, стоящих на позициях, соответствующих целевым классам.

2.4.2 Обучение моделей

Для обучения нейронных моделей использовался фреймворк «keras» [12].

2.4.2.1 Обучение на общем наборе триграмм

В первом случае обучалась простейшая feed forward архитектура с одним промежуточным слоем.

На вход поступает числовой вектор размерности 74 799, который проходит через один полносвязный слой размерности 50 с активацией «relu». Применение полносвязного слоя к вектору x можно выразить следующим образом: $Wx + b$, где W – обучаемая матрица весов, а b – обучаемое смещение; «relu» есть нелинейное преобразование, описываемое формулой: $relu(x) = \max(x, 0)$. Этот слой позволяет выучить векторные представления для триграмм.

Далее применяется ещё один полносвязный слой размерности 16 с «softmax» активацией:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}}.$$

Таким образом, полученная нейронная сеть содержит 3 740 816 настраиваемых параметров.

Для обучения в качестве функции потерь использовалась категориальная кросс-энтропия, а примеры группировались в пачки по 256 единиц для дополнительной регуляризации. После первой эпохи точность классификации на тестовом наборе составляла 98.57%, после девяти эпох (пяти часов) – 99.23%.

2.4.2.2 Обучение на отдельных наборах триграмм

Основное отличие этого подхода от предыдущего заключается в том, что знания о триграммах были разделены на три отдельных входных вектора.

Архитектура нейронной сети похожа на предыдущую за исключением того, что первое преобразование применяется трижды. При этом матрица весов и смещение, используемые в первом слое, одинаковы для всех входных векторов. Преобразованные векторы конкатенируются, и затем к ним применяется ещё один полносвязный слой.

Параметры обучения были аналогичны предыдущему случаю. За девять часов обучения модели прошло 4 эпохи по данным, и модель смогла достигнуть точности предсказаний в 99.45% на валидационном наборе, начиная с 99.18% точности после первой эпохи обучения.

2.4.2.3 Обучение на отдельных наборах символов и биграмм

В данном подходе были исключены триграммы, но добавлены символы и символьные биграммы. Решение было обусловлено тем, что большинство выражений состоят из небольшого количества символов, и для достижения лучшего покрытия имеет смысл использовать n-граммы меньшей размерности.

Ещё одно изменение заключается в увеличение размера промежуточных векторов до 100, для того, чтобы извлекать больше информации.

В итоге модель содержит 786 016 настраиваемых параметров, что позволило за семь часов провести 45 обучающих эпох по данным. На начальном этапе модель показывала точность 99.12%, а после 45-ой эпохи – 99.74%. На текущем этапе ошибка обучения у модели продолжает падать, а это значит, что в случае необходимости можно понизить долю ошибок классификации.

Аналогичная модель была обучена и для русского набора данных. Количество настраиваемых параметров у модели – 772 015, что сравнимо с англоязычной моделью. После 40-ой эпохи на тестовом наборе данных модель показала точность классификации 99.91%.

2.4.3 Полученные результаты

В результате экспериментов была получена модель, дающая точность предсказания на англоязычном валидационном наборе, равную 99.74%, на русскоязычном – 99.91%.

2.5 Сравнение подходов

Очевидным преимуществом использования градиентного бустинга является простота и скорость применения обученной модели. Среди недостатков можно выделить сложность представления данных и низкую скорость обучения.

Преимуществом второго подхода является его масштабируемость, высокая скорость обучения и высокое качество предсказаний.

Помимо прочего важно отметить, что модель градиентного бустинга выдаёт один, наиболее вероятный класс, в то время как нейронная сеть предскажет вероятность каждого из классов, что может быть использовано для дальнейшего улучшения алгоритмов нормализации текстов.

2.6 Анализ ошибок классификации

Для анализа была выбрана лучшая из полученных моделей, основанная на использовании символов и символьных биграмм.

В таблице ниже приведена статистика самых распространённых выявленных ошибок классификации (таблица 2.3).

Таблица 2.3 — Наиболее распространённые ошибки классификации

Корректный класс	Предсказанный класс	Доля ошибок такого вида
«LETTERS»	«PLAIN»	0.46
«PLAIN»	«LETTERS»	0.1
«ORDINAL»	«CARDINAL»	0.07
«CARDINAL»	«ORDINAL»	0.06
«PUNCT»	«PLAIN»	0.027
«PLAIN»	«PUNCT»	0.022

Если проанализировать те входные данные, на которых чаще всего совершали ошибки, можно отметить, что в этих данных недостаточно большой контекст, например, центральное выражение ограничено двумя знаками препинания. Таким образом, ещё одной точкой роста качества является увеличение размера контекста.

2.7 Выводы

В данной главе была проделана следующая работа:

- реализован известный алгоритм классификации слов и выражений, основанный на применении градиентного бустинга;
- предложен способ представления входных данных для нейронной сети;
- предложены и рассмотрены три архитектуры нейронных сетей для решения задачи классификации;
- приведено сравнение реализованных алгоритмов;
- проведён анализ ошибок;
- приведены возможные направления работы для улучшения качества классификации;
- зафиксирован метод, классифицирующий англоязычные выражения с точностью 99.74%, русскоязычные – с 99.91%.

3 ПРИМЕНЕНИЕ МЕТОДОВ НОРМАЛИЗАЦИИ СЛОВ И ВЫРАЖЕНИЙ

3.1 Подготовка данных для обучения моделей

Во второй главе был произведён краткий обзор состава данных и способов подготовки обучающего и тестового множества. Для экспериментов, проводимых в рамках этого раздела, будет использоваться тот же набор данных, разделённый в соотношении 9 : 1 по предложениям. В то же время каждая словопозиция каждого предложения будет являться отдельным обучающим примером. Таким образом, общее количество обучающих примеров совпадает с общим количеством выражений в отобранных для обучения предложениях.

3.1.1 Формат входных данных

Для лучшей работы кодировщика, который принимает и предобрабатывает входные данные, помимо самого выражения на вход необходимо подавать и его контекст. С учётом результатов, полученных в предыдущей главе, был выбран контекст размера 3, т.е. для каждого выражения для обучения будут использоваться три предшествующих и три последующих слова или выражения, если таковые присутствуют в предложении.

Все выделенные выражения объединяются некоторыми пробельными символами, отсутствующими в словаре (в работе использовался символ @), и подаются как единая последовательность. Выражение, подлежащее нормализации, ограничивается тегами «NORM».

Учитывая небольшой объём обучающих данных, а именно небольшое количество вариантов употребления аналогичных слов, не представляется возможным выучить достаточно качественные векторные представления слов и выражений. Таким образом, данные представляются именно как последовательности символов. Каждый символ определяется вектором из нулей с единственной единицей в позиции, соответствующей индексу символа в общем словаре доступных для обучения символов. Общая длина вектора (размерность пространства) совпадает с размерностью словаря. Иными словами, для кодировки символов используется one-hot-encoding.

Размер словаря символов, составленный по обучающей части выборки, составил 250 символов для обоих языков.

3.1.2 Фильтрация входных данных

3.1.2.1 Фильтрация по классу

Описанный выше способ подготовки данных не учитывает наличие во входных данных выражений, чья длина в несколько раз превышает среднюю длину других выражений. В рамках исследований было выделено два типа таких текстов:

1. Адреса веб-страниц в интернете. Длина некоторых URL-адресов доходит до тысячи символов, а нормализованное представление превышает две тысячи символов. При этом такие типы выражений можно безошибочно нормализовывать, используя правиловый подход.
2. Названия, записанные с помощью CamelCase (рус. Верблюжим регистром). Длины таких записей составляют порядка 500 символов, при нормализации эти выражения остаются неизменными.

Таким образом максимальная длина одного обрабатываемого выражения была сокращена до 61 символа.

3.1.2.2 Фильтрация по типу нормальной формы

В ходе анализа имеющихся данных в предшествующей главе было приведено наблюдение о несбалансированности входных данных, а именно следующее: 90% входных данных не изменяют форму при нормализации, в частности «PLAIN» данные. Таким образом, основной задачей становится выучивание нетривиальных преобразований. Для достижения наилучших результатов на каждой эпохе обучения данные фильтровались таким образом, чтобы на каждые 10 нетривиальных примеров был один тривиальный.

3.1.3 Использование в экспериментах

Описанный процесс подготовки входных данных использовался для проведения всех указанных далее экспериментов. Проведённые эксперименты отличались форматами выходных данных и архитектурами моделей.

3.2 Проведение и оценка экспериментов

3.2.1 Аппаратное обеспечение

Все эксперименты по обучению моделей проводились на видеокарте NVIDIA GeForce GTX 1080 Ti.

3.2.2 Метрика качества

Для тестирования была выбрана метрика «accuracy», то есть точность предсказания. Результат принимался за правильный, если предсказанное нормализованное значение совпадало с ожидаемым с точностью до пробельных символов и регистров.

3.2.3 Базовое качество

Базовое решение, предоставленное вместе с данными, обеспечивает точность 92.3% для английского языка, 84.0% – для русского.

3.3 Применение генеративных нейронных сетей

3.3.1 Однослойная рекуррентная посимвольная сеть

3.3.1.1 Формат выходных данных

В первом эксперименте нейронная сеть получала на вход последовательность символов и возвращала последовательность символов. Для этого декодер получал на вход информацию от кодировщика и специальный зарезервированный символ начала генерации результата. Далее предсказываются символы нормальной формы до тех пор, пока не будет выведен символ конца генерации, также специально зарезервированный. Максимальный размер выходной последовательности был дополнительно ограничен 145 символами. Символы, используемые для генерации результата извлекаются из словаря, размера 250 для английского языка (и соразмерного – для русского), построенного по обучающим данным.

3.3.1.2 Обучение

В рамках данного эксперимента была реализована архитектура нейронной сети, в которой и кодировщик, и декодировщик представляют собой однослойные LSTM модели, с латентным слоем размерности 256. В качестве функции потерь использовалась категориальная кросс-энтропия применённая к результату работы декодировщика после softmax активации.

Общее время обучения модели составило порядка одних суток (примерно 22 часа). Модель за это время совершила 15 эпох обучения с 32 000 примерами обрабатываемыми за одну эпоху.

Ошибка на обучении во время обучения была сопоставима с ошибкой на валидации, что означает то, что входные данные были подготовлены правильно. В начале обучения значение лосс-функции было порядка 0.8, к концу первой

эпохи – 0.07, и это значение продолжало падать, но не так значительно, на последующих эпохах.

3.3.1.3 Тестирование

На выбранных для тестирования 10% исходных данных была достигнута точность в 96.6% для англоязычной модели и 95.8% – для русскоязычной.

3.3.1.4 Анализ ошибок

Среди основных ошибок предсказания можно выделить следующие (таблица 3.1).

Таблица 3.1 — Наиболее распространённые ошибки нормализации

Описание	**Доля ошибок такого вида**
Расстояние Левенштейна между оригиналом и результатом от 1 до 3, а нормализация не нужна	0.4
Иероглифы и символы других алфавитов, редко встречающиеся во входных данных, нормализуются в другие случайные символы	0.08
Ошибочно называются цифры и разряды	0.07
Аббревиатуры воспринимаются, как слова, и наоборот	0.06 0.04
Изменяется регистр	0.04

Рассмотрим несколько возможностей их устранения.

- Для устранения ошибок первого и второго видов предлагается использовать информацию о классе выражения при обучении и применении, что может позволить нейронной сети понять, что данное выражение нужно оставить без изменений. Либо можно ввести отдельную разметку для неизменяемых примеров и научить модель предсказывать факт неизменяемости.
- Для ошибок третьего вида есть два варианта улучшений: дообучение отдельной модели на таком классе выражений, и её применение в случае срабатывания классификатора, или же дополнительные эпохи по всем данным, чтобы модель лучше заучила слова.

В рамках анализа ошибок нормализации аббревиатур, предлагается уделить отдельное внимание ошибкам разметки, которые могли привести к ошибкам такого рода. Предлагается рассмотреть несколько примеров.

- "Letter from Vincent van Gogh to Theo van Gogh Saint **Rémy** , 2 July 1889 ". – [**r e acute m y**]. Имя собственное нормализуется как аббревиатура, что не верно.
- Аббревиатура [**ISLAND**] нормализуется в [**island**]. В то же время: **NASA's** Space Shuttle program had specific standards and conditions relating to GAS payloads . – [**NASA's**]. В каких-то случаях регистр меняется, а в некоторых нет, по неочевидным причинам.
- The call is a moderately loud "chuch **chrrr** - chuck chuck chuck ". – [c h r r r]. Нижнерегистровое слово chrrr нормализуется как аббревиатура.
- Helen **Pidd** (18 October 2009) . – [**p i d d**].

Также предлагается отдельно рассмотреть ошибки именования чисел.

- In **1999** won the gold medal again at the games , once again defeating New Zealand in the final . – [**one thousand nine hundred ninety nine**]. При этом модель предсказывает [**nineteen ninety nine**], что, исходя из правил языка, является более подходящим вариантом нормализации.
- European Union , April **2012** BADIA , Carmina ; CLOP , Merce i JUAREZ , Francisco . – [**two o one two**].
- USS PC- **1264** was a PC- **461** - class submarine chaser built for the United States Navy during World War II . – [**one two six four]**, [**four hundred sicty one**], разная логика для аналогичных случаев.

Таким образом, некоторые ошибки может быть почти невозможно устранить, т.к. аналогичные ошибки возникают во многих примерах как обучающих, так и тестовых данных.

3.3.2 Рекуррентная пословная сеть

3.3.2.1 Формат выходных данных

В этом эксперименте нейронная сеть получала на вход последовательность символов, а возвращала последовательность слов произвольной длины, либо зарезервированный символ «SELF», обозначающий то, что слово не подлежит изменению (используется для того, чтобы избежать ошибки первого типа из анализа ошибок для предыдущей модели). На вход поступал символ начала генерации, далее генерировались слова из словаря размера 1 000 для английского языка, и 2 000 – для русского языка, пока не был получен символ конца нормализации. Используемые словари включали не все слова, имеющиеся в используемых данных, однако покрывали 99.99% примеров из обучающей

выборки. Полный словарь не использовался из-за большого объёма, увеличивающего время обучения.

3.3.2.2 Обучение

Использовалась архитектура нейронной сети, в которой и кодировщик, и декодировщик также представляют собой LSTM модели. Тут были опробованы также многослойные варианты (2-3 рекуррентных слоя). Остальные настройки экспериментов аналогичны предыдущему подходу.

Модель обучалась до тех пор, пока ошибка на обучении не перестала падать. Время обучения составило порядка 30 часов для многослойных вариантов сетей, и сопоставимое с предыдущими экспериментами время для однослойного варианта. Количество и размеры эпох были аналогичны предыдущим экспериментам для упрощения сравнения результатов.

3.3.2.3 Тестирование

Лучшего качества достигли двух и трёхслойные модели. Трёхслойная модель давала прирост в качестве на уровне погрешности, поэтому по соотношению качества и скорости применения был зафиксирован двухслойный вариант. Точность такой англоязычной модели составила 98.0%, точность русскоязычной – 96.3%.

3.3.2.4 Анализ ошибок

В данной секции предлагается рассмотреть выбранные за оптимальные двухслойные генеративные нейронные сети для русского и английского языка. Для начала приведём таблицу точностей предсказания моделей по всем известным классам (таблица 3.2). Указанные значения были получены посредством применения моделей к отложенным 10% обучающих данных.

Таблица 3.2 — Точность предсказания по классам

Класс	**Англоязычная модель**	**Русскоязычная модель**
«PLAIN»	0.99	0.97
«PUNCT»	0.99	0.99
«DATE»	0.96	0.85
«LETTERS»	0.79	0.9
«CARDINAL»	0.92	0.85
«VERBATIM»	0.96	0.99
«MEASURE»	0.56	0.55
«ORDINAL»	0.85	0.75
«DECIMAL»	0.79	0.27
«MONEY»	0.39	0.04
«DIGIT»	0.31	0.64
«ELECTRONIC»	0.02	0.02
«TELEPHONE»	0.0	0.0
«TIME»	0.66	0.21
«FRACTION»	0.0	0.0
«ADDRESS»	0.0	–

По таблице можно сделать следующие наблюдения.

- Класс «PUNCT» является достаточно простым для выучивания, так как модели для обоих языков с высокой точностью могут обрабатывать такие выражения.
- Основные ошибки приходятся на классы с минимальной долей в обучающей выборке. Класс «PLAIN», для которого наблюдался большой процент ошибок в предыдущем подходе, теперь обрабатывается со сравнительно высокой точностью для обоих языков.
- По основным классам русскоязычная заметно уступает в точности англоязычной модели. Но данное наблюдение можно с высокой уверенностью объяснить тем, что русский язык обладает более сложной морфологией.

3.3.3 Рекуррентная пословная сеть с использованием механизма внимания

3.3.3.1 Формат выходных данных

В данном эксперименте, как и в предыдущем, нейронная сеть получала на вход последовательность символов, а возвращала последовательность слов произвольной, либо символ, обозначающий неизменяемость, – «SELF». Размеры и состав словарей совпадает с предшествующим экспериментом.

3.3.3.2 Обучение

Архитектура нейронной сети в этом эксперименте представляет собой двухслойную LSTM модель (так как этот вариант показал лучший результат в предыдущем эксперименте) с применением механизма внимания (attention). Построение используемого вектора внимания подробно описано в первой главе данной работы.

Также использовался подход под названием «teacher forcing» [13], подразумевающий использование правильного слова для подачи на очередной шаг работы декодировщика, а не предсказанного.

Время обучения такой модели составило 40 часов при неизменном количестве и размере эпох.

3.3.3.3 Тестирование

Точность англоязычной модели с использованием механизма внимания составила 98.7%, точность русскоязычной – 97.9%.

3.3.3.4 Анализ ошибок

Для этого, усовершенствованного, подхода также проанализируем точность по классам выражений (таблица 3.3).

Таблица 3.3 — Точность предсказания по классам для модели с вниманием

Класс	**Англоязычная модель**	**Русскоязычная модель**
«PLAIN»	0.99	0.98
«PUNCT»	0.99	1.0
«DATE»	0.98	0.92
«LETTERS»	0.95	0.96
«CARDINAL»	0.97	0.9
«VERBATIM»	0.98	0.99
«MEASURE»	0.86	0.78
«ORDINAL»	0.91	0.84
«DECIMAL»	0.97	0.76
«MONEY»	0.80	0.23
«DIGIT»	0.73	0.85
«ELECTRONIC»	0.53	0.2
«TELEPHONE»	0.39	0.37
«TIME»	0.65	0.84
«FRACTION»	0.0	0.05
«ADDRESS»	0.5	–

Легко видеть, что в значительной степени повысилась точность предсказания для редких классов, при этом точность для частотных классов в среднем не ухудшилась. Таким образом, можно заключить, что механизм внимания позволяет модели лучше обрабатывать разнообразные редкие и сложные примеры.

Также можно отметить, что лучший прирост в качестве наблюдается для русскоязычной модели. Как было указано ранее, в русском языке контекст имеет большое влияние на то, какой будет нормальная форма (например, на падеж), и механизм внимания позволил модели лучше извлекать такую информацию из контекста.

В то же время точность по таким классам, как «MONEY», «DIGIT», «TELEPHONE» и т.п. всё ещё меньше 90% для обоих языков, хотя для большей части примеров из этих классов можно формально описать правила нормализации.

3.3.4 Использование лучевого поиска

Как описывалось ранее, применение лучевого поиска на этапе применения модели позволяет получать правильный вариант, даже если правильные первые слова не являлись самыми вероятными с точки зрения модели.

Данный подход был применён к моделям с использованием механизма внимания (лучшим на данный момент). Одновременно поддерживалось 5 наиболее вероятных гипотез (что увеличило время применения модели в 5 раз). На отложенной части обучающей выборки удалось исправить каждый десятитысячный пример, улучшив тем самым качество моделей на 0.01%.

Рассмотрим несколько примеров, которые исправились за счёт использования такого подхода (таблица 3.4).

Таблица 3.4 — Точность предсказания по классам для модели с вниманием

Пример	**Результат лучшей модели**	**Результат с лучевым поиском**
only in the 1330s .	one thirty s	thirteen thirties
SIR RONALD AYLMER	r o n a l d d	RONALD
of 1.6 m (5.2 ft)	five a two feet	five point two feet
declared on Sunday, 3 September	october third	sunday the third of september
5728701760	пять миллиардов семьсот двадцать восемь миллионов семьсот о один тысяч девятьсот шестьдесят	пять миллиардов семьсот двадцать восемь миллионов семьсот одна тысяча семьсот шестьдесят
в начале 2007 года	двух тысяч семи у	две тысячи седьмого года

3.4 Гибридный подход

Идея гибридного подхода берёт своё начало в задаче машинного перевода. На данный момент основные интернет ресурсы совмещают правиловый и нейронные методы для перевода с целью достижения наилучшего качества.

Далее предлагается рассмотреть классы, для которых возможна разработка правиловых алгоритмов, и сравнить качество правиловых и нейронных моделей на этих классах, а также качество нейронной и гибридной модели на всех данных.

Данный подход рассматривается только для английского языка, так как нормализация описанных типов выражений почти наверняка не зависит от контекста, в отличие от русского языка, где падеж слов всегда зависит от контекста и разработка правил становится практически невозможной.

3.4.1 Обзор правил

- «LETTERS». Символы из входной последовательности нужно вывести через пробел.
- «CARDINAL». Из базовых слов для степеней десятки и чисел от 1 до 19 формируется прописное название числа. Римские числа преобразовываются в десятичную форму и обрабатываются аналогично.
- «MEASURE». Составляется словарь известных единиц измерений и приставок порядка, которые используются для обработки текста. Числа обрабатываются правилом из предыдущего пункта.
- «TELEPHONE». Числа называются посимвольно, пробельные символы обозначаются зарезервированным словом для описания тишины (паузы). Слова произносятся как есть.
- «MONEY». Числа обрабатываются правилом «CARDINAL» (и целая, и дробная части). Для валют хранится словарь, также составляется словарь соответствий валют и дробных сумм.
- «ELECTRONIC». Слова читаются посимвольно, для символов есть словарь соответствий, числа обрабатываются как и в других правилах.
- «DATE». Хранится словарь месяцев и дней недели, числа обрабатываются правилом «CARDINAL». Есть несколько шаблонов для порядка названий, в зависимости от исходной записи.
- «DIGIT». Названия цифр записываются через пробел.

3.4.2 Структура гибридной модели

Модель работает в два этапа:

1. на первом шаге работает модель классификации из второй главы данной магистерской работы;
2. на втором шаге используется одна из нейронных моделей предыдущей секции, если предсказан класс, для которого не описано правило, иначе используется правило.

3.4.3 Анализ качества работы модели

Для начала рассмотрим сравнительную таблицу качества правил и лучшей нейронной модели, использующей механизм внимания (таблица 3.5).

Таблица 3.5 — Сравнение точности предсказания по выделенным классам

Класс	**Нейронная модель**	**Правиловая модель**
«DATE»	0.98	0.997
«LETTERS»	0.95	0.9996
«CARDINAL»	0.97	0.9997
«MEASURE»	0.86	0.9991
«MONEY»	0.80	0.9993
«DIGIT»	0.73	1.0
«ELECTRONIC»	0.53	0.93
«TELEPHONE»	0.39	0.98

Как видно из таблицы, для указанных классов простыми правилами удалось по крайней мере на процент превзойти качество лучшего из имеющихся машиннообученных подходов.

Итоговое качество гибридной модели для пословной модели составило 98.7%, что на 0.7% выше качества самой модели. Точность гибридной модели с нейронной модели с вниманием составило 99%, что на 0.3% превышает полученный ранее результат.

3.5 Другие эксперименты

В рамках экспериментов с моделями нормализации было проведено ещё некоторое количество экспериментов, которые не превосходили имеющиеся на тот момент модели, поэтому не несли интереса с точки зрения анализа

ошибок, тем не менее опишем эти подходы и полученное качество, чтобы отразить всю полноту экспериментов.

- Использование вектора предсказаний классификаторов, обученных во второй главе, в качестве дополнительного входа генеративных моделей. Данный подход должен был подсказать модели классы нормализуемых выражений, однако точность составила лишь 95.1%.
- Совместное обучение нормализатора и классификатора. Ожидалось, что такое обучение поможет модели извлекать больше разнообразной информации из входных данных. Полученная точность составила 94.3%.
- Изменение порога уверенности классификатора в гибридных моделей. Предполгагалось, что при низкой уверенности классификатора лучше использовать нейронный подход, а не правиловый, но качество падало, а не повышалось при выставлении порога уверенности отличного от нуля.

3.6 Выводы

В данной главе была проведена следующая работа:

- описан способ организации и балансировки данных;
- предложена метрика для оценки качества нормализации;
- рассмотрено несколько подходов к нормализации текстов с помощью рекуррентных нейронных сетей, с последовательным усовершенствованием архитектуры;
- на примерах показано, что использование лучевого поиска может повышать качество предсказания нейронных моделей;
- предложен гибридный подход;
- описан набор правил для простых классов;
- рассмотрен способ применения полученных ранее моделей классификации для реализации гибридных моделей;
- проведена оценка качества работы всех реализованных алгоритмов;
- рассмотрены и проанализированы различные ошибки нормализации;
- проанализированы обнаруженные ошибки разметки.

ЗАКЛЮЧЕНИЕ

В рамках магистерской диссертации были достигнуты следующие результаты.

1. В первой главе рассмотрена общая постановка задачи нормализации текстов.

2. Подробно изучены общие подходы к решению задачи обработки текстов, предложены способы применения методов к задаче нормализации текстов на естественном языке.

3. Выделена и сформулирована связанная задача классификации слов и выражений, описаны известные подходы к её решению.

4. Во второй главе описано применение методов машинного обучения к поставленной задаче классификации.

5. Предложена архитектура нейронной сети и способ организации входных данных, обеспечивающие высокую точность классификации.

6. Проведены анализ и сравнение полученных результатов, описано применение результатов для решения конечной задачи нормализации текстов на естественном языке.

7. В третьей главе описаны эксперименты по применению методов нормализации текстов.

8. Предложено несколько архитектур рекуррентных нейронных сетей для генерации нормализованных форм, через генерацию символов и генерацию слов.

9. Показано, как использование лучевого поиска на этапе предсказания может повышать точность результатов.

10. Описан правиловый подход для выделенных проблемных для нейронной нормализации классов.

11. Реализована логика построения гибридной модели нормализации, на основе классификаторов, обученных во второй главе, и нейронных моделей, обученных в начале третьей главы.

12. Выбрана метрика качества нормализации, проведена оценка всех предложенных моделей, проанализированы ошибки.

13. Перечислены другие проведённые эксперименты, не принесшие прироста качества, однако применимые для других задач.

Дальнейшие направления работы:

- исследование других методов классификации слов и выражений, в частности, методов использующих рекуррентные нейронные сети;
- оптимизация реализованных методов классификации;
- исследование зависимости размера используемого контекста и качества предсказания классов;
- увеличение размера контекста в задаче нормализации выражений;
- подача сопутствующей информации про все слова из контекста нормализуемого выражения;
- подача нормализуемого текста отдельным входом;
- обучение более глубоких нейронных сетей;
- использование дополнительных данных для обучения из других источников;
- исследование возможностей повышения качества нормализации за счёт более длительного обучения или использования более глубоких нейронных сетей.

Литература

1. Taylor P. Text-to-Speech Synthesis. Cambridge: Cambridge University Press, 2009. 627 p.
2. Seung-Hoon Na. Two-Stage Document Length Normalization for Information Retrieval // ACM Transactions on Information Systems (TOIS), vol. 33, no. 2. 2015. 40 p.
3. ASR Normalization for Machine Translation / H. Huang, C. Feng, J. Wang et al. // 10.1109/IHMSC.2010.122, vol. 2. 2010. 12 p.
4. Grangier D. Yarats D. Dauphin Y. Auli M. Gehring J. Convolutional Sequence to Sequence Learning. 2017.
5. Koutník J. Steunebrink B.R. Schmidhuber J. Srivastava R.K. Greff K. LSTM: A Search Space Odyssey. 2015.
6. Le Q. Vinyals O. Sutskever I. Sequence to Sequence Learning with Neural Networks. 2014.
7. Bengio Y. Cho K. Bahdanau D. Neural Machine Translation by Jointly Learning to Align and Translate. 2014.
8. Jaitly N. Sproat R. RNN Approaches to Text Normalization: A Challenge. 2016.
9. Rohatgi Sh. Zare M. DeepNorm-A Deep Learning Approach to Text Normalization. 2017.
10. XGBoost Documentation. 2018. https://xgboost.readthedocs.io.
11. Jianfeng Gao et al. Xiaodong He Po-Sen Huang. Learning deep structured semantic models for web search using clickthrough data // Proceedings of the 22nd ACM international conference on Conference on information and knowledge management. 2013. P. 2333–2338.
12. Keras: Deep Learning library for Theano and TensorFlow. 2018. https: keras.io.
13. R. J. Williams, D. Zipser. A learning algorithm for continually running fully recurrent neural networks. 1989.