

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
“Национальный исследовательский университет ИТМО”

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

ИСПОЛЬЗОВАНИЕ МАШИННОГО ОБУЧЕНИЯ ДЛЯ
АВТОМАТИЗИРОВАННОГО РАСШИРЕНИЯ ФОРМАЛЬНЫХ
ГРАММАТИК, ИСПОЛЬЗУЮЩИХСЯ ДЛЯ КЛАССИФИКАЦИИ
ТЕКСТОВ В ДИАЛОГОВЫХ СИСТЕМАХ

Автор Ионов Дмитрий Павлович _____ (Подпись)
(Фамилия, Имя, Отчество)

Направление подготовки (специальность) 01.03.02 Прикладная
математика и информатика _____
(код, наименование)

Квалификация бакалавр _____
(бакалавр, магистр, инженер)*

Руководитель ВКР Ульянцев В.П., к. т. н. _____ (Подпись)
(Фамилия, И., О., ученое звание, степень)

Санкт-Петербург, 2020 г.

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
"НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО"

УТВЕРЖДАЮ

Руководитель ОП

_____ (Фамилия, И.О.)

_____ (подпись)

« _____ » « _____ » 20 ____ г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Обучающийся Ионов Дмитрий Павлович

(ФИО полностью)

Группа M3437 **Факультет/институт/кластер** ИТиП

Квалификация _____ бакалавр

(магистр, инженер, бакалавр)**

Направление подготовки 01.03.02, прикладная математика и информатика

(код, название направления подготовки)

Направленность (профиль) образовательной программы

Математические модели и алгоритмы в разработке программного обеспечения

Специализация _____

Тема ВКР Использование машинного обучения для автоматизированного расширения формальных грамматик, использующихся для классификации текстов в диалоговых системах

Руководитель Ульянцев Владимир Игоревич, канд. т. н., доцент факультета информационных технологий и программирования университета ИТМО

(ФИО полностью, место работы, должность, ученая степень, ученое звание)

технологий и программирования университета ИТМО

2 Срок сдачи студентом законченной работы до « _____ » « _____ » 20 ____ г.

3 Техническое задание и исходные данные к работе

Требуется разработать и реализовать алгоритм, позволяющий повысить уровень автоматизации составления формальных грамматик, использующихся для классификации текстов в диалоговых системах. Алгоритм должен находить множество выделяемых из нетерминалов слов в исходной грамматике и для каждого из них добавлять в грамматику семантически близкие слова. Реализацию необходимо протестировать на реальных грамматиках, использующихся для классификации текстов в диалоговых системах.

4 Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов)

Описание существующих подходов к решению задачи классификации текстов в диалоговых системах, разбор их преимуществ и недостатков. Анализ различных методов

поиска семантически близких слов, выбор одного либо нескольких из них для использования в алгоритме. Разработка и реализация алгоритма, позволяющего добавить в исходную грамматику семантически близкие слова. Разработка способа тестирования качества классификации расширенных алгоритмов грамматик и проведения тестирования на реальных данных.

5 Перечень графического материала (с указанием обязательного материала)

Графические материалы и чертежи работой не предусмотрены

6 Исходные материалы и пособия

7 Дата выдачи задания « ____ » « _____ » 20 ____ г.

Руководитель ВКР _____
(подпись)

Задание принял к исполнению _____ « ____ » « _____ » 20 ____ г.
(подпись)

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
"НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО"

АННОТАЦИЯ

ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Обучающийся Ионов Дмитрий Павлович

(ФИО)

Наименование темы ВКР: Использование машинного обучения для автоматизированного расширения формальных грамматик, использующихся для классификации текстов в диалоговых системах

Наименование организации, где выполнена ВКР Университет ИТМО

ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

1 Цель исследования Повышение уровня автоматизации составления формальных грамматик, использующихся для классификации текстов в диалоговых системах

2 Задачи, решаемые в ВКР:

- а) изучение существующих подходов к решению задачи классификации текстов в диалоговых системах
- б) анализ методов поиска семантически близких слов
- в) разработка и реализация алгоритма расширения формальных грамматик, используя методы поиска семантически близких слов
- г) тестирование реализованного алгоритма на реальных данных

3 Число источников, использованных при составлении обзора 17

4 Полное число источников, использованных в работе 28

5 В том числе источников по годам

Отечественных			Иностранных		
Последние 5 лет	От 5 до 10 лет	Более 10 лет	Последние 5 лет	От 5 до 10 лет	Более 10 лет
3	1	6	2	2	7

6 Использование информационных ресурсов Internet да, число ресурсов: 7

(Да, нет, число ссылок в списке литературы)

7 Использование современных пакетов компьютерных программ и технологий (Указать, какие именно, и в каком разделе работы)

Пакеты компьютерных программ и технологий	Раздел работы
Среда разработки PyCharm и язык программирования Python 3	3.1, 3.2
Среда разработки VSCode	3.1, 3.2
Сервис RusVectores	2.1, 3.2
Rulemma	3.1, 3.2
Gensim	3.1, 3.2
Томиита-парсер	3.1, 3.2

8 Краткая характеристика полученных результатов

Произведены изучение существующих подходов к решению задачи классификации в диалоговых системах и анализ методов поиска семантически близких слов. Разработан и реализован

алгоритм, позволяющий добавлять к выделяющимся из нетерминалов словам в формальных грамматиках семантически близкие слова. В результате тестирования на реальных данных получено приемлемое качество классификации. Эти результаты, а также отзывы лингвистов, полученный после тестирования ими реализованного алгоритма, показали, что существует возможность использования полученной реализации для повышения уровня автоматизации составления формальных грамматик.

9 Полученные гранты, при выполнении работы _____
(Название гранта)

10 Наличие публикаций и выступлений на конференциях по теме выпускной работы Да
(Да, нет)

а) 1 _____
(Библиографическое описание публикаций)

2 _____

3 _____

б) 1 Ионов Д.П. Использование машинного обучения для автоматизированного расширения
(Библиографическое описание выступлений на конференциях)

формальных грамматик, использующихся для классификации текстов в диалоговых системах – IX Конгресс молодых учёных, апрель 2020.

2 _____

3 _____

Обучающийся Ионов Д.П. _____
(ФИО) (подпись)

Руководитель ВКР Ульянцев В.И. _____
(ФИО) (подпись)

“ _____ ” 20 ____ г.

ОГЛАВЛЕНИЕ

ОГЛАВЛЕНИЕ.....	4
ВВЕДЕНИЕ	6
ТЕРМИНЫ И ПОНЯТИЯ	8
ГЛАВА 1. ОБЗОР СУЩЕСТВУЮЩИХ ПОДХОДОВ	9
1.1. ФОРМАЛЬНЫЕ ГРАММАТИКИ	9
1.2. МЕТОДЫ КЛАССИФИКАЦИИ НА ОСНОВЕ МАШИННОГО ОБУЧЕНИЯ	12
1.3. ИДЕЯ УЛУЧШЕНИЯ СУЩЕСТВУЮЩЕГО ПОДХОДА.....	13
1.4. МЕТОДЫ ПОИСКА СЕМАНТИЧЕСКИ БЛИЗКИХ СЛОВ	14
1.4.1. Векторное представление слов	14
1.4.2. Электронный тезаурус	16
1.4.3. Другие методы	17
1.5. ПОСТАНОВКА ЦЕЛИ И ЗАДАЧ ВКР	18
Выводы по главе 1	19
ГЛАВА 2. ОПИСАНИЕ РАЗРАБОТАННОГО РЕШЕНИЯ.....	20
2.1. ТЕОРЕТИЧЕСКОЕ ОПИСАНИЕ НОВОГО ПОДХОДА.....	20
2.1.1. Обоснование выбора метода для поиска семантически близких слов	20
2.1.2. Поиск близких слов в векторной модели.....	21
2.1.3. Структура ансамблирования моделей.....	22
2.1.4. Описание запроса	24
2.2. АНАЛИЗ НОВОГО ПОДХОДА.....	25
2.2.1. Описание характеристик	25
2.2.2. Сравнение с существующими подходами	26
2.2.3. Использование электронного тезауруса для поиска близких слов	27
2.2.4. Описание алгоритма расширения грамматик.....	28

ВЫВОДЫ ПО ГЛАВЕ 2	29
ГЛАВА 3. РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ АЛГОРИТМА	30
3.1. РЕАЛИЗАЦИЯ АЛГОРИТМА РАСШИРЕНИЯ	30
3.1.1. Описание реализации.....	30
3.1.2. Пример работы расширения.....	31
3.2. ТЕСТИРОВАНИЕ АЛГОРИТМА.....	31
3.2.1. Сбор данных и выбор инструментов тестирования.....	32
3.2.2. Выбор гиперпараметров и подготовка грамматик.....	33
3.2.3. Простые грамматики	34
3.2.4. Сложные грамматики.....	37
3.3. ОТЗЫВ О РАБОТЕ РЕАЛИЗОВАННОГО АЛГОРИТМА.....	39
ВЫВОДЫ ПО ГЛАВЕ 3	40
ЗАКЛЮЧЕНИЕ.....	41
БИБЛИОГРАФИЧЕСКИЙ СПИСОК.....	42

ВВЕДЕНИЕ

Диалоговыми системами, взаимодействующими с пользователем на естественном языке, ежедневно пользуются миллионы людей по всему миру. Это чат-боты в мессенджерах и социальных сетях, с которыми можно поговорить на различные темы, голосовые ассистенты, которые могут отвечать на запросы и выполнять некоторые команды, заложенные в функционал, и другие. Одной из задач, которую решают в этих системах, является задача классификации смысла фразы человека, для того чтобы система смогла дать пользователю максимально близкий к ожидаемому ответ.

Для решения этой задачи существуют два основных подхода. Один из подходов заключается в написании ряда правил, так называемой грамматики, по которой фразу можно отнести к одному или нескольким классам. Специалист, знакомый с предметной областью, может составить грамматику, которая затем применяется к текстам для классификации. Этот подход имеет предсказуемое поведение, так как грамматика составляется вручную, и он не требует больших объёмов данных для работы. Однако этот подход требует проделать много ручной работы и занимает большое количество времени.

Другой подход заключается в обучении классификатора, используя один из алгоритмов машинного обучения. В этом подходе критерий принятия решения о выборе класса фразы вычисляется классификатором, обученным на заранее собранной выборке. При наличии размеченных данных развёртывание этого подхода намного быстрее чем написание грамматик и практически не требует ручной работы. Но при отсутствии данных подход работать не сможет, так как не на чем будет обучить классификатор. Также для обучения и получения хороших результатов классификации необходимо подбирать архитектуры, методы обучения и гиперпараметры алгоритмов.

Актуальность решения задачи классификации в диалоговых системах и недостатки в существующих подходах приводят к необходимости усовершенствования одного из используемых подходов. Алгоритм, рассматриваемый в данной работе, позволит повысить уровень автоматизации

составления формальных грамматик и тем самым уменьшит затрачиваемые на составление грамматики ручную работу и временные затраты.

Данная работа проводилась на базе платформы Dasha, разработанной в компании Dasha.AI, с использованием реальных данных для тестирования.

В первой главе описана постановка задачи, приводится анализ существующих подходов к решению задачи классификации фраз, рассматриваются различные методы поиска семантически близких слов.

Во второй главе обоснован выбор метода поиска, используемого в новом подходе в качестве основного, описан принцип работы разработанного подхода, приводятся анализ нового подхода и описание алгоритма расширения грамматик.

В третьей главе описаны выбор программных средств, реализация разработанного алгоритма, его тестирование на реальных данных и отзыв лингвистов о качестве работы реализованной версии алгоритма.

ТЕРМИНЫ И ПОНЯТИЯ

Формальная грамматика – способ описания формального языка, представляющий из себя четверку (стартовый нетерминал, множество терминалов, множество нетерминалов, набор правил вывода)

Формальный язык – множество конечных слов над конечным алфавитом.

Терминал – объект, присутствующий в словах языка, который описывает грамматика, и имеющий конкретное значение.

Нетерминал – объект, обозначающий сущность описываемого языка и не имеющий конкретного значения.

Семантическая близость – численная мера, обозначающая степень схожести двух объектов. Обычно выражается в виде скалярной величины в диапазоне $[0; 1]$. Для пар слов в семантических отношениях (синонимы, общее и частное и т. д.) значение будет близким к единице, для остальных пар оно будет нулевым.

Текстовый корпус – подобранный и обработанный набор текстов, использующийся для исследования языка.

Векторное представление слов – общее название различных методов и представлений в обработке естественного языка, направленных на сопоставление словам из словаря векторов чисел большой размерности.

Тезаурус – в лингвистике, особый словарь, в котором хранятся лексические единицы и семантические отношения между ними.

Синсет – набор слов со схожим значением, содержит также определение и примеры употребления слов в контексте. Между собой синсеты связаны семантическими отношениями.

Лемма – словарная форма слова.

Лемматизация – процесс приведения словоформы к лемме.

ГЛАВА 1. ОБЗОР СУЩЕСТВУЮЩИХ ПОДХОДОВ

Задача классификации встречается в достаточно большом количестве мест помимо диалоговых систем. Она применяется для фильтрации спама, составления интернет-каталогов, распознавания эмоциональной окраски текстов, показа более релевантной рекламы и в некоторых других задачах информационного поиска. Для решения этой задачи существуют два основных подхода, которые рассмотрены ниже.

1.1. ФОРМАЛЬНЫЕ ГРАММАТИКИ

Данный подход заключается в написании ряда правил, составляющих грамматику, по которой текст можно отнести к одному, либо нескольким классам. Например, одно из правил может выглядеть следующим образом: если обрабатываемый текст содержит слова «поздравляю» или «с днём рождения», то отнести его к категории «Поздравления». В качестве искомым в тексте слов могут выступать любые символы рассматриваемого языка, в том числе и одиночные. Пример грамматики для языка правильных скобочных последовательностей показан на рисунке 1.

```
S -> '(' S ')';
S -> S S;
S -> e;
```

Рисунок 1 – Формальная грамматика для языка правильных скобочных последовательностей

Здесь S – нетерминал, использующийся для вывода строк, '(' и ')' – терминалы, из которых состоит реализуемый язык, e – пустой символ, нужен для возможности завершения вывода.

По иерархии Хомского [1] формальные грамматики делятся на четыре типа: неограниченные, контекстно-зависимые, контекстно-свободные, регулярные. Каждый следующий является подмножеством предыдущего. Примеры правил для каждого типа можно увидеть на рисунке 2. Здесь A, B –

нетерминалы, a – терминал, $\alpha, \beta, \gamma, \delta$ – строки, состоящие из терминалов или нетерминалов, все кроме γ могут быть пустыми.

Тип 0	$\alpha A \beta \rightarrow \delta$
Тип 1	$\alpha A \beta \rightarrow \alpha \gamma \beta$
Тип 2	$A \rightarrow \alpha$
Тип 3	$A \rightarrow aB$

Рисунок 2 – Примеры правил для разных типов грамматик

Для классификации текстов в диалоговых системах используются контекстно-свободные грамматики, поэтому в дальнейшем в работе будут описываться и использоваться только они.

Для определения класса данного на вход текста выполняется двухэтапный алгоритм.

На первом этапе происходит лексический анализ, так называемая «токенизация», в процессе которой, входной текст разбивается на распознанные группы – лексемы, из которых затем получают идентифицированные последовательности, называемые «токенами». Этот этап проводится с точки зрения формального языка, именно он задаёт набор лексем, которые могут встретиться при обработке текста. Каждая лексема идентифицируется какому-либо токену, определённого грамматикой языка. Если при этом встречается лексема, которая не может быть идентифицирована в токен языка, она рассматривается как токен-ошибка. Основная цель этого этапа — подготовка текста для дальнейшей обработки и избавление его от ненужных в дальнейшем лексических подробностей. Результат токенизации строчки кода, написанного на языке программирования C++ представлен на рисунке 3.

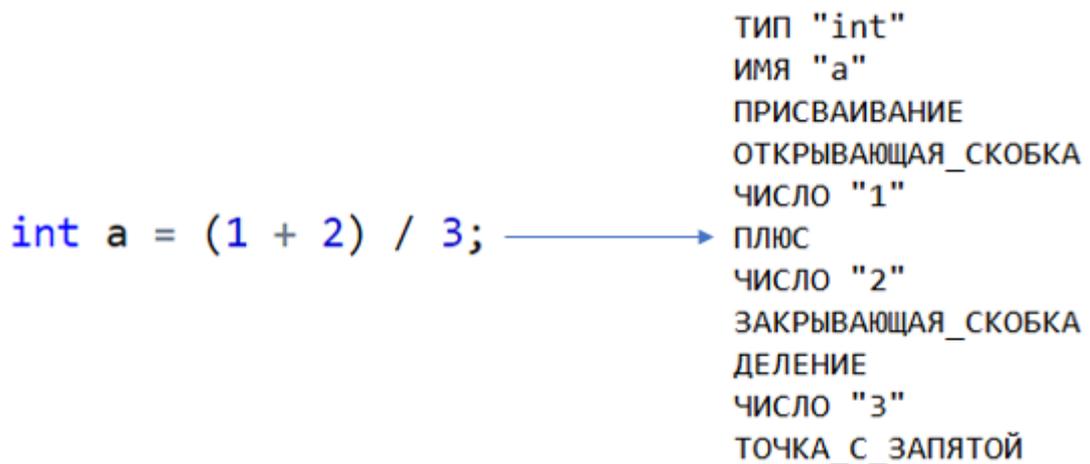


Рисунок 3 – Пример токенизации C++ кода

На втором этапе происходит сопоставление последовательности токенов с составленной формальной грамматикой, называемое синтаксическим анализом или «парсингом». Различают два типа парсеров, выполняющих данный этап: восходящие и нисходящие [2]. Помимо типа, алгоритмы парсеров отличаются между собой подмножествами грамматик, которые парсер может рассматривать, способами восстановления после ошибок и временем работы.

В случае если текст подходит под написанную грамматику парсинг будет успешным и можно утверждать, что данный текст соответствует грамматике и он относится к одному либо нескольким классам, которые грамматика реализует.

Преимущества:

- 1) Предсказуемое поведение, т. е. при составлении грамматики по множеству текстов для какого-то класса, можно быть уверенным в том, что для любого текста из этого множества парсинг с использованием составленной грамматики будет успешным.
- 2) В среднем высокая скорость работы. Например, при использовании GLR-парсера [3] и детерминированной грамматики, время работы парсинга – $O(n)$.
- 3) Отсутствие необходимости в размеченных данных, грамматика составляется экспертом вручную.

Недостатки:

- 1) Много ручной работы и большие временные затраты на составление грамматики, в частности на продумывание вариаций фраз и описывание множества ключевых слов для классов.
- 2) Плохая масштабируемость – при необходимости добавлении в грамматику нового класса текстов, возникает проблема с сохранением корректных результатов для других классов, может резко увеличиться число false positive и false negative.

1.2. МЕТОДЫ КЛАССИФИКАЦИИ НА ОСНОВЕ МАШИННОГО ОБУЧЕНИЯ

Подход заключается в обучении классификатора, используя алгоритмы машинного обучения. В качестве алгоритма обучения можно выбирать разные алгоритмы классификации, например, метод опорных векторов [\[4\]](#).

Перед началом обучения происходит предобработка текста. В неё входят: перевод всех букв в тексте в один регистр, удаление пунктуации, пробельных символов и стоп-слов, замена цифр на текстовый эквивалент и т. д. После обучения можно отдавать тексты классификатору и получать класс, либо набор классов, к которым принадлежит текст по мнению классификатора. Для оценки качества работы обученного классификатора вычисляются precision, recall и F-score [\[5\]](#).

Преимущества:

- 1) При наличии размеченных данных, т. е. образцов текстов для каждого класса, обучение займёт намного меньше времени чем составление грамматики для набора классов.
- 2) Автоматическое создание классификатора без вмешательства человека, т. е. при наличии размеченных данных ручная работа практически отсутствует.
- 3) В отличие от подхода с грамматиками нет необходимости учитывать все вариации фраз, которые могут встретиться в тексте.

Недостатки:

- 1) При отсутствии размеченных данных обучить классификатор и решить задачу не выйдет.
- 2) Классификатор может медленно обрабатывать тексты.
- 3) Для получения хороших результатов классификации необходимо подбирать архитектуры, методы обучения и гиперпараметры алгоритмов, в зависимости от набора классов, под который обучается классификатор.

1.3. ИДЕЯ УЛУЧШЕНИЯ СУЩЕСТВУЮЩЕГО ПОДХОДА

В диалоговых системах задача классификации сводится к классификации смысла фразы человека. Для этого решается задача *multi-label classification* [6] – разновидность задачи классификации, в которой каждому тексту может быть сопоставлен набор классов. Формально, после обработки входного текста алгоритмом, на выходе получается бинарный вектор, в котором для каждого рассматриваемого класса поставлены: 1 — если текст соответствует классу, 0 — если нет. Два описанных выше подхода применяются для решения этой задачи, но каждый из них обладает рядом своих недостатков.

В подходе, который использует для классификации формальные грамматики, одними из существенных недостатков являются большая трудоёмкость и большие временные затраты при разработке грамматики. Они возникают, в частности, из-за необходимости лингвисту продумывать различные ключевые слова, которые могут встретиться во фразе человека. При этом многие из этих ключевых слов часто имеют схожий смысл.

```
Contract -> "контракт" | "договор" | "соглашение" | "сделка" | "документ" |
            "акт" | "договоренность" | "пакт" | "бумаги" | "обязательство";
```

Рисунок 4 – Пример правила для терминала "Контракт"

На рисунке 4 изображено правило для нахождения в тексте терминала «Контракт». Поскольку само слово «контракт» в диалоге может быть сказано другим словом со схожим смыслом, лингвисту приходится описывать множества полностью. Часто для описания даже одного класса текстов приходится

составлять несколько десятков таких правил, из-за чего составление грамматики становится весьма трудоёмким.

Можно воспользоваться тем, что в некоторых правилах описываются множества слов близких по смыслу, и попытаться создать алгоритм, который по введённому слову может либо добавить все нужные слова в грамматику, либо подсказать лингвисту какие ещё близкие по смыслу слова стоило бы описать в грамматике вместе с выбранным словом.

1.4. МЕТОДЫ ПОИСКА СЕМАНТИЧЕСКИ БЛИЗКИХ СЛОВ

Для реализации описанной выше идеи необходим какой-либо способ нахождения семантически близких слов по запросу. Рассмотрим различные методы, которые решают эту задачу.

1.4.1. Векторное представление слов

Данный метод заключается в представлении слова в виде контекстного вектора признаков. Каждому слову сопоставляется вектор чисел, обычно от нуля до единицы. Множество векторов образует словесное векторное пространство. Для получения этого пространства используются различные инструменты получения векторов признаков из текстовых корпусов. Любой набор чисел является допустимым вектором, однако, чтобы он был полезным, набор должен отражать значения слов, отношения между словами и контекст различных слов.

Опишем необходимые характеристики векторного представления слов:

- 1) Каждому слову сопоставлен уникальный вектор, являющийся списком чисел для этого слова.
- 2) Похожие по смыслу слова имеют близкие значения в векторах.

На практике для обеспечения векторного представления этими характеристиками, должны выполняться следующие условия [7]:

- 1) Векторное пространство должно быть многомерно, для хороших моделей размерность лежит в диапазоне от пятидесяти до пятиста.
- 2) Для каждого слова вектор должен отражать «смысл» этого слова.

Простейшим примером векторного представления слов является One-hot encoding. В нём размерность пространства равна числу слов в словаре, каждый вектор почти полностью заполнен нулями, единица стоит только в ячейке, соответствующей номеру этого слова в словаре.

Пример построения схемы One-hot encoding изображён на рисунке 5 [7]. Очевидным недостатком этого метода является отсутствие возможности судить о схожести смысла слов по таким векторам. Вследствие этого данный метод на практике является неэффективным.

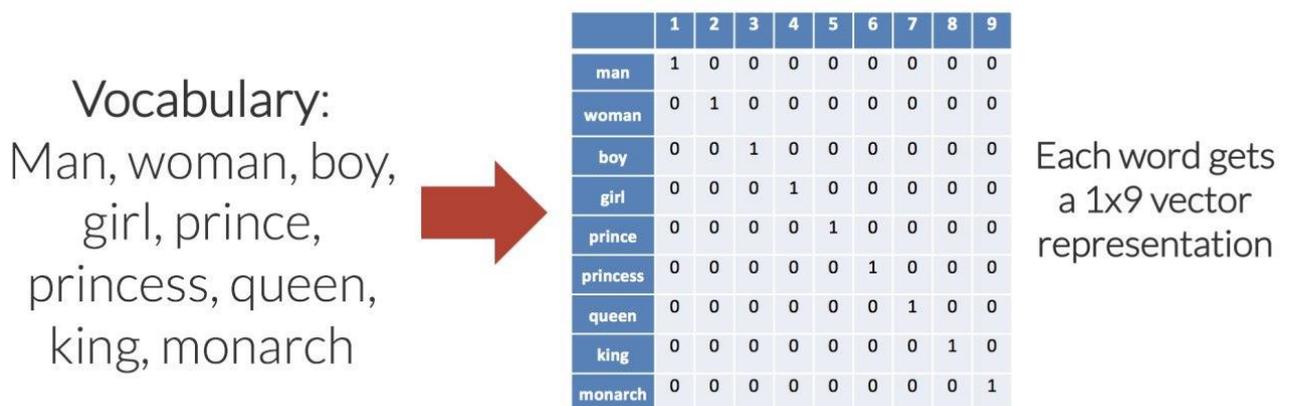


Рисунок 5 – Пример построения схемы One-hot encoding для девяти слов

Одним из наиболее часто используемых для анализа семантики естественных языков алгоритмом построения векторного пространства является word2vec [8-9], использующий нейронные сети. Построение векторного пространства работает следующим образом: принимая в качестве входных данных большой текстовый корпус, алгоритм сопоставляет каждому слову вектор, выдавая на выходе координаты слов в многомерном пространстве. Сначала создаётся словарь корпуса, а затем в процессе “обучения” на входных текстах, вычисляется векторное представление слов. Оно основывается на контекстной близости: слова, которые встречаются в тексте рядом с одинаковыми словами, будут иметь высокую косинусную близость.

В word2vec используются две основных модели обучения: CBoW – Continuous bag of words и Skip-gram. В первой из них при обучении текущее слово предсказывается исходя из окружающего его контекста. Вторая схема

действует наоборот: текущее слово используется для предугадывания окружающего контекста.

Недостатком word2vec является то, что если некоторое слова не встретилось в обучающем корпусе, то оно не будет представлено в построенном векторном пространстве. Примером алгоритма, в котором эта проблема решена, является fastText [10]. Для её решения в этом алгоритме используются N-граммы символов. Например, для слова “работа”, 3-граммами будут: {раб, або, бот, ота}. Векторные представления строятся для N-грамм, а для получения представления слова, нужно сложить сумму векторных представлений всех его N-грамм. Поскольку части слов могут встречаться и в словах отсутствующих в обучающем корпусе, появляется возможность получить векторное представление и для них.

После обработки корпуса алгоритмом, на выходе получается модель, которая позволяет исследовать отношения между словами. Помимо вычисления семантической близости между наборами слов и нахождения близких слов к данному, модель позволяет выполнять алгебраические операции над векторами слов, строить аналогии слов, рисовать семантические карты, находить лишние слова в некотором наборе и т. д.

1.4.2. Электронный тезаурус

В этом методе для поиска семантически близких слов используются электронные тезаурусы, собранные либо вручную, либо с помощью специальных инструментов, и хранящие в себе большой набор слов какого-то языка и связи между словами. Одним из первых собранных тезаурусов для английского языка является разработанный в Пристонском университете WordNet [11]. Он представляет структуру языка в целом, вместо отдельных лексических областей. Состоит из четырёх основных частей речи: существительные, глаголы, прилагательные и наречия. В WordNet базовой словарной единицей является не слово, а синонимический ряд, так называемый синсет (Synset), объединяющий слова, имеющие схожее значение в один узел семантической сети. Каждый синсет дополнен определением и примерами употребления. Слова могут встречаться в нескольких синсетах и иметь

несколько частей речи. Между собой синсеты связаны различными семантическими отношениями: гиперонимия, гипонимия, синонимия, антонимия и т. д.

Пример результата запроса к WordNet для английского слова «fruit» показан на рисунке 6. Как видно, это слово может быть как существительным, так и глаголом, и в обоих случаях оно может иметь несколько смыслов.

The image shows a screenshot of WordNet results for the word "fruit". It is divided into two sections: "Noun" and "Verb".

Noun

- **S: (n) fruit** (the ripened reproductive body of a seed plant)
- **S: (n) yield, fruit** (an amount of a product)
- **S: (n) fruit** (the consequence of some effort or action) *"he lived long enough to see the fruit of his policies"*

Verb

- **S: (v) fruit** (cause to bear fruit)
- **S: (v) fruit** (bear fruit) *"the trees fruited early this year"*

Рисунок 6 – Пример запроса к WordNet для слова "fruit"

WordNet позволяет решать разные задачи, такие как: определение значения слова, поиск близких по смыслу слов, построение тематических тезаурусов, вычисление логичности и связности предложений в тексте и т. д. WordNet можно свободно использовать в научных целях, для работы с ним существуют различные интерфейсы и API. Для русского языка было несколько попыток создать электронный тезаурус, в том числе были попытки взять за основу WordNet, и даже перевести его на русский язык. Примером качественного и актуального на сегодняшний день русского тезауруса является YARN [12], разрабатываемый усилиями представителей нескольких российских университетов.

1.4.3. Другие методы

Также существуют другие методы, не используемые в рамках данной работы. Одним из таких методов является анализ ссылок. На основе ссылочного пространства какого-либо ресурса строится граф вычислений. Для этого используется структура гиперссылок ресурса и учитываются весовые

коэффициенты, назначенные каждому документу. Это позволяет вычислять важность документов внутри набора. Для построения графа используются различные алгоритмы [13], которые можно поделить на два вида.

- 1) Явное задание ссылок с помощью гиперссылок (HITS [14], PageRank [15], WLVM [16]).
- 2) Предварительное построение ссылок (Similarity Flooding [17]).

После реализации алгоритм даёт возможность по запросу выполнять поиск близких по значению слов.

1.5. ПОСТАНОВКА ЦЕЛИ И ЗАДАЧ ВКР

Целью работы является повышение уровня автоматизации составления грамматик. Для этого необходимо создать алгоритм поиска семантически близких ключевых слов и добавления их в формальную грамматику. На основе такого алгоритма возможно реализовать помощника лингвисту в составлении грамматик. Помощник будет по запросу лингвиста выдавать набор слов, семантически близких к слову из запроса, из которых лингвист может выбрать какие слова добавить в грамматику. Это уменьшит время на составление грамматики, так как не нужно будет продумывать различные вариации слов, и упростит работу лингвистов.

Для достижения поставленной цели необходимо решить следующие задачи:

- 1) Выбор одного или нескольких решений задачи поиска семантически близких слов для дальнейшего использования.
- 2) Разработка и реализация алгоритма расширения формальных грамматик, т. е. добавляющего в грамматику семантически близкие слова для выбранных слов из грамматики.
- 3) Тестирование реализованного алгоритма на реальных данных, взятых из диалоговой системы Dasha.AI.

ВЫВОДЫ ПО ГЛАВЕ 1

В данной главе проведён обзор существующих подходов к решению задачи классификации. Описаны недостатки этих подходов, которые приводят к необходимости их усовершенствования. Сформулирована основная идея усовершенствования подхода, связанного с использованием формальных грамматик. Рассмотрены некоторые существующие методы решения задачи поиска семантически близких слов.

ГЛАВА 2. ОПИСАНИЕ РАЗРАБОТАННОГО РЕШЕНИЯ

2.1. ТЕОРЕТИЧЕСКОЕ ОПИСАНИЕ НОВОГО ПОДХОДА

2.1.1. Обоснование выбора метода для поиска семантически близких слов

После анализа возможностей использования методов поиска семантически близких слов, описанных в разделе 1.4, для применения в алгоритме расширения грамматик, в качестве основного метода было выбрано векторное представление слов.

Основными факторами для выбора этого метода стали:

- Возможность использовать значение семантической близости каждого слова для исключения абсолютно неподходящих слов из выборки и показа более релевантных слов в отсортированном по близости порядке. Это поможет лингвисту увидеть сверху списка наиболее близкие слова из множества и решить, какие из них стоит добавить в грамматику.
- Большой выбор предварительно обученных моделей, использующих для обучения различные алгоритмы и параметры обучения, причём как для русского, так и для английского языка
- Возможность улучшить точность поиска семантически близких слов, с помощью использования нескольких моделей в ансамбле. В результате выборка подходящих слов будет увеличена, а шум, возникающий при использовании одной модели, будет уменьшен.
- Возможности моделей, которые не ограничиваются только поиском близких слов, но и позволяют выполнять некоторые другие семантические операции, которые могут быть полезны лингвисту, например, вычисление близости между двумя wybranными словами.
- Простота работы с предварительно обученными моделями и удобные библиотеки для использования возможностей моделей в своём коде.

Метод электронного тезауруса был выбран дополнительным. Для него также был реализован алгоритм расширения, на нём проводились тестирования

и сравнения результатов с грамматикой до расширения и с векторным представлением слов.

2.1.2. Поиск близких слов в векторной модели

Перед тем как отдавать слово на вход обученной модели и получить от модели множество семантически близких слов, необходимо сделать «лемматизацию» слова. Для лемматизации используются специально написанные инструменты или библиотеки, например, Treetagger [18-19] или UDPipe [20]. Затем в зависимости от используемой модели может потребоваться добавить к слову «тэг» части речи, поскольку в некоторых алгоритмах часть речи используется при обучении модели. Некоторые API для работы с векторными моделями выполняют всю предобработку слова самостоятельно.

Семантические ассоциаты для **разум** (ALL)

Частотность слова

Высокая Средняя Низкая

НКРЯ и Wikipedia

1. **рассудок** NOUN 0.78
2. **ум** NOUN 0.65
3. **здравый** ADJ 0.64
4. **разум** PROPN 0.62
5. **разумение** NOUN 0.62
6. **сознание** NOUN 0.61
7. **истина** NOUN 0.61
8. **познание** NOUN 0.60
9. **мудрость** NOUN 0.59
10. **мышление** NOUN 0.59

Рисунок 7 – Пример запроса на поиск семантически близких слов в сервисе RusVectores

На рисунке 7 [21] изображен результат поиска семантически близких слов для слова «разум» в сервисе RusVectores [22]. Использовалась модель, обученная на НКРЯ [23] и русской Википедии. Результатом запроса является набор из 10 слов, отсортированных по семантической близости к слову из запроса. Для

каждого найденного слова указана его часть речи. Кроме того, присутствует возможность выбрать, слова какой встречаемости нужно показывать в ответе.

2.1.3. Структура ансамблирования моделей

Довольно часто результат поиска семантически близких слов содержит в себе слова, которые на самом деле не являются близкими к данному слову, т. е. являются шумом в модели. Одним из способов борьбы с этим является использование заданного порога близости, т. е. в ответе на запрос будут лишь слова, для которых значение близости превысит порог. Однако это неизбежно ведёт к уменьшению числа слов в результирующем множестве, что также является проблемой, так как уменьшает число слов, которые мы можем добавить в грамматику.

Одним из возможных решений возникшей проблемы является ансамблирование моделей. Основная идея этого подхода в использовании нескольких обученных моделей и агрегировании их результатов по какому-либо правилу, для получения окончательного результата. Помимо уменьшения шума и увеличения числа слов в результирующем множестве, это также помогает избежать переобучения/недообучения конкретной модели и даёт возможность совместить в одном запросе результаты от моделей, обученных с использованием разных алгоритмов обучения.

Рассмотрим три основных правила агрегирования результатов разных моделей:

- Объединение (union) – множества слов, полученные путём запроса близких слов в каждую модель, объединяются в одно результирующее множество.
- Пересечение (intersection) – слово добавляется в результирующее множество, только если оно встретилось хотя бы в N моделях из ансамбля, N в этом случае будет изменяемым параметром. Назовём этот параметр `intersection_number`, и в дальнейшем будем использовать это обозначение.

- Взвешивание (weighted) — вместо использования порога близости для ограничения множества слов у одной модели, будем использовать его для ограничения результирующего множества. Сперва получим от каждой модели некоторое множество семантически близких слов, не используя при этом порог близости. Затем пробежимся по всем словам из множеств и найдём сумму значений близости для каждого слова по каждой модели. Пример такого суммирования показан на рисунке 8. После суммирования добавим в результирующее множество слова, для которых значение близости будет больше чем порог, умноженный на число моделей.

Результаты запроса для слова
«разговор»

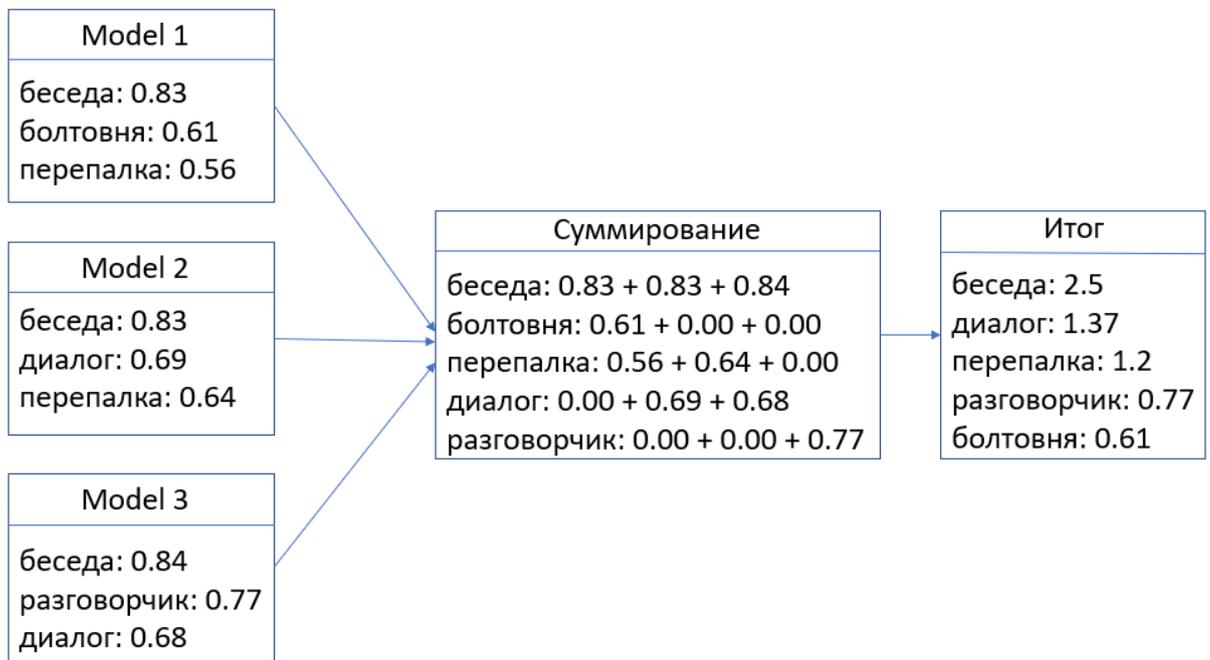


Рисунок 8 – Пример суммирования значений близости для метода
взвешивания

2.1.4. Описание запроса

Обобщим описанное выше в полное описание запроса для одного слова. На рисунке 9 изображена схема поиска семантически близких слов для одного слова при использовании правил объединения или пересечения. Сперва в каждой модели делается запрос на поиск близких слов внутри этой модели, причём для уменьшения шума также учитывается заданный порог косинусной близости. Получившиеся множества слов от каждой модели преобразуются в результирующее множество, используя выбранное правило агрегирования результатов. Получившееся множество и будет являться ответом на исходный запрос.

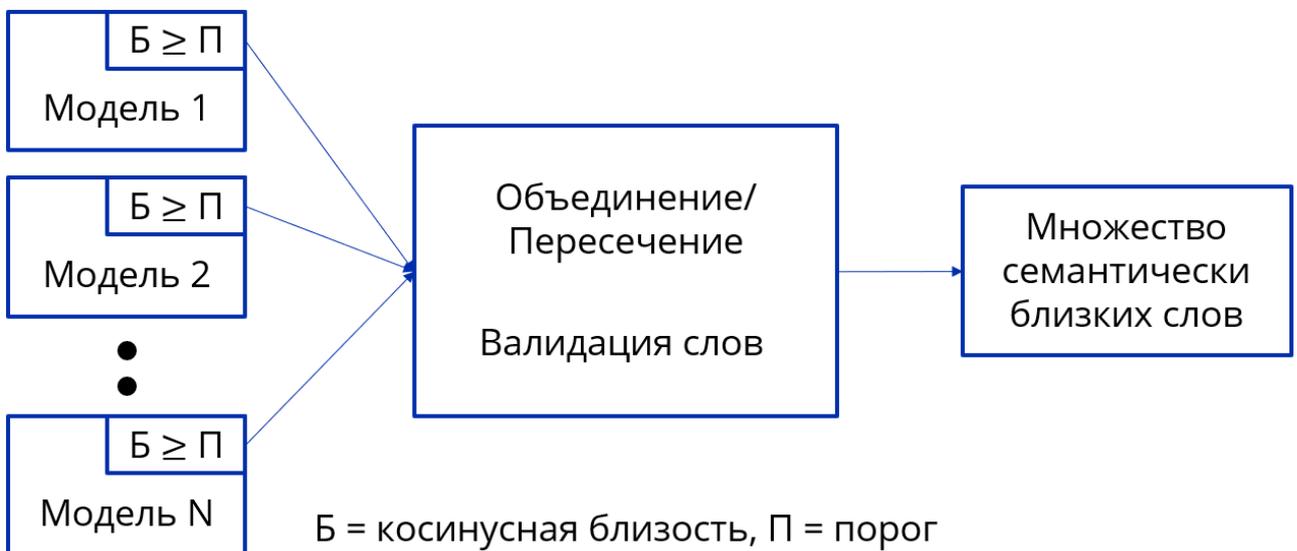


Рисунок 9 – Визуализация запроса на поиск близких слов для одного слова при использовании методов объединения и пересечения

При использовании взвешивания запрос будет проведён схожим образом, будет только два изменения. Во-первых, не будет использоваться порог близости при получении множества близких слов из одной модели, будут просто взяты несколько слов с наибольшим значением близости. Во-вторых, процедура агрегирования результатов будет немного усложнена дополнительным суммированием значения близости для каждого слова из каждой модели и использованием порога близости для получения результирующего множества. Схема для этого метода представлена на рисунке 10.

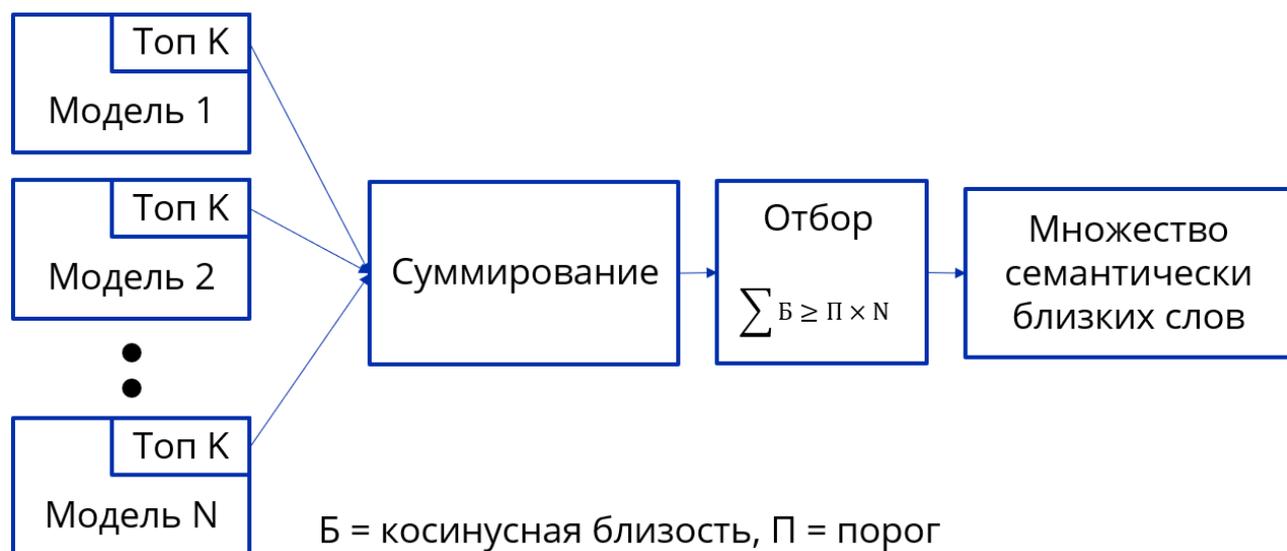


Рисунок 10 – Визуализация запроса на поиск близких слов для одного слова при использовании метода взвешивания

2.2. АНАЛИЗ НОВОГО ПОДХОДА

2.2.1. Описание характеристик

Описанный в разделе 2.1 подход, с использованием векторного представления для поиска семантически близких слов, позволяет упростить написание грамматик для решения задачи классификации. В отличие от алгоритма расширения грамматик путём добавления новых правил на основе синтаксических деревьев, этот подход предполагает полное сохранение структуры грамматики до расширения, что достаточно редко встречается в работах на тему расширения грамматик. Он добавляет семантически близкие слова к описанным заранее правилам, выполняя поиск близких слов в предобученных векторных моделях.

В новом подходе лингвисту не нужно продумывать все варианты слова, которые могут встретиться в фразе, используемой для классификации. Достаточно составить некоторую исходную грамматику, в которой нужно описать некоторой небольшой набор слов, а затем отдать составленную грамматику на вход алгоритма расширения. Получившуюся после работы

алгоритма расширенную грамматику будем использовать для классификации фраз с помощью синтаксического анализатора.

Преимущества:

- 1) В среднем качество классификации расширенной грамматики будет выше, чем у исходной, так как после расширения в грамматике будет описано большее количество нужных для классификации слов.
- 2) Меньше ручной работы на придумывание и выписывание различных вариаций фраз.
- 3) Уменьшились временные затраты на составление грамматик.

Недостатки:

- 1) Осталась необходимость в предварительном сборе данных: либо нужно собрать размеченные данные для обучения векторных моделей, либо найти и выбрать предобученные модели, подходящие для своей задачи.
- 2) Для получения хорошего качества классификации при использовании расширенных грамматик, нужно подбирать оптимальные правила агрегирования и параметры расширения на основе экспериментов.
- 3) В зависимости от выбранных правил и параметров алгоритм может работать медленно.

2.2.2. Сравнение с существующими подходами

Опишем отличия нового подхода от подхода с составлением формальных грамматик, включающих описание всех нужных слов вручную:

- Уменьшенные временные затраты и ручная работа на составлении грамматики.
- После расширения в грамматике будет большее число нужных для классификации слов, чем в составленной за небольшое время исходной грамматике.
- Для составления финальной версии грамматики необходимо провести настройку алгоритма: сбор/обучение моделей, подбор оптимальных параметров расширения.

- Ухудшенное качество классификации, по сравнению с грамматикой, включающей в себя полное описание всех нужных для классификации слов.

Опишем отличия нового подхода от методов классификации с использованием машинного обучения:

- Меньший размер необходимой для работы подхода выборки размеченных данных. Если для обучения классификатора на основе машинного обучения необходимо иметь хорошую выборку фраз для каждого встречающегося класса, то в новом подходе необходимы лишь обученные векторные модели, которые можно обучить на существующих корпусах (Википедия, НКРЯ [23]).
- Меньшее число методов обучения и гиперпараметров, которые необходимо подбирать для работы подхода.
- Увеличение необходимой ручной работы, связанное с необходимостью составления исходной грамматики, которая будет расширяться новым подходом.

2.2.3. Использование электронного тезауруса для поиска близких слов

Запрос на поиск семантически близких слов в электронном тезаурусе идейно не отличается от запроса в методе векторного представления. Перед запросом также необходимо провести лемматизацию. В случае использования тезауруса, построенного на основе системы используемой в WordNet [11], ответом на запрос будет множество синсетов, в которых встречается выбранное слово. Для каждого синсета могут быть указаны дополнительные атрибуты, поддерживаемые тезаурусом: часть речи, описание значения слов в выбранном синсете, обозначения предметной области синсета и т. д. Однако в отличие от векторного представления, в этом методе нет значения близости, по которому можно было бы определить нужно ли добавлять слово в результирующее множество или нет. В случае если у синсета есть некоторые атрибуты, например,

часть речи, можно воспользоваться им, но в общем случае единственным способом является объединение всех слов из полученных синсетов и выдача этого множества как ответ на исходный запрос.

Опишем основные отличия поиска близких слов в электронном тезаурусе от поиска с использованием векторного представления слов:

- Запрос в тезаурусе происходит намного быстрее, так как обращение к одному тезаурусу быстрее, чем обращение к множеству векторных моделей и агрегирование их результатов.
- Ввиду того, что тезаурус собирается вручную, он может содержать редко встречающиеся в реальных текстах слов, которые, однако, важно учесть.
- Практически полностью отсутствует возможность фильтрации близости слов, есть риск выдать большое число лишних слов, что существенно понизит качество классификации.

2.2.4. Описание алгоритма расширения грамматик

Используя описание и анализ нового подхода, опишем алгоритм расширения грамматик:

- 1) Фиксирование используемого метода поиска семантически близких слов: векторное представление либо электронный тезаурус.
- 2) В случае векторного представления также фиксируются: порог косинусной близости, метод ансамблирования моделей (объединение, пересечение, взвешивание) и в случае использования метода пересечения – `intersection_number`.
- 3) Выделение из полученной на вход грамматики множества слов, выделяемых из нетерминалов, с сохранением позиций этих слов, для последующего расширения.
- 4) Лемматизация слов из множества, получение леммы и части речи для каждого слова.
- 5) Запрос поиска семантически близких слов для каждого слова из множества. В случае использования векторного представления запрос

производится к каждой используемой модели, с учётом зафиксированных гиперпараметров. В случае использования электронного тезауруса запрос производится лишь к самому тезаурусу.

- 6) В случае векторного представления: агрегирование множеств близких слов, полученных от каждой модели с использованием выбранного метода ансамблирования.
- 7) Добавление множества полученных близких слов в грамматику, используя сохранённые позиции слов во входной грамматике.

ВЫВОДЫ ПО ГЛАВЕ 2

В данной главе описан выбор наиболее подходящего метода поиска семантически близких слов для решаемой задачи. Предложен подход, позволяющий эффективно использовать ансамбль моделей для уменьшения шума. Приведено сравнение нового подхода с существующими подходами для решения задачи классификации, описан алгоритм расширения грамматик. Помимо использования алгоритма для добавления множества семантически близких слов в грамматику, найденные слова можно показывать лингвисту в отсортированном по близости порядке, чтобы он решил стоит ли добавлять найденные слова в грамматику.

ГЛАВА 3. РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ АЛГОРИТМА

Для реализации и тестирования описанного в подразделе 2.2.4 алгоритма, автором был выбран язык программирования Python 3 [24]. С его помощью удобно реализовывать код, на нём написано большое количество библиотек для работы с машинным обучением и векторными моделями. Это позволяет делать запросы к векторным моделям напрямую из кода, что значительно упрощает работу.

3.1. РЕАЛИЗАЦИЯ АЛГОРИТМА РАСШИРЕНИЯ

3.1.1. Описание реализации

На вход реализованной версии алгоритма поступает файл грамматики, которую необходимо расширить, добавив семантически близкие слова. Из грамматики собирается множество слов, выделяемых из нетерминалов, с сохранением их позиций для последующего добавления близких слов. Лемматизация проводится с помощью библиотеки `rulemma` [25], также с её помощью находится часть речи слова, которая затем при необходимости добавляется к слову. Запросы к векторным моделям проводятся с использованием библиотеки `gensim` [26], которая позволяет загрузить модель в память и производить различные операции. Загрузка модели в память небыстрый процесс, поэтому каждая модель загружается в память один раз. После загрузки у неё запрашиваются ближайшие слова для каждого слова из собранного в грамматике множества. Результаты всех запросов сохраняются в памяти, модель выгружается из памяти, затем рассматривается следующая модель. Таким образом делаются запросы ко всем моделям из ансамбля. После работы с моделями над получившимися множествами производится агрегирование результатов, затем итоговые множества слов добавляются в грамматику. В случае использования для поиска близких слов электронного тезауруса, алгоритм работает идентично, только запросы идут не к векторным моделям, а к предварительно загруженным в память синсетам.

3.1.2. Пример работы расширения

На рисунке 11 и рисунке 12 можно увидеть пример работы реализованного алгоритма на небольшой формальной грамматике. Для каждого

```
#encoding "utf-8"

S -> Nearby | Wrong | Forest | Contract;

Nearby -> "ближайший";

Wrong -> "ошибка";

Forest -> "кустарник";

Contract -> "контракт";
```

Рисунок 11 – Исходная грамматика до расширения

```
#encoding "utf-8"

S -> Nearby | Wrong | Forest | Contract;

Nearby -> "ближайший" | "близкие" | "близко";

Wrong -> "ошибка" | "неточность" | "промах" | "просчет" | "оплошность";

Forest -> "кустарник" | "подлесок" | "заросль" | "куст";

Contract -> "контракт" | "договор" | "соглашение";
```

Рисунок 12 – Полученная после расширения грамматика

из 4 выделяемых из нетерминалов слов, алгоритм сделал запросы к моделям, агрегировал полученные результаты и добавил их в грамматику, тем самым расширив её.

3.2. ТЕСТИРОВАНИЕ АЛГОРИТМА

В рамках тестирования введём некоторые обозначения:

- *Полная грамматика* — составленная вручную лингвистом грамматика, содержащая все слова, которые необходимо выделить для каждого класса. Её составление требует выполнить много ручной работы и занимает

большое количество времени. В рамках тестирования качество для этой грамматики всегда будет равно единице.

- *Исходная грамматика* — по структуре она аналогична полной грамматике, однако в ней описан существенно меньший набор слов относительно полной грамматики. Составляется быстрее чем полная грамматика, из-за отсутствия необходимости в описании всех слов. Именно она отдаётся на вход алгоритму расширения.
- *Расширенная грамматика* — исходная грамматика после расширения тестируемым алгоритмом, содержит добавленные семантически близкие слова. Изменяется в зависимости от используемых методов поиска близких слов, также при использовании векторного представления, от правил агрегирования и значения порога.

Основными задачами тестирования были: попытка получения прироста качества классификации расширенной грамматики относительно исходной и подбор оптимальных гиперпараметров для алгоритма расширения, которые позволят максимально приблизить качество классификации к результатам полной грамматики.

3.2.1. Сбор данных и выбор инструментов тестирования

Для тестирования алгоритма автором были собраны следующие данные:

- Ансамбль из 7 векторных моделей, выложенных в открытый доступ на ресурсе RusVectors [\[27\]](#). Выбирались модели, обученные на разных корпусах (НКРЯ [\[23\]](#), Википедия, Тайга) и использующие разные алгоритмы обучения (Word2vec [\[8–9\]](#), FastText [\[10\]](#)).
- Последняя версия электронного тезауруса YARN [\[12\]](#) для русского языка, выложенного в открытый доступ на официальном сайте проекта. Этот тезаурус был выбран, так как он оказался наиболее продвинутым из тех, с которыми у автора была возможность поработать. Он достаточно актуален и имеет некоторое разбиение на части речи, которое можно использовать для уменьшения лишних слов, добавляемых в грамматику. По разным

причинам (устаревание, закрытый доступ, отсутствие русского языка) некоторые другие существующие тезаурусы с большим количеством понятий, не были использованы в данной работе. Также не рассматривались тезаурусы, собранные не вручную, а с помощью алгоритмов, так как работа с ними по сути идентична работе с методом векторного представления слов.

- Набор фраз из реальных логов диалогов людей с голосовым ассистентом, разработанным компанией Dasha.AI. Каждая фраза была размечена, т. е. ей были добавлены метки классов, которые в ней встречаются. Разметка производилась частично людьми, частично с помощью парсинга фраз по полным грамматикам.
- Грамматики, используемые для классификации фраз в реальном голосовом ассистенте и описывающие некоторый набор классов, которые могли бы встретиться в собранном наборе фраз.

В качестве парсера фраз по формальным грамматикам был выбран находящийся в свободной доступе Томита-парсер [28]. Обусловлено это простотой как синтаксиса его грамматик, так и процедуры запуска самого парсера. Также немаловажным было время работы: в качестве основы в Томита-парсере используется GLR-парсер [3], который в случае детерминированной грамматики работает за линейное время. Код самого парсера написан на C и C++. Всё это сильно ускоряет процесс тестирования, что особенно актуально при использовании больших наборов фраз.

3.2.2. Выбор гиперпараметров и подготовка грамматик

Непосредственно перед расширением фиксировались гиперпараметры алгоритма:

- Порог косинусной близости, который нужно преодолеть слову из векторной модели, чтобы добавить его в множество слов этой модели и рассматривать в дальнейшем при агрегировании результатов.

- Метод ансамблирования моделей: объединение, пересечение или взвешивание, описанные в подразделе 2.1.3.
- В случае выбора метода «пересечение» используется третий параметр (`intersection_number`): число моделей, в скольких множествах результатов слово должно встретиться, для того чтобы быть добавленным в агрегированное множество слов. Должно быть больше одного и меньше либо равно числу моделей в ансамбле.

Исходная грамматика расширялась с помощью реализованного алгоритма, используя зафиксированные параметры. Затем с помощью выбранного парсера, в каждой фразе из набора выделялись описанные в расширенной грамматике классы. Получившиеся в результате метки классов сравнивались с реальной разметкой. После тестирования всех фраз из набора, используя собранные сведения, считались `precision`, `recall` и `f1-score` [5].

3.2.3. Простые грамматики

В этом тестировании использовалась грамматика, выделяющая 58 классов. В каждом из классов выделялось небольшое множество слов. Был собран набор из 344 фраз, каждая из которых выделяла от одного до нескольких классов. `f-score` для фиксированных гиперпараметров был получен как среднее по всем `f-score`, посчитанным для каждого класса.

На рисунке 13 показан график с результатами тестирования. Он показывает изменение значения среднего `f-score` от изменения порога косинусной близости. Для каждого метода ансамблирования и параметра `intersection_number`, при использовании метода пересечения, изображена отдельная кривая. Синяя пунктирная линия снизу графика – тестирование исходной грамматики до расширения, зелёная пунктирная линия сверху – тестирование полной грамматики.

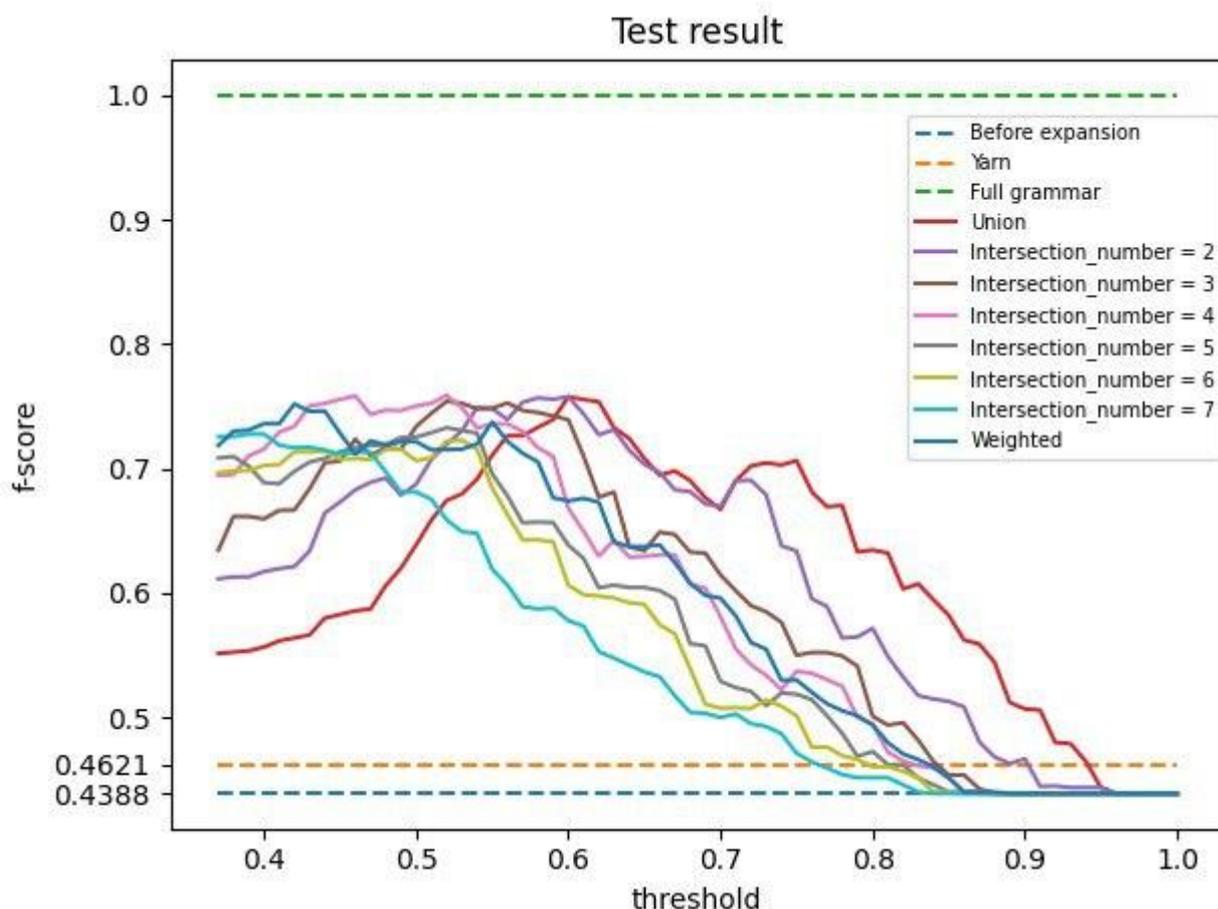


Рисунок 13 – Результаты тестирования алгоритма расширения на простых грамматиках

Тестирование проводилось с изменением значения порога от 0,35 до единицы. Меньшие значения порога на близость не рассматривались по двум причинам. Во-первых, при уменьшении значения порога, к каждому слову, которое выделяется из нетерминала, пришлось бы добавлять всё больше слов из словаря модели. В итоге при пороге близком к нулю, пришлось бы добавить весь словарь, размер которого в зависимости от модели составляет от ста до двухсот тысяч слов. При запуске на такой грамматике парсер начнёт строить деревья для всех этих слов, что неизбежно приведёт к большому потреблению памяти и вылету программы с ошибкой. Во-вторых, тестирование на небольших значениях порога имеет мало смысла, так как при уменьшении порога и добавлении всё большего числа слов, большинство из которых к тому же будут

являться шумом, начнет увеличиваться число false positive для каждого класса, что в итоге приведёт к существенному уменьшению общего f-score.

Рассмотрим кривые, соответствующие методу объединения и пересечения, с параметром `intersection_number` от двух до четырёх. Сперва f-score этих кривых растёт с увеличением порога, достигая при некотором значении порога наивысшей точки для кривой. Увеличение происходит из-за того, что при росте порога уменьшается шум и меньше лишних слов попадает в расширенную грамматику, уменьшается число false positive. Однако при дальнейшем увеличении значения порога, f-score кривых начинает постепенно уменьшаться, достигая в итоге такого же значения, как и у исходной грамматики. Это происходит, потому что через порог перестают проходить нужные для классификации слова и увеличивается число false negative. Достижение кривой значения, соответствующего исходной грамматике, значит, что при таком пороге в грамматику не было добавлено ни одного слова.

Рассмотрим кривые, соответствующие методу пересечения с параметром `intersection_number` от пяти до семи. От кривых, описанных выше, они отличаются тем, что их f-score не возрастает и потом падает, а сперва немного колеблется, после чего начинает постепенно падать с увеличением порога. Причина этого в том, что для добавления слова в грамматику в этом методе необходимо, чтобы слово встретилось в нескольких моделях, а с увеличением порога множество слов каждой модели начинает уменьшаться, что неизбежно приводит к уменьшению добавляемого множества слов. Чем больше параметр `intersection_number`, тем больше моделей должны содержать слово для валидации, а следовательно, валидацию пройдёт ещё меньше слов. На графике это отражено тем, что кривые с большим значением этого параметра находятся ниже, чем кривые с меньшим его значением, при увеличении порога близости.

Поведение кривой, соответствующей методу взвешивания, лишь немного отличается от кривых для метода пересечения с большим `intersection_number`. Эта кривая колеблется на промежутке от 0,35 до 0,55, после чего при увеличении порога начинает падать к результатам исходной грамматики. Её главное отличие

– это достижение высокого максимального значения f-score, причём при достаточно низком пороге.

При использовании для поиска семантически близких слов электронного тезауруса YARN [12], удалось добиться совсем небольшого прироста значения f-score. Этот метод на графике изображён оранжевой пунктирной линией. Поскольку все синсеты в тезаурусе составляются вручную, он содержит в себе лишь небольшой набор слов всего языка. Из-за этого для многих слов близкие просто не были найдены и добавлены, и их f-score относительно исходной грамматики не изменился, что и привело лишь к малому увеличению общего f-score.

По результатам этого тестирования лучший результат показал метод пересечения со значением порога равным 0,52 и параметром intersection_number равным 4. Очень близкие к лучшему результату значения, при разных порогах косинусной близости, показали также методы объединения, взвешивания и пересечения с параметром 2 и 3. Прирост качества классификации расширенной грамматики относительно исходной, составил приблизительно 0,32.

3.2.4. Сложные грамматики

В этом тестировании набор классов был уменьшен до десяти, но существенно усложнена внутренняя структура грамматики. В каждый из классов выделялась фраза, состоящая из нескольких слов и имеющая определённый смысл. Набор фраз был расширен до пятидесяти трёх тысяч, в каждой фразе по-прежнему выделялся один либо несколько классов.

Была добавлена кросс-валидация: после выбора метода ансамблирования выборка фраз делилась на обучающую и тестовую, затем для обучающей выборки находилось оптимальное значение порога из диапазона от 0,4 до 0,7. Диапазон был выбран исходя из результатов первого тестирования. После нахождения оптимального порога алгоритм запускался для тестовой выборки. Получившиеся значения f-score для обучающей и тестовой выборки сохранялись, кросс-валидация продолжалась. Всего проводилось пять итераций

кросс-валидации для каждого метода ансамблирования. Итоговые значения f-score по результатам кросс-валидации усреднялись.

Таблица 1 – Результаты тестирования алгоритма расширения на сложных грамматиках

	Полная грамматика	Исходная грамматика	Объединение	Взвешивание	YARN
Train score	1	0,5411	0,8072	0,7379	0,6606
Test score			0,8071	0,7377	

Таблица 2 – Результаты тестирования алгоритма расширения на сложных грамматиках

	Пересечение					
Intersection number	2	3	4	5	6	7
Train score	0,7925	0,7369	0,7365	0,7336	0,7279	0,7155
Test score	0,7923	0,7368	0,7363	0,7335	0,7278	0,7154

На таблицах 1 и 2 показаны результаты для всех тестируемых методов. Как видно лучшие результаты и для обучающей и для тестовой выборки показал метод объединения. Метод пересечения в среднем показал себе аналогично первому тестированию: почти везде значения постепенно уменьшались при увеличении параметров intersection_number. Результаты метода взвешивания в этом тестировании немного ухудшились.

В данном тестировании метод электронного тезауруса показал улучшенный прирост f-score по сравнению с тестированием на простых грамматиках, однако его результаты и в этом тестировании хуже, чем результаты метода векторного представления слов.

Прирост качества классификации расширенной грамматики относительно исходной при рассмотрении метода с лучшим результатом, составил приблизительно 0,26.

3.3. ОТЗЫВ О РАБОТЕ РЕАЛИЗОВАННОГО АЛГОРИТМА

Помимо тестирования реализованного алгоритма для расширения грамматик, алгоритм также был отдан для тестирования лингвистам, с целью получения их мнения, относительно возможности применения программы для автоматизации их работы.

Вот основные моменты полученного от лингвистов отзыва:

- Из всех методов ансамблирования лучше всех себя показал метод пересечения, с параметром `intersection_number` равным 4 и использованием достаточно высокого порога близости, в среднем больше 0,6. При таких параметрах добавляется не слишком много слов и большинство из них действительно являются близкими по смыслу к запрашиваемому слову.
- При использовании малых значений порога, независимо от выбранного метода ансамблирования, алгоритм начинает добавлять слишком много лишних слов, включая даже антонимы, например к слову «близко» может добавиться слово «далеко».
- Метод объединения оказался малоприменимым для реального использования ввиду того, что при его использовании добавляется много слов, которые порой имеют много смыслов. А при расширении лингвистам важно в первую очередь выдержать один смысл.
- Метод пересечения в целом работает гораздо точнее и добавляет слова с одним смыслом. Он выглядит так, что с его помощью можно было бы пополнить используемые словари.
- Программа неаккуратно работает с видами глаголов, бывает, что это важно и нужно учитывать при описании слов. Например, необходимо учитывать, что при смене приставки в глаголе, смене его возвратности или при смене вида – меняется и значение, а также сочетаемость глагола. Для достижения

этого в программу можно добавить более тщательный анализ грамматических характеристик слов (в том числе части речи), которые были описаны заранее.

Подводя итог, было отмечено, что текущая версия программы может быть полезна там, где не нужна тонкая настройка синонимического ряда, но есть необходимость предусмотреть все варианты, как пользователь может назвать сущность. В качестве примера был приведён набор слов «документы», который из набора слов: {«документы», «бланк», «бумага»}, был расширен до набора: {«документы», «бланк», «бумага», «анкета», «картон», «документ», «паспорт», «бланка», «конверт», «справка», «удостоверение», «бумажка», «листок», «документация», «квитанция»}. Алгоритм может быть полезен при составлении новых локальных грамматик, его работа позволит повысить уровень автоматизации составления таких грамматик, а также предусмотреть различные варианты описания смысла некоторого слова.

Выводы по главе 3

В данной главе описана реализация алгоритма для расширения формальных грамматик и используемые в ней библиотеки. Предложен алгоритм проверки качества классификации грамматик, расширенных алгоритмом. Приведены результаты тестирования реализованного алгоритма на грамматиках разной сложности, с использованием разного набора классов и классифицируемых фраз. Описан отзыв, полученный от лингвистов, которые протестировали реализованный алгоритм на используемых при классификации наборах слов.

ЗАКЛЮЧЕНИЕ

В рамках работы был проведен обзор существующих подходов к решению задачи классификации фраз. Описаны преимущества и недостатки их применения. Поставлена задача усовершенствования подхода, связанного с использованием формальных грамматик.

При формулировке идеи для нового подхода были описаны различные методы поиска семантически близких слов, и обусловлен выбор одного из них в качестве основного для использования в алгоритме. Предложен сам алгоритм расширения, основанный на использовании векторного представления слов и ансамблирования обученных векторных моделей.

Алгоритм был реализован на языке Python 3, с использованием нескольких существующих для него библиотек. Для тестирования реализованного алгоритма были взяты реально использующиеся для классификации фраз формальные грамматики и фразы из логов диалогов людей с голосовым ассистентом Dasha.AI. В качестве парсера текстов по формальным грамматикам использовался находящийся в свободном доступе Томиита-парсер. Были проведены эксперименты с грамматиками разной сложности. Также реализованный алгоритм был отдан лингвистам, для проведения его тестирования. Полученные результаты позволили заявить о возможности использования реализованного алгоритма для повышения уровня автоматизации составления формальных грамматик.

По результатам работы сделано выступление на Конгрессе молодых учёных в Университете ИТМО, в рамках секции «Технологии программирования, искусственный интеллект, биоинформатика».

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Chomsky N.* Three models for the description of language // IRE Transactions on Information Theory. — 1956. — Vol. 2. — P. 113–124. — URL: <http://www.chomsky.info/articles/195609--.pdf>.
2. *Compilers: Principles, Techniques, and Tools (2nd Edition) / A. V. Aho [et al.]*. — USA: Addison-Wesley Longman Publishing Co., Inc., 2006. — ISBN 0321486811.
3. *Lang B.* Deterministic Techniques for Efficient Non-Deterministic Parsers // Vol. 14. — 07/1974. — P. 255–269. — DOI: 10.1007/3-540-06841-4_65.
4. *Cortes C., Vapnik V.* Support-Vector Networks // Machine Learning. — 1995. — P. 273–297.
5. *Goutte C., Gaussier E.* A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation // Advances in Information Retrieval / ed. by D. E. Losada, J. M. Fernández-Luna. — Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. — P. 345–359. — ISBN 978-3-540-31865-1.
6. *Tsoumakas G., Katakis I.* Multi-Label Classification: An Overview // International Journal of Data Warehousing and Mining. — 2009. — Sept. — Vol. 3. — P. 1–13. — DOI: 10.4018/jdwm.2007070101.
7. Get Busy with Word Embeddings [Электронный ресурс]. — URL: <https://www.shanelynn.ie/get-busy-with-word-embeddings-introduction/> (дата обращения: 25.04.2020)
8. Efficient Estimation of Word Representations in Vector Space / Т. Mikolov [и др.] // CoRR. — 2013. — Т. abs/1301.3781. — URL: <http://dblp.uni-trier.de/db/journals/corr/corr1301.html#abs-1301-3781>.
9. *Mikolov T., Yih W.-t., Zweig G.* Linguistic Regularities in Continuous Space Word Representations // Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. — Atlanta, Georgia: Association for Computational Linguistics, 06/2013. — P. 746–751. — URL: <https://www.aclweb.org/anthology/N13-1090>.

10. Bag of Tricks for Efficient Text Classification / A. Joulin [и др.] // arXiv preprint arXiv:1607.01759. — 2016.
11. *Fellbaum C.* WordNet: An Electronic Lexical Database. — Bradford Books, 1998.
12. YARN: Spinning-in-progress / P. Braslavski [и др.] // Proceedings of the 8th Global WordNet Conference, GWC 2016. — United States: Global WordNet Association, 2016. — С. 58–65. — ISBN 9789730207286.
13. *Тищенко С. О., Добрица В. П.* Системы распознавания семантически близких слов // Auditorium. — 2018. — 1(17). — URL: <https://cyberleninka.ru/article/n/sistemy-raspoznavaniya-semanticheski-blizkih-slov>.
14. *Kleinberg J. M.* Authoritative sources in a hyperlinked environment // Journal of the ACM. — 1999. — May 1997. — P. 668–677.
15. *Langville A. N., Meyer C. D.* Deeper Inside PageRank // Internet Mathematics. — 2004. — Vol. 1, no. 3. — P. 335–380. — DOI: 10.1080/15427951.2004.10129091. — eprint: <https://www.tandfonline.com/doi/pdf/10.1080/15427951.2004.10129091>. — URL: <https://www.tandfonline.com/doi/abs/10.1080/15427951.2004.10129091>.
16. *Krizhanovsky A.* Automatic forming lists of semantically related terms based on texts rating in the corpus with hyperlinks and categories (In Russian). — 2006. — arXiv: 0606128 [cs]. — URL: <http://arxiv.org/abs/cs/0606128>.
17. *Melnik S., Garcia-Molina H., Rahm E.* Similarity Flooding: A Versatile Graph Matching Algorithm (Extended Technical Report) : Technical Report / Stanford InfoLab. — 06/2001. — No. 2001–25. — URL: <http://ilpubs.stanford.edu:8090/497/>.
18. *Schmid H.* Probabilistic Part-of-Speech Tagging Using Decision Trees // Proceedings of the International Conference on New Methods in Language Processing. — Manchester, UK, 1994.
19. *Schmid H.* Improvements in part-of-speech tagging with an application to German. — 1995.
20. *Straka M., Straková J.* Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe // Proceedings of the CoNLL 2017 Shared Task: Mul-tilingual

- Parsing from Raw Text to Universal Dependencies. — Vancouver, Canada: Association for Computational Linguistics, 08/2017. — P. 88–99. — URL: <http://www.aclweb.org/anthology/K/K17/K17-3009.pdf>.
21. Вычисление семантических ассоциатов [Электронный ресурс]. — URL: <https://rusvectors.org/ru/associates/>. (дата обращения: 15.03.2020)
22. *Kutuzov A., Kuzmenko E.* WebVectors: A Toolkit for Building Web Interfaces for Vector Semantic Models // Analysis of Images, Social Networks and Texts: 5th International Conference, AIST 2016, Yekaterinburg, Russia, April 7-9, 2016, Revised Selected Papers / ed. by D. I. Ignatov [et al.]. — Cham: Springer International Publishing, 2017. — P. 155–161. — ISBN 978-3-319-52920-2. — DOI: 10.1007/978-3-319-52920-2_15. — URL: http://dx.doi.org/10.1007/978-3-319-52920-2_15.
23. Национальный корпус русского языка [Электронный ресурс]. — URL: <http://www.ruscorpora.ru/>. (дата обращения 05.05.2020)
24. Python 3 documentation [Электронный ресурс]. — URL: <https://docs.python.org/3/> (дата обращения 19.02.2020)
25. Rulemma [Электронный ресурс]. — URL: <https://github.com/Koziev/rulemma>. (дата обращения 10.03.2020)
26. *Řehůřek R., Sojka P.* Software Framework for Topic Modelling with Large Corpora // Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. — Valletta, Malta: ELRA, 05/2010. — P. 45–50. — <http://is.muni.cz/publication/884893/en>.
27. Модели RusVectors [Электронный ресурс]. — URL: <https://rusvectors.org/ru/models/>. (дата обращения 14.03.2020)
28. Томита-парсер [Электронный ресурс]. — URL: <https://yandex.ru/dev/tomita/>. (дата обращения 03.03.2020)