Санкт-Петербургский государственный университет

Факультет прикладной математики - процессов управления
Кафедра технологии программирования

Сердюк Юлиан Анатольевич

Выпускная квалификационная работа бакалавра

Классификация статей в системе УДК

Направление 010500
Прикладная математика и информатика

Заведующий кафедрой
к.ф.-м.н., доцент — Сергеев С.Л.

Научный руководитель
к.ф.-м.н., доцент — Добрынин В.Ю.

Рецензент
Инженер по автоматизации,
ООО "Яндекс" — Липенков Г. А.

Санкт-Петербург
2014

Оглавление

ПРИНЯТЫЕ СОКРАЩЕНИЯ И ОБОЗНАЧЕНИЯ **4**

АННОТАЦИЯ **5**

ВВЕДЕНИЕ **6**

Глава 1. О задаче **8**
1.1 Постановка основной задачи 8
1.2 Дополнительные задачи 8
1.3 Описание задачи 9
1.4 Алгоритм поиска релевантного класса 12
1.5 Обзор литературы 13

Глава 2. Подготовка данных для решения задачи **16**
2.1 Составление графа УДК 16
2.2 Отбор обучающих и тренировочных статей. 17
2.2.1 Предобработка статей. 18
2.2.2 Метод фильтрации с использованием LDA. 20
2.2.3 Метод фильтрации с использованием косинусной меры. 23
2.2.4 Сравнение фильтров. 24

Глава 3. Построение классификаторов **25**
3.1 Описание классификаторов 25
3.1.1 Метод опорных векторов 25
3.1.2 Случайный лес 27
3.1.3 Метод Роккио 27
3.1.4 Метод тематики слов 28
3.2 Сравнение классификаторов 29
3.2.1 Использование локального контекста для метода тематики слов 30
3.2.2 Типичные ошибки 31
3.3 Метод выявления ошибок на одном из уровней. 33
3.4 Результат использования тематики слов. 33
3.5 Выбор ключевых слов. 34

3.6 Поиск похожих статей. 36

ВЫВОДЫ **37**

ЗАКЛЮЧЕНИЕ **39**

СПИСОК ЛИТЕРАТУРЫ **40**

Принятые сокращения и обозначения

В данном списке перечисляются сокращения, используемые в работе:

SVM - Support Vector Machines (метод опорных векторов);
RF - Random Forest (случайный лес)
kNN - k Nearest Neighbors (k ближайших соседей);
TF - Term Frequency (частота слова);
DF - Document Frequency (документная частота);
IDF - Invert Document Frequency (обратная документная частота);
MI - Mutual Information (взаимная информация);
LDA - Latent Dirichlet Allocation (латентное размещение Дирихле);

$\propto$ - символ пропорциональности.

Аннотация

В данной работе рассматривается вопрос иерархической классификации научных статей в системе УДК. В качестве обучающих используются статьи, полученные с помощью поисковой системы «Яндекс». Эксперименты проводятся с использованием популярных методов машинного обучения, таких как «метод опорных векторов», «случайный лес» и модификация алгоритм Роккио. Также предлагается новый метод классификации, основанный на математическом ожидании количества слов, посвященных каждому из классов. В качестве вспомогательных подходов используются поиск взаимной информации, построение модели LDA и расчет меры perplexity, сравнение статей через косинусную меру. Рассматривается вопрос поиска ключевых слов, основанный на равномерности распределения слов. Описывается метод поиска похожих статей для статьи запроса, основанный на разбиении множества статей на кластеры.

Введение

С давних времен люди, в целях облегчения поиска, систематизировали хранимую текстовую информацию. К примеру, книги в библиотеке найти проще, если они отсортированы по автору или по названию. Также люди разбивают хранимые ресурсы на классы. Художественную литературу по жанрам, а научные статьи по разделам.

С ходом развития науки количество статей увеличивалось, как и количество различных научных областей. Поэтому встал вопрос о создании общепринятой системы классификации, которая могла бы развиваться, отвечая всем современным требованиям. Одной из попыток её создания стала опубликованная в 1876 году Мелвилом Дьюи «Классификация и предметный указатель для каталогизации и расположения книг и брошюр в библиотеке», или ДКД – десятичная классификация Дьюи (DDC – Dewey Decimal Classification). В наше время ДКД, с изменениями, принята в западных странах.

В 1897 Поль Отле и Анри Лафонтен опубликовали свою версию классификации, являющуюся переработкой ДКД. Эта система получила название УДК – универсальная десятичная классификация (UDC – Universal Decimal Classification). УДК иногда называют "европейской версией ДКД". Она широко используется во всём мире, включая Россию. Многие издания (включая «Вестник СПбГУ») требуют, чтобы авторы при подаче статьи указывали её номер УДК. Однако, это происходит не во всех издательствах, поэтому, при публикации их статей в электронных библиотеках возникают определенные трудности, связанные с тем, что тематика отдельно взятой статьи не известна. Многим издательствам, которые не требуют указания номера УДК, удобно иметь автоматизированное средство определения тематики статьи, чтобы отправить её к нужному рецензенту. Также, авторам статей, которые впервые сталкиваются с УДК, может быть сложно найти требуемый раздел. К примеру, чтобы найти раздел распознавания лиц, нужно пройти следующий путь:

"Общий отдел. Наука и знание. Информация. Документация..."
"Общие вопросы науки и культуры"
"Информационные технологии. Вычислительная техника. Обработка данных"

"Прикладные информационные (компьютерные) технологии..."
"Распознавание и преобразование образов".

Для новичка подобный путь может оказаться неочевидным.

Целью данной работы является исследование методов классификации статей среди основных разделов системы УДК. УДК имеет иерархическую структуру: её можно представить в виде графа без циклов. На первом уровне классификации находятся 9 наиболее общих разделов, такие как «Общественные науки», «Математика. Естественные науки» , «Искусство. Развлечения. Зрелища. Спорт». Далее каждый раздел разбивается на подклассы. К примеру, потомками «общественных наук» в иерархии являются «Методы общественных наук», «Политика», «Этнография. Жизнь народа. Обычаи. Образ жизни. Фольклор» и т.д. Данный вид классификации принято называть иерархическим. При такой систематизации для статьи-запроса обычно находят наиболее релевантный класс на первом уровне. Затем поиск происходит по классам, которые являются потомками релевантного класса на первом уровне и т.д.

Результатом работы метода будет являться код в УДК класса, который был определен как релевантный и который не имеет потомков.

Следует отметить, что в данной дипломной работе понятия «класс» и «вершина» часто рассматриваются как эквивалентные.

Глава 1

О задаче

1.1 Постановка основной задачи

Пусть задано множество обучающих документов $D = \{d_1, d_2, ..., d_n\}$. Пусть также задано дерево $G = (C, E)$, где E - множество ребер данного графа, а C - множество вершин, причем каждая вершина соответствует одному классу. Каждой вершине графа G отнесено несколько документов их D. С помощью этих данных требуется построить классификатор, который сможет входной документу Q поставить в соответствие 1 или несколько вершин из G, не имеющих потомков, причем считается, что если Q релевантен классу $c \in C$, то он также релевантен всем предкам c.

Качество полученного метода будет оцениваться с помощью тестовых документов, для которых заранее известен их класс из C. В качестве оценок метода будут выступать точность и F-мера.

1.2 Дополнительные задачи

Часто оказывается, что определения класса УДК для статьи недостаточно для того, чтобы определить её тему и содержание. Поэтому, в качестве дополнительных задач было решено поставить задачи определения ключевых слов для статьи, а также поиск похожих статей.

Постановка задачи поиска ключевых слов

Пусть задано множество обучающих документов $D = \{d_1, d_2, ..., d_n\}$. На основе данных, полученных из обучающих статей, выделить из входного документа Q список слов $\{w_1, w_2, ..., w_k\}$, которые в наибольшей мере характеризуют данный документ.

Постановка задачи поиска похожих статей

Пусть задано множество документов $D = \{d_1, d_2, ..., d_n\}$. Для входного документа Q найти подмножество документов $\{d_{i_1}, d_{i_2}, ..., d_{i_m}\}$, посвященных той же теме, что и документ Q.

1.3 Описание задачи

В информационном поиске можно выделить два типа классификаций: «плоская классификация» (или просто, классификация), при которой требуется выделить 1 или несколько классов из ряда равнозначных, и «иерархическая классификация», в которой классы имеют иерархическую структуру и могут быть представлены в виде графа. Иерархическая классификация обычно сводится к последовательности «плоских» классификаций. Поскольку УДК имеет иерархическую структуру, большинство классов имеют предков и потомков, то классификация статей в УДК будет являться иерархической.

Вопрос иерархической классификации не раз поднимался в литературе. В постановке задачи данной дипломной работы говорится, что граф G является деревом. Вообще говоря, это не всегда верно. В некоторых классификациях классы могут быть представлены в виде ориентированного графа без циклов, в котором, если из вершины $c_1 \in C$ идёт дуга в вершину $c_2 \in C$, то c_1 считается предком c_2. Преимущество данного вида представления иерархической классификаций в том, что у вершин могут быть несколько предков.

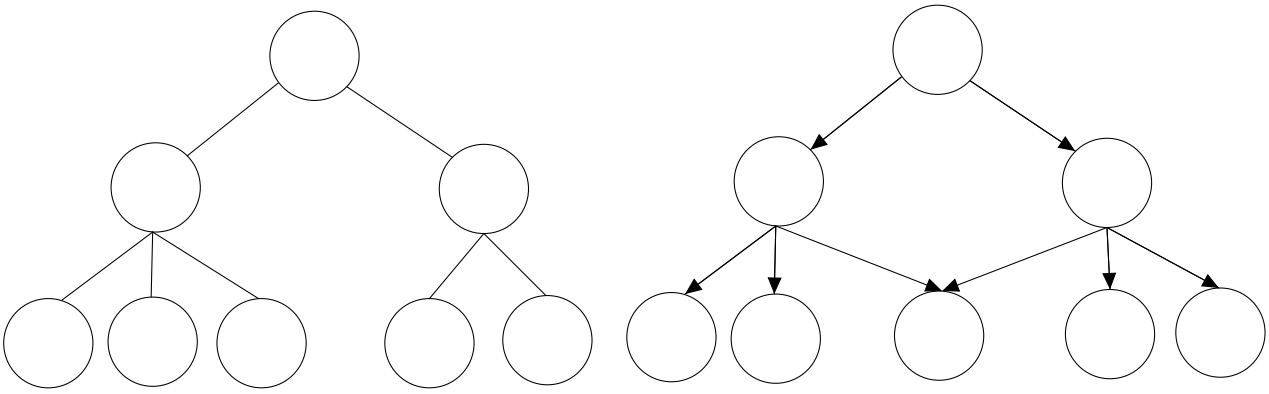

Рис. 1.1: Пример классификации, представляемой в виде дерева(слева) и в виде ориентированного графа(справа).

Можно выделить два основных подхода к постройке иерархических классификаторов. В первом подходе вся модель обучается в целом и иерархическая классификация сводится к одной глобальной плоской классификации среди всех вершин графа. В данном случае ответ находится за один шаг. К преимуществам данного подхода можно отнести его скорость и простоту. Однако, он не имеет широкого применения, поскольку обладает низкой точностью и не использует все преимущества, которые предоставляет иерархическая структура классов.

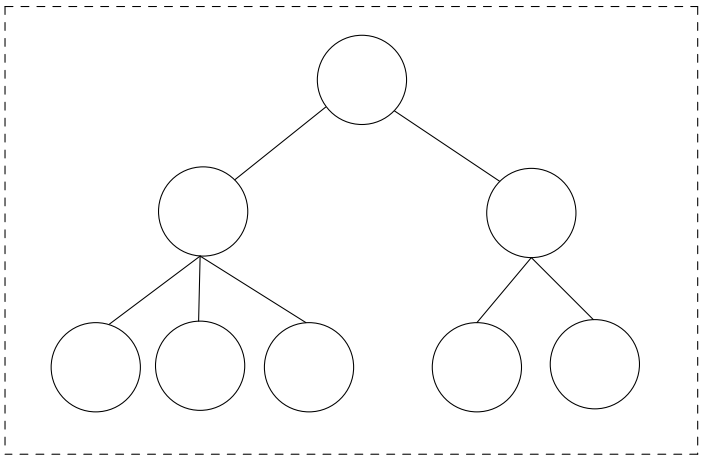

Рис. 1.2: Обучение всей модели в целом.

Второй метод является более стандартным для данного типа задач. В литературе его обычно называют «спуском сверху вниз». Процесс классификации начинается с корня. Сначала рассматриваются классы первого уровня. Потом, в зависимости от результатов, полученных на первом уровне, переходят к некоторым классам второго уровня и т. д.

Здесь также можно выделить два основных подхода. При первом подходе, наиболее популярном для данного класса задач, для каждой вершины строится два множества документов: «положительное» множество документов, каждый документ которого соответствует данной вершине, и «отрицательное» множество, документы которого не соответствуют данной вершине. Различные методы используют различные подходы к выбору документов для «отрицательного» множества. Для каждой вершины строится бинарный классификатор (к примеру, метод опорных векторов), который определяет, соответствует ли данный документ рассматриваемой вершине (был ли он отнесен к «положительному» множеству). Если документ определяется как соответствующий, осуществляется переход к потомкам данного класса.

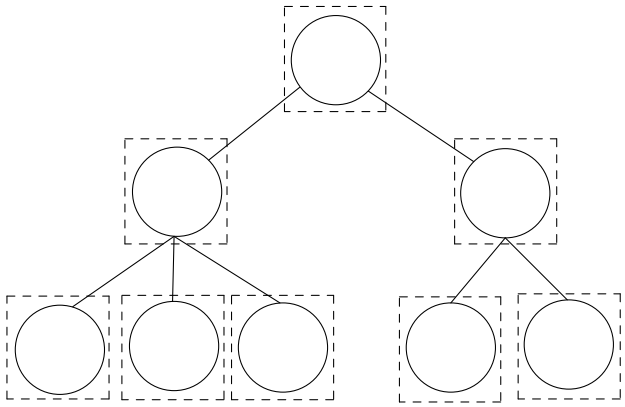

Рис. 1.3: Обучение каждой вершины отдельно.

При втором подходе метода спуска сверху вниз, проводится плоская классификация среди потомков рассматриваемой вершины. Процесс классификации начинается с корня графа, потомками которого считаются классы первого уровня. В данной работе используется именно этот метод.

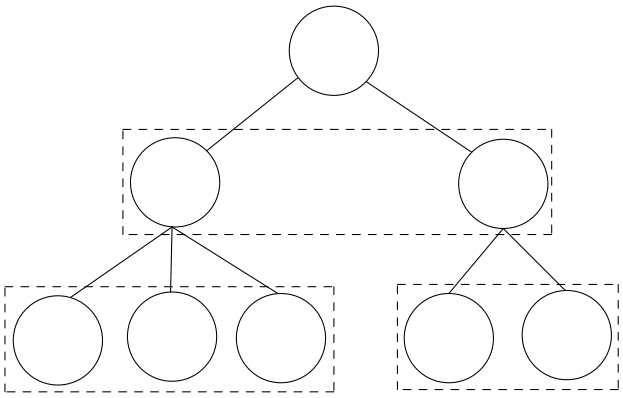

Рис. 1.4: Обучение по уровням.

Также в формулировке задачи было сказано, что процесс классификации заканчивается тогда, когда у текущей вершины нет потомков. В общем случае это также неверно, поскольку в некоторых задачах иерархической классификации может потребоваться остановка на промежуточном уровне. Данное условие связано со структурой УДК: каждый класс разбивается на набор подклассов, которые в совокупности дают исходный класс. Поэтому используется утверждение, что если удалось определить, что документ соответствует вершине с потомками, то получится уточнить, к какому из потомков относится данная статья (к какому разделу математики, если это математика; какому языку, если это статья по филологии; какой религии и т.д.).

1.4 Алгоритм поиска релевантного класса

Ниже приведен псевдо-код алгоритма, который ищет релевантный класс для документа Q. Преимущество иерархической классификации — возможность определить ошибку на одном из шагов. Если уверенность в верности выбора класса на шаге n необоснованно ниже, чем уверенность в верности выбора класса на шаге $n-1$,тогда, возможно, была допущена ошибка на шаге $n-1$ и следует выбрать другой класс.

Используемые функции:
Предок(N) - возвращает предка вершины N. Если вершина N - корень, возвращает N.
Потомки(N) - возвращает список потомков вершины N.
Мера(Q, N) - возвращает число от 0 до inf, отражающее релевантность

документа Q классу N.
Принятие(Мера1, Мера2) - возвращает ИСТИНА или ЛОЖЬ. Мера1 - значение меры на предыдущем шаге, Мера2 - на текущем.

Входные данные : Запрос Q
Выходные данные: Класс N
Переменные : N - текущий класс;
max - текущее максимальное значение меры;
$C_{\texttt{max}}$ - потомок с максимальной мерой;
p_max - предыдущее максимальное значение;
E - множество исключенных из рассмотрения классов;

```
1  N = корень дерева;
2  p_max = 0;
3  while Множество {Потомки(N) \E} не пустое do
4  |   max = 0;
5  |   C_max = NULL;
6  |   for C в {Потомки(N) \E} do
7  |   |   if Мера(Q, C) > max then
8  |   |   |   C_max = C;
9  |   |   |   max = Мера(Q, C);
10 |   |   end
11 |   end
12 |   if Принятие(p_max, max) then
13 |   |   N = C_max;
14 |   |   p_max = max;
15 |   else
16 |   |   Поместить N в E;
17 |   |   N = Предок(N);
18 |   end
19 end
20 return N
```

Алгоритм 1: Псевдо-код поиска класса.

Цель данной работы - нахождение оптимальных функций Мера(Q, N) и Принятие(Мера1, Мера2).

1.5 Обзор литературы

Одной из русскоязычных статей, посвященных иерархической классификации, является статья [1], написание которой поддерживалось компанией «Яндекс». В данной работе рассматривается вопрос автоматического пополнения веб-каталога англоязычными страницами. В качестве набора

документов и веб-каталога взяты реальные данные, предоставленные компанией «Яндекс». Автор статьи рассматривает многие популярные методы машинного обучения (SVM, kNN, RF и т.д.), а также несколько методов предобработки текстов (стемминг, отбор существительных, удаление стопслов и т.д.). Проводятся эксперименты, на основе которых делаются выводы о предпочтительных методах классификации и обработки на каждом из уровней.

В статье [2] проводится исследование методов машинного обучения и предобработки данных для решения задачи иерархической классификации. В качестве данных взяты статьи, предоставляемые международного проекта по исследованию методов иерархической классификации текстов (LSHTC, Large Scale Hierarchical Text classification). В статье определяется ряд закономерностей качества и времени работы алгоритмов в зависимости от используемых методов. Также, приводятся описание алгоритмов-победителей LSHTC.

Хороший обзор англоязычной литературы, посвященной иерархической классификации, представлен в [3]. В работе рассматриваются современные подходы к решению различных вариантов задачи иерархической классификации, проводится сравнение и перечисление преимуществ и недостатков различных методов, находятся границы применимости.

В работе [4] задача мультиклассовой иерархической классификации сводится к использованию жадного алгоритма, который выделяет оптимальный подграф. Ключевая идея заключается в создании “супервершин” — вершин, которые конденсируют в себя несколько вершин. Предлагается алгоритм поиска такой супервершины, что добавление к ней новой вершины не повышает значение меры релевантности для рассматриваемого документа. В этом случае полученная супервершина считается результатом классификации, а вершины, входящие в неё – подграфом. Вершины найденного графа будут являться классами, релевантными рассматриваемому документу.

В статье [5] описывается алгоритм классификации текстов для для классификаций-деревьев. Идея работы заключается в наблюдении о том, что классы на низком уровне имеют много общих признаков, в то время как на верхних уровнях классы отличаются довольно сильно. Метод, описанный в статье, призван решил проблему классификации на низких уровнях. Авторы для каждой из вершин выделяют набор признаков так, чтобы каждый класс имел набор признаков как можно более ортогональный к набору признаков остальных классов на том же уровне.

В статье [6] описывается применение иерархической классификации для определения возраста человека по фотографии. Авторы в своих экспериментах получили, что методы определения возраста за 1 шаг имеют более низкую точность, чем метод иерархического определения, когда все возрасты собираются в группы (к примеру, 15-25 лет, 25-40 лет и т.д.). Также

авторы исследуют методы улучшения классификации. В качестве классификаторов используются SVM и SVR.

Глава 2

Подготовка данных для решения задачи

2.1 Составление графа УДК

Как следует из постановки задачи, требуется составить граф $G = (C, E)$, который соответствует классификации УДК. Всего классификация УДК включает в себя более 100.000 разделов, из которых 1913 считаются основными. В связи с тем, что получить качественную обучающую коллекцию, покрывающую все классы, достаточно сложно, было решено воспользоваться только основными разделами УДК, которые находятся в открытом доступе на официальном сайте УДК. Для каждого раздела из числа основных (за исключением 9 главных) указан раздел, который является его родителем, а также все его потомки (если таковые имеются). Имея эти данные можно легко составить граф G. Однако, граф, составленный на основе основе этих данных, не будет являться деревом, поскольку в нём не существует пути между разделами, не имеющих ни общих предков, ни общих потомков (к примеру, 1 и 8). Также следует учесть, что классификатор, выбирая подходящий раздел, на каждом шаге будет выбирать среди потомков текущего раздела. Поэтому для удобства к графу G добавляется фиктивная вершина N, которая не соответствует ни одному классу из разделов УДК, и которая является предком всех 9 главных разделов. После добавления вершины N граф G становится деревом, и алгоритм поиска класса будет начинаться именно с вершины N.

Построенный граф обладает следующими свойствами:

Средняя глубина дерева:	5
Количество вершин:	1913
Количество листьев:	1465
Среднее число потомков:	4,25

2.2 Отбор обучающих и тренировочных статей.

В соответствии с постановкой задачи, каждой вершине графа G требуется поставить в соответствие множество статей. Для этих целей было решено взять статьи с номерами УДК, указанными авторами. На этом моменте следует учесть ту особенность УДК, что на 1 и 2 уровнях расположены классы, которые имеют достаточно общее название. Подбирая номера УДК для своих статей, авторы обычно используют классы, расположенные не выше 3 уровня. Поэтому было решено разбить процесс присваивания документов вершинам на несколько этапов. На первом этапе документы присваиваются тем вершинам, которые не имеют потомков и номера которых авторы присваивают своим статьям. На втором этапе происходит отсеивание шумовых и ошибочно присвоенных статей. На последующих этапах происходит "обратное наследование" статьей от потомков к предкам. Таким образом, каждой вершине, имеющей потомков, должны быть присвоены наиболее типичные документы её потомков (к примеру, коду "51 - Математика" присваивать наиболее общие статьи из теории вероятностей, геометрии и т. д.).

В качестве источников научных статей выступало два ресурса: электронная библиотека «Cyberleninka» и поисковая система «Яндекс». «Cyberleninka» специально для целей данной работы предоставила 5000 научных статей, около 2200 из которых имеют номер УДК. Эти статьи использовались в качестве тестовых. Однако, для обучения классификатора требовалось не менее 45000 статей, поэтому было решено воспльзоваться автоматическими средствами, а именно поисковой системой «Яндекс». В начале экспериментов использовались все документы, возвращаемые в качестве результатов, но, как показали тесты, документы формата .html имеют низкое качество и методы не способны обучаться на них. Поэтому, в качестве результатов принимались только те документы, которые имели формат .pdf, .doc и .docx. Запросы имели следующий вид: описание + " УДК "+ номер раздела. К примеру, для получения статей по теории вероятностей был составлен следующий запрос: "Теория вероятностей и математическая статистика УДК 519.2". К сожалению, полученные таким образом статьи всё равно содержали много шума: некоторые статьи имели иной УДК, а некоторые, имеющие требуемый УДК, не говорили о том, что указывалось в описании УДК.

Встает вопрос: какую использовать меру для отбора наиболее "типичных" статей для данной области. Для этих целей было решено, после стадии предобработки документов, сравнить два различных метода, один из которых основан на использовании модели LDA и применении меры perplexity,

а второй основыван на косинусной мере сравнения векторов.

2.2.1 Предобработка статей.

Перед тем, как начать работу с полученными статьями, требуется сделать начальную предобработку загруженных статей. Она включает в себя:

- Извлечение текста из статей в формате .doc, .docx, .pdf.
- Токенизация текстов.
- Стемминг или лемминг.
- Составление словаря коллекции.

Извлечение текста: поскольку выделить текст из файлов в формате .doc, .docx и .pdf намного сложнее, чем из простого .txt файла, было решено воспользоваться сторонними библиотеками, предоставляющими возможность извлечения текста. Для работы с файлами .doc и .docx была использована библиотека *Apache POI*, а для работы с .pdf – *jPOD*.

Токенизация текстов: одна из первых задач, встающих на стадии предобработки текстов — токенизация текста, разбиение их на слова. Обычно, эта задача сводится к приведению текста к единому регистру, удалению большинства знаков препинания (за исключением некоторых символов, к примеру, дефис "-") и разбиение на слова по пробелам. Однако, в случае работы с научными статьями эта задача будет нескольку усложнена тем, что в текстах используются переносы слов, которых бывает достаточно много (если, к примеру, статья написана в не одной широкой колонке, а в двух).

Если во время обработки в тексте было найдено слово, оканчивающееся символом "-" и стоящее на последней позиции в строке, то это слово, вместе с первым словом следующей строки, помещается в особый список слов. После обработки всех текстов, требуется вернуться к этому списку, и на основе слов, которые были найдены и точно определены, принять решение о том, принять это слово как перенос или как слово, внутри которого стоит дефис (к примеру, гамма-функция).

Стемминг и лемминг: поскольку в русском языке многие слова имеют несколько различных форм, которые требуется рассматривать как одно и то же слово, встает вопрос о нахождении слов, которые являются словоформами одного слова, и приведения этих слов к общей форме. Для этих целей обычно используются алгоритмы стемминга и лемминга.

Стемминг – это процесс удаления окончаний слов и выделение его основной части по набору правил, составленных так, чтобы различные словоформы одного слова приводились к одной форме. Наиболее популярный

алгоритм стемминга – стеммер Портера, который имеет реализации для многих языков.

Лемминг – это процесс нормализации текста, приведение его слов к нормальной форме. Этот процесс обычно требует больших затрат, чем стемминг. Также, для лемматизации требуется наличие словаря языка. Однако, он дает намного более качественные результаты, чем стемминг.

После проведения ряда тестов, было выявлено, что применение стемминга для научных русскоязычных статей в большинстве случаев не приводит к требуемым результатам, а именно различные словоформы не приводятся к единой форме. Это связано с тем, что в русском языке используется большое множество суффиксов, окончаний, приставок, и создание единого алгоритма приведения слов к единой форме не представляется возможным. Поэтому для исследований применялась лемматизация.

Для лемматизации русскоязычных текстов часто используется программа «MyStem», которая на данный момент принадлежит компании «Яндекс» и доступна для некоммерческого использования. Преимущество данной программы в том, что она не только находит начальную форму слов присутствующих в словаре, но также и строит гипотезы для отсутствующих слов [9].

Составление словаря требуется для того, чтобы у каждого слова, которое может быть найдено в тексте, был свой порядковый номер. Тогда все статьи, как которые имеются в базе, так и те, для которых требуется найти класс, можно будет отобразить в точки единого пространства R^n, где n – размер словаря. Также, для каждого слова из словаря находятся 2 значения: количество вхождений слова в документы (tf) обучающей выборки и количество документов, содержащих данное слово (df).

В итоге, после предобработки статей, получаются новые документы, состоящие из начальных форм слов исходных текстов, а также словарь коллекции. В таблице ниже показаны примеры размеров словарей для различных данных. В документы “Поиск, все форматы” включены документы, которые выдаются поисковой в качестве результатов запроса, независимо от их формата. В категорию “Поиск, только .pdf .doc .docx” включены документы, выдаваемые поисковой системой в качестве результатов и, имеющие соответствующее расширение.

Данные	**Количество документов**	**Размер словаря**
Статьи из cyberleninka	5000	75 650
Поиск, все форматы	16819	201 086
Поиск, только .pdf .doc .docx	48394	491 511

Таблица 2.1: Размеры словарей для различных данных

Стоит отметить, что в словарь входят не только слова русского языка, но и английского (поскольку некоторые статьи были написаны на английском языке).

2.2.2 Метод фильтрации с использованием LDA.

LDA

Модель LDA — это порождающая модель, которая с помощью неявных групп помогает объяснить, почему некоторые части наблюдений схожи[7][8]. К примеру, если рассматривается коллекция документов, то в LDA каждый документ будет рассматриваться как набор различных тематик. С помощью LDA можно для всех слов, встречающихся в документах коллекции, найти вероятности того, что это слово относится к той или иной тематике. Для этого требуется выбрать количество тем K и значения для гиперпараметров α и β. Низкое значение α говорит, что имеется предположение о том, что каждый документ содержит небольшое количество тем. Высокое значение α говорит, что каждый документ, скорее всего, содержит большинство тематик. Соответственно, низкий параметр β говорит, что каждая тема должна включать небольшое количество слов. Опытным путем было получено, что часто оптимальными значениями являются $\alpha = 50/K$ и $\beta = 0.01$. Определив количество тем и гиперпараметры распределения, можно распределить слова по тематикам. Для начала, каждое слово относится к случайной теме. Положим M - количество документов в коллекции, N_m - длина документа m, N - сумма всех N_m, V - размер словаря коллекции, $\vec{w}$ - вектор, в начале которого стоят по порядку слова первого документа с учетом повторений, затем слова второго документа и т. д., $\vec{z}$ - вектор, на i-ой позиции которого стоит тема, к которой относится слово w_i. Рассмотрим слово t, стоящее на позиции i документа m, и для каждой темы k находим вероятности того, что это слово относится к данной теме:

$$p(z_i = k|\vec{z}_{\neg i}, \vec{w}) = \frac{p(\vec{w}, \vec{z})}{p(\vec{w}, \vec{z}_{\neg i})} = \frac{p(\vec{w}|\vec{z})}{p(\vec{w_{\neg i}}|\vec{z}_{\neg i}) \cdot p(w_i)} \cdot \frac{p(\vec{z})}{p(\vec{z}_{\neg i})}$$

$$\propto \frac{n^{(t)}_{k,\neg i} + \beta}{\sum_{t=1}^{V} \left(n^{(t)}_{k,\neg i} + \beta\right)} \cdot \frac{n^{(k)}_{m,\neg i} + \alpha}{\sum_{k=1}^{K} \left(n^{(k)}_{m} + \alpha\right) - 1}$$

$$\propto \frac{n_{k,\neg i}^{(t)} + \beta}{\sum_{t=1}^{V}\left(n_{k,\neg i}^{(t)} + \beta\right)} \cdot \left(n_{m,\neg i}^{(k)} + \alpha\right),$$

где $\vec{w} = \{w_i = t, \vec{w}_{\neg i}\}, \vec{z} = \{z_i = k, \vec{z}_{\neg i}\}$;
$n_k^{(t)}$ - количество раз, с которым слово t встречается с темой k;
$n_m^{(t)}$ - количество раз, с которым слово t встречается в документе m;
$n_m^{(k)}$ - количество раз, с которым тема k встречается в документе m.
Символ $\neg i$ означает, что рассматриваются все позиции, кроме i - ой. После нахождения вероятности для каждой темы, это слово случайно относится к одной из тем в соответствии с полученными вероятностями. Далее рассматривается следующее слово $(i+1)$ документа (m). Потом рассматриваем все слова второго документа $(m+1)$ и так далее, затем процесс начинается заново, и так несколько раз. Распределив слова по темам, находятся значения для матриц Θ размерности $M \times K$, распределения тем по документам, и Φ размерности $K \times V$, распределения слов по темам, по формулам:

$$\varphi_{k,t} = \frac{n_k^{(t)} + \beta}{\sum_{t=1}^{V}\left(n_k^{(t)} + \beta\right)} \quad \vartheta_{m,k} = \frac{n_m^{(k)} + \alpha}{\sum_{k=1}^{K}\left(n_m^{(k)} + \alpha\right) - 1}.$$

Будем говорить, что $\mathcal{M} = \{\Theta, \Phi\}$ - модель нашего языка.

Perplexity

Вместе с LDA часто рассматривается мера Perplexity, характеризующая способность модели $\{\Theta, \Phi\}$ генерировать документы, или, проще говоря, говорит, какова правдоподобность того, что коллекция документов $\mathcal{W}$ могла появиться в языке, описываемого моделью $\mathcal{M}$. Perplexity имеет два основных применения — оценка качества модели $\mathcal{M}$ и соответствия коллекции $\mathcal{W}$ этой модели. В качестве примера первого применения можно рассмотреть случай, когда мы, имея коллекцию из 10.000 документов, построим модель $\mathcal{M}$ с помощью 8.000 документов, скрыв 2.000 документов, а потом будем проверять, насколько хорошо данная модель описывает документы, которые были первоначально скрыты. Второе применение возможно, когда имеется модель языка, которая хорошо описывает документы какой-то тематики, и требуется проверить, соответствует ли коллекция $\mathcal{W}$ этой тематике. Мера Perplexity для коллекции $\mathcal{W}$ при учете модели $\mathcal{M}$ определяется как

$$P(\mathcal{W}|\mathcal{M}) = \prod_{m=1}^{M} p(\vec{w}_m|\mathcal{M})^{-\frac{1}{N}} = \exp\left(-\frac{\sum_{m=1}^{M} \log p(\vec{w}_m|\mathcal{M})}{\sum_{m=1}^{M} N_m}\right),$$

$$p(\vec{w}_m|\mathcal{M}) = \prod_{t=1}^{V} \left(\sum_{k=1}^{K} \varphi_{k,t} \cdot \vartheta_{m,k}\right)^{n_m^{(t)}}.$$

Чем ниже значение Perplexity, тем более коллекция $\mathcal{W}$ соответствует модели $\mathcal{M}$.

Для поиска типичных статей и отсеивания шумовых было решено применить следующий метод: пусть $\mathcal{D}$ - множество документов для вершины N. Рассмотрим произвольный документ $d \in \mathcal{D}$. Построим модель LDA $\mathcal{M}$ для множества $\mathcal{D}\backslash d$. После этого находится значение меры Perplexity для модели $\mathcal{M}$ и документа d (в данном случае предполагается, что коллекция $\mathcal{W}$ состоит из одного документа). Найденное значение меры Perplexity будет говорить о том, насколько документ d соответствует модели вершины N.

Используя описанный метод, находятся значения меры Perplexity для каждой из вершин и отсеиваются некоторые статьи с наибольшим значением (а иногда целые разделы, если значения Perplexity для документов данного раздела намного больше среднего значения по другим разделам).

Как уже было сказано выше, LDA способен самостоятельно отбирать слова по темам, собирая неважные слова в одни темы (к примеру, стоп-слова). Однако, у LDA есть один существенный недостаток – качественные модели строятся достаточно долго. Для того, чтобы модель $\mathcal{M}$ стала качественной, обычно нужно провести тысячи итераций. В случае, если модель строится на большой коллекции документов и/или эти документы имеют большие размеры, то процесс может занять продолжительный период времени. По оценочным расчетам, для того, чтобы посчитать меру Perplexity для каждого документа из обучающего множества, делая для каждой модели по 1000 итераций, потребовалось бы 2 месяца непрерывной работы. Поэтому встал вопрос о методах снижения размеров документов. Для этих целей было решено использовать взаимную информацию между словами и классами.

Взаимная информация

Понятие меры взаимной информации пришло из теории информации. MI — это статистическая функция двух величин, равная разности энтропии случайной величины и условной энтропии двух случайных величин:

$$\mathtt{MI}(x,y) = H(x) - H(x|y) = H(y) - H(y|x) = H(x) + H(y) - H(x,y)$$

Предположим, что имеется множество документов $\mathcal{D}$, каждый из которых присвоен одному классу из $\mathcal{C}$. Пусть $\mathcal{V}$ – множество слов, встречаемых в документах из $\mathcal{D}$. Рассмотрим $t \in \mathcal{V}$ и $c \in \mathcal{C}$. Взаимная информация $\mathrm{MI}(t,\ c)$ между словом t и классом c будет определяться по следующей формуле:

$$\mathtt{MI}(t,c) = \sum_{e_t \in \{0;1\}} \sum_{e_c \in \{0;1\}} \mathtt{P}\left(U = e_t, C = e_c\right) \log_2 \frac{\mathtt{P}\left(U = e_t, C = e_c\right)}{\mathtt{P}\left(U = e_t\right) \mathtt{P}\left(C = e_c\right)}.$$

U и C – случайные переменные, которые принимают значения 0 и 1. $U = 1$ означает, что документ содержит слово t, и $U = 0$ иначе. $C = 1$ означает, что документ принадлежит классу c, и $C = 0$ иначе. К примеру, вероятность $\mathtt{P}\left(U = 0, C = 1\right)$ будет равна вероятности того, что случайно взятый документ из $\mathcal{D}$ не будет содержать слово t и будет принадлежать классу c.

В нашем случае $\mathcal{C}$ – это множество категорий, а $\mathcal{D}$ – множество документов для этих категорий. Формула взаимной информации будет иметь следующий вид:

$$\mathtt{MI}(t,c) = \frac{N_{11}}{N} \log_2 \frac{N\ N_{11}}{N_{1*}\ N_{*1}} + \frac{N_{10}}{N} \log_2 \frac{N\ N_{10}}{N_{1*}\ N_{*0}} + \\ + \frac{N_{01}}{N} \log_2 \frac{N\ N_{01}}{N_{0*}\ N_{*1}} + \frac{N_{00}}{N} \log_2 \frac{N\ N_{00}}{N_{0*}\ N_{*0}},$$

где N – общее количество документов, N_{11} – количество документов класса c, в которых встречается слово t, N_{10} – количество документов класса НЕ c, в которых встречается слово t, N_{01} – количество документов класса c, в которых НЕ встречается слово t, N_{00} - количество документов класса НЕ c, в которых НЕ встречается слово t.

2.2.3 Метод фильтрации с использованием косинусной меры.

В информационном поиске в качестве меры сравнения документов часто используют косинусную меру.

Предположим, что имеется два документа d_1 и d_2 и множество слов $\mathcal{V} = \{w_1, w_2, ..., w_n\}$ такое, что любое слово t, входящее в документ d_1 или d_2, входит также в множество $\mathcal{V}$.

Представим документы d_1 и d_2 в виде векторов v_1 и v_2 размерностью n. Значение k-го коэффициента вектора v_1 равно количеству вхождений слова w_k в документ d_1. Аналогично для вектора v_2 и документа d_2.

Пусть $\alpha(v_1, v_2)$ – угол между векторами v_1 и v_2. В качестве меры схожести документов примем функцию

$$\texttt{sim}(d_1, d_2) = \cos\alpha(v_1, v_2) = \frac{v_1 v_2}{|v_1||v_2|}.$$

В соответствии со свойствами функции $\cos\alpha$, получаем, что $0 \leqslant \texttt{sim}(d_1, d_2) \leqslant 1$, причем $\texttt{sim}(d_1, d_2) = 1$ в том случае, если тексты имеют один и тот же набор слов(векторы совпадают по направлению), и $\texttt{sim}(d_1, d_2) = 0$, когда документы не имеют ни одного общего слова (векторы перпендикулярны).

Данной мерой можно воспользоваться для определение шумовых документов. Предположим, что вершине N соответствует множество документов $\mathcal{D} = \{d_1, d_2, ..., d_n\}$. Пусть v_i – вектор, соответствующий документу d_i. Для каждого документа $d_i \in \mathcal{D}$ найдем значение $\cos\alpha\left(v_i, \overline{V}_i\right)$, где $\overline{V}_i = \sum_{j=1, j\neq i}^{n} v_j$. Чем больше значение $\cos\alpha\left(v_i, \overline{V}_i\right)$, тем более документ d_i соответствует рассматриваемому классу.

2.2.4 Сравнение фильтров.

Меры, найденные фильтрами, можно использовать для ранжирования документов по релевантности данной вершине. Документы сортируются по мере в порядке возрастания или убывания, в зависимости от фильтра. Предполагается, что документы, стоящие на первых позициях, должны быть релевантны данной вершине, а стоящие в конце должны быть наиболее далеки от тематики данного класса.

После ранжирования документов и сравнения двух фильтров, было получено, что фильтр, использующий косинусную меру, показывает лучший результат, по сравнению с тем, что показывает LDA. Связано это с тем, что LDA и MI оказались неспособными выделить нужные слова при размере коллекции в пару десятков документов. Также оказывалось, что для некоторых классов чуть меньше половины документов – шумы, поэтому и LDA, и MI часто отбрасывали важные слова и оставляли несущественные.

В то же время, косинусная мера показала хороший результат. Рассматривая все слова как равные и используя усредненные показатели, она смогла достаточно качественно выделять релевантные статьи. Поэтому для экспериментов было решено использовать именно эту меру.

В итоге, вершинам, которым не были присвоены статьи с помощью поисковых запросов, присваивались документы их потомков, имеющие наилучшее значение меры релевантности.

Глава 3

Построение классификаторов

3.1 Описание классификаторов

После того, как была пройдена стадия предобработки и каждой вершине были присвоены документы, можно переходить к стадии постройки классификаторов. SVM и RF — два метода, которые часто использую для решения задач классификации. Также было решено исследовать одну из модификаций алгоритма Роккио, который в экспериментах, проводимыми авторами, показал намного лучшие результаты, чем SVM (54,45% точности против 0,18 %). Низкую точность SVM авторы объяснили большой размерностью пространства свойств (большим размером словаря). Кроме того, в качестве классификатора предлагается новый алгоритм, основанный на предположениях о тематиках слов, входящих в документ.

3.1.1 Метод опорных векторов

Метод опорных векторов – это бинарный классификатор, который был описан Владимиром Вапником[10]. Идея метода состоит в построении разделяющей гиперплоскости с максимальным зазором между классами. Если в исходном пространстве невозможно построить такую гиперплоскость, то данные отображают в пространство большей размерности. Алгоритм работает на предположении о том, что чем больше зазор между классами – тем меньше средняя ошибка классификатора. Новые данные классифицируются на основании своего положения в пространстве относительно разделяющей гиперплоскости.

Наиболее популярным является метод с мягким зазором (soft-margin), при котором некоторые данные из обучающего множества могут оказаться

данными-нарушителями и находиться в полуплоскости противоположного класса(шумовые данные). В общем виде, решение задачи с жестким зазором (hard-margin), при котором все обучающие данные классифицируются верно, не существует.

Одной из популярных реализаций SVM является библиотека *libsvm*, которая изначально входила в программное обеспечение для анализа данных Weka (Waikato Environment for Knowledge Analysis), но после была вынесена как отдельный пакет, который можно подключить к основной программе. На текущий момент в Weka реализован алгоритм SVM, описанный в [11].

Поскольку SVM изначально бинарный классификатор, то для использования SVM в мультиклассовой классификации нужно сделать дополнительные модификации. Существует две основные модификации SVM для данного случая:

1). Построение SVM для каждой вершины с помощью положительных и отрицательных множеств документов.

2). Построение SVM для пар классов, среди которых проходит классификация. В качестве результата выбирается тот класс, за который "проголосует" наибольшее число классификаторов.

Для выбора подхода были проведены два теста на первом уровне иерархии:

1). Для каждой вершины строилось два множества. Первое множество состояло из документов, приписанных данной вершине. Этот класс имел значение "1". Во второе множество входило несколько статей с наибольшим значением `sim` из каждой соседней вершины. Этот класс имел значение "-1". Считалось, что метод сработал верно, если статьи из тренировочного множества относились к "1" в вершинах, соответствующим тем УДК, которые приписывались статье авторами.

2). Строилось несколько классификаторов из пар классов второго уровня иерархии. Считалось, что метод сработал верно, если за тот УДК, к которому приписал статье автор, проголосует наибольшее число классификаторов.

Ниже представлены результаты для данных методов:

	Первый подход	Второй подход
Точность	23%	41%

Как видно из таблицы, второй подход обладает большей точностью по сравнению с первым. Это можно объяснить тем, что во втором методе в каждом из множеств используются статьи, отнесенные к одной вершине, в то время как для первого метода в отрицательное множество входят разные статьи, из-за чего построение разделяющей полуплоскости усложняется. В связи с этим было решено использовать второй подход.

3.1.2 Случайный лес

Случайный лес – метод машинного обучения, основанный на построении множества независимых решающих деревьев [12]. При классификации текстов, каждое дерево из леса голосует за 1 из классов. В качестве результата возвращается класс, за который проголосовало наибольшее число деревьев. Многие исследователи отмечают высокое качества результатов, получаемых методом RF, сравнимое с SVM. Однако, также отмечается, что RF склонен к переобучению и страдает от зашумленных данных [13].

Предположим, что в обучающей выборке имеется N документов, размерность пространства признаков равна M. Тогда из обучающих документов делается выборка с повторением размером n (часто $n = N$). Этот процесс обычно называют bootstrap aggregating . Из M признаков также случайным образом выбирают m признаков (обычно $m = \sqrt{M}$). На основе выбранных документов и свойств строится дерево решений.

Поскольку деревья решений изначально могут работать с любым количеством классов, то для проведения экспериментов не требуется производить дополнительных улучшений.

Реализация RF также есть в Weka.

3.1.3 Метод Роккио

Идея метода Роккио, подобно идее SVM, основывается на построении разделяющей гиперплоскости, но с использованием центроид классов [14]. В статье [15] описывается модификация данного метода, которая для поиска центроид использует tf и idf слов.

Предположим, что вершинам $N_1, N_2, ...N_m$, имеющих общего родителя, соответствует множество документов $\mathcal{D} = \{d_1, d_2, ..., d_n\}$. Рассмотрим слово i документа d_j, который принадлежит классу $c = N_c$. Вес слова i найдем по формуле:

$$w_{ij} = \ln\left(\mathtt{tf}_{ic} + 1\right)\ \mathtt{idf}_i, \tag{3.1}$$

где $\mathtt{tf}_{ic}$ – суммарное число вхождений слова в документы класса c, а $\mathtt{idf}_i = \dfrac{N}{\mathtt{df}_i}$. После этого находятся веса слов для классов:

$$wt_{ic} = \sum_j w_{ij}.$$

Поскольку все тренировочные документы имеют различную длину, то требуется нормализация весов. Нормальные веса для классов находятся по формуле

$$nwt_{ic} = \frac{wt_{ic}}{\sqrt{\sum_i wt_{ic}^2}}.$$

В результате находится вектор весов слов для каждого класса.

Для классификации документа Q требуется составить вектор весов слов $\bar{w_q}$ по формуле (3.1). Далее, вектор $\bar{w_q}$ сравнивается с векторами весов классов, используя косинусную меру, и выбирается наилучший класс.

3.1.4 Метод тематики слов

В данной работе в качестве метода, решающего задачу классификации статей, предлагается новый метод, который основывается на предположении о том, что класс документа определяется не столько тем, какие слова в нём встречаются, сколько тем, какой теме посвящены эти слова. В данном случае каждый класс рассматривается как отдельная тема, которой могут быть посвящены слова. Бахтин в своих трудах писал, что слова имеют множество смыслов, и конкретное значение слово может обрести только в контексте. Это говорит о том, что нам нельзя приписать каждому слову один единственный смысл, нужно рассматривать возможность того, что в разных документах одно и то же слово может встречаться по-разному.

Предположим, что вершинам $N_1, N_2, ...N_m$, имеющих общего родителя, соответствует множество документов $\mathcal{D} = \{d_1, d_2, ..., d_n\}$. Рассмотрим слово i и класс $c = N_c$. Эмпирическая условная вероятность того, что слово i употребляется в контексте c равна:

$$\mathtt{P}(c|i) = \frac{\mathtt{P}(i|c)\mathtt{P}(c)}{\mathtt{P}(i)} = \frac{\frac{tf_{ic}}{tf_c}\frac{tf_c}{V}}{\frac{tf_i}{V}} = \frac{tf_{ic}}{tf_i},$$

где tf_{ic} – количество вхождений слова i в документы класса c, tf_c – суммарное вхождение всех слов в документы класса c, tf_i – суммарное количество вхождений слова i во все документы, V – сумма количеств слов в документах из $\mathcal{D}$.

При классификации документа $Q = \{w_{q1}, w_{q2}, ..., w_{qk}\}$ в качестве результата возвращается такое значение $c_q \in N_1, N_2, ...N_m$, что

$$\mathcal{L}(c_q|Q) = \max_c \mathcal{L}(c|Q) = \max_c \sum_i \mathtt{P}(c|w_{qi}).$$

Таким образом, величина $\mathcal{L}(c|Q) = \sum_i \mathtt{P}(c|w_{qi})$ есть математическое ожидание количества слов в документе Q, посвященных теме c.

3.2 Сравнение классификаторов

Для определения лучшего классификатора проводились эксперименты по плоской классификации на каждом из уровней классификации. Классификатор, показавший лучшие результаты на каком-то из уровней предполагалось использовать в дальнейшем для работы с данным уровнем.

В ходе экспериментов было обнаружено, что все классификаторы показывают низкие результаты на первом уровне иерархии, на котором находятся 9 главных категорий. Это связано со структурой УДК. К примеру, потомками пятого раздела "*Математика. Естественные науки.*" являются "*Математика*", "*Физика*", "*Зоология*", "*Ботаника*", "*Химия*" и т. д. Создать набор статей, на основе которого классификатор смог бы точно относить все статьи этих тем к этому разделу достаточно сложно. Поэтому было решено начать классификацию не с первого уровня иерархии, а со второго, поскольку на этом уровне в большинство тем уже хорошо выделены. На втором уровне иерархии находится 69 категорий.

В таблице ниже приведены результаты сравнения различных классификаторов для второго уровня иерархии. Каждый классификатор возвращает список классов в порядке убывания релевантности. На данном уровне имеется 69 классов. Для тестов было отобрано 1173 документов. Для определения точности методов использовалось 2 подхода: в первом подходе требовалось, чтобы раздел, указанный автором, был на 1 месте. Во втором подходе требовалось, чтобы указанный автором раздел входил в число 3 классов, определенных как ближайшие.

Метод	**SVM**	**RF**	**Rocchio**	**Тематики слов**
Точность(1 класс)	40,8%	2,1 %	43,4%	56,5%
Точность(3 класса)	59,4%	10,0 %	68,1%	77,4%
F-мера(1 класс)	0,094	0,008	0,187	0,208
Время обучения (сек)	124,80	158,8	36,3	1352,6
Среднее время классификации (сек)	1,3	1,12	1,45	2,92

Таблица 3.1: Результаты для различных классификаторов на втором уровне иерархии.

Следует заметить, что столь низкая точность метода RF – аномалия второго уровня. На более низких уровнях иерархии он показывает более качественные результаты.

Как видно из результатов выше, метод использования тематики слов показал лучший результат. И для объяснения этого предлагается рассмотреть случай классификации математических статей (раздел 51). В тестовой коллекции имеется 73 статьи по математике. Перед каждым классификатором ставится задача верно определить разделы математики, к которому

относятся статьи. Как говорилось ранее, для отбора статей используется косинусная мера. Рассмотрим случай ограничения статей, входящих в обучающее множество, задавая параметр α и требуя, чтобы $\texttt{sim} > \alpha$. В таблице ниже приведены результаты.

Значение α	**SVM**	**RF**	**Rocchio**	**Тематики слов**
0	4,2%	16,4%	38,3%	58,9%
cos(1,3)	12,3%	20,5 %	56,1%	64,3%
cos(1,2)	28,7%	43,8 %	47,9%	64,3%
cos(1,15)	39,7%	46,5 %	47,9%	61,6%
cos(1,1)	32,8 %	43,8 %	42,4%	60,2%

Таблица 3.2: Результаты для математического раздела при различных ограничениях на множество обучающих статей.

Из таблицы 3.2 можно сделать вывод о том, что SVM и RF страдают от шумовых данных, и если уменьшить количество данных, убирая некачественные, то точность классификации заметно возрастает. В то же время можно сделать вывод о том, что метод тематических классификаций не только дает лучший результат, но и также меньше подвержен влиянию шумовых документов: отсеивание некачественных документов влияет на его качество меньше, чем на остальные классификаторы. Подобная тенденция наблюдается во всех разделах УДК.

В результате описанных экспериментов было решено остановиться на методе тематических классификаций текстов.

3.2.1 Использование локального контекста для метода тематики слов

До этого при определении вероятностей тематики слов в документе использовались только эмпирические вероятности встречи слова в рассматриваемом контексте. Однако, можно выдвинуть предположение о том, что тематика слова также зависит от слов, стоящих рядом с ним, т.е. его локального контекста.

Рассмотрим документ $D = \{w_1, w_2, ..., w_n\}$. Предположим, что тематика слова w_k зависит от тематики слов $w_{k-i}, w_{k-i+1}, ..., w_{k+i-1}, w_{k+i}$. Добавим к документу D в начале и в конце i раз слово $\tilde{w}$, при этом положим, что $\tilde{w}$ встречается с равной вероятностью в любом контексте. Тогда положим, что вероятность $\widetilde{P}(c|w_k)$, что слово w_k принадлежит тематике c, пропорционально величине

$$\widetilde{P}(c|w_k) \propto \sum_{t=k-i}^{k+i} \left(\frac{1}{|t-k|+1} \right)^{\alpha} P(c|w_t) \qquad (3.2)$$

Параметр α определяет влияние слов по мере их отдаления от слова w_k. При $\alpha = 0$ все слова влияют в равной мере, при $\alpha = 1$ влияние обратно пропорционально расстоянию. Поскольку сумма вероятностей $\sum_c \widetilde{P}(c|w_k)$, вычисленных по формуле (3.2), не будет равняться единице, то требуется нормализовать эти величины.

Ниже приведены результаты использования различных параметров α и i на втором уровне иерархии (69 классов) с поиском 1 лучшего класса.

	$\alpha = 0$	$\alpha = 1$	$\alpha = 2$	$\alpha = 10$
$i = 1$	56,5%	56,5 %	56,5%	55,7%
$i = 2$	56,5%	55,2 %	55,2%	54,1%
$i = 3$	55,2%	53,1 %	52,4%	51,9%
$i = 4$	53,7 %	53,1 %	51,9%	51,9%

Таблица 3.3: Результаты для метода тематик слов при использовании локального контекста.

Как видно из таблицы, выдвижение предположений зависимости тематики слов от локального контекста не улучшает результат, полученный без них.

3.2.2 Типичные ошибки

Среди ошибок, допускаемых методом тематики слов, можно выделить 2 преобладающих класса ошибок.

1). ***Субъективность выбора УДК***. Поскольку вопрос выбора кода УДК для статьи не имеет однозначного ответа, то любое мнение по поводу выбора УДК можно считать субъективным. Поэтому однозначно доверять тому коду УДК, которое было указано автором, нельзя.

В качестве примера из списка ошибочно классифицированных документов случайным образом была выбрана статья “ПРОБЛЕМА ЖЕНЩИНЫ В «КОРОЛЕВСКИХ ИДИЛЛИЯХ»ТЕННИСОНА”. Данной статье автор присвоил УДК 008 – “Цивилизация. Культура. Прогресс”. Выделенные автором ключевые слова - “культура; социум; литература; викторианство; Теннисон”.

С помощью метода тематик слов на втором уровне классификации было выделено 3 следующих класса:

39. Этнография. Этнология. Нравы. Обычаи. Образ жизни. Фольклор.

82. Литература. Литературоведение.
21/29. Религиозные системы. Религии, верования.

Видно, что найденные УДК близки теме документа и нельзя однозначно говорить об ошибке классификатора.

2). ***Ошибки структуры УДК***. УДК, как и любая другая классификация, не имеет идеальную структуру: исторически может сложиться так, что 2 раздела, схожих по тематике, не имеют общих предков (к примеру, информатика 004 и математика 51). Из-за этого многие статьи по информатике вместо раздела 00 присваиваются к 51. Можно уменьшить количество ошибок, если несколько перестроить классификацию так, чтобы родственные категории находились ближе.

Был составлен алгоритм, который перестраивал дерево УДК в зависимости от ошибок. Предположим, что документы класса c_{11}, который является предком класса c_1, часто ошибочно относят к классу c_2. Тогда класс c_{11} можно сделать предком c_2, если алгоритм будет верно различать документы, относящиеся к предкам c_2 и классу c_{11}.

Ниже представлено несколько примеров перенесенных классов.

Название раздела	***Бывший предок***	***Новый предок***
Информационные технологии.	Общие вопросы науки и культуры	Математика
Горное дело. Горные предприятия (рудники, шахты, карьеры). Добыча нерудных ископаемых	Инженерное дело. Техника в целом	Науки о земле. Геологические науки
Общее машиностроение. Ядерная технология. Электротехника. Технология машиностроения	Инженерное дело. Техника в целом	Физика.
Организация производства. Управление. Экономика предприятий	Управление предприятиями, организация производства, торговли и транспорта	Экономика. Народное хозяйство. Экономические науки
Просодия. Стихосложение. Вспомогательные науки и источники филологии	Общие вопросы лингвистики, литературы и филологии	Языкознание и языки. Лингвистика
Точная механика	Отрасли промышленности и ремесла для изготовления и обработки различных изделий	Математика
Развитие и способности психики. Сравнительная психология	Психология	Воспитание. Обучение. Образование

Таблица 3.4: Изменения в структуре УДК.

Использование улучшенной версии УДК помогло повысить результат на 9,2 %.

3.3 Метод выявления ошибок на одном из уровней.

Как уже говорилось выше, требуется разработать метод, способный определять ошибки, допущенные на предыдущих уровнях. Предположим, что на одном из шагов классификатор выбрал класс $\bar{N}$. Далее требуется выбрать один из классов, являющихся потомками $\bar{N}$. Для выявления того, что документ был ошибочно присвоен к классу $\bar{N}$, используется следующих подход:

Рассмотрим множества вершин $N_1, N_2, ..., N_m$, являющихся потомками $\bar{N}$. Для $\bar{N}$ составляется два множества документов: в первом множестве находились верно классифицированные документы в $\bar{N}$, во втором — ошибочно. Для каждого документа d находится его значение $M_N = \texttt{sim}(d, \bar{N})$ и $M_A = \dfrac{\sum_k \texttt{sim}(d, N_k)}{m}$. Далее строится SVM, которому на вход подаются документы в виде точек (M_N, M_A), и классом “1”, если документ классифицирован верно, и “-1”, если ошибочно.

Далее, для запроса Q, когда он был присвоен классу $\bar{N}$, находятся значения M_N и M_A, которые поступают на вход построенному SVM, и в зависимости от результата (возвращено 1 или -1) гипотеза о верности классификации принимается или отвергается.

С помощью использования данного метода удалось определить 12.1% ошибок на стадии классификации, в итоге точность увеличилась на 6%.

3.4 Результат использования тематики слов.

С помощью метода тематики слов для данной статью были выделены следующие УДК:

1). *Прикладные информационные (компьютерные) технологии. Методы основанные на применении компьютеров.*
2). *Ориентация процесса обработки данных.*
3). *Исследование операций*

В целом для тренировочных для 64.3% среди 3 классов встречался УДК, указанный автором.

3.5 Выбор ключевых слов.

Данный метод был описан в статьях [16, 17]. Допустим, что в нашем распоряжении находится статья (или коллекция статей) и задача заключается в выделении ключевых слов. Для этой цели разобьем текст на кластеры, или, как их еще можно назвать, текстовые единицы (textual units). Осуществить это можно различными способами, исходя из удобства разделения или каких-то логических побуждений. К примеру, статью можно разбивать его по $3 - 5$ предложений или помещая в каждый кластер по определенному количеству идущих подряд слов. Если имеется коллекции статей (к примеру, энциклопедия), то можно так же разбить каждую статью, если они достаточно крупные, можно принимать каждую статью как отдельный кластер или объединять их вместе по каким-то признакам (дате, наименованию и проч.).

После разделения документа необходимо составить словарь коллекции, причем каждому терму будем ставить в соответствие две числовые величины: то количество раз, с которым слово встречалось в документе (tf) и количество кластеров, в которых слово присутствовало. Можно заметить, что некоторые термы коллекции распределены равномерно по всему тексту. Такими словами обычно являются стоп-слова, которые не несут какой-то смысловой нагрузки и встречаются в тексте независимо от его содержания. Однако могут быть и исключения. К примеру, если взять коллекцию статей, посвященную информационному поиску, то слова «информационный» и «поиск» могут встречаться достаточно часто, и тогда эти слова будут считаться как нехарактеризующие коллекцию. С другой стороны, если требуется выделить ключевые слова каждой отдельно взятой статьи, то информация о том, что какая-то отдельная часть посвящена информационному поиску, не окажется полезной.

В то же время, некоторые термы будут встречаться неравномерно, т. е. встречаться достаточно большое количество раз, но в малом количестве кластеров. Такие слова будем принимать как ключевые. К примеру, если в коллекции про информационный поиск будет несколько статей про поиск ключевых слов, то именно в них термы «ключевые» и «слова» будут употребляться наиболее часто, а в других статьях могут и не использоваться вовсе. Тогда можно сказать, что эти два слова представляют ценность для коллекции и этих статей, в частности.

Таким образом, идея основывается на том, что ключевые слова распределены неравномерно по документу, в то время как неключевое слово может встретиться с достаточно большой вероятностью в любом случайно взятом кластере.

Данный метод позволяет выделить слова, которые представляют наибольшую ценность для коллекции в целом. Впоследствии, можно отобрать

какое-то количество наиболее важных слов для коллекции, или выделить ключевые слова для отдельных кластеров.

На основе вышесказанного можно сделать вывод, что нам требуется выбрать критерий, на основе которого можно будет объективно судить о равномерности распределения каждого отдельно взятого слова в документе. В качестве такого критерия возьмем вероятность того, что наш термин содержится ровно в N или менее кластерах при принятии гипотезы о равномерности распределения данного терма.

Предположим, что текст разбит на D кластеров. Для каждого терма известно N — число кластеров, содержащих данный терм, T — число вхождений терма в документ с учетом повторений (term frequency).

Тогда вероятность $p(n, T)$ того, что ровно n кластеров содержат наш терм может быть получена вычислением

$$p(n, T) = \frac{n! \binom{D}{n} \left\{ {T \atop n} \right\}}{D^T},$$

где выражение в фигурных скобках — число Стирлинга второго рода.

Таким образом, можно вычислить вероятность $P(N, T)$ того, что ровно N или менее кластеров содержат термин. Эта вероятность должна быть достаточно мала для большинства ключевых слов. Таким образом, вероятность $P(N, T)$ вычисляется как

$$P(N, T) = \sum_{n=1}^{N} p(n, T).$$

Сравнивая все наши термы по значению $P(N, T)$, можно судить о равномерности их распределения в тексте, и, соответственно, о степени значимости для документа. Слова с наименьшим $P(N, T)$ будут представлять для нас наибольшее значение. Для выделения ключевых слов кластера необходимо сравнить эти значения для входящих в него термов и выделить из них наиболее важные.

С помощью этого метода для данной дипломной работы были выделены следующие ключевые слова:

классификация
удк
лемминг
стемминг
лемматизация
роккио
отсеивание
шумов
косинусная
гиперплоскость

Таблица 3.5: Ключевые слова для данной дипломной работы.

3.6 Поиск похожих статей.

Для того, чтобы описать статью наиболее полно, желательно привести список статей, которые совпадают по тематике с рассматриваемой. В качестве меры схожести была выбрана косинусная мера. Однако, возникает вопрос о том, какие статьи проверять на схожесть: статьи всех разделов УДК или только того раздела, к которому была определена статья. Поскольку, к сожалению, процент ошибок классификатора существенен, то нельзя полагаться на тот раздел, который был определен. В то же время, поиск похожих статей среди всех доступных занимает много времени. Поэтому было решено собрать статьи в кластеры, находить ближайший кластер и вести поиск похожих статей среди входящих в него.

Предположим, что имеется коллекция документов $\mathcal{D} = \{d_1, d_2, ..., d_n\}$. Зададим параметр α и выделим из $\mathcal{D}$ такое подмножество документов $\mathcal{B} = \{d_{i_1}, d_{i_2}, ..., d_{i_n}\}$, что $\forall d_j, d_k \in \mathcal{B} : \texttt{sim}(d_j, d_k) < \alpha$. Документы из $\mathcal{B}$ будем называть базисными документами. Далее, каждую статью из $\mathcal{D}$ поставим в соответствие одной или нескольким ближайшим из базисных статей.

Для поиска ближайших документов к Q сначала требуется найти ближайшую базисную статью d_{i_j}, а далее вести поиск по статьям, отнесенным к d_{i_j}.

С помощью описанного метода из 5000 статей было выделено 274 базисных статей. Эксперименты показали, что результаты, получаемые с помощью полного перебора статей и с использованием $\mathcal{B}$ идентичны. Время поиска близких статей уменьшился более, чем в 20 раз.

Выводы

В таблице ниже показаны все использованные данные.

Тип данных	**Источник**	**Размер**
Обучающие данные	«Яндекс»	48394
Тестовые данные для классификаторов	Cyberleninka	1173
Статьи для поиска похожих	Cyberleninka	5000

Таблица 3.6: Использованные данные

В ходе экспериментов было определено, что в связи со структурой УДК поиск релевантного класса лучше начинать не на 1, а на 2 уровне иерархии.

В таблице ниже приведены результаты сравнения классификаторов на втором уровне.

Метод	**SVM**	**RF**	**Rocchio**	**Тематики слов**
Точность(1 класс)	40,8%	2,1 %	43,4%	56,5%
Точность(3 класса)	59,4%	10,0 %	68,1%	77,4%
F-мера(1 класс)	0,094	0,008	0,187	0,208
Время обучения (сек)	124,80	158,8	36,3	1352,6
Среднее время классификации (сек)	1,3	1,12	1,45	2,92

Таблица 3.7: Результаты для различных классификаторов на втором уровне иерархии.

На основе этой таблицы можно сделать вывод о том, что метод, основанный на определении тематик слов, дает лучшие результаты. Его точность на 13,1% выше, чем у алгоитма Роккио, на 15,7% выше, чем у SVM и на 54,4% выше, чем у RF.

Также было определено, что низкие показатели точности SVM и RF связаны с качеством набора обучающих статей, поскольку эти методы страдают от шумов. В таблице ниже показана точность методов, при обучении на всех статьях из раздела “математика”, максимальная точность, полученная после фильтрации данных и разность максимальной и начальной точностей:

Метод	**SVM**	**RF**	**Rocchio**	**Тематики слов**
Начальная точность	4,2 %	16,4 %	38,3 %	58,9 %
Максимальная точность	39,7 %	46,5 %	56,1 %	64,3 %
Разность	35,5 %	30,1 %	17,8 %	5,4 %

Таблица 3.8: Точности при различных данных.

Как видно из таблице выше, метод, использующий тематики слов меньше подвержен влиянию шумов, чем остальные методы.

Также было определено, что использование локального контекста не приводит к улучшению результатов.

Было определено, что доля совершенных ошибок объясняется структурой УДК, когда похожие классы имеют различных предков. Перестроив структуру УДК удалось повысить точность на 6%.

В результате всех улучшений для 64.3% тренировочных документов среди 3 классов, определенных методом тематик слов как ближайшие, встречался УДК, указанный автором.

Заключение

В работе была рассмотрена задача классификации текстов в системе УДК. Был построен граф, соответствующий данной системе классификации. Рассмотрен способ получения обучающих статей для ветвей данного графа, метод предобработки статей, а также метод, с помощью которого для каждой вершины можно выделить статьи, в наибольшей мере ей соответствующие. Рассмотрено несколько существующих классификации, среди которых SVM, RF, алгоритм Роккио. Также предложен новый метод классификации, основанный на тематиках слов. Эксперименты показали, что метод тематики слов показывает наилучший результат. Предложено несколько модификаций метода. Предложен метод изменения структуры УДК для повышения точности классификаторы. Предложен метод выявления ошибок на уровнях при классификации. Рассмотрен способ выделения ключевых слов для статей и метод поиска похожих статей. Разработан программный комплекс для решения описанных задач, создан сайт, на котором им можно воспользоваться онлайн.

В дальнейшем планируется увеличение числа разделов УДК, для которых производится классификация. Расширение базы обучающих статей, дополнение её более качественными статьями. Улучшение метода классификации текстов. Улучшение методов поиска ключевых слов и похожих статей. Изменение структуры УДК.

Список литературы

1. Киселев М. В. Оптимизация процедуры автоматического пополнения веб-каталога. – 2005.
   http://lvk.cs.msu.su/∼bruzz/articles/classification/08_Kiselev_102710.pdf
2. Токарева Е. И., Дьяконов А. Г. Иерархическая классификация текстов.
   http://alexanderdyakonov.narod.ru/Diplom2010TokarevaE.pdf
3. Silla Jr C. N., Freitas A. A. A survey of hierarchical classification across different application domains //Data Mining and Knowledge Discovery, 2011. Vol. 22, №. 1-2, P. 31-72.
4. Bi W., Kwok J. T. Multi-label classification on tree-and dag-structured hierarchies //Proceedings of the 28th International Conference on Machine Learning (ICML-11), 2011. P. 17-24.
5. Xiao L., Zhou D., Wu M. Hierarchical classification via orthogonal transfer //Proceedings of the 28th International Conference on Machine Learning (ICML-11), 2011. P. 801-808.
6. Choi S. E. et al. Age estimation using a hierarchical classifier based on global and local facial features //Pattern Recognition, 2011. Vol. 44, №6, P. 1262-1281.
7. Heinrich G. Parameter estimation for text analysis. Technical report Fraunhofer IGD Darmstadt, Germany, 2009. 32 p.
   http://faculty.cs.byu.edu/∼ringger/CS601R/papers/Heinrich-GibbsLDA.pdf
8. Blei D. M. , Ng A. Y., Jordan M. I. Latent dirichlet allocation // Journal of Machine Learning Research, 2003. Vol. 3, P. 993–1022.
9. Segalovich I. A Fast Morphological Algorithm with Unknown Word Guessing Induced by a Dictionary for a Web Search Engine //MLMTA, 2003. P. 273-280.
10. Vapnik V. The nature of statistical learning theory. Springer, 2000. P. 314.
11. R.-E. Fan, P.-H. Chen, and C.-J. Lin. Working set selection using second order information for training SVM. // Journal of Machine Learning Research, 2005. Vol. 6, P. 1889-1918.
12. Amit Y., Geman D. Shape quantization and recognition with randomized trees //Neural computation, 1997. Vol. 9, №. 7, P. 1545-1588.

13. Segal M. R. Machine learning benchmarks and random forest regression. 2004.
http://escholarship.org/uc/item/35x3v9t4
14. Маннинг К., Рагхаван П., Шютце Х. Введение в информационный поиск //М.: Вильямс, 2011. 528 с.
15. Gauch S., Chandramouli A., Ranganathan S. Training a hierarchical classifier using inter document relationships //Journal of the American Society for Information Science and TechnologyЮ 2009. Vol. 60, №1, P. 47-58.
16. Bookstein A., Klein S. T., Raita T. Detecting content-bearing words by serial clustering //Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 1995. P. 319-327.
17. Сердюк Ю. А. Алгоритм поиска ключевых слов методом кластеризации и его экспериментальное исследование // Процессы управления и устойчивость: Труды 43-й международной научной конференции аспирантов и студентов / Под ред. А. С. Ерёмина, Н. В. Смирнова. СПб.: Издат. Дом С.-Петерб. гос. ун-та, 2012. С. 397–402.