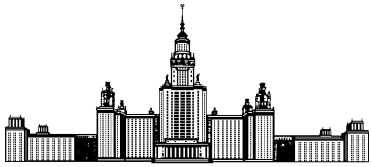
Московский государственный университет имени М. В. Ломоносова



Факультет Вычислительной Математики и Кибернетики

Кафедра Математических Методов Прогнозирования

ДИПЛОМНАЯ РАБОТА СТУДЕНТА 517 ГРУППЫ

«Обучаемые методы извлечения наукометрической информации из коллекций научных публикаций»

Выполнил:

студент 5 курса 517 группы

Полежаев Валентин Александрович

Научный руководитель:

д.ф-м.н., профессор

Воронцов Константин Вячеславович

Москва, 2013

Содержание

Аннотация

В работе рассматривается задача построения графа цитирований по коллекции научных документов. Предлагается технология, основанная на решении цепочки подзадач анализа текстов трех типов: разметки, сегментации и мэтчинга. Для решения подзадач предлагаются новые методы или адаптируются существующие. Описанный алгоритм представляет собой законченный технологический процесс для обработки больших коллекций документов и извлечения наукометрической информации.

1 Введение

Рост объёмов научной литературы, доступной в электронном виде, приводит к необходимости автоматического выделения ссылок и построения графа цитирований для нужд информационного поиска и наукометрии. Широко известные системы агрегации научного контента CiteSeerX, Google Scholar, MS Academic Search решают эти задачи всё ещё с недостаточной точностью. Кроме того, в них слабо представлен русскоязычный контент.

В данной работе представлена технология автоматического построения графа цитирования по коллекции научных документов, разрабатываемая в рамках проекта интеллектуального поиска, классификации и агрегации научной информации в пополняемых мультидисциплинарных коллекциях текстовых документов.

Предлагаемая технология обработки документа включает шесть основных этапов: предварительная обработка, выделение метаописания (если оно присутствует в тексте документа), выделение списков библиографии, разбиение каждого списка на записи, выделение из каждой записи полей, связывание записей-дубликатов.

Методы решения некоторых задач по отдельности описаны в литературе, но не рассматривались вместе в рамках единой технологии. Целью данной работы является обзор существующих методов, выбор и адаптация наилучшего сочетания методов. В будущем: будут выполнены эксперименты по всем подзадачам, для некоторых подзадач предложены новые методы решения.

1.1 Объекты предметной области

Основным объектом предметной области в данном исследовании является электронный документ (далее просто «документ»). Документ может быть представлен в различных форматах, таких как txt, html, pdf. Рассматриваемые документы ограничены лишь требованием возможности извлечения текстового содержимого. Причем при извлечении текста из некоторых форматов, возможны различные «артифакты», которые усложняют дальнейшую обработку.

К таким «артифактам» относятся некорректно распознанное содержимое, если исходный документ был представлен в виде сканированных изображений, нераспо-

знанные символы при извлечении текста из pdf, а также номера страниц и колонтитулы. Задача предварительной обработки текстов решается на начальном этапе и служит для приведения текста к лучшей форме.

Граф цитирования — направленный граф, в котором вершинами являются документы, а ребра соответствуют отношению цитирования.

В контексте задачи построения графа цитирования имеет смысл рассматривать только документы научного характера, то есть содержащие наукометрическую информацию — библиографию и метаописание.

Библиография — множество библиографических записей, каждая из которых обозначает ссылку на некоторую научную работу. Метаописание представляет собой информацию о документе, такую как автор работы, название, издательство. И если библиография чаще всего представлена одним или несколькими частями в тексте (блоками), то метаописание зачастую не представлено в тексте, а если представлено, то обычно имеет низкую полноту (представлены только авторы и название).

С точки зрения логической структуры, метаописание документа и библиографическая запись идентичны — их можно считать структурами с одинаковым набором полей.

Построить граф цитирования по коллекции документов означает «связать» метаописания одних документов с библиографическими записями в других документах.

Таким образом, для построения графа цитирования необходимо извлечь метаописание и библиографию. После извлечения блоков библиографии необходимо выделить отдельные элементы — библиографические записи (или ссылки), а затем выполнить их разбор по полям, то есть выделить авторов, название и другие поля. Все эти задачи ввиду многообразия форматов представлений целесообразно решать методами машинного обучения.

Зачастую, одни и те же библиографические записи и метаописания могут быть записаны по-разному. Это зависит от выбранного формата оформления, а также предпочтений автора документа. Из этого следует, что задача связывания ссылок и метаописаний также не может решаться «жестким» сравнением строк, для ее решения используются специальные методы машинного обучения.

Более того, обработка больших коллекций документов вычислительно трудоемка, поэтому внимание будет также сосредоточено на использовании вычислительно оптимальных алгоритмов в подзадачах, где это критично.

1.2 Задачи анализа текстов

В области анализа текстов, можно выделить три класса задач, возникающих при построении графа цитирования. Это задачи разметки, сегментации текста и связывания объектов.

1.2.1 Разметка

Пусть заданы входной и выходной алфавиты $\mathbb{A}$ и $\mathbb{B}$ соответственно. Задача разметки заключается в построении поэлементного отображения последовательности входных символов $a_1 a_2 \dots a_n$, $a_i \in \mathbb{A}$ на последовательность выходных символов $b_1 b_2 \dots b_n$, $b_i \in \mathbb{B}$.

Под символом входного алфавита может пониматься некоторый элемент, который описывает объект предметной области, например буква, слово или вектор признаков. Под символом выходного алфавита обычно понимается метка класса.

Таким образом, задача разметки может рассматриваться как задача классификации, где объектами являются последовательности входных элементов, а результатом классификации является последовательность меток классов.

В простейшем случае каждый элемент может быть классифицирован независимо от других с помощью известных методов классификации, предполагающих независимость объектов в выборке. Но в таком случае не учитываются взаимосвязи между элементами в последовательности, что, как правило, критично для решения задачи разметки. Последовательный характер данных задает структурные ограничения.

Использование широко известных методов. Простым способом учета последовательной структуры входных данных является использование скользящего окна при построение признакового описания. При этом признаковое описание расширяется контекстными признаками, значения которых соответствуют значениям базовых признаков для соседних объектов.

Такой прием позволяет использовать известные методы SVM [15], Winnow [22] или решающие деревья для отдельной классификации каждого элемента входной последовательности.

Сложности возникают, когда помимо признаков требуется учитывать метки классов объектов-соседей, которые известны на момент обучения, но не известны на момент классификации. В таком случае применяются эвристики, например голосование.

Методы на основе графических моделей. Более совершенные методы основаны на математическом аппарате графических моделей. Графические модели позволяют описывать вероятностные распределения со сложными зависимости между переменными, которые представляются в виде графа. Графические модели позволяют естественным образом учитывать структурные ограничения.

В терминах задачи разметки, каждой позиции i входной последовательности ставится в соответствии переменная x_i, область значений которой совпадает с множеством значений $\mathbb{A}$. Аналогично, для каждой позиции i выходной последовательности ставится в соответствие переменная y_i с областью значений $\mathbb{B}$.

Переменные x_i называются наблюдаемыми, y_i — скрытыми. Значения скрытых и наблюдаемых переменных, соответствующие некоторым входным и выходным последовательностям образуют конфигурации наблюдаемых и скрытых переменных соответственно.

Тогда задача разметки состоит в поиске наиболее вероятной конфигурации скрытых переменных при условии заданной конфигурации наблюдаемых переменных.

Первыми широкое применение получили скрытые марковские модели (Hidden Markov Models, HMM [19]), однако они позволяют учитывать лишь простейшие зависимости между переменными и ограничивают в использовании признакового описания. Были разработаны некоторые обобщения, которые частично устранили эти недостатки [20, 21].

На данный момент для решения задачи разметки в анализе текстов наибольшую популярность приобрели модели условных случайных полей (Conditional Random Fields, CRF [16]). В CRF удалось решить характерные для графических моделей

проблемы вывода, что позволило использовать полноценное признаковое описание и учитывать сложные зависимости между переменными.

1.2.2 Сегментация

Задача сегментации схожа с задачей разметки, но анализируемые последовательности имеют характерную блочную структуру. Выходная последовательность $b_1 b_2 \dots b_n$ разбивается на части-блоки, внутри каждого блока все символы совпадают. Также, в отличие от разметки, методы решения задачи сегментации обычно используют кластеризацию. Общие для большинства методов сегментации черты заключаются в определении функции схожести на блоках и задании функционала качества сегментации. В [2, 6, 5] описаны методы сегментации текстов, основанные на представлении блоков, как «мешка слов».

1.2.3 Связывание

В различных текстовых документах один и тот же объект может быть представлен по-разному. Например, в названиях журналов, конференций могут использоваться сокращения, а в именах авторов может быть изменен порядок слов.

Задача связывания (Record Linkage, RL) — это задача идентификации различных представлений одного объекта или поиска дубликатов представлений. В литературе также упоминается как «matching». Связывание можно выполнять для объектов самой разной природы, в данном случае будет рассматриваться случай, когда объекты представлены в виде структур с текстовыми полями (например, метаописание документа с полями «автор», «название» и т.д.).

Обзор техник связывания дан в [13]. Стандартной техникой для определения дубликатов представлений, является решение задачи классификации, в которой объектами распознавания являются пары структур, с классами «дубликаты», «не дубликаты». В качестве классификатора обычно используется SVM.

В качестве признаков рассматриваются различные функции сравнения двух строк. Так как объекты представлены в виде структур с текстовыми полями, то при заданных K функциях сравнения строк, применяя каждую из функций к каж-

дому полю, можно составить признаковое описание размерности $M \cdot K$, M — число полей.

Функции сравнения строк Согласно [9, 3] выделяются следующие типы функций сравнения строк.

1. Статические:
   (a) Character-based. Основаны на посимвольном сравнении. Расстояние Левенштейна, расстояние Смита-Ватермана, функция Джаро и другие.
   (b) Token-based. Основаны на сравнении строк, как «мешка слов». Функция Монжа-Элкана, схожесть Джаккарда, TF-IDF.
   (c) Hybrid. Двухуровневые. Представляют собой token-based функции, но схожесть токенов определяется по character-based метрике второго уровня. Адаптированная функция Монжа-Элкана, Soft-TFIDF.
   (d) Предикаты. Являются некоторым логическим утверждением для пары строк, например «в строках одинаковое число слов».
2. Обучаемые. Основаны на применении алгоритмов машинного обучения для лучшей настройки параметров метрики (например, веса операций в расстоянии Левенштейна) под конкретные данные.

Применение обучаемых функций требует дополнительных затрат на подготовку обучающих данных и сам процесс обучения. В то же время, показано, что использование только статических функций позволяет добиться приемлемого качества.

Вычислительная оптимизация Обычно требуется искать объекты-дубликаты внутри множества или на паре множеств. Простейший способ — классификация всех возможных пар объектов — вычислительно не эффективен. Обычно используются методы, позволяющие заранее отбросить пары, заведомо не являющиеся дубликатами. Обзор методов дан в [7].

2 Предварительная обработка

Перед решением основной задачи выделения и анализа списков библиографии необходимо очистить входные данные для повышения качества распознавания. Предполагается, что входными являются не форматированные тексты, выделяемые из файлов распространённых форматов, таких как pdf, doc, html, ps, djvu, rtf. В результате выделения текста из форматов, имеющих физическую, а не логическую разметку (pdf, ps), а также полученных в результате сканирования, могут появляться различные артефакты — колонтитулы, номера страниц, фрагменты формул, таблиц и графиков, мешающие правильному распознаванию списков библиографии.

2.1 Удаление номеров страниц

Сделаем несколько предположений: (1) каждый колонтитул с номером страницы представляется в тексте отдельной строкой, (2) в одном документе номера страниц оформлены в одном стиле, (3) номера страниц могут окружаться некоторым текстом, например «= 12 =» или «Page 12 of 16», длина которого не превышает заданного порога ℓ_0.

Введем три вспомогательные функции от двух строк s, t, содержащих номера страниц.

$L(s,t)$ — модифицированное расстояние Левенштейна [9], в котором замены букв штрафуются больше, чем замены цифр, и каждая следующая замена штрафуется больше, чем предыдущая.

$S(s,t)$ — функция штрафа за различия чисел в строках.

$$S(s,t) = \begin{cases} C_1\big(\sigma(t) - \sigma(s)\big), & \sigma(t) > \sigma(s), \\ C_2, & \sigma(t) \le \sigma(s), \end{cases}$$

где $\sigma(t)$ — сумма всех целых чисел в строке t.

$F(s,t)$ — функция штрафа за большое удаление строк в тексте.

$$F(s,t) = \begin{cases} 0, & C_3 < N(t) - N(s) < C_4, \\ C_5, & \text{иначе}, \end{cases}$$

где $N(t)$ — порядковый номер строки t в тексте, $C_4 > C_3 \ge 0$.

Наконец, определим *функцию порядка* $P(s,t) = L(s,t) + S(s,t) + F(s,t)$, значение которой тем меньше, чем более правдоподобно, что s и t — две последовательные строки с номерами. Параметры функции $P(s,t)$ подбираются экспериментально.

Для удаления строк с номерами страниц сначала выбираются все *строки-кандидаты*, содержащие цифры и имеющие длину не более $\ell_0 = 20$. Затем строятся всевозможные *последовательности-кандидаты* из строк-кандидатов. Строится матрица A, $A_{ij} = P(s_i, s_j)$, где s_i — строка-кандидат с номером i. Если значение A_{ij} больше заданного порога, то, будем считать, что строка s_i заведомо не может располагаться перед s_j, и полагать $A_{ij} = N$, где N — специальная метка. В дальнейшем при построении последовательностей, метки N будут игнорироваться, что уменьшит вычислительные затраты. Последовательности-кандидаты строятся по матрице A путем добавления новых элементов таким образом, чтобы минимизировать значение функции порядка от последнего элемента в последовательности и добавляемого элемента. В качестве первых элементов последовательностей выбираются те, которым соответствуют столбцы матрицы A, состоящие только из меток N.

Полученные последовательности-кандидаты можно отфильтровать (например, по минимальной длине). Окончательно выполняется удаление из текста строк, составляющих полученные последовательности.

Особенности реализации Зачастую оказывается, что последовательности-кандидаты имеют существенное число общих фрагментов. Это приводит к тому, что несмотря на небольшое число уникальных элементов последовательностей, число уникальных последовательностей оказывается очень большим. Таким образом явное нахождение всех последовательностей вычислительно неэффективно.

Практически более эффективный алгоритм основан на представлении исходных данных в виде ориентированного графа с весами, в котором вершины соответствуют строкам-кандидатам, а ребра с весами определяются матрицей A (метка N обозначает отсутствие ребра).

Для первой начальной вершины строится дерево похожее на дерево поиска в ширину (breadth first search), но с той разницей, что при обходе очередной вершины присоединяются только её ближайшие в смысле веса сыновья. Если при обходе у

каждой вершины u есть только один сын v, такой что ребро (u, v) имеет минимальный вес среди всех ребер вида (u, c), то дерево вырождается в цепочку.

Затем берется вторая вершина и для неё выполняются те же действия. В результате будет либо построено новое дерево (если последовательности, которые можно построить из двух первых начальных вершин не имеют общих элементов), либо будет дополнено дерево, построенное ранее. Во втором случае построенная структура уже не будет деревом (есть две начальные вершины).

Повторяя те же действия для остальных начальных вершин, будет построен ориентированный граф с множеством начальных вершин (не имеющих родителей), соответствующих множеству начальных вершин последовательностей и с множеством вершин, совпадающим с множеством элементов всех последовательностей-кандидатов.

Так как сложность обхода графа методом поиска в ширину есть $O(|V| + |E|)$ (V — множество вершин графа, E — множество ребер), а предложенный алгоритм обходит лишь подмножество всех вершин, то его сложность также не более $O(|V| + |E|)$. Исходный граф часто оказывается разреженным, поэтому рекомендуется представление в виде списков смежных вершин, а не матрицы в явном виде.

На практике оказалось, что данный алгоритм, помимо последовательности номеров страниц, находит также последовательности подписей к рисункам и таблицам. Поэтому целесообразно удалять все найденные последовательности без выделения какой-либо одной из них.

2.2 Удаление колонтитулов

Как и в случае с номерами страниц, будем предполагать, что колонтитулы представляются отдельными строками в тексте. Необходимо учесть, что колонтитулы в одном тексте могут незначительно варьироваться, чередоваться и изменяться.

Определим бинарную *функцию сходства* строк $B(s, t)$. Если длины строк s, t отличаются более чем на 5 символов, то $B = 0$. Если строки полностью совпадают, то $B = 1$. Если у строк не совпадают ни префиксы, ни суффиксы, $B = 0$. Иначе вычисляется значение меры сходства Джаро-Винклера [9], и если оно превышает 0.95, то $B = 1$, иначе $B = 0$.

Аналогично удалению номеров страниц, для поиска последовательностей строк колонтитулов сначала отбираются строки-кандидаты, длина которых превышает заданный порог. Затем вычисляется значение функции сходства на всех парах строк-кандидатов и множество всех строк-кандидатов разбивается на группы, содержащие вместе с каждым объектом как минимум один объект, схожий с ним.

Эффективная практическая реализация основана на представлении исходных данных в виде неориентированного графа без весов с вершинами, соответствующими строкам-кандидатам. В графе имеется ребро (u, v) тогда и только тогда, когда $B(u, v) = 1$.

В такой постановке задача сводится к поиску связных компонент на построенном графе. Решение может быть достигнуто, например при помощи поиска в глубину со сложностью $O(|V| + |E|)$.

На практике оказалось, что данный алгоритм, помимо группы строк колонтитулов, находит и другие подпоследовательности схожих строк, в частности, группы похожих формул. Поэтому целесообразно удалять все найденные группы.

3 Выделение метаописания

При обработке документов, размещённых в открытом доступе в Интернете, *метаописание документа*, содержащее библиографические данные (заголовок, список авторов, название журнала, том, страницы, и т. д.) не всегда может быть получено из того контекста, где была обнаружена ссылка на данный документ. Эти данные могут содержаться внутри самого документа: для статьи — в заголовке или в колонтитуле; в случае книги, отчёта, диссертации — на первых страницах.

Задача выделения метаописания состоит в формировании структуры с библиографической информацией из текста самого документа.

3.1 Базовый алгоритм

В [14] предложен метод выделения полей метаописания из текста документа, основанный на решении задачи классификации, в которой объектами являются строки, классами — типы полей (заголовок, список авторов, и т. д.), признаками — харак-

теристики строк (номер строки в тексте, доля цифр в строке, число слов в строке, число слов из контекстного словаря, и т. д.). Контекстный словарь составляется для каждого типа полей (и, возможно, некоторые другие) и содержит слова, часто встречающиеся в полях данного типа. Наличие контекстных словарей определяется технически-организационными возможностям и наличием соответствующих источников данных.

Для каждого класса строится отдельный классификатор SVM. Таким образом, каждая строка может быть отнесена к нескольким классам. Обучающая выборка формируется из документов, в которых метаописания размечены вручную. Процесс классификации является итерационным: на каждой итерации выполняется контекстная классификация, учитывающая классификацию на предыдущем шаге.

Так как отдельные классификаторы SVM отмечают строки на принадлежность классам целиком, то необходимо выполнить выделение подстрок, соответствующих искомым значениям полей. Например, если некоторая строка отмечена, как содержащая поле ББК, то она может содержать код ББК только как часть строки.

Для этого используются «финализаторы». Финализаторы образуют цепочку, в которой за каждым финализатором закрепляется класс (тип поля). Финализаторы применяются к каждой строке, если она отмечена тем же классом, в порядке нахождения в цепочке.

Финализаторы могут использовать регулярные выражения, словари и иные эвристические алгоритмы. Если финализатор находит в строке некоторую подстроку, то она вырезается из исходной строки, отметка о соответствующем классе снимается, и измененная строка передается следующем по списку финализатору. Очевидно, можно использовать несколько финализаторов с одинаковыми закрепленынми за ними классами.

Используемые финализаторы:

1. Выделение подстроки, находящейся в контекстном словаре
2. На основе регулярных выражений (например, для УДК, ББК, года издания)
3. Эвристические алгоритмы выделения имен авторов

Таблица 1: Качество (F_1-мера) по полям выделения метаописания для коллекции книг.

Автор	Название	Издатель	ББК	Город
0.94	0.906	0.901	0.957	0.954
ISBN	УДК	Год	Аннотация	ISSN
0.995	0.975	0.945	0.995	0.989

Таблица 2: Качество (F_1-мера) по полям выделения метаописания для коллекции авторефератов.

Автор	Название	Организация
0.925	0.919	0.895
Город	Код специальности	Год
0.967	0.905	0.921

Если в результате применения цепочки финализаторов в строке все равно остается пометка о более чем одном классе, тогда применяется эвристический алгоритм на основе вероятностей классификации, описанный в [14].

Эксперименты Эксперименты проводились на двух коллекциях, составленных из книг и авторефератов. В обоих случаях выборки составлялись из первых 30 строк каждого документа. Использовалось 40 признаков, полный список которых имеется в приложении. Для контекстных классификаторов размер скользящего окна равен 5.

Параметры каждой из моделей SVM настраивались индивидуально.

Эксперименты проводились по схеме 10-fold CV. Оценкой качества является F1-мера по каждому из классов. Оценивая качество таким образом, предполагается, что финализаторы работают достаточно хорошо.

Результаты экспериментов для двух коллекций приведены в таблицах 1 и 2.

3.2 Расширенный алгоритм

Далее приводятся идеи, относящиеся к практической применимости описанного выше алгоритма.

Определение типа текста Экспериментально было отмечено, что описанный выше алгоритм лучше работает на однородных коллекциях, которые составлены преимущественно из текстов схожей структуры. Поэтому целесообразно при распознавании очередного текста предварительно определять его тип. Другая практическая мотивация — в текстах разных типов могут присутствовать различные наборы типов полей, что означает использование различного набора классификаторов.

В предыдущем разделе эксперименты проводились над коллекциями двух типов — «книги» и «авторефераты».

Это задача многоклассовой классификации, в которой объектами являются тексты, а классами — типы текстов. Предполагается, что для каждого типа найдется обученная композиция классификаторов SVM, необходимая для применения алгоритма, описанного выше.

Дополнительная информация Зачастую колонтитулы содержат информацию об издании. Более того, часто эта информация отсутствует в основном тексте.

В таком случае можно использовать алгоритмы предварительной обработки текстов, описанные в предыдущем разделе для поиска колонтитулов. Анализировать выделенные колонтитулы можно при помощи того же описанного ранее метода, но в данном случае строки колонтитулов не являются соседями в тексте и контекстные классификаторы не нужны.

Определение частей в сборнике На практике может быть необходимость анализировать тексты, являющиеся сборниками, состоящими из нескольких статей. Тогда необходимо применять алгоритм выделения метаописания для каждого из фрагментов, представляющих отдельные статьи.

Задачу разделения сборника на части можно считать задачей классификацией на строках с целью отыскать строки, являющиеся разделителями.

Для определения границ можно руководствоваться следующими соображениями: а) если в колонтитулах записывается название текущей статьи, то изменения колонтитула является признаком окончания текущей статьи, б) если в статьях сборника указывается список библиографии, то это также является признаком окончания текущей статьи (алгоритмы выделения списков библиографии описаны далее), в) сме-

на страниц является также несет дополнительную информацию, так как зачастую статьи начинаются с новой страницы.

4 Выделение списков библиографии

Списки библиографии выделяются для дальнейшего их разбиения на библиографические записи и анализа.

В общем случае тексты могут иметь самую разную логическую структуру, например список библиографии может находиться не в конце документа, если текст выделен из сборника, то списков может быть несколько. Сами же библиографические записи могут быть и обычно оформлены в разных стилях.

Это обуславливает использование методов машинного обучения.

4.1 Определение типа текста

Если текст получен из документа с физической разметкой (например, pdf), то он разбит на строки по ширине страницы или колонки. Библиографические записи обычно оформляются как абзацы, но в данном случае абзацы оказываются разделенными на несколько частей-строк. Если же структура абзацев сохранена, то можно считать, что каждая библиографическая запись представлена одной строкой. Задача состоит в определении типа текста — сохранена ли структура абзацев.

Также, факт разбиения абзацев влияет на способ генерации признакового описания для задачи выделения списков библиографии, что будет показано далее.

Предлагается эвристический алгоритм, использующий только информацию о длинах строк текста. Он основан на нескольких наблюдениях: (1) если текст разбит по строкам, то большинство строк в тексте относительно короткие и примерно одинаковой длины; (2) однако могут встречаться и заметно более длинные строки, вызванные артефактами извлечения; (3) если текст разбит по абзацам, то распределение длин строк имеет дисперсию, сравнимую с квадратом средней длины абзаца; (4) строки малой длины (меньше 30) можно игнорировать.

Определение типа осуществляется с помощью решающего правила: если дисперсия длин строк менее 1000, то считать, что текст разбит по строкам, иначе — по абзацам. В экспериментах это правило показало 99% точности.

4.2 Алгоритм выделения списка библиографии

Задача выделения списков библиографии ставится как задача разметки или сегментации на строках текста с классами «принадлежит библиографии», «не принадлежит библиографии».

Классификатор и признаковое описание. Рассматривались два алгоритма — стандартный алгоритм классификации и алгоритм, основанный на графических моделях. В первом случае в качестве классификатора было выбрано решающее дерево C4.5 , как показавшее высокое качество вместе с хорошей скоростью работы, во втором — CRF.

использовался набор из 33 признаков, из которых наиболее информативными оказались: число знаков препинания в строке; начинается ли строка с цифры; число слов, начинающихся с заглавных букв; число двух подряд идущих слов, начинающихся с заглавных букв; число инициалов; число цифр; число точек; число заглавных букв; число ключевых слов («С.», «no», «vol.», и др.) число четырёхзначных чисел; длина строки относительно средней длины библиографической записи (220 символов); нормированное расстояние до предыдущего ключевого слова, например «References» или «Литература».

Полный список признаков приводится в приложении.

Для учета структурных ограничений задачи в случае использования дерева C4.5 [18] можно воспользоваться схемой скользящего окна при формировании признакового описания, либо использовать контекстную классификацию.

Контекстная классификация выполняется на результатах базовой классификации, полученной решающим деревом C4.5 по правилу: строка относится к тому классу, к которому отнесено большинство из k соседних строк. Экспериментально было найдено оптимальное значение параметра $k = 10$.

В качестве модели CRF использовалась линейная модель второго порядка с признаками, построенными для окна размера 5. Несмотря на то, что модель CRF учитывает последовательную структуру данных, использование дополнительно контекстной классификации позволяет улучшить качество.

Разбиение абзацев. Допустим, в результате определения типа текста оказалось, что абзацы разбиты по ширине страницы или колонки, то есть библиографические записи оказываются разделены на несколько частей.

Рассмотрим реальный пример:

> Mohri, M. (2000). Minimization algorithms for sequential transducers. Theoretical Computer Science, 234, 177–201.

Вторая строка вне контекста слабо отличима от обычного текста, следовательно, в поставленной задаче классификации на строках, признаковое описание таких строк слабо отличается от строк вне списков библиографии. Это означает пересечение классов и ухудшение качества распознавания.

Для решения этой проблемы можно опять же воспользоваться скользящим окном при генерации признаков, либо использовать суммирование значений соответствующих признаков соседних строк из скользящего окна ширины S, определяемой для каждого текста индивидуально.

Пусть, например, каждая библиографическая запись в тексте в среднем представлена тремя соседними строками. Условно, соответствующие части можно обозначить как A, B, C. A — первая часть, обычно содержит авторов, номер записи в списке, B — следующая часть, обычно содержит название работы, C — последняя, содержит остальную информацию вроде издательства, года, страниц. Очевидно, признаковое описание будет существенно отличаться для записей каждого из этих типов.

Последовательность строк с обозначенными типами в тексте может иметь примерно следующий вид:

$$...CABCABCA...$$

Выбирая размер окна суммирования равным 3, получается, что признаковое описание для первой строки типа A (вторая строка в примере выше) есть сумма призна-

ковых описаний самой строки типа A, левой строки типа C и правой строки типа B. Аналогично, для третьей строки (типа B) признаковое описание будет складываться из признакового описания трех строк типов A, B, C.

В результате проблема пересечения классов разрешается.

В экспериментах полагалось $S = [L/W]$, где $L = 220$ — средняя длина записи, W — средняя длина строки, оценённая с помощью робастного алгоритма.

Оценка средней длины строки. Для оценки средней длины строки предлагается следующий алгоритм.

Строится гистограмма длин строк текста $Y(X)$, которая затем нормируется на отрезок $X \in [0; 100]$. После этого выполняется сглаживание и домножение значения гистограммы $Y(X)$ в каждой точке на значение X. В качестве искомого значения выбирается $argmax\ Y(X)$ с учетом выполненной ранее нормировки на отрезок $[0; 100]$.

Этот алгоритм на практике оказался устойчивым к различным выбросам и артефактам извлечения текста.

4.3 Эксперименты

Эксперименты выполнялись на размеченной коллекции из 70 текстов, включающих англоязычные тексты по computer science и русскоязычные авторефераты, по схеме скользящего контроля leave-one-out. Каждый из N текстов по очереди рассматривался как контрольный, на объектах из остальных $N - 1$ текстов происходило обучение. Оценка качества вычислялась как среднее по всем контрольным текстам значение F_1-меры для класса «принадлежит блоку библиографии».

Результаты экспериментов приведены в таблицах 3, 4. Метки S и M обозначают способы генерации признаков: S — скользящее окно, M — суммирование значений; + означает использование контекстной классификации.

Эксперименты показали, что CRF лучше решает задачу, чем C4.5, но требует больше времени для обучения и классификации. Метод, не использующий скользящее окно и суммирование значений при генерации признаков, работает существенно хуже, значение F_1-меры ниже 70%.

Таблица 3: Качество (F_1-мера) выделения списка библиографии для текстов, разбитых по строкам.

	C4.5				CRF	
	S	M	+S	+M	+S	+MS
сред.	0.811	0.833	0.886	**0.937**	0.924	**0.948**
мин.	0.144	0.147	0.199	0.500	0.295	0.504
макс.	0.967	1.000	1.000	1.000	1.000	1.000
откл.	0.161	0.145	0.086	0.093	0.137	0.109

Таблица 4: Качество (F_1-мера) выделения списка библиографии для текстов с сохраненной структурой абзацев.

	C4.5 S	C4.5 +S	CRF +S
сред.	0.943	**0.974**	**0.982**
мин.	0.838	0.925	0.926
макс.	0.992	0.993	1.000
откл.	0.046	0.019	0.017

5 Разбиение списков на записи

Подзадача разбиения списка библиографии на библиографические записи актуальна только для текстов, разбитых по строкам, в противном случае сами абзацы и будут записями.

Задача ставится как задача разметки, в которой объектами являются строки, классов два — «первая строка записи» и «не первая строка». Это минимальный набор классов, который позволяет по разметке разбить списки на записи. Рассматривались и некоторые избыточные случаи (например, добавление класса «последняя строка записи»), но выигрыша в качестве они не дали.

Для решения использовались те же методы C4.5, CRF и тот же набор признаков. Контекстная классификация, скользящее окно и суммирование значений не использовались.

Эксперименты выполнялись по аналогичной схеме. Оценкой качества являлась средняя по классам F_1-мера. В таблице 5 приведены результаты экспериментов на всей коллекции. Теперь классификатор C4.5 показал лучшее качество по сравнению с CRF.

Таблица 5: Качество (F_1-мера) выделения записей из списка библиографии.

	C4.5	CRF
сред.	**0.972**	0.854
мин.	0.877	0.543
макс.	1.000	1.000
откл.	0.031	0.092

6 Стандартизация

На этапе стандартизации выделенные на предыдущем этапе текстовые строки, представляющие библиографические записи, необходимо преобразовать в структуры с библиографическими полями «автор», «название», «издание», «том», «номер», «организация», «конференция», «издатель», «редактор», «место», «год», «страницы».

Стандартизация является задачей разметки где объектами распознавания являются выделенные текстовые последовательности, в которых элементами являются слова и символы пунктуации, а метки классов представлены библиографическими полями.

Первые системы стандартизации представляли собой вручную составленные экспертные правила, позднее стал активно применяться аппарат графических моделей.

Сначала использовались простые системы на основе одной модели HMM [8], затем появились более сложные системы, использующие композицию из нескольких HMM [4, 1].

Основная трудность задачи стандартизации связана с многообразием форматов библиографических записей.

На данный момент наилучшими считаются следующие два метода [11, 12].

Метод на основе CRF. Среди графических моделей наилучшим способом для задачи стандартизации библиографических записей показал себя алгоритм, основанный на CRF. Модель CRF второго порядка обеспечивает качество выделения большинства ключевых полей порядка 97% на тестовых данных [12].

Как правило, в подобных задачах используется модель CRF, обучаемая с учителем, то есть требующая вручную размеченных обучающих данных.

На практике качество распознавания существенно зависит от того насколько много различных форматов представления записей, которые встречаются в текстах.

Метод FLUX-CiM не требует размеченных вручную данных. Вместо этого предлагается собрать для каждого извлекаемого поля словарь, которые когда-либо встречались в этом поле. Например, для поля «автор» — это, очевидно, будет коллекция из имен и фамилий. Также хранится информация о числе вхождений каждого из слов. Во время распознавания решение о принадлежности слова полю принимается на основе этой информации.

Для создания такой базы можно пользоваться внешними источниками (библиографические базы DBLP, PubMed, словари названий журналов, конференций, издательств и т.д.), главное требование — возможность выделить из данных соответствующие поля.

На известных текстовых коллекциях FLUX-CiM показал лучшее качество [11], чем алгоритм, основанный на CRF. Однако качество работы FLUX-CiM напрямую зависит от полноты его базы, создание которой является организационно-технической проблемой.

7 Общая схема построение графа цитирования

В результате решения предыдущих подзадач для каждого поступившего документа выполнено выделение метаописания, выделены и разобраны по полям библиографические записи. На последнем этапе выполняется построение графа цитирования.

Граф цитирования — это направленный граф, вершины которого соответствуют документам, ребра — отношению цитирования.

Для формирования графа цитирования необходимо выполнить связывание метаописаний документов с выделенными в пристатейных списках литературы библиографическими записями.

Как структура с полями, метаописание является расширением библиографической записи, а значит метаописания и записи можно сравнивать по общему подмножеству полей. Таким образом для построения графа цитирования можно пользоваться стандартным способом связывания объектов, описанным в разделе 1.2.3.

Тем не менее, есть ряд особенностей, которые требуют адаптации базового алгоритма связывания. Во-первых, процесс построения графа цитирования инкрементный, то есть при поступлении новых документов в граф должны добавляться новые вершины и ребра. Во-вторых, обновление графа при добавлении пакета может быть реализовано эффективнее, чем при добавлении документов по одному.

В качестве базового алгоритма для процесса обновления графа цитирования предлагается использовать модификацию метода адаптивного мэтчинга (Adaptive matching, AM) и метода вычислительной оптимизации Canopies, описанных в [17, 10].

7.1 Адаптивный мэтчинг

Рассмотрим два множества $\mathbb{A}$, $\mathbb{B}$, их подмножества $A \subset \mathbb{A}$, $B \subset \mathbb{B}$. Объектами являются пары $X \subseteq A \times B$, а решением множество пар $Y \subseteq X$.

Адаптивный мэтчинг выполняет связывание элементов a и b множеств A, B, в общем случае разной природы, чтобы минимизировать заданную функцию потерь. Связанные элементы обозначаются парой (a, b).

Для предлагаемого решения Y^* и правильного решения Y функцию потерь можно ввести как мощность симметрической разности множеств Y^* и Y.

Базовый алгоритм Алгоритм [10] основан на использовании стандартной техники — бинарного классификатора на объектах парах $(a, b) : a \in A, b \in B$. Классификатор принимает решение о связывании (a, b) элементов a и b. То есть решение Y для объектов X может быть получено классификацией всех возможных пар $(a, b) \in A \times B$.

В данном случае, признаковое описание составлено из функций сравнения пары строк, как описано в разделе 1.2.3.

Для оптимизации вычислений используется метод Canopies [17].

Если представить объекты множеств A и B точками на плоскости, а пару (a, b) ребром, соединяющим точки a, b, то результатом работы алгоритма будет двудольный граф.

Модификация алгоритма Основное отличие, которое требуется для применения адаптивного мэтчинга к задаче построения графа цитирования заключается в необ-

ходимости связывания элементов не только между двумя множествами A и B, но и внутри одного из множеств, пусть для определенности B.

Как было показано, несмотря на то, что связываются объекты разной природы (метаописания и библиографические записи), их представление как структур идентично, то есть $\mathbb{A} = \mathbb{B}$. Тем не менее, для удобства мы будем их отличать.

Предлагаемый алгоритм адаптивного мэтчинга описан в Алгоритме 1, модификация вспомогательного алгоритма Canopies — в Алгоритме 2.

Метод Canopies использует вычислительно не трудоемкую метрику для определения пар-кандидатов, которые затем подаются бинарному классификатору.

Algorithm 1 Модифицированный алгоритм адаптивного мэтчинга

TRAIN

Require: $\mathbb{Y}^* = \{(a,b) \ : \ a \in A, b \in B\}$ — объекты пары (решение) для обучения;

$LearningSet = \varnothing$;

$SetOfPairs = Canopies(A, B)$;

for $(a,b) \in SetOfPairs$ **do**

$$(a,b).Label = \begin{cases} +\ , if\ \ (a,b) \in Y^* \\ -\ , otherwise \end{cases}$$

$LearningSet.Add((a,b))$;

return $TrainClassifier(LearningSet)$;

MATCH

Require: A, B, Alg — элементы для связывания, обученный классификатор;

$SetOfPairs = Canopies(A, B)$;

$G = (V, E) \ : \ V = \{v \ : \ v \in A \cup B\}, E = \emptyset\}$

for $(a,b) \in SetOfPairs$ **do**

if $Alg(a,b) = +$ **then**

$E.Add((a,b))$;

$C = Components(G)$; — вернуть компоненты связности, внутри каждой компоненты любая пара объектов является дубликатами

return C;

Первое отличие от базового алгоритма в том, что модификация Canopies (Алгоритм 2) строит пары-кандидаты не только как подмножество $A \times B$. Для каждого элемента a множества A выбираются близкие к нему элементы b из множества B. Далее для каждого выбранного элемента b выбирается множество близких к нему элементов на том же множестве b. Выбранные таким образом элементы образуют результирующее множество пар, подмножество $(A \times B) \cup (B \times B)$.

Второе отличие заключается в добавлении пар, которые не были отмечены классификатором непосредственно. Считается, что внутри каждой компоненты связности любая пара объектов является дубликатами.

Как будет показано далее, мэтчинг между множествами A и B соответствует мэтчингу уже находящихся в базе объектов и новых, а сопоставление на множестве B соответствует поиску дубликатов среди новых объектов.

Algorithm 2 Модифицированный алгоритм Canopies

Require: $A, B, Distance, T_{loose}$ — множества A, B, вычислительно не трудоемкая метрика, параметр, определяемый эвристическим путем;

$CandidatePairs = \emptyset$;
$PossibleCentres = A$;
while $PossibleCentres \neq \emptyset$ **do**
  $a = PickRandom(PossibleCentres)$;
  $Canopy(a) = \{(a, b) \ : \ b \in B$ **and** $Distance(a, b) \leq T_{loose}\}$;
  $CandidatePairs.AddAll(Canopy(a))$;
  for $(a, b) \in Canopy(a)$ **do**
    $PairsToAdd = \{(b, x) \ : \ x \in B)$ **and** $Distance(b, x) \leq T_{loose}\}$;
    $CandidatePairs.AddAll(PairsToAdd)$;
  $PossibleCenters.Remove(a)$;
return $CandidatePairs$;

7.2 Алгоритм обновления графа цитирования

Наконец, опишем процесс обновления графа цитирования по коллекции документов.

Пусть O — объекты-метаописания уже включенные в граф цитирования, S — библиографические записи в хранилище, для которых не были найдены соответствующие документы (не удалось найти подходящие метаописания), N — новые библиографические записи выделенные из поступивших документов, I — новые метаописания выделенные из поступивших документов.

Обозначим процедуру адаптивного мэтчинга (Алгоритм 1) на множествах A, B за $AM(A, B)$.

На первом шаге выполняется $AM(O, I)$. Это соответствует проверке на наличие в графе цитирования документов, которые добавляются. Данный шаг можно заменить просто «жестким» сравнением, однако использование адаптивного мэтчинга является более гибким.

По результатам первого шага выполняется включение новых документов в граф.

Затем выполняется $AM(O \cup S, N)$. Связываются новые выделенные библиографические записи (N) с теми, что уже включены в граф цитирования и есть в хранилище. Это или сами документы (O), или если не получается, то другие записи, для которых не было найдено документов (S). Если соответствий не найдено, они добавляются в S.

Наконец, выполняется $AM(I, S)$. Теперь связываются те записи, которые находятся в хранилище (которые не удалось соотнести с метаописаниями) (S) с новыми метаописаниями (I).

Ключевой особенностью является возможность применять один и тот же алгоритм связывания многократно для решения однотипных подзадач.

8 Конкретная схема построения графа цитирования

Описанный выше алгоритм иллюстрирует общую схему, которая не уточняет реализации конкретных шагов. Далее предлагается один из возможных вариантов, детализирующий описанную схему.

Документы, метаописания и библиографические записи Документ является совокупностью следующих объектов:

1. Основное метаописание

2. Дубликаты метаописания
3. Агрегированное метаописание
4. Ссылки на другие документы
5. Ссылки других документов на данный
6. Прочая информация, не участвующая в процессе построения графа цитирования

Основное метаописание (далее просто «метаописание») является первичным объектом, который определяет документ. Далее будет показано, что основное метаописание может являться метаописанием документа (в терминах предыдущих разделов), либо может быть сформировано на основе библиографической записи.

В описанном ранее алгоритме определяется множество S библиографических записей без документов, на практике, однако, удобнее создавать «пустые» документы, метаописание которых формируется из библиографических записей.

При связывании библиографических записей и метаописаний можно использовать библиографические записи в качестве дубликатов метаописания. Пусть документ a имеет библиографическую запись (ссылку) b на документ c, тогда если удалось связать запись b и основное метаописание c, можно считать что b является дубликатом c.

Дубликаты полезны во-первых для связывания — имея различные варианты записи метаописания, связав хотя бы один из них, можно установить цитирование. Во-вторых, при помощи дубликатов можно формировать агрегированное метаописание для случаев, когда основное метаописание не полно. Агрегированное метаописание полезно как наиболее полная информации о документе для конечного пользователя.

8.1 Включение одного документа

Начнем формальное описание алгоритма с простого случая, когда документы включаются по одному.

Обозначения Используем следующие обозначения: O — документы включенные в граф цитирования, d — документ, $d.bib$ — библиографические записи документа d, $d.meta$ — метаописание документа d, $d.dup$ — дубликаты метаописания документа d.

Как и ранее будет использоваться процедура адаптивного мэтчинга $AM(A, B)$, возвращающая пары объектов-дубликатов (пары объектов можно считать ребрами графа, поэтому такое представление эквивалентно компонентам связности). Далее отождествляются понятия вершины графа и объекта, которому эта вершина соответствует.

Как и ранее, считается, что внутри компоненты связности любая пара объектов считается дубликатами. Связывание документов выполняется по основному метаописанию.

В некоторых случаях множество B будет состоять из одного элемента, а если в качестве аргументов выступают множества документов, то считается, что вместе с каждым документом d в связывании участвуют все записи из $d.dup$. Пусть некая запись (или метаописание) c оказалась связана с $b \in d.dup$, тогда в результирующем наборе будет представлена пара (c, d) (а не (c, b)). Это позволит упросить рассуждения.

Будут использоваться дополнительные переменные $MatchedDocuments$ — множество пар связанных документов, $Components$ — компоненты связности, полученные в результате процедуры адаптивного мэтчинга $AM(O, d.bib)$, подмножество $(O \times d.bib) \cup (d.bib \times d.bib)$.

Также используем вспомогательную функцию $Verticies(C)$, которая возвращает множество вершин в компоненте связности C.

Алгоритм Алгоритм включения одного документа в граф цитирования описан псевдокодом Алгоритма 3.

Рассмотрим подробнее отдельные этапы алгоритма.

Основная задача оператора $MatchedDocuments = AM(O, d)$ — найти дубликаты добавляемого документа d. Если будет найден один документ-дубликат x, то после выполнения всех операций, необходимо объединить документы x и d. Проблема возникает, когда в результате будет найдено более одного дубликата, например x и y. Считая, что свойство «идентичности» транзитивно, x и y также являются дублика-

Algorithm 3 Алгоритм включения одного документа в граф цитирования

Require: O, d — множество документов графа цитирования O, включаемый документ d;

```
MatchedDocuments = AM(O, d);
Components = AM(O, d.bib);
for C ∈ Components do
    V = Verticies(C);
    M = V ∩ O;
    if |M| = 0 then
        CreateDocument(V, d);
        continue
    if |M| ≥ 2 then
        MatchedDocuments = MatchedDocuments ∪ (M × M)
    for m ∈ M do
        Link(d, m, V/M);
Merge(MatchedDocuments);
```

тами друг для друга. То есть в данном случае необходимо выполнить слияние уже трех документов d, x, y, двое из которых уже включены в граф цитирования.

Если предположить, что сначала в граф цитирования был включен документ x, а после него документ y, то при включении y документ x не был найден как дубликат для y (иначе вместо x и y был бы другой документ, полученный их слиянием). Дубликатами x и y оказались только после включения «документа-посредника» d через которого они оказались связаны. Можно предположить, что такая ситуация достаточно редка.

Переменная $MatchedDocuments$ будет накапливать информацию для слияния документов.

После вызова $Components = AM(O, d.bib)$ в переменной $Components \subseteq (O \times d.bib) \cup (d.bib \times d.bib)$ хранятся компоненты связности, соответствующие группам связанных объектов. Далее в цикле обрабатывается каждая такая компонента.

Множество $V = Verticies(C)$ состоит из двух подмножеств объектов, соответствующих документам из O (множество M) и библиографическим записям из $d.bib$. Необходимо рассмотреть отдельно следующие случаи.

Если множество M пусто, то в компоненте C нет ни одного документа из O. То есть C состоит из связанных библиографических записей (причем так как все они выделены из одного документа, в большинстве случаев $|C| = 1$). В таком случае создается новый документ (Алгоритм 4) и итерация цикла завершается.

Если M содержит два и более элемента, то повторяя рассуждения выше, можно считать, что документы элементы множества M являются дубликатами и необходимо выполнить их слияние. Для этого пока только добавляется информация в виде вызова $MatchedDocuments = MatchedDocuments \cup (M \times M)$. Вообще говоря, достаточно пополнить $MatchedDocuments$ множеством пар, таким, чтобы оно связывало все объекты из M. Можно использовать, например граф-цепочку, тогда вместо $M \times M$ будет $\{(m_1, m_2), (m_2, m_3), ..., (m_{|M|-1}, m_{|M|}) | m_i \in M\}$.

Для случаев непустого множества M выполняется установка цитирования (Алгоритм 5) между документами множества M и документом d. На самом деле достаточно установить цитирование только с одним любым документом из M так как в конце концов все они сливаются в один документ и ссылки цитирования объединяются.

Наконец, по завершению итераций внешнего цикла выполняется слияние документов по накопленной информации (Алгоритм 6). Множество пар документов-дубликатов, хранящихся в $MatchedDocuments$ задает некоторый граф, дубликатами в нем являются документы принадлежащие одной компоненте связности. Слияние выполняется в конце всего процесса, так как после очередного обновления $MatchedDocuments$ может происходить объединение некоторых компонент связности.

При слиянии документов одной компоненты связности необходимо объединить информацию о цитировании и списки дубликатов, и позаботиться о том, чтобы не было цитирования внутри множества сливаемых документов (это маловероятно, но в общем случае исключать нельзя).

В результате новый документ будет включен в сеть цитирования, возможно, произойдет слияние некоторых документов, возможно, появятся новые «пустые» доку-

Algorithm 4 Функция CreateDocument

Require: V, d — множество V дубликатов библиографических записей, добавляемый документ d;

$NewDoc = newDocument()$ — создание нового документа;

$m = SelectMain(V)$ — выбор одной из записей для формирования основного метаописания;

$NewDoc.meta = m$ — основное метаописание;

$Link(d, NewDoc, V/\{m\})$ — связывание документов цитированием;

Algorithm 5 Функция Link

Require: $d1, d2, dups$ — документы $d1$ и $d2$, которые нужно связать цитированием, дубликаты основного метаописания для документа $d2$, фактически библиографические записи, выделенные из документа $d1$;

$SetCites(d1, d2)$ — установить, что $d1$ ссылается на $d2$;

$SetCited(d2, d1)$ — установить, что на $d2$ ссылается $d1$;

$d2.dup = dups$ — установить дубликаты метаописания для $d2$;

Algorithm 6 Функция Merge

Require: $MatchedDocuments$ — множество пар документов-дубликатов;

$Components = Components(MatchedDocuments)$ — построить компоненты связности в графе, задаваемом парами $MatchedDocuments$;

for $C \in Components$ **do**

$MergeDocuments(C)$

менты, сформированные на основе библиографических записей, которые не удалось связать с документами, включенными в граф цитирования.

8.2 Реализация метода Canopies

Время работы алгоритма включения одного документа определяется тем насколько эффективно реализована процедуры адаптивного мэтчинга $AM(A, B)$, остальные операции выполняются за ограниченное время.

Из псевдокода Алгоритма 2 ясно, что значение функции $Distance(a, b)$ вычисляется для всех пар $(a, b) \in A \times B$, эти вычисления необходимо организовать максимально эффективно. Выбор способа представления данных зависит от выбора функции $Distance(a, b)$.

Выбор функции расстояния Независимо от того объекты какой природы связываются (метаописания или библиографические записи), в обоих случаях они представимы в виде мешка слов. Для этого достаточно «пройтись» по всем полям объектов и строки-значения каждого поля разбить на слова. Далее станет ясно почему такое представление данных удобно.

В таком случае среди множества функций расстояний правильно выбрать token-based функцию сравнения строк (см. раздел 1.2.3).

Рассмотрим функцию схожести Джаккарда $J(a, b)$.

$$J(a, b) = \frac{|tokens(a) \cap tokens(b)|}{|tokens(a) \cup tokens(b)|}$$

$tokens(a)$ — множество слов в объекте a.

Организация данных Для эффективной реализации метода Canopies с функцией схожести Джаккарда необходимо организовать инвертированный индекс, в котором ключами являются слова, а значениями — списки объектов, в описаниях которых встречается слово-ключ.

Инвертированные индексы могут быть эффективно реализованы с помощью деревьев или хеш-таблиц.

Тогда для вычисления $J(a, b)$ для фиксированного объекта a и всех объектов b необходимо 1) получить слова $tokens(a)$, 2) для каждого слова $w \in tokens(a)$ полу-

чить с помощью инвертированного индекса множество объектов B_w, 3) объединить множества B_w для всех $w \in tokens(a)$, получить множество B, 4) явно посчитать значение функции $J(a, b)$ для всех $b \in B$.

Особенностью является тот факт, что множество B не велико, а для остальных объектов $c \notin B$, $J(a, c) = 0$ так как $tokens(a) \cap tokens(c) = \emptyset$, что и обеспечивает высокую эффективность.

Такой подход может использовать и другие token-based функции сравнения строк, например TF-IDF. В таком случае может потребоваться хранить дополнительную информацию.

О других техниках вычислительной оптимизации написано в обзорной статье [7].

8.3 Пакетное включение документов

Включение документов пакетами (то есть набором) потенциально может быть более эффективно, если в соответствующем случае реализация процедуры адаптивного мэтчинга $AM(A, B)$ позволяет использовать вычислительные ресурсы более оптимально.

Алгоритм включения пакета документов получается модификацией алгоритма включения одного документа.

Воспользуемся вспомогательными функциями $Bibs(D)$ — получение множества библиографических записей всех документов в пакете D, $Documents(B)$ — для множества библиографических записей B получение множества документов, в которых они содержатся.

Алгоритм Новый алгоритм описан псевдокодом Алгоритма 7.

Рассмотрим модификации по порядку.

Оператор $MatchedDocuments = AM(O, D)$ теперь ищет дубликаты еще и внутри пакета включаемых документов D.

Необходимо обнаруживать возможное цитирование внутри пакета включаемых документов, поэтому $Components$ определяется как $Components = AM(O \cup D, B)$, $B = Bibs(D)$.

Algorithm 7 Алгоритм включения пакета документов в граф цитирования

Require: O, D — множество документов графа цитирования O, включаемый пакет документов D;

$MatchedDocuments = AM(O, D)$;

$B = Bibs(D)$;

$Components = AM(O \cup D, B)$;

for $C \in Components$ **do**

$V = Verticies(C)$;

$M = V \cap (O \cup D)$;

if $|M| = 0$ **then**

$CreateDocument(V, Documents(V))$;

continue

if $|M| \geq 2$ **then**

$MatchedDocuments = MatchedDocuments \cup (M \times M)$

for $(d, m) \in (Documents(V/M) \times M)$ **do**

$Link(d, m, V/M)$;

$Merge(MatchedDocuments)$;

В цикле при обработке компонент связности множество M теперь должно содержать документы из пакета включаемых документов $M = V \cap (O \cup D)$.

Если множество M пусто, то при создании нового документа необходимо учитывать, что библиографические записи дубликаты множества V могут принадлежать разным документам из D, поэтому устанавливать цитирование необходимо между новым созданным документом и документами $Documents(V)$.

Последняя модификация касается внутреннего цикла, устанавливающего цитирование. В данном случае множество дубликатов библиографических записей V/M связаны с множеством документов M, поэтому нужно установить цитирование между всеми документами, которым принадлежат записи V/M со всеми документами M, то есть $Link(d, m, V/M)$, где $(d, m) \in Documents(V/M) \times M$.

9 Заключение

Основным результатом работы является создание технологии автоматического построения графа цитирования по коллекции научных документов, включающую решение серии задач распознавания.

Разработаны алгоритмы предварительной обработки текстов, необходимые для повышения качества распознавания при обработке текстов. Разработаны алгоритм выделения метаописания и алгоритм выделения библиографических записей, показавшие высокое качество распознавания на различных русскоязычных коллекциях. Разработан алгоритм построения графа цитирования, использующий процедуру идентификации и простые алгоритмы и структуры данных для работы с графами.

Изучены три типа задач распознавания: разметка, сегментация и связывание (мэтчинг, идентификация), которые представляют самостоятельные области машинного обучения со своими прикладными задачами и методами их решений, не имеющие прямого отношения к задаче построения графа цитирования.

Ценность текущей работы состоит в том числе в изучении этих областей — истории их развития, специфичных проблемах, современных методах решения задач. В результате выбраны наилучшие методы для решения описанных подзадач основной задачи. Выбранные методы адаптированы и объединены с собственными разработка-

ми, что и позволило построить технологический процесс для решения поставленной задачи, составленный из наиболее современных методов.

10 Библиография

[1] Eugene Agichtein and Venkatesh Ganti. Mining reference tables for automatic text segmentation. In *In proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining*, pages 20–29. ACM Press, 2004.

[2] Doug Beeferman, Adam Berger, and John Lafferty. Statistical models for text segmentation. In *Machine Learning*, pages 177–210, 1999.

[3] Mikhail Bilenko, Raymond Mooney, William Cohen, Pradeep Ravikumar, and Stephen Fienberg. Adaptive name matching in information integration. *IEEE Intelligent Systems*, 18:16–23, September 2003.

[4] Vinayak Borkar, Kaustubh Deshmukh, and Sunita Sarawagi. Automatic segmentation of text into structured records, 2001.

[5] Thorsten Brants. Topic-based document segmentation with probabilistic latent semantic analysis. In *In Proceedings of CIKM (McLean*, pages 211–218. ACM Press, 2002.

[6] Freddy Y. Y. Choi, Peter Wiemer-Hastings, and Johanna Moore. Latent semantic analysis for text segmentation. In *In Proceedings of EMNLP*, pages 109–117, 2001.

[7] Peter Christen. A survey of indexing techniques for scalable record linkage and deduplication. *IEEE Transactions on Knowledge and Data Engineering*, 24:1537–1555, 2012.

[8] Peter Christen, Tim Churches, and Justin Xi Zhu. Probabilistic name and address cleaning and standardisation, 2002.

[9] William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. A comparison of string distance metrics for name-matching tasks. pages 73–78, 2003.

[10] William W. Cohen and Jacob Richman. Learning to match and cluster large high-dimensional data sets for data integration, 2002.

[11] Eli Cortez, Altigran S. da Silva, Marcos André Gonçalves, Filipe Mesquita, and Edleno S. de Moura. Flux-cim: flexible unsupervised extraction of citation metadata. In *JCDL '07: Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries*, pages 215–224, New York, NY, USA, 2007. ACM.

[12] Isaac G. Councill, C. Lee Giles, and Min yen Kan. Parscit: An open-source crf reference string parsing package. In *International language resources and evaluation*. European Language Resources Association, 2008.

[13] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. Duplicate record detection: A survey. *Transactions on knowledge and data engineering*, page 2007, 2007.

[14] Hui Han C. Lee Giles, Eren Manavoglu, Hongyuan Zha, Zhenyue Zhang, and Edward A. Fox. Automatic document metadata extraction using support vector machines. In *In JCDL '03: Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 37–48, 2003.

[15] Taku Kudoh and Yuji Matsumoto. Use of support vector learning for chunk identification. In *In Proceedings of CoNLL-2000 and LLL-2000*, pages 142–144, 2000.

[16] John Lafferty. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. pages 282–289. Morgan Kaufmann, 2001.

[17] Andrew McCallum, Kamal Nigam, and Lyle H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching, 2000.

[18] J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[19] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. 77(2):257–286, 1989.

[20] Marios Skounakis, Mark Craven, and Soumya Ray. Hierarchical hidden markov models for information extraction. In *In Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 427–433. Morgan Kaufmann, 2003.

[21] Shun-Zheng Yu. Hidden semi-markov models. *Artificial Intelligence*, 2010.

[22] Tong Zhang, Fred Damerau, and David Johnson. Text chunking based on a generalization of winnow. *Journal of Machine Learning Research*, 2:615–637, 2001.

[23] Mikhail Bilenko and Raymond J. Mooney. Learning to combine trained distance metrics for duplicate detection in databases. Technical report, 2002.

[24] A. McCallum C. Sutton. An introduction to conditional random fields for relational learning. *Introduction to Statistical Relational Learning*, 2006.

[25] J. Connan and C. W. Omlin. Bibliography extraction with hidden markov models. Technical report, 2000.

[26] Olivier Ferret. Finding document topics for improving topic segmentation, 2007.

[27] Shai Fine and Yoram Singer. The hierarchical hidden markov model: Analysis and applications. In *MACHINE LEARNING*, pages 41–62, 1998.

[28] Aidan Finn and Nicholas Kushmerick. Learning to classify documents according to genre. In *In IJCAI-03 Workshop on Computational Approaches to Style Analysis and Synthesis*, 2003.

[29] P. Fragkou, V. Petridis, and Ath. Kehagias. A dynamic programming algorithm for linear text segmentation. *Journal of Intelligent Information Systems*, 2002.

[30] Dayne Freitag and Andrew Kachites Mccallum. Information extraction with hmms and shrinkage. In *In Proceedings of the AAAI-99 Workshop on Machine Learning for Information Extraction*, pages 31–36, 1999.

[31] F. Golcher. Statistical text segmentation with partial structure analysis. *Proceedings of 8th Conference on Natural Language Processing (KONVENS 2006)*, 2006.

[32] Anna Kazantseva and Stan Szpakowicz. Linear text segmentation using affinity propagation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 284–293, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics.

[33] Tomanek K. Klinger, R. *Classical Probabilistic Models and Conditional Random Fields.* Dortmund University of Technology, 2007.

[34] Taku Kudo and Yuji Matsumoto. Chunking with support vector machines, 2001.

[35] Andrew Mccallum and Kedar Bellare. A conditional random field for discriminatively-trained finite-state string edit distance. In *In Conference on Uncertainty in AI (UAI*, 2005.

[36] Andrew Mccallum and Dayne Freitag. Maximum entropy markov models for information extraction and segmentation. pages 591–598. Morgan Kaufmann, 2000.

[37] Alvaro Monge and Charles Elkan. The field matching problem: Algorithms and applications. In *In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 267–270, 1996.

[38] Alvaro Monge and Charles Elkan. An efficient domain-independent algorithm for detecting approximately duplicate database records, 1997.

[39] Raul Abella Perez, Jose Eladio, and Medina Pagola. An incremental text segmentation by clustering cohesion.

[40] Eric Sven Ristad and Peter N. Yianilos. Learning string edit distance, 1997.

[41] Sunita Sarawagi and William W. Cohen. Semi-markov conditional random fields for information extraction. In *In Advances in Neural Information Processing Systems 17*, pages 1185–1192, 2004.

[42] Kristie Seymore, Andrew Mccallum, and Ronald Rosenfeld. Learning hidden markov model structure for information extraction. In *In AAAI 99 Workshop on Machine Learning for Information Extraction*, 1999.

[43] Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. pages 213–220, 2003.

[44] T. F. Smite and M. S. Waterman. Identification of common molecular subsequences.

[45] Greig Shmueli Vans Staelin, Elad. Biblio: automatic meta-data extraction. 2006.

[46] J. P. Yamron, I. Carp, L. Gillick, S. Lowe, and P. van Mulbregt. A hidden markov model approach to text segmentation and event tracking. In *Proc. IEEE Int Acoustics, Speech and Signal Processing Conf*, volume 1, pages 333–336, 1998.

[47] Seamus Ross Yunhyong Kim. The naming of cats : Automated genre classification. *International Journal of Digital Curation*, 2008.

11 Собственные публикации

[48] П.Ю. Кудинов and В.А. Полежаев. Инкрементное обучение деревьев решений в задаче распознавания стрктуры статистических таблиц. In *Всероссийская конференция Математические методы распознавания образов (ММРО-15).*, 2011.

[49] П.Ю. Кудинов В.А. Полежаев. Динамическое обучение распознаванию статистических таблиц. In *Международная конференция Интеллектуализация обработки информации (ИОИ-8).*, 2010.

[50] П.Ю. Кудинов В.А. Полежаев. Композиция случайных инкрементных деревьев и восстановление структуры таблиц. *Бизнес-информатика*, 4(18):22–29, 2011.

[51] В.А. Полежаев. Задачи и методы автоматического построения графа цитирования по коллекции научных документов. In *Международная конференция Интеллектуализация обработки информации (ИОИ-9).*, 2012.

[52] В.А. Полежаев. Задачи и методы автоматического построения графа цитирований по коллекции научных документов. *Компьютерные исследования и моделирование.*, 4(4):707–720, 2012.

12 Приложение

12.1 Список признаков для задачи выделения списков библиографии

1. Средняя длина слов $\in \mathbb{R}_+$.
2. Есть ли «библиографический номер» (вида [1] или 1.) в первых k символах (k = 20) $\in \{0, 1\}$.
3. Число слов, начинающихся с заглавных букв $\in \mathbb{Z}_+$.
4. Число пар слов, начинающихся с заглавных букв $\in \mathbb{Z}_+$.
5. Число пар слов, начинающихся с заглавных букв в первых k символах (k = 30) $\in \mathbb{Z}_+$.
6. Число пар слов, начинающихся с заглавных букв в последних k символах (k = 30) $\in \mathbb{Z}_+$.
7. Число запятых $\in \mathbb{Z}_+$.
8. Нормированное расстояние до предыдущего ключевого слова (Например, References) $\in [0; 1]$, измеряемое в строках.
9. Число инициалов $\in \mathbb{Z}_+$.
10. Число инициалов в первых k символах (k = 20)$\in \mathbb{Z}_+$.
11. Число инициалов в последних k символах (k = 20) $\in \mathbb{Z}_+$.
12. До или после ключевого слова $\in \{0, 1\}$.
13. Есть ли цифры в первых k символах (k = 20) $\in \{0, 1\}$.
14. Число цифр $\in \mathbb{Z}_+$.
15. Число цифр в первых k символах (k = 20) $\in \mathbb{Z}_+$.
16. Число цифр в последних k символах (k = 20) $\in \mathbb{Z}_+$.

17. Соотношение числа цифр и слов $\in \mathbb{R}_+$.
18. Число сокращений «страница» $\in \mathbb{Z}_+$.
19. Число точек $\in \mathbb{Z}_+$.
20. Число точек в первых k символах (k = 20) $\in \mathbb{Z}_+$.
21. Число точек в последних k символах (k = 20) $\in \mathbb{Z}_+$.
22. Относительная длина строки (относительная, в смысле как отношение к прогнозируемой ширине полосы текста, об этом далее) $\in \mathbb{R}_+$.
23. Относительная длина предыдущей строки $\in \mathbb{R}_+$.
24. Число сокращений «секция» $\in \mathbb{Z}_+$.
25. Число служебных символов (скобки, тире, кавычки и прочие) $\in \mathbb{Z}_+$.
26. Число служебных символов в первых k символах (k = 20) $\in \mathbb{Z}_+$.
27. Число служебных символов в последних k символах (k = 20) $\in \mathbb{Z}_+$.
28. Число заглавных букв $\in \mathbb{Z}_+$.
29. Число заглавных букв в первых k символах (k = 20) $\in \mathbb{Z}_+$.
30. Число заглавных букв в последних k символах (k = 20) $\in \mathbb{Z}_+$.
31. Число сокращений «том» $\in \mathbb{Z}_+$.
32. Число слов $\in \mathbb{Z}_+$.
33. Число идущих подряд четырех цифр $\in \mathbb{Z}_+$.

12.2 Список признаков для задачи выделения метаописания

1. Строка содержит слово «Аннотация» $\in \{0, 1\}$.
2. Средняя длина слов в строке $\in \mathbb{R}_+$.
3. Строка содержит слово «ББК» $\in \{0, 1\}$.

4. Строка содержит слова «Литература», «Список литературы», «Библиография» с произвольным регистром букв $\in \{0, 1\}$.

5. Все буквы в строке — заглавные $\in \{0, 1\}$.

6. Количество заглавных букв $\in \mathbb{Z}_+$.

7. Количество слов, начинающихся с заглавных букв $\in \mathbb{Z}_+$.

8. Наличие слов в строке в словаре с названиями городов $\in \{0, 1\}$.

9. Количество запятых $\in \mathbb{Z}_+$.

10. Отношение количества цифр к количеству слов $\in \mathbb{R}_+$.

11. Строка содержит слова «совета» и «Д» $\in \{0, 1\}$.

12. Количество точек $\in \mathbb{Z}_+$.

13. Строка содержит слова «доктора» или «кандидата» и наук $\in \{0, 1\}$.

14. Строка содержит email $\in \{0, 1\}$.

15. Число четырех идущих подряд цифр $\in \mathbb{Z}_+$.

16. Номер группы, в которую входит строка (строки объединяются в группы согласно порядку) $\in \mathbb{Z}_+$.

17. Строка содержит слова «рис», «илл», «рисунков», «иллюстраций» с произвольным регистром букв $\in \{0, 1\}$.

18. Число инициалов $\in \mathbb{Z}_+$.

19. Строка содержит фразу «ключевы слова» в разных вариациях $\in \{0, 1\}$.

20. Строка заканчивается числом $\in \{0, 1\}$.

21. Строка содержит фразу «на правах рукописи» с произвольным регистром букв $\in \{0, 1\}$.

22. Количеств цифр $\in \mathbb{Z}_+$.

23. Номер строки $\in \mathbb{Z}_+$.

24. Строка содержит фразу «официальные оппоненты» с произвольным регистром букв $\in \{0, 1\}$.

25. Строка содержит слова «стр», «с», «страниц», «р», «pages» $\in \{0, 1\}$.

26. Строка содержит слово «телефон» в разных вариациях $\in \{0, 1\}$.

27. Наличие слов в строке в словаре с названиями издательств $\in \{0, 1\}$.

28. Строка содержит слово «Издательство» с произвольным регистром букв $\in \{0, 1\}$.

29. Строка содержит слово «Рекомендован» с произвольным регистром букв $\in \{0, 1\}$.

30. Строка содержит слова «редактор», «корректор», «ред», «иллюстратор» с произвольным регистром букв $\in \{0, 1\}$.

31. Строка содержит фразу «Научный руководитель» с произвольным регистром букв $\in \{0, 1\}$

32. Строка содержит фразу «Ученый секретарь» с произвольным регистром букв $\in \{0, 1\}$

33. Число служебных символов $\in \mathbb{Z}_+$.

34. Строка содержит слово «Специальность» с произвольным регистром букв $\in \{0, 1\}$

35. Строка содержит фразу «Алфавитный(предметный) указатель» с произвольным регистром букв $\in \{0, 1\}$

36. Строка содержит слово «Автореферат» с произвольным регистром букв $\in \{0, 1\}$

37. Строка содержит слова «Оглавление», «Содержание» с произвольным регистром букв $\in \{0, 1\}$

38. Строка содержит слово «УДК» $\in \{0, 1\}$

39. Строка содержит URL $\in \{0, 1\}$

40. Количество слов $\in \mathbb{Z}_+$.