ИСП Российская Академия наук
Институт Системного Программирования

ISSN 2079-8156 (Print)
ISSN 2220-6426 (Online)

Труды Института Системного Программирования РАН

Proceedings of the Institute for System Programming of the RAS

Том (Volume) 26

выпуск (issue) 4

Москва 2014

ИСП Российская Академия наук
Институт Системного Программирования

Труды Института Системного Программирования РАН

Proceedings of the Institute for System Programming of the RAS

Том (Volume) 26
Выпуск (issue) 4

Под редакцией
академика РАН В.П. Иванникова
Edited by
Academician V.P. Ivannikov

Москва 2014

УДК004.45

Труды Института системного программирования: Том 26, выпуск 4. /Под ред. Академика РАН В.П. Иванникова/ – М.: ИСП РАН, 2014.

Proceedings of the Institute for System Programming: Volume 26, issue 4. /Edited by Academician V.P. Ivannikov/ – M.: ISP RAS, 2014.

Четвертый выпуск 26-го тома Трудов ИСП РАН посвящен управлению данными, обработке и анализу данных. В этом выпуске содержатся статьи, описывающие результаты исследований, выполненных сотрудниками отдела Информационных систем ИСП РАН в 2014 г., а также расширенные тексты докладов, представленных авторами на конференциях SYRCoDIS’2013 и RuSSIR YSC 2013.

The fourth issue of the 26th volume of the Proceedings of the Institute for System Programming of the RAS concentrated on data management, data processing and analysis. This issue contains articles written by researchers of Information Systems Department of ISP RAS in 2014 and advanced reports from SYRCoDIS'2013 and RuSSIR YSC 2013.

ISSN 2079-8156 (Print)
ISSN 2220-6426 (Online)

© Институт Системного Программирования РАН, 2014
© Institute for System Programming of the RAS, 2014

Содержание

П р е д и с л о в и е

Четвертый выпуск 26-го тома Трудов ИСП РАН посвящен управлению данными, обработке и анализу данных. В первых четырех статьях выпуска описываются некоторые результаты исследований, выполненных сотрудниками отдела Информационных систем ИСП РАН в 2014 г.

Следующие две статьи содержат расширенные тексты докладов, представленных авторами на конференции по тематике баз данных и информационных систем SYRCoDIS'2013 (http://syrcodis.ispras.ru/2013/), и завершают выпуск три статьи с расширенными текстами докладов на конференции по тематике информационного поиска RuSSIR YSC 2013 (http://romip.ru/russir2013/section.php?id=149). В организации этих конференций и их работе принимали активное участие сотрудники ИСП РАН.

С.Д. Кузнецов

Автоматическое извлечение терминов из коллекции текстов предметной области с помощью Википедии

Н.А. Астраханцев <astrakhantsev@ispras.ru>
ИСП РАН, 109004, Россия, г. Москва, ул. А. Солженицына, дом 25

Аннотация. Автоматическое извлечение терминов является важной задачей во многих приложениях, связанных с обработкой текстов предметной области. В настоящее время существует множество методов извлечения терминов, однако они недостаточно полным образом используют внешние ресурсы, в частности – интернет-энциклопедию Википедия. Кроме того, существующие методы сильно зависят от языка и предметной области входной коллекции текстов. В данной работе предлагаются два новых признака: «Вероятность быть гиперссылкой» – нормализованная частота, с которой кандидат в термины является гиперссылкой в статьях Википедии; и «Близость к ключевым концептам» – среднее арифметическое значений семантической близости к ключевым понятиям заданной предметной области, определяемым автоматически на основе входной коллекции текстов предметной области. Также в данной работе предлагается новый автоматический метод извлечения терминов, основанный на алгоритме частичного обучения и не требующий размеченных данных. Схема метода состоит в извлечении лучших 100-300 кандидатов, присутствующих в Википедии, с помощью специального метода и последующем использовании этих кандидатов как положительных примеров для построения модели алгоритма обучения на основе положительных и неразмеченных примеров. Проведенное экспериментальное исследование на четырех предметных областях (настольные игры, биомедицина, информатика, сельское хозяйство) показывают значительное превосходство предложенного метода и его независимость от предметной области: средняя точность возросла на 5-17% по сравнению с лучшим из существующих методов для конкретного набора данных.

Ключевые слова: извлечение терминологии; Википедия; обучение на основе положительных и неразмеченных примеров.

1. Введение

Автоматическое извлечение терминов, то есть слов и словосочетаний, обозначающих определенные понятия заданной предметной области, является важным этапом во многих задачах, связанных с обработкой текстов предметной области. К таким задачам относятся, например, построение и

обогащение глоссариев, тезаурусов или онтологий [1], информационный поиск [2], извлечение информации [3], классификация и кластеризация документов [4] и т. п.

К настоящему времени разработано множество методов автоматического извлечения терминов, однако, как и во многих других задачах автоматической обработки текстов, большая часть методов существенно зависит от языка и предметной области входных текстов, что естественным образом сужает практическую применимость метода.

Кроме того, источником данных в большинстве методов является исключительно коллекция текстовых документов предметной области. Некоторые методы также используют внешние ресурсы, такие как корпуса текстов других предметных областей, поисковые машины или онтологии, созданные экспертами, однако все эти ресурсы обладает своими недостатками.

Так, внешние текстовые документы, в том числе найденные поисковыми машинами, лишены структуры и позволяют использовать только статистическую информацию о встречаемости слов и словосочетаний вне рассматриваемой предметной области; использование специализированных онтологий, например Генной онтологии [5], практически исключает возможность переноса метода на другие предметные области; универсальные онтологии, например WordNet [6] или РуТез [7], обладают малым объемом (порядка 100-150 тыс. терминов) и покрывают лишь самые общие понятия предметных областей.

Указанных недостатков лишена многоязычная интернет-энциклопедия Википедия[1]. Она содержит структурную информацию в виде соответствия статей понятиям реального мира, а также в виде гиперссылок между статьями; Википедия ежедневно пополняется сообществом пользователей, обладает очень большим размером (английская версия насчитывает более 4.5 млн. статей) и сочетает в себе универсальность и предметную специфичность.

За последние годы появилось несколько методов извлечения терминов на основе Википедии, однако в качестве возможных терминов они рассматривают только названия статей Википедии, что заведомо ограничивает полноту извлечения терминов (например, предметная область «Сельское хозяйство» покрывается Википедией только на 50% [8]).

В данной работе предлагается автоматический метод извлечения терминов из коллекции текстов с использованием Википедии, не ограничивающий набор возможных терминов и не зависящий от предметной области и языка.

Данная статья устроена следующим образом. Следующий раздел посвящен обзору существующих методов извлечения терминов. Раздел 3 представляет новый метод извлечения терминов на основе Википедии. В разделе 4 описывается проведенное экспериментальное исследование. В заключении подводятся итоги работы и предлагаются направления будущих исследований.

[1] http://wikipedia.org

2. Обзор существующих работ

Методы извлечения терминов, как правило, состоят из трех последовательных этапов:

- **Сбор кандидатов**: фильтрация слов и словосочетаний, извлеченных из коллекции документов, по статистическим и лингвистическим критериям.
- **Подсчет признаков**: перевод каждого кандидата в вектор признакового пространства.
- **Вывод на основе признаков**: классификация кандидатов на термины и не термины либо сортировка всех кандидатов по вероятности быть термином и взятие заранее определенного числа кандидатов.

2.1 Сбор кандидатов

Методы сбора кандидатов также состоят из нескольких шагов. На первом шаге применяются лингвистические фильтры, цель которых – оставить только существительные и именные группы, то есть словосочетания с существительным в роли главного слов. Для этого применяется поверхностный синтаксический разбор (shallow parsing, chunking) или фильтрация N-грамм по шаблонам частей речи.
На последующих шагах сбора кандидатов с целью снижения шума производится дополнительная фильтрация по частоте либо содержанию стоп-слов из заранее составленного списка.

2.2 Признаки для извлечения терминов

Подавляющее большинство признаков для извлечения терминов основано на частоте вхождения кандидатов в рассматриваемую коллекцию текстовых документов. К таковым относятся, например, TF-IDF [9], LexicalCohesion [10], CValue [11], DomainConsensus [12], статистические критерии Стьюдента (TTest) и логарифмического правдоподобия (Loglikelihood) [9], методы на основе тематических моделей [13]. Некоторые признаки также учитывают контекст вхождений, например NCValue [14] и Domain Model [15]. В других признаках используется частота вхождений во внешнюю коллекцию документов, не принадлежащую какой-либо предметной области: Weirdness [16], DomainRelevance [17], Relevance [18].

2.3 Вывод на основе признаков

На данном этапе, в случае использования нескольких признаков, возникает задача преобразования вектора признаков в число, показывающее уверенность метода в том, что данный кандидат является термином.

Наиболее простым способом является линейная комбинация, применяемая, например, в методе TermExtractor [17].

В работе [19] предлагается метод на основе алгоритма голосования:

$$V(t) = \sum_{i=1}^{n} \frac{1}{r(f_i(t))}$$

где t – кандидат в термины, n – количество признаков, $r(f_i(t))$ – позиция кандидата t среди всех кандидатов, отсортированных по значению признака $f_i(t)$. Данный метод не требует нормализации признаков и показывает в среднем лучшие результаты [19].

При наличии размеченных данных становится возможным применять алгоритмы машинного обучения с учителем, в частности AdaBoost [20], Ripper [21], машину опорных векторов (SVM) [22]. Как было показано в работе [23], классификаторы на основе машинного обучения достигают лучшей средней точности.

3. Метод извлечения терминов на основе Википедии

В данном разделе описываются новые признаки, основанные на использовании информации Википедии, а также новый метод, применяющий алгоритм обучения на основе положительных и неразмеченных примеров.

3.1 Признаки на основе Википедии

Признак *«Вероятность быть гиперссылкой»* (LinkProb) представляет собой нормализованную частоту, с которой кандидат в термины является гиперссылкой в статьях Википедии. Значение этого признака будет близко к нулю для слов и словосочетаний, являющихся частью общей лексики, то есть не принадлежащих какой-либо предметной области. Таким образом, мотивация использования этого признака заключается в фильтрации таких слов и словосочетаний, поскольку они, скорее всего, не принадлежат и к предметной области, для которой извлекаются термины. Стоит также отметить, что данный признак используется в методах разрешения лексической многозначности [24].

Например, словосочетание *Last card* (последняя карта) встречается 332 раза в статьях Википедии и всего лишь 4 раза в виде гиперссылки (на статью про карточную игру с одноименным названием). Таким образом, значение признака составит 0.012 и при прочих равных данное словосочетание не будет отнесено к терминам.

Признак «*Близость к ключевым концептам*» (KeyRel) основан на семантической близости к ключевым понятиям заданной предметной области и вычисляется по формуле:

$$Key\,\mathrm{Re}\,l(t) = \max_{c \in C(t)} \frac{1}{n} \sum_{i=1}^{n} sim(c, k_i)$$

где t – кандидат в термины; $C(t)$ – возможные концепты кандидата, то есть все статьи Википедии с названием, совпадающим с t с точностью до лемматизации; k_i – ключевые концепты предметной области, т.е. статьи Википедии, описывающие наиболее важные понятия для предметной области; $sim(c,k)$ – функция семантической близости, определенная для любой пары концептов и имеющая значения от 0 до 1: чем ближе значение функции к 1, тем больше общего между концептами.

Значение этого признака будет близко к нулю для слов и словосочетаний, обозначающих понятия, далекие по смыслу от ключевых понятий предметной области. Данный признак также используется в методах разрешения лексической многозначности [25].

Семантическая близость вычисляется с помощью системы Текстерра [24], в которой используется взвешенный алгоритм Дайса. Ключевые концепты для предметной области определяются на основе ключевых концептов, извлеченных для каждого документа из заданной коллекции: для каждого ключевого концепта подсчитывается, во скольких документах он был выбран ключевым, и отбирается 3 наиболее частых. Для извлечения ключевых концептов из документа используется метод KPMiner [26], реализация которого также взята из системы Текстерра.

Рассмотрим в качестве примера все то же словосочетание *Last card* применительно к предметной области «Настольные игры». Допустим, из коллекции текстов про настольные игры были извлечены следующие ключевые концепты:

- Board game (собственно, Настольная игра)
- Card game (Карточная игра)
- Hasbro Inc. (Компания, занимающаяся производством игрушек и настольных игр)

Как уже упоминалось выше, в Википедии существует статья *Last card* про одноименную игру; значения семантической близости этого концепта с указанными в списке составляют 0.001, 0.037 и 0, соответственно. Таким образом, значение признака составляет 0.0127. В то же время для термина Gene (Ген) семантическая близость к указанным концептам будет равна нулю и, таким образом, термин *Last card* является более вероятным термином предметной области «Настольные игры» с точки зрения признака «Близость к ключевым концептам».

3.2 Подход на основе частичного обучения

Идея метода основана на наблюдении, что небольшая верхняя часть списка кандидатов, полученных с помощью многих существующих методов извлечения терминов, представляет собой в основном действительно правильно определенные термины, которые могут рассматриваться в качестве обучающих данных для извлечения остальных терминов. Кроме того,

покрытие Википедии для большинства предметных областей позволяет с помощью простых методов найти небольшое число правильных терминов.

Более точно, схема метода состоит в определении лучших 100-300 кандидатов, присутствующих в Википедии, с помощью специального метода извлечения терминов и использовании этих кандидатов как положительных примеров для построения модели алгоритма обучения на основе положительных и неразмеченных примеров (Positive-unlabeled learning, PU-learning) – частного случая алгоритма частичного обучения (Semi-supervised learning). В данном случае неразмеченными примерами служат все остальные кандидаты в термины.

3.2.1 Метод извлечения положительных примеров

В данной работе используется собственный метод ModBasic в качестве метода извлечения кандидатов для последующего обучения. ModBasic представляет собой модификацию метода Basic, являющегося частью метода DomainModel [15] и, в свою очередь, модификацией широко распространенного метода CValue [11]. Метод Basic вычисляется по формуле:

$$b(t) = |t| \log f(t) + \alpha e_t$$

где t – кандидат в термины, $|t|$ - длина кандидата t, $f(t)$ – частота вхождений t в коллекции текстов, e_t – количество кандидатов, объемлющих кандидата t.

Метод Basic предназначен для извлечения «средне-специфичных» терминов, то есть терминов, описывающие распространенные понятия заданной предметной области, например, «колода карт» можно назвать средне-специфичным термином для предметной области «Настольные игры».

В методе ModBasic используется та же формула, однако вместо числа объемлющих кандидатов e_t обозначает количество кандидатов, содержащихся в данном кандидате t. Легко заметить, что данный метод предпочитает более специфичные термины, чем метод Basic, например термины «полная колода карт» или «сокращенная колода карт».

Кроме того, как было отмечено выше в описании подхода, в целях снижения шума производится фильтрация по наличию термина в названиях статей Википедии.

3.2.2 Алгоритм обучения на основе положительных и неразмеченных примеров

К настоящему времени разработано множество алгоритмов обучения на основе положительных и неразмеченных примеров; в данной работе рассмотрены следующие алгоритмы: Traditional PU learning [27], GradualReduction [28], Spy-EM [29] и PairwiseRanking SVM [30].

В качестве признаков для обучения использовались «Вероятность быть гиперссылкой», «Близость к ключевым концептам», Relevance и DomainModel.

4. Эксперименты

Извлечение терминов из коллекции текстов предметной области подразумевает следующую методологию оценки качества: для входной коллекции текстов формируется эталонное множество терминов, с которым и сравнивается результат работы системы извлечения терминов с помощью стандартных метрик полноты, точности и средней точности. Следует отметить, что на практике зачастую невозможно получить эталонное множество терминов, в точности соответствующее коллекции текстов. В таких случаях точность и полнота оцениваются приближенно.

Далее в этом разделе описываются используемые наборы данных, методика тестирования и результаты тестирования.

4.1 Наборы данных

К сожалению, в настоящее время нет общепринятого набора данных для тестирования методов извлечения терминов. Многие наборы данных и методы обладают спецификой, не позволяющей проводить тестирования в других работах; в некоторых случаях наборы данных не могут распространяться из-за нарушения интеллектуальных прав.

В работе [15] проводилось тестирование на трех открытых наборах данных: GENIA [31], Krapivin [32] и FAO [33].

GENIA можно назвать одним из наиболее популярных наборов данных; он представляет собой коллекцию из 2000 размеченных документов биомедицинской тематики.

FAO состоит из 780 размеченных вручную отчетов Продовольственной и сельскохозяйственной организации ООН (Food and Agriculture Organization).

Krapivin представляет собой 2304 научные статьи по информатике; в качестве эталонного множества терминов используются ключевые слова, выделенные авторами статей. При тестировании в данной работе к этому множеству был добавлен словарь предметной области «Вычислительная техника» (Computing), использованный в качестве эталона в системе Protodog [34].

Кроме того, для тестирования также использовался набор данных Board game [35] – коллекция из 1300 документов – описаний и рецензий настольных игр. 35 документов были размечены вручную, и для тестирования использовались только термины, имеющие хотя бы одно вхождение в размеченные документы.

4.2 Методика тестирования

Качество оценивалось с помощью средней точности:

$$AvP(N) = \sum_{i=1}^{N} P(i)(R(i) - R(i-1))$$

где *N* – общее количество оцениваемых кандидатов, или длина верхней части списка отсортированных кандидатов, *P(i)* – точность для *i* лучших кандидатов, *R(i)* – полнота для *i* лучших кандидатов. Данная метрика широко используется при оценке качества извлечения терминов, поскольку позволяет учитывать точность для всех срезов финального списка терминов.

Для набора данных Board game число N равнялось 500, для остальных наборов данных – 5000.

Кандидаты для всех оцениваемых методов представляли собой N-граммы (от 1 до 4) с фильтрацией по шаблонам частей речи и по частоте (пороговые значения для Board game и Genia – 2; для Krapivin и FAO – 3).

В методах на основе положительных и неразмеченных примеров число лучших кандидатов, используемых для обучения, равнялось 100 для Board game и 300 для остальных наборов данных.

В качестве программной реализации алгоритмов логистической регрессии, Наивного Байеса и машины опорных векторов, лежащих в основе методов Traditional, Gradual Reduction, Spy-EM и PairwiseRanking SVM, использовалась библиотека машинного обучения Weka [36]. Для реализации метода на основе тематических моделей [13] использовался открытый фреймворк[2] для построения многоязыковых регуляризованных робастных тематических моделей.

4.3 Результаты тестирования

Результаты представлены в табл. 1. Учитывая очень медленную работу и низкую эффективность алгоритма PairwiseRanking SVM на наборах данных Board game и Genia, этот алгоритм не тестировался на остальных наборах данных.

Табл. 1. Результаты тестирования

	Board game	GENIA	Krapivin	FAO
Существующие методы				
TermExtractor	0.35597	**0.79171**	0.35821	0.11126
CValue	0.36259	0.76916	0.41915	**0.29431**
Weirdness	0.30592	0.53960	0.29464	0.20923
DomainModel (DM)	0.36377	0.72815	**0.42182**	0.27956
Relevance	**0.38854**	0.55043	0.34864	0.27535
NovelTopicModel	0.34091	0.76585	0.10855	0.07801
Методы, предлагаемые в данной работе				

[2] https://github.com/ispras/tm

LinkProb	0.41065	0.81170	0.18491	0.02362
KeyRel	0.56113	0.79691	0.21254	0.08250
Voting (KeyRel+LinkProb+DM+Relevance)	0.46896	0.79963	0.33703	0.20577
Traditional PU	0.52806	**0.84843**	0.55675	**0.36508**
Spy-EM	0.49429	0.83089	0.47577	0.34436
PairwiseRanking SVM	0.30871	0.65465	-	-
GradualReduction	**0.56159**	**0.84843**	**0.55786**	0.34775

Как видно из таблицы, метод на основе положительных и неразмеченных примеров показывает лучшие результаты, из которых, в свою очередь, лучшим на 3 из 4 наборов данных является алгоритм GradualReduction.

Стоит отметить, что алгоритм DomainModel, продемонстрировавший одни из лучших показателей среди существующих методов, предназначен для извлечения средне-специфичных терминов, в то время как остальные методы, в том числе предложенный в настоящей работе, ставит целью извлечение всех терминов предметной области.

5. Заключение

В данной работе был представлен новый полностью автоматический метод извлечения терминов из коллекции текстов предметной области, использующий Википедию и алгоритм частичного обучения. В частности, были предложены два новых признака, использующих граф ссылок Википедии, а также специальный метод для извлечения лучших 100-200 кандидатов в термины, служащих в качестве положительных примеров для алгоритма обучения на основе положительных и неразмеченных примеров.

Экспериментальное исследование показало значительное превосходство предложенного метода по сравнению с существующими: средняя точность возросла на 5-16% (для разных наборов данных).

Среди основных направлений дальнейших исследований стоит выделить проведение более детальной оценки качества, в частности – отбора признаков и подбора параметров алгоритмов. Кроме того, представляет интерес разработка методов, повторяющих более одного раза описанную схему, то есть извлечение лучших кандидатов в термины и использование их как положительных примеров для последующего обучения.

Список литературы

[1]. Н.А. Астраханцев, Д.Ю. Турдаков. Методы автоматического построения и обогащения неформальных онтологий. Программирование, Т.39, №1, стр. 23-34, 2013.

[2]. Y. Lingpeng, J. Donghong, Z. Guodong, N. Yu. Improving retrieval effectiveness by using key terms in top retrieved documents. Advances in Information Retrieval, Springer, 2005, P. 169–184.
[3]. R. Yangarber, R. Grishman, P. Tapanainen, and S. Huttunen. Automatic acquisition of domain knowledge for information extraction. Proceedings of the 18th conference on Computational linguistics-Volume 2, 2000, P. 940–946.
[4]. Д. Д. Голомазов. Методы и средства управления научной информацией с использованием онтологий. Диссертация на соискание ученой степени кандидата физико-математических наук. МГУ им. Ломоносова, 2012, 154 стр.
[5]. The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. Nat Genet, 25(1), 2007.
[6]. G. A. Miller. WordNet: A Lexical Database for English. Communications of the ACM Vol. 38, No. 11. P. 39-41, 1995.
[7]. А.А. Алексеев, Б.В. Добров, Н.В. Лукашевич. Лингвистическая онтология тезаурус РуТез. Труды конференции Open Semantic Technologies for Intelligent Systems - OSTIS, P. 153–158, 2013.
[8]. D. Milne, O. Medelyan, I. H. Witten. Mining domain-specific thesauri from wikipedia: A case study. Proceedings of the 2006 IEEE/WIC/ACM international conference on web intelligence, 2006, P. 442–448.
[9]. C.D. Manning and H. Schutze. Foundations of statistical natural language processing. MIT press, Cambridge, MA, USA. 1999. 680 p.
[10]. Y. Park, R.J. Byrd, and B.K. Boguraev. Automatic glossary extraction: beyond terminology identification. In Proceedings of the 19th international conference on Computational linguistics-Volume 1, P. 1–7. Association for Computational Linguistics, 2002.
[11]. K.T. Frantzi and S. Ananiadou. Extracting nested collocations. Proceedings of the 16th conference on Computational linguistics-Volume 1, P. 41–46. Association for Computational Linguistics, 1996.
[12]. R. Navigli and P. Velardi. Semantic interpretation of terminological strings. In Proc. 6th Intl Conf. Terminology and Knowledge Eng, 2002, P. 95-100.
[13]. S. Li, J. Li, T. Song, W. Li, B. Chang. A novel topic model for automatic term extraction. In Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval. ACM, 2013, P. 885-888.
[14]. K. Frantzi, S. Ananiadou, H. Mima. Automatic recognition of multi-word terms: the c-value/nc-value method. International Journal on Digital Libraries, vol. 3, no. 2, P. 115–130, 2000.
[15]. G. Bordea, P. Buitelaar, T. Polajnar. Domain-independent term extraction through domain modeling. 10th International Conference on Terminology and Artificial Intelligence (TIA 2013), Paris, France, 2013.
[16]. K. Ahmad, L. Gillam, L. Tostevin, et al. University of surrey participation in trec8: Weirdness indexing for logical document extrapolation and retrieval (wilder). In The Eighth Text REtrieval Conference (TREC-8), 1999.
[17]. F. Sclano and P. Velardi. Termextractor: a web application to learn the shared terminology of emergent web communities. Enterprise Interoperability II, 2007, P. 287-290.
[18]. A. Penas, F. Verdejo, J. Gonzalo, et al. Corpus-based terminology extraction applied to information access. In Proceedings of Corpus Linguistics, volume 2001. Citeseer, 2001.

[19]. Z. Zhang, C. Brewster, F. Ciravegna. A comparative evaluation of term recognition algorithms. In Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC08), Marrakech, Morocco, 2008.
[20]. A. Patry and P. Langlais. Corpus-based terminology extraction. In Terminology and Content Development–Proceedings of 7th International Conference On Terminology and Knowledge Engineering, Litera, Copenhagen, 2005.
[21]. J. Foo, Term extraction using machine learning. Linkoping University, LINKOPING, 2009.
[22]. A. Judea, H. Schütze, S. Brügmann. Unsupervised Training Set Generation for Automatic Acquisition of Technical Terminology in Patents. The 25th International Conference on Computational Linguistics (COLING 2014), Dublin, Ireland, 2014, P. 290-300.
[23]. D. Fedorenko, N. Astrakhantsev, D. Turdakov. Automatic recognition of domain-specific terms: an experimental evaluation. Proceedings of SYRCoDIS 2013, 2013, P. 15-23.
[24]. Д. Турдаков, Н. Астраханцев, Я. Недумов, А. Сысоев, И. Андрианов, В. Майоров, Д. Федоренко, А. Коршунов, С. Кузнецов. Texterra: инфраструктура для анализа текстов. Труды Института системного программирования РАН, том 26, 2014 г. Выпуск 1. Стр. 421-438. DOI: 10.15514/ISPRAS-2014-26(1)-18.
[25]. Д. Ю. Турдаков, С. Д. Кузнецов. Автоматическое разрешение лексической многозначности терминов на основе сетей документов. Программирование, Том. 36, Номер 1, стр. 11-18, 2010.
[26]. S. R. El-Beltagy, A. Rafea. KP-Miner: A keyphrase extraction system for English and Arabic documents. Information Systems, 34(1), P. 132-144, 2009.
[27]. M. Montes, H. J. Escalante. Novel representations and methods in text classification. 7th Russian Summer School in Information Retrieval. Kazan, Russia, 2013.
[28]. M. Montes-y-Gómez, P. Rosso. Using PU-Learning to Detect Deceptive Opinion Spam. WASSA 2013, p. 38, 2013.
[29]. B. Liu, W. S. Lee, P. S. Yu, X. Li. Partially supervised classification of text documents. ICML, 2002, vol. 2, P. 387–394.
[30]. S. Sellamanickam, P. Garg, S. K. Selvaraj. A pairwise ranking based approach to learning with positive and unlabeled examples. Proceedings of the 20th ACM international conference on Information and knowledge management, 2011, P. 663–672.
[31]. J.-D. Kim, T. Ohta, Y. Tateisi, J. Tsujii. GENIA corpus--a semantically annotated corpus for bio-textmining. Bioinformatics, vol. 19, no. Suppl 1, P. 180–182, 2003.
[32]. M. Krapivin, A. Autaeu, M. Marchese. Large dataset for keyphrases extraction. 2009.
[33]. O. Medelyan, I. Witten. Domain-independent automatic keyphrase indexing with small training sets. Journal of the American Society for Information Science and Technology, 59.7, 2008, P. 1026-1040.
[34]. S. Faralli, R. Navigli. Growing Multi-Domain Glossaries from a Few Seeds using Probabilistic Topic Models. EMNLP, 2013, P. 170–181.
[35]. N. Astrakhantsev, D. Fedorenko, D. Turdakov. Automatic Enrichment of Informal Ontology by Analyzing a Domain-Specific Text Collection. Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference “Dialogue”, Issue 13, 2014, P. 29-42.
[36]. M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten. The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1, 2009.

Automatic term acquisition from domain-specific text collection by using Wikipedia

N. Astrakhantsev <astrakhantsev@ispras.ru>
ISP RAS, 25 Alexander Solzhenitsyn Str., Moscow, 109004, Russian Federation

Abstract. Automatic term acquisition is an important task for many applications related to domain-specific texts processing. At present there are many methods for automatic term acquisition, but they are highly dependent on language and domain of input text collection. Also these methods, in general, use domain-specific text collection only, while many external resources are underutilized. We argue that one of the most promising external resources for automatic term acquisition is the online encyclopedia Wikipedia. In this paper we propose two new features: "Hyperlink probability" - normalized frequency showing how often the candidate terms is a hyperlink in Wikipedia articles; and "Semantic relatedness to the domain key concepts" - arithmetic mean of semantic relatedness to the key concepts of a given domain; those key concepts are determined automatically on the basis of input domain-specific text collection. In addition, we propose a new method for automatic term acquisition. It is based on semi-supervised machine learning algorithm, but it does not require labeled data. Outline of the method is to extract the best 100-300 candidates presented in Wikipedia by using a special method for term acquisition, and then to use these candidates as positive examples to construct a model for a classifier based on positive-unlabeled learning algorithm. An experimental evaluation conducted for the four domains (board games, biomedicine, computer science, agriculture) shows that the proposed method significantly outperforms existed one and is domain-independent: the average precision is higher by 5-17% than that of the best method for a particular data set.

Keywords: automatic term acquisition, automatic term recognition, Wikipedia, positive-unlabeled learning.

References

[1]. N.A. Astrakhantsev, D.Yu. Turdakov. Automatic construction and enrichment of informal ontologies: A survey, 2013, published in Programmirovanie, 2013, Vol. 39, No. 1.

[2]. Y. Lingpeng, J. Donghong, Z. Guodong, N. Yu. Improving retrieval effectiveness by using key terms in top retrieved documents. Advances in Information Retrieval, Springer, 2005, P. 169–184.

[3]. R. Yangarber, R. Grishman, P. Tapanainen, and S. Huttunen. Automatic acquisition of domain knowledge for information extraction. Proceedings of the 18th conference on Computational linguistics-Volume 2, 2000, P. 940–946.

[4]. D. D. Golomazov. Metody i sredstva upravleniya nauchnoj informatsiej s ispol'zovaniem ontologij [Methods and tools for management of scientific information by using ontologies]. Ph.D. Thesis. Lomonosov MSU, 2012, 154 p.

[5]. The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. Nat Genet, 25(1), 2007.
[6]. G. A. Miller. WordNet: A Lexical Database for English. Communications of the ACM Vol. 38, No. 11. P. 39-41, 1995.
[7]. А.А. Алексеев, Б.В. Добров, Н.В. Лукашевич. Лингвистическая онтология тезаурус РуТез. Труды конференции Open Semantic Technologies for Intelligent Systems - OSTIS, P. 153–158, 2013.
[8]. D. Milne, O. Medelyan, I. H. Witten. Mining domain-specific thesauri from wikipedia: A case study. Proceedings of the 2006 IEEE/WIC/ACM international conference on web intelligence, 2006, P. 442–448.
[9]. C.D. Manning and H. Schutze. Foundations of statistical natural language processing. MIT press, Cambridge, MA, USA. 1999. 680 p.
[10]. Y. Park, R.J. Byrd, and B.K. Boguraev. Automatic glossary extraction: beyond terminology identification. In Proceedings of the 19th international conference on Computational linguistics-Volume 1, P. 1–7. Association for Computational Linguistics, 2002.
[11]. K.T. Frantzi and S. Ananiadou. Extracting nested collocations. Proceedings of the 16th conference on Computational linguistics-Volume 1, P. 41–46. Association for Computational Linguistics, 1996.
[12]. R. Navigli and P. Velardi. Semantic interpretation of terminological strings. In Proc. 6th Intl Conf. Terminology and Knowledge Eng, 2002, P. 95-100.
[13]. S. Li, J. Li, T. Song, W. Li, B. Chang. A novel topic model for automatic term extraction. In Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval. ACM, 2013, P. 885-888.
[14]. K. Frantzi, S. Ananiadou, H. Mima. Automatic recognition of multi-word terms: the c-value/nc-value method. International Journal on Digital Libraries, vol. 3, no. 2, P. 115–130, 2000.
[15]. G. Bordea, P. Buitelaar, T. Polajnar. Domain-independent term extraction through domain modeling. 10th International Conference on Terminology and Artificial Intelligence (TIA 2013), Paris, France, 2013.
[16]. K. Ahmad, L. Gillam, L. Tostevin, et al. University of surrey participation in trec8: Weirdness indexing for logical document extrapolation and retrieval (wilder). In The Eighth Text REtrieval Conference (TREC-8), 1999.
[17]. F. Sclano and P. Velardi. Termextractor: a web application to learn the shared terminology of emergent web communities. Enterprise Interoperability II, 2007, P. 287-290.
[18]. A. Penas, F. Verdejo, J. Gonzalo, et al. Corpus-based terminology extraction applied to information access. In Proceedings of Corpus Linguistics, volume 2001. Citeseer, 2001.
[19]. Z. Zhang, C. Brewster, F. Ciravegna. A comparative evaluation of term recognition algorithms. In Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC08), Marrakech, Morocco, 2008.
[20]. A. Patry and P. Langlais. Corpus-based terminology extraction. In Terminology and Content Development–Proceedings of 7th International Conference On Terminology and Knowledge Engineering, Litera, Copenhagen, 2005.
[21]. J. Foo, Term extraction using machine learning. Linkoping University, LINKOPING, 2009.
[22]. A. Judea, H. Schütze, S. Brügmann. Unsupervised Training Set Generation for Automatic Acquisition of Technical Terminology in Patents. The 25th International

Conference on Computational Linguistics (COLING 2014), Dublin, Ireland, 2014, P. 290-300.
[23]. D. Fedorenko, N. Astrakhantsev, D. Turdakov. Automatic recognition of domain-specific terms: an experimental evaluation. Proceedings of SYRCoDIS 2013, 2013, P. 15-23.
[24]. D. Turdakov, N. Astrakhantsev, YA. Nedumov, A. Sysoev, I. Andrianov, V. Majorov, D. Fedorenko, A. Korshunov, S. Kuznetsov. Texterra: infrastruktura dlya analiza tekstov [Texterra: A Framework for Text Analysis]. Trudy ISP RAN [Proceedings of ISP RAS], 26(1), 2014. P. 421-438. DOI: 10.15514/ISPRAS-2014-26(1)-18.
[25]. D. Turdakov, S. Kuznetsov. Automatic word sense disambiguation based on document networks. Programming and Computer Software, Volume 36, Number 1, 11-18, 2010.
[26]. S. R. El-Beltagy, A. Rafea. KP-Miner: A keyphrase extraction system for English and Arabic documents. Information Systems, 34(1), P. 132-144, 2009.
[27]. M. Montes, H. J. Escalante. Novel representations and methods in text classification. 7th Russian Summer School in Information Retrieval. Kazan, Russia, 2013.
[28]. M. Montes-y-Gómez, P. Rosso. Using PU-Learning to Detect Deceptive Opinion Spam. WASSA 2013, p. 38, 2013.
[29]. B. Liu, W. S. Lee, P. S. Yu, X. Li. Partially supervised classification of text documents. ICML, 2002, vol. 2, P. 387–394.
[30]. S. Sellamanickam, P. Garg, S. K. Selvaraj. A pairwise ranking based approach to learning with positive and unlabeled examples. Proceedings of the 20th ACM international conference on Information and knowledge management, 2011, P. 663–672.
[31]. J.-D. Kim, T. Ohta, Y. Tateisi, J. Tsujii. GENIA corpus--a semantically annotated corpus for bio-textmining. Bioinformatics, vol. 19, no. Suppl 1, P. 180–182, 2003.
[32]. M. Krapivin, A. Autaeu, M. Marchese. Large dataset for keyphrases extraction. 2009.
[33]. O. Medelyan, I. Witten. Domain-independent automatic keyphrase indexing with small training sets. Journal of the American Society for Information Science and Technology, 59.7, 2008, P. 1026-1040.
[34]. S. Faralli, R. Navigli. Growing Multi-Domain Glossaries from a Few Seeds using Probabilistic Topic Models. EMNLP, 2013, P. 170–181.
[35]. N. Astrakhantsev, D. Fedorenko, D. Turdakov. Automatic Enrichment of Informal Ontology by Analyzing a Domain-Specific Text Collection. Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference “Dialogue”, Issue 13, 2014, P. 29-42.
[36]. M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten. The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1, 2009.

Улучшение качества разбиения графа с помощью многоуровневой оптимизации[1]

Р.К. Пастухов <pastukhov@ispras.ru>
А.В. Коршунов <korshunov@ispras.ru>
Д.Ю. Турдаков <turdakov@ispras.ru>
С.Д. Кузнецов <kuzloc@ispras.ru>
ИСП РАН, 109004, Россия, г. Москва, ул. А. Солженицына, дом 25

Аннотация. Разбиение графа необходимо для решения задач, связанных с обработкой графов, данные которых распределены по нескольким дискам или вычислительным узлам. Эта задача хорошо изучена, но большинство ее решений не подходит для обработки графов с миллиардами вершин на вычислительных кластерах, т.к. эти решения предназначены для вычислительных машин с общей памятью либо для суперкомпьютеров с возможностью посылать сообщения с минимальными задержками. Один из подходов, позволяющий решать задачу разбиения графа на кластерах, – это метод Balanced Label Propagation, основанный на алгоритме распространения меток. В данной работе предлагается метод, позволяющий использовать многоуровневую оптимизацию для улучшения качества разбиений, получаемых с помощью алгоритма Balanced Label Propagation.

Ключевые слова: разбиение графов, распространение меток, многоуровневая оптимизация, социальные сети, распределенные вычисления.

1. Введение

Задача разбиения графа возникает в таких областях, как параллельные и распределенные вычисления, научные вычисления и проектирование электронных схем. Задача заключается в том, что требуется разбить множество вершин графа на k непересекающихся частей примерно одинакового размера так, чтобы количество ребер (или их суммарный вес, если граф взвешенный) с концами в разных частях было минимальным. Эта задача является NP-трудной даже при $k = 2$ [1], поэтому используются приближенные решения.

Задачу разбиения графа можно формализовать следующим образом: для данного графа $G = (V, E)$ нужно таким образом разбить множество вершин на

[1] Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта по гранту №13-07-12134

k частей $V_1, V_2, \dots, V_k$ таких, что $V_i \cap V_j = \emptyset$ при $i \neq j$, $\cup_i V_i = V$, $L_i \leq |V_i| \leq U_i$, чтобы минимизировать мощность разреза $W_{cut} = |\{(a,b) \in E: a \in V_i, b \in V_j, i \neq j\}|$. Параметры L_i и U_i задают ограничения на размеры частей. Часто ограничения для всех частей одинаковы, например, $L_i = \lfloor |V|/k \rfloor, U_i = \lceil |V|/k \rceil$. Если вершинам графа назначены веса, то вместо $|V_i|$ следует использовать $W(V_i) = \sum_{v \in V_i} w(v)$, где $w(v)$ – вес вершины v. Аналогично, если граф взвешен, то $W_{cut} = \sum_{e \in E_{cut}} w(e)$, где $E_{cut} = \{(a,b) \in E: a \in V_i, b \in V_j, i \neq j\}$, а $w(e)$ – вес ребра e. В случае с заданными весами вершин жесткие ограничения, задаваемые параметрами L_iи U_i, достаточно трудно удовлетворить (возникает задача, похожая на задачу о рюкзаке). Поэтому часто эти условия делают менее жесткими, добавляя коэффициент мягкости $f > 0$: $L_i = \lfloor (1-f)W(V)/k \rfloor, U_i = \lceil (1+f)W(V)/k \rceil$.

Такое разбиение также называют разрезом по ребрам. При распределенной обработке графов также используются разрезы по вершинам [12], когда на части разбивается не множество вершин, а множество ребер графа. Формально эта задача эквивалентна разрезу по ребрам реберного графа, но обычно для ее решения используют специализированные алгоритмы. Разрезы по вершинам в данной работе не рассматриваются.

Один из подходов [2], показывающий наилучшие результаты, основан на алгоритме Кернигана-Лина [4] и многоуровневой оптимизации. Этот алгоритм достаточно эффективен и позволяет находить хорошие разбиения, но не подходит для распределенных систем.

Для разбиения больших социальных графов был предложен алгоритм Balanced Label Propagation [3]. Он основан на алгоритме распространения меток из [5] и может использоваться в распределенных системах, поддерживающих парадигму Map-Reduce [6].

В следующем разделе кратко описываются алгоритм Balanced Label Propagation и суть метода многоуровневой оптимизации. Затем предлагается подход, в котором комбинируется применение Balanced Label Propagation и многоуровневой оптимизации. В четвертом разделе обсуждаются результаты экспериментов применения подхода авторов, реализованного в среда Sparc [7], к разбиению двух реальных графов социальных сетей. В заключении отмечаются преимущества предложенного подхода и формулируется ряд нерешенных исследовательских проблем.

2. Используемые алгоритмы и методы

2.1 Balanced Label Propagation

Алгоритм поиска сообществ [5], основанный на распространении меток, показал эффективность такого подхода в рамках распределенных систем. В нем каждой вершине присваивается уникальная метка, затем несколько раз производится следующая операция: метка каждой вершины заменяется на

наиболее часто встречающуюся метку среди ее соседей (если таких меток несколько, одна из них выбирается случайно). Таким образом, каждая вершина стремится попасть в группу вершин, в которой количество внутренних ребер, инцидентных вершине, будет минимально. При этом вычисления можно разбить на два этапа: генерация сообщений (меток) и получение сообщений. При этом до окончания этапа генерации доставка сообщений не требуется. Это позволяет эффективно реализовать данный алгоритм на основе парадигмы Map-Reduce.

Получаемое разбиение на сообщества не является решением задачи разбиения графа, потому что количество сообществ нельзя задать заранее, и их размеры не сбалансированы. Для обеспечения разбиения графа у алгоритма Balanced Label Propagation имеются два отличия от описанного алгоритма, основанного на распространении меток:

- при инициализации метки назначаются таким образом, чтобы ограничения по балансировке соблюдались (например, случайно);
- для изменения в соответствии с алгоритмом распространения меток выбирается только часть вершин графа таким образом, чтобы ограничения по балансировке при этом не были нарушены.

Выбор вершин, метки которых изменяются, происходит с помощью решения задачи линейного программирования, с целью минимизации количества ребер между частями при соблюдении ограничений по балансировке.

Алгоритм достаточно быстро сходится; для большинства графов достаточно около 10 итераций.

2.2 Многоуровневая оптимизация

Многие алгоритмы разбиения графа, в частности, алгоритм Кернигана-Лина и Balanced Label Propagation, основаны на улучшении существующих разбиений и по сути являются алгоритмами локальной оптимизации: они неплохо оптимизируют существующее разбиение, но не всегда могут его существенно изменить. Многоуровневая оптимизация позволяет локальным оптимизациям работать с более крупными частями графа путем стягивания ребер в вершины. Ее можно разделить на три этапа: огрубление графа, разбиение огрубленного графа и уточнение.

Во время огрубления из исходного графа G формируется последовательность графов $G = G_0, G_1 \dots G_n$ такая, что граф G_{i+1} получается за счет стягивания некоторых ребер в графе G_i, образующих паросочетание. Обычно множество стягиваемых ребер выбирается так, что они образуют максимальное паросочетание. При этом, даже в исходном графе G не было весов, в генерируемых графах G_i как вершины, так и ребра должны быть взвешены. Это связанно с тем, что при объединении нескольких ребер или вершин вес образуемого элемента графа должен равняться сумме весов исходных элементов. Если после стягивания ребер вершина v в графе G_i переходит в

вершину u в графе G_{i+1}, будем называть u образом v и обозначать это $u = I(v)$. У каждой вершины в графах $G_i, 0 \le i < n$ имеется ровно один образ.

После этого задача разбиения решается на графе G_n. Увеличивая n, количество вершин в графе G_n можно сделать сколь угодно малым. Поэтому при получении разбиения этого графа можно (но не обязательно) использовать более сложные алгоритмы, чем при улучшении разбиений во время перехода от G_{i+1} к G_i на следующем шаге.

На этапе уточнения разбиения огрубленных графов преобразуются в разбиения этих графов до огрубления. Чтобы получить разбиение графа G_i, имея разбиение графа G_{i+1}, достаточно разбить вершины в соответствии с разбиением их образов в G_{i+1}. То есть если набор V_i является разбиением графа G_{i+1}, то соответствующее ему разбиение G_i будет состоять из частей $V_i' = \{v: I(v) \in V_i\}$. Легко видеть, что у разбиения графа G_i веса частей и мощность разреза будут такими же, как и у разбиения G_{i+1}. После получения разбиения графа G_i оно улучшается с помощью используемого алгоритма локального улучшения разбиений. Эти действия повторяются, пока не получится разбиение графа $G_0 = G$.

3 Предлагаемый подход

Основная проблема, возникающая при распределенной многоуровневой оптимизации, состоит в сложности выбора пар вершин для стягивания на этапе огрубления графа. В [2] показано, что стягивание ребер с наибольшим весом позволяет улучшить качество разбиения по сравнению со стягиванием случайно выбранных ребер. В распределенной системе подобный подход сложно реализовать, поскольку данные о ребрах графа распределены по разным вычислительным узлам: один узел не может самостоятельно выбрать ребро для стягивания, так как другой вычислительный узел может выбрать смежное ему ребро, создавая конфликт (например, если разные узлы решат стягивать смежные ребра, стягиваемые ребра не будут образовывать паросочетание). С другой стороны, согласование выбора ребра со всеми вычислительными узлами, содержащими смежные ему ребра, приведет к существенной деградации производительности.

Достаточно простое решение – разбить вершины на группы (используя некоторое начальное разбиение графа, например, случайное) и выбирать для стягивания только такие ребра, чтобы обе инцидентные им вершины принадлежали одной группе – показывает хорошие результаты. Это позволяет производить стягивание ребер на каждом вычислительном узле независимо при условии, что памяти каждого вычислительного узла достаточно для обработки любой группы.

При этом может оказаться, что некоторые вершины не инцидентны никаким ребрам, целиком содержащимся в их группе. Это приводит к необходимости стягивать не только ребра, но и произвольные пары вершин без ребра между ними – доля таких вершин может быть достаточно большой, а для

эффективной работы желательно, чтобы количество вершин в графах G_i сокращалось экспоненциально.

Самый простой способ реализации такого подхода в распределенной системы, поддерживающей Map-Reduce, состоит в группировке вершин по идентификатору их группы в операции map с последующим стягиванием ребер внутри групп вершин в операции reduce. Оптимальное количество групп равно общему числу процессоров на всех машинах в кластере: меньшие значения не позволят использовать все ресурсы кластера, а при больших значениях будет расти количество вершин, изолированных в группах вершин, которые назначаются одному вычислительному узлу.

Если в начальном разбиении столько же частей, сколько в результирующем разбиении, то многоуровневую оптимизацию можно производить итеративно: результат разбиения, полученный с помощью многоуровневой оптимизации, можно использовать как начальное разбиение для следующей итерации. Теоретически это может улучшить качество разбиения, но подобные эксперименты в рамках данной работы не проводились.

Таким образом, предложенный метод состоит из следующих шагов.

1. Разбиение вершин исходного графа на группы. Для этого выбирается желаемое количество групп, и граф разбивается на подграфы с помощью небольшого количества итераций Balanced Label Propagation.
2. Огрубление графа. На этом шаге генерируется последовательность огрубленных графов (а также дополнительные данные, которые будут использоваться при уточнении). Сначала в каждой группе вершин выбираются ребра (или пары вершин) для стягивания. Результат выбора ребер для стягивания можно представить в виде отображения (набора пар) идентификаторов вершин в идентификаторы их образов в огрубленном графе. Затем создается огрубленный граф путем отождествления вершин, инцидентных стягиваемым ребрам. Детали этого этапа могут зависеть от возможностей используемой платформы распределенных вычислений. В общем случае достаточно заменить идентификаторы вершин на идентификаторы их образов и затем объединить одинаковые вершины и ребра, соединяющие одинаковые вершины. Имеется несколько способов реализовать это действие в с использованием Map-Reduce, но подходы, позволяющие избежать соединения данных графа с полученным отображением вершин на их образы, позволяют достигать лучшей производительности. Например, поддержка кеширования в системе Spark [7] позволяет сохранить группы вершин в памяти кластера, выбрать ребра для стягивания в пределах каждой группы, разослать эту информацию всем узлам кластера (здесь предполагается, что у каждого вычислительного узла в кластере достаточно памяти чтобы хранить информацию об образах всех вершин) и затем генерировать огрубленный граф, используя

данные из кеша.

3. Разбиение наименьшего из огрубленных графов с помощью алгоритма Balanced Label Propagation.
4. Уточнение графов. При этом после получения разбиения очередного уточненного графа (для этого достаточно обратить отображения вершин на их образы; это можно легко сделать как с помощью Map-Reduce, так и в памяти, если ее достаточно, чтобы хранить это отображение) оно улучшается с помощью нескольких итераций Balanced Label Propagation.
5. После получения в результате уточнения исходного графа с улучшенным разбиением вывести это разбиение как результат работы алгоритма.

4 Вычислительные эксперименты

Для тестирования алгоритм Balanced Label Propagation и его вариант, в котором используется многоуровневая оптимизация были реализованы с использованием системы Spark. Сравнение производилось на следующих наборах данных:

- граф связей в сети LiveJournal [8, 9]; содержащий 4.8 млн. вершин и 68.9 млн. ориентированных ребер; для тестирования граф был преобразован в неориентированный, количество ребер после этого составило 42.8 млн.;
- часть графа связей сети Facebook, полученная с помощью алгоритма MHRW [10, 11]; содержащая 0.9 млн. вершин и 1.79 млн. ребер.

В качестве меры качества разбиения использовалась доля ребер, обе вершины которых оказались в одном разделе. Если суммарный вес ребер графа (поскольку все графы в тестировании были без весов, он равен количеству ребер) W, а суммарный вес ребер в разрезе – W_{cut}, то доля внутренних ребер выражается формулой $(W - W_{cut})/W$. Видно, что чем меньше суммарный вес разреза, тем больше доля внутренних ребер. Но за счет нормализации этот параметр удобнее при сравнении результатов на различных графах.

При тестировании алгоритма, использующего многоуровневую оптимизацию, огрубление графа производилось до тех пор, пока количество вершин не станет меньше или равно 10000.

Перед началом огрубления вершины разбивались на разделы с помощью применения трех итераций Balanced Label Propagation к случайному разбиению. Для графа LiveJournal наилучшие результаты достигаются при 0..1 итерациях. Для части графа Facebook качество возрастает при увеличении количества итераций в интервале 0..10. Влияние начального разбиение на качество разбиения – это одно из возможных направлений дальнейших исследований.

После каждого перехода от меньшего графа к большему, разбиение оптимизировалось с помощью двух итераций Balanced Label Propagation. Увеличение количества итераций на этом этапе улучшает качество разбиения, но также увеличивает время работы.

На рис. 1 видно, что качество разбиения Balanced Label Propagation сходится достаточно быстро и после 10 итераций уже практически не меняется. В сравнительных тестах проводилось 20 итераций.

На рис. 2 показано, что многоуровневая оптимизация позволяет значительно улучшить качество разбиения при всех рассмотренных значениях количества разделов от 8 до 56.

Во всех тестах многоуровневая версия работала приблизительно в два раза дольше. Теоретически, многоуровневая версия может быть быстрее оригинальной, если оптимизировать процесс огрубления графов, но пока не ясно, насколько сильно можно улучшить производительность этого этапа. Также актуальной темой для дальнейших исследований является определение оптимального количества итераций Balanced Label Propagation на этапе уточнения графов и влияние их количества на производительность.

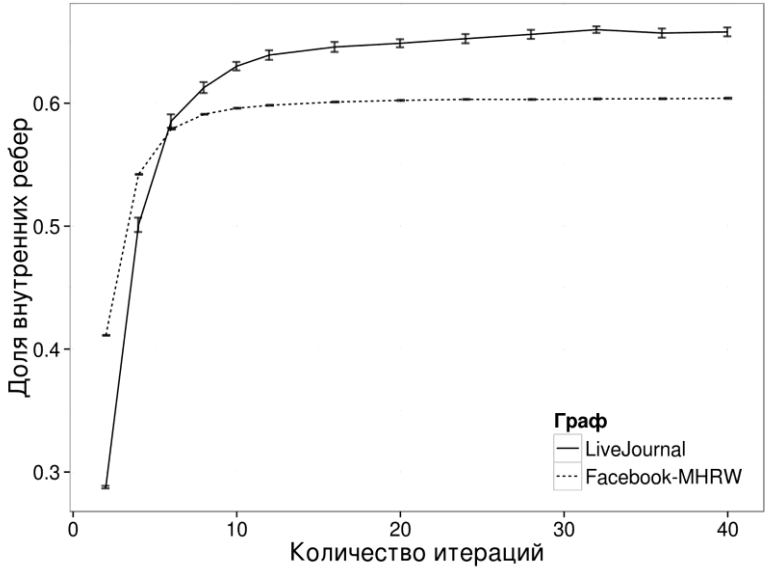

Рис. 1. Сходимость Balanced Label Propagation.

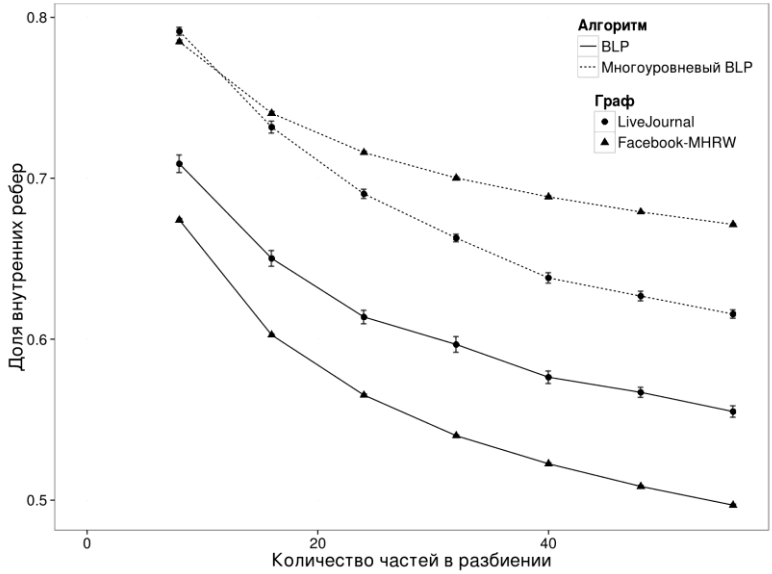

Рис. 2. Сравнение качества разбиений

5 Заключение

Было показано, что многоуровневая оптимизация позволяет заметно улучшить качество разбиений, получаемых с помощью алгоритма Balanced Label Propagation. Одна из выявленных проблем подхода связана с тем, что в исходном разбиении, используемом для генерации последовательности графов путем стягивания ребер, в каждом разделе может быть довольно много изолированных вершин, что может оказывать влияние на качество результирующего разбиения. Кроме того ребра для стягивания брались произвольно, хотя известно, что более сложные алгоритмы выбора ребер для стягивания могут улучшить качество разбиений [2]. Таким образом, актуальны следующие направления для дальнейших исследований:

- исследование возможного выигрыша от использования более сложных алгоритмов выбора ребер для стягивания в рамках одной группы вершин, например, алгоритма поиска паросочетаний большого веса из [2];
- исследование возможности распределенного стягивания ребер графа без ограничений на расположение вершин ребра относительно вычислительных узлов, например, за счет разрешения ситуаций, когда ребра не образуют паросочетания;
- исследование влияния начального разбиения на результаты, в том числе того, насколько лучших результатов можно достичь с помощью

итеративной многоуровневой оптимизации;

- улучшение производительности алгоритма.

Список литературы

[1]. M. R. Garey, D. S. Johnson, and L. Stockmeyer. “Some simplified NP-complete graph problems”. In: Theoretical computer science 1.3 (1976), pp. 237–267.

[2]. G. Karypis and V. Kumar. “A fast and high quality multilevel scheme for partitioning irregular graphs”. In: SIAM Journal on scientific Computing 20.1 (1998), pp. 359–392.

[3]. J. Ugander and L. Backstrom. “Balanced Label Propagation for Partitioning Massive Graphs”. In: Proceedings of the Sixth ACM International Conference on Web Search and Data Mining. WSDM ’13. Rome, Italy: ACM, 2013, pp. 507–516. isbn: 978-1-4503-1869-3. doi: 10.1145/2433396.2433461.

[4]. B. W. Kernighan and S. Lin. “An Efficient Heuristic Procedure for Partitioning Graphs”. In: Bell System Technical Journal 49.2 (1970), pp. 291–307. doi: 10.1002/j.1538-7305.1970.tb01770.x.

[5]. U. Raghavan, R. Albert, and S. Kumara. “Near linear time algorithm to detect community structures in large-scale networks”. In: Physical Review E 76.3 (2007). doi: 10.1103/physreve.76.036106.

[6]. J. Dean and S. Ghemawat. “MapReduce: Simplified Data Processing on Large Clusters”. In: OSDI 2004. 2004, pp. 137–150.

[7]. M. Zaharia et al. “Resilient Distributed Datasets: A Fault-tolerant Abstraction for In-memory Cluster Computing”. In: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation. NSDI’12. San Jose, CA: USENIX Association, 2012.

[8]. Stanford Network Analysis Project, LiveJournal social network. 2006. url: http://snap.stanford.edu/data/soc-LiveJournal1.html.

[9]. L. Backstrom et al. “Group Formation in Large Social Networks: Membership, Growth, and Evolution”. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD ’06. Philadelphia, PA, USA: ACM, 2006, pp. 44–54. isbn: 1-59593-339-5. doi: 10.1145/1150402.1150412.

[10]. Sampling Online Social Networks, Facebook Social Graph. 2009. url: http://odysseas.calit2.uci.edu/doku.php/public:online_social_networks#facebook_social_graph_-_mhrw_uni.

[11]. M. Gjoka et al. “Walking in Facebook: A Case Study of Unbiased Sampling of OSNs”. In: Proceedings of the 29th Conference on Information Communications. INFOCOM’10. San Diego, California, USA: IEEE Press, 2010, pp. 2498–2506. isbn: 978-1-4244-5836-3. doi: 10.1109/infcom.2010.5462078.

[12]. J. E. Gonzalez et al. “PowerGraph: Distributed Graph-parallel Computation on Natural Graphs”. In: Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation. OSDI’12. Hollywood, CA, USA: USENIX Association, 2012, pp. 17–30. isbn: 978-1-931971-96-6.

Improving quality of graph partitioning using multilevel optimization

R. Pastukhov <pastukhov@ispras.ru>
A. Korshunov <korshunov@ispras.ru>
D. Turdakov <turdakov@ispras.ru>
S. Kuznetsov <kuzloc@ispras.ru>
ISP RAS, 25 Alexander Solzhenitsyn Str., Moscow, 109004, Russian Federation

Abstract. Graph partitioning is required for solving tasks on graphs that need to be split across disks or computers. This problem is well studied, but most results are not suitable for processing graphs with billons of nodes on commodity clusters, since they require shared memory or low-latency messaging. One approach suitable for cluster computing is Balanced Label Propagation, based on distributed label propagation algorithm for community detection. In this work we show how multi-level optimization can be used to improve partitioning quality of Balanced Label Propagation. One of major difficulties with distributed multi-level optimization is finding a matching in the graph. The matching is needed to choose pairs of vertices for collapsing in order to produce a smaller graph. As this work shows, simply splitting graph into several parts and finding matching in these parts independently is enough to improve the quality of partitioning generated by Balanced Label Propagation. Proposed algorithm can be implemented within any framework that supports MapReduce. In our experiments, when graphs were partitioned into 32 parts, ratio of edges that don't cross partitions increased from 54-60% to 66-70%. One of significant problems of our implementation is performance – work time of multi-level algorithm was approximately twice that of the original algorithm. It seems likely that implementation can be improved so that multi-level algorithm would achieve better computational performance as well as partitioning quality.

Keywords: graph partitioning, label propagation, multi-level optimization, social networks, cluster computing.

References

[1]. M. R. Garey, D. S. Johnson, and L. Stockmeyer. "Some simplified NP-complete graph problems". In: Theoretical computer science 1.3 (1976), pp. 237–267.

[2]. G. Karypis and V. Kumar. "A fast and high quality multilevel scheme for partitioning irregular graphs". In: SIAM Journal on scientific Computing 20.1 (1998), pp. 359–392.

[3]. J. Ugander and L. Backstrom. "Balanced Label Propagation for Partitioning Massive Graphs". In: Proceedings of the Sixth ACM International Conference on Web Search and Data Mining. WSDM '13. Rome, Italy: ACM, 2013, pp. 507–516. isbn: 978-1-4503-1869-3. doi: 10.1145/2433396.2433461.

[4]. B. W. Kernighan and S. Lin. "An Efficient Heuristic Procedure for Partitioning Graphs". In: Bell System Technical Journal 49.2 (1970), pp. 291–307. doi: 10.1002/j.1538-7305.1970.tb01770.x.

[5]. U. Raghavan, R. Albert, and S. Kumara. “Near linear time algorithm to detect community structures in large-scale networks”. In: Physical Review E 76.3 (2007). doi: 10.1103/physreve.76.036106.

[6]. J. Dean and S. Ghemawat. “MapReduce: Simplified Data Processing on Large Clusters”. In: OSDI 2004. 2004, pp. 137–150.

[7]. M. Zaharia et al. “Resilient Distributed Datasets: A Fault-tolerant Abstraction for In-memory Cluster Computing”. In: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation. NSDI’12. San Jose, CA: USENIX Association, 2012.

[8]. Stanford Network Analysis Project, LiveJournal social network. 2006. url: http://snap.stanford.edu/data/soc-LiveJournal1.html.

[9]. L. Backstrom et al. “Group Formation in Large Social Networks: Membership, Growth, and Evolution”. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD ’06. Philadelphia, PA, USA: ACM, 2006, pp. 44–54. isbn: 1-59593-339-5. doi: 10.1145/1150402.1150412.

[10]. Sampling Online Social Networks, Facebook Social Graph. 2009. url: http://odysseas.calit2.uci.edu/doku.php/public:online_social_networks#facebook_social_graph_-_mhrw_uni.

[11]. M. Gjoka et al. “Walking in Facebook: A Case Study of Unbiased Sampling of OSNs”. In: Proceedings of the 29th Conference on Information Communications. INFOCOM’10. San Diego, California, USA: IEEE Press, 2010, pp. 2498–2506. isbn: 978-1-4244-5836-3. doi: 10.1109/infcom.2010.5462078.

[12]. J. E. Gonzalez et al. “PowerGraph: Distributed Graph-parallel Computation on Natural Graphs”. In: Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation. OSDI’12. Hollywood, CA, USA: USENIX Association, 2012, pp. 17–30. isbn: 978-1-931971-96-6.

Автоматическое создание виртуальных кластеров Apache Spark в облачной среде Openstack[1]

О. Д. Борисенко <al@somestuff.ru>
Д. Ю. Турдаков <turdakov@ispras.ru>
С. Д. Кузнецов <kuzloc@ispras.ru>
ИСП РАН, 109004, Россия, г. Москва, ул. А. Солженицына, дом 25

Аннотация. В работе описывается процесс создания системы автоматического создания виртуальных кластеров Apache Spark в среде Openstack. Также в работе приводится краткий обзор различий между предоставляемыми метаданными в средах Openstack и Amazon EC2.

Ключевые слова: Apache Spark, Openstack, Amazon EC2, Map-Reduce, HDFS, виртуальные кластеры, облачные вычисления, Big Data.

1. Введение

На текущий момент технологии вычислений в программной модели Map-Reduce становятся все более востребованными. Существует множество реализаций данного подхода [1], [2], [3], [4], и проект Apache Spark [5] является одной из таких реализаций. Проект Apache Spark изначально разрабатывался институтом Беркли и после открытия исходных кодов в 2010 году активно развивается множеством независимых разработчиков. Apache Spark является одной из самых быстрых реализаций Map-Reduce [6], [7].

В то же время все существующие проекты, реализующие Map-Reduce, обладают сложной архитектурой, и процесс настройки кластерного окружения для их использования является сложным и запутанным. Умение программировать в парадигме Map-Reduce совершенно не обязывает разработчика и исследователя уметь настраивать сложные кластерные окружения.

[1] Работа выполнена при поддержке гранта РФФИ №14-07-00602 А «Исследование и разработка методов автоматизации масштабирования и разворачивания виртуальных кластеров для обработки сверхбольших объёмов данных в облачной среде Openstack»

На момент начала работы над данным проектом существовало лишь несколько проектов для автоматизации развертывания окружения для распределенных вычислений в парадигме Map-Reduce, и ни один из них не предполагал использование открытых облачных сред в качестве платформы для развертывания вычислительного кластера [8], [9]; более того, все эти проекты подразумевают использование лишь классического стека технологий Apache Hadoop или Cloudera Hadoop. Единственный проект, ориентированный на открытые облачные среды (Openstack Sahara [10]) был в самом начале цикла разработки, и он также подразумевал использование Apache Hadoop в качестве реализации Map-Reduce. Использование Apache Spark появилось в планах лишь недавно, и проект очень далек от завершения.

Таким образом, задача автоматического построения виртуальных кластеров с готовым окружением для выполнения задач в Apache Spark кажется очень актуальной. Кроме того, такой подход позволяет очень точно делить аппаратные ресурсы между разработчиками и выделять им ровно столько ресурсов, сколько им требуется, без привязки к нижележащему аппаратному обеспечению. Целью данной работы является предоставление решения для автоматизации создания виртуальных кластеров Apache Spark в облачной среде Openstack.

2. Компоненты Apache Spark

В этом разделе приводится краткий обзор основных компонентов Apache Spark.

2.1 Уровень хранения

В качестве системы хранения данных Apache Spark использует распределенную файловую систему HDFS [11]. Разработчики фреймворка предполагают единовременное использование двух файловых систем HDFS: одну предполагается использовать для долговременного хранения результатов работы программ, вторую же предполагается использовать для оперативного хранения промежуточных данных вычислений и для распределения заданий между вычислительными узлами кластера. Apache Spark поддерживает все известные реализации и версии файловой системы HDFS.

2.2 Вычислительные узлы

В Apache Spark используется разделение ролей вычислительных узлов. Один из узлов назначается управляющим и называется «мастером». Остальные узлы предназначаются для выполнения вычислений и называются «рабочими» (workers). Программа запускается на «мастер»-узле, и он автоматически формирует задания для «рабочих» узлов. После выполнения заданий «рабочие» узлы уведомляют «мастер» узел о результате исполнения заданий.

2.3 Слежение за жизнедеятельностью кластера

От версии к версии в проекте Apache Spark менялась модель слежения за работоспособностью вычислительных узлов. В первых версиях проекта использовалось разделение ответственности: слежение за работоспособностью узлов производилось при помощи выделенных узлов, на которых специальным образом настраивался ZooKeeper [12]. В более поздних версиях от этого подхода отказались, и был реализован собственный протокол опроса узлов о доступных им ресурсах и слежения за работоспособностью. В данной работе используются только встроенные в Apache Spark механизмы слежения за работоспособностью кластера.

Отдельно стоит отметить, что для слежения за использованием ресурсов узлами вычислительной сети, разработчки проекта Apache Spark предлагают использовать специализированный инструмент для слежения за нагрузкой под названием Ganglia [13]. В рамках данной работы также используется этот инструмент.

3. Существующие способы создания кластеров Apache Spark в облачных средах

На момент начала работы над данным проектом существовало ровно два способа создания кластеров Apache Spark.

Первый способ предполагает ручную конфигурацию всех компонентов системы. То есть для создания кластерного окружения необходимо задать конфигурацию для следующих подсистем:

- Apache Hadoop (v1/2.x) в двух экземплярах
- Окружений JVM и Scala
- Ganglia
- Apache Spark

Настройка JVM и Scala подразумевает установку всех необходимых пакетов и настройку переменных окружения для дальнейшего их использования в Apache Spark и Apache Hadoop.

Apache Hadoop и Apache Spark конфигурируются при помощи множества файлов XML, которые должны содержать роли каждого из узлов, IP-адреса и порты для соединений между узлами, уровни репликации HDFS, конфигурацию нижележащего уровня под файловой системой HDFS и так далее.

Ganglia также подразумевает настройку с разделением ролей: на каждом узле должен существовать конфигурационный файл с указанием тех ресурсов, за которыми должен следить демон, и должен существовать конфигурационный файл на «мастер»-узле для слежения за ресурсами кластера в целом.

Второй способ предназначается для настройки кластеров в облачной среде Amazon EC2 [14][15]. Настройка окружения подразумевает использование заранее сконфигурированного образа виртуальной машины с установленными компонентами без конфигурации и специального набора скриптов для настройки компонентов. Образ виртуальной машины использует два типа блочных устройств для хранения данных в HDFS: устройство для перманентного хранения данных и устройство для оперативных данных HDFS и раздела для своппинга. Второе блочное устройство называется термином «ephemeral» и уничтожается при остановке кластера.

Скрипты настройки выполняют следующую последовательность действий:

- при помощи программных интерфейсов Amazon EC2 (далее – API Amazon EC2) инициируется процесс запуска необходимого количества виртуальных машин из заранее настроенного образа на базе ОС Red Hat Enterprise Linux 5.3;
- с помощью API Amazon EC2 настраиваются правила ограничений сетевого доступа к сетевым портам каждой из виртуальных машин;
- скрипт впадает в режим ожидания ровно на 5 минут;
- после истечения пяти минут скрипт «узнает» выданные IP-адреса для каждой из виртуальных машин при помощи API Amazon EC2; в случае, если виртуальные машины не успели запуститься, скрипт выдает ошибку;
- с использованием полученных данных заполняются конфигурационные файлы для всех компонентов сиситем. «Мастер» узел назначается NameNode для обеих файловых систем HDFS;
- «мастер»-узлу передаются все конфигурационные файлы при помощи ssh-подключения;
- «мастер»-узел располагает полученные файлы в зараннее определенных папках и по ssh раздает дочерним узлам соответствующие им конфигурационные файлы;
- «мастер»-узел инициирует запуск служб HDFS и Apache Spark на всех узлах.

После выполнения перечисленных действий управление возвращается на терминал пользователя, и пользователю сообщаются необходимые данные для подключения к свежесозданному кластеру.

Ключевые особенности второго способа:

- используется API закрытой облачной системы Amazon EC2;
- используется заранее настроенный образ операционной системы RHEL 5.3, который нельзя использовать как из-за лицензионных ограничений данного дистрибутива, так и из-за невозможности извлечь образы виртуальных машин из среды Amazon EC2.

Таким образом, оба способа неприменимы для использования в облачной среде Openstack из-за сложности настройки отдельных компонентов для первого способа и из-за явной привязки к Amazon EC2 для второго способа.

4. Построение решения

Существует несколько подходов для достижения поставленной задачи:

1. переиспользование существующих скриптов настройки кластера в Amazon EC2 при помощи адаптеров API в среде Openstack (подсистема Heat);
2. использование систем оркестрации;
3. адаптация (портирование) решения для среды Openstack.

Рассмотрим эти подходы подробнее.

4.1 Использование подсистемы Openstack Heat

На момент начала работы над проектом в проекте Openstack существовал специальный побочный проект под названием Heat [16]. Целью проекта являлось обеспечение совместимости с Openstack скриптов, предназначенных для использования в Apache EC2. Предполагалось, что при использовании этой прослойки Openstack сможет «притворяться» Amazon EC2 для конечного пользователя и обеспечивать точно такое поведение. Тем не менее, в тот момент проект находился в зачаточном состоянии и предоставлял менее половины от вызовов к API Amazon EC2.

Необходимо отдельно отметить, что Amazon EC2 и Openstack реализуют совершенно разные модели архитектуры облачных сервисов. В частности, обе облачные платформы предлагают пользователю хранить определенный набор метаданных, связанных с виртуальными машинами, но различается как и сам набор метаданных, так и уровни доступа к ним. Openstack предоставляет модель, в которой каждая виртуальная машина может иметь доступ только к собственным метаданным, в то время как Amazon EC2 позволяет использовать метаданные и «снаружи» виртуальной машины.

Кроме того, облачные сервисы по-разному реализуют виртуальную сетевую инфраструктуру. В рамках Amazon EC2 каждая виртуальная машина получает два IP-адреса: один является «внешним» и используется для доступа из любой точки сети Интернет, второй адрес является «внутренним» и доступен только в пределах виртуальной сети между виртуальными машинами конкретного пользователя. Эти настройки являются фиксированными, и повлиять на них нельзя, так что и вызовов в API Amazon EC2 для настройки сети не существует.

В то же время, Openstack позволяет очень гибко настраивать сетевое окружение. Каждая сеть в Openstack обладает собственным именем и идентификатором, который задается на этапе настройки среды Openstack. При создании виртуальной машины необходимо явно указывать, какие именно

сети будет использовать виртуальная машина; если этого не сделать, то виртуальная машина получает сетевые интерфейсы по умолчанию, которые задаются администратором Openstack, и настройки этих сетей нужно знать заранее, поскольку Heat не предоставляет данных о конфигурации сети.

Таким образом, адаптер Heat в рамках этого проекта использовать было невозможно с самого начала по двум причинам.

- Поскольку Heat в точности повторяет вызовы Amazon EC2, с его помощью невозможно использовать сети, отличные от заданных администратором Openstack в качестве сетей по умолчанию. Это накладывает ограничения на пользователя, так как ему необходимо иметь непосредственный контакт с администратором Openstack для назначения нужных настроек по умолчанию, а этот вариант не всегда доступен.
- Недостаточный уровень совместимости вызов Heat с API Amazon EC2. В частности, в данном случае оказалось, что система именования виртуальных машин в Openstack API и Heat API отличается: Heat API не способен установить hostname виртуальной машины, и виртуальные машины, созданные при помощи API Heat получают свой внутренний IP-адрес в качестве hostname. В то же время, Apache Spark для настройки узлов требует именования узлов в соответствии с FQDN, и в случае, если узел имеет hostname, не соответствующий FQDN, настройки считаются некорректными. Более того, ответ на вызов API Heat, соответствующий запросу «какой у меня hostname», всегда возвращается пустым.

Таким образом, мы не можем переиспользовать существующий набор утилит для настройки виртуального кластера без изменений.

4.2 Использование систем оркестрации

На момент начала работы над проектом, существовало несколько распространенных систем оркестрации: Puppet [17], Chef [18], Salt [19] и Ansible [20].

Puppet, Chef и Salt используют разделение ролей на управляющие узлы и дочерние узлы и предполагают установку специальных агентов на дочерние узлы. Это накладывает существенные ограничения на использование этих систем, поскольку в таком случае необходимо специальным образом настраивать управляющий узел. Кроме того, необходимо использовать специально настроенный образ операционной системы на дочерних узлах, в котором будет содержаться настроенный агент системы оркестрации. Это кажется избыточным шагом, требует особых знаний от пользователя и усложняет систему в целом, поэтому от Puppet, Chef и Salt было решено отказаться. Кроме того, указанные системы являются коммерческими и ограничивают возможности использования бесплатной версии.

Ansible выгодно отличается от этих систем.

- Ansible не накладывает никаких функциональных ограничений на бесплатную версию.
- В качестве зависимостей Ansible использует только ssh и python 2.6 или выше.
- Ansible очень прост в освоении.
- Сценарии настройки можно переиспользовать в любой среде: настройка дочерних узлов и настройка окружения (будь то облачный сервис или физические сервера) не взаимосвязаны. То есть возможно создать один большой сценарий настройки Apache Spark и отдельное множество маленьких сценариев для настройки нижележащей инфраструктуры, и сценарий Apache Spark будет работать везде одинаково.

К сожалению, на момент начала проекта в Ansible еще не было поддержки Openstack (из сред виртуализации была поддержка только Amazon EC2 и Vagrant). Но на момент публикации поддержка всех открытых облачных сред уже есть, так что перенос решения на Ansible с целью унификации и отстранения от вопросов инфраструктуры, кажется перспективным направлением для дальнейшей работы.

4.3 Адаптация решения для среды Openstack

В качестве последнего подхода для достижения поставленной задачи, предлагается полная адаптация существующего решения для Amazon EC2 к среде Openstack. Именно этот способ и был реализован в данной работе.

Процесс адаптации состоит из двух этапов:

1. Создание базового образа виртуальной машины на основе CentOS 6.4.
2. Портирование скриптов для создания виртуального кластера Apache Spark с использованием нативного API Openstack.

Создание базового образа не представляет особой сложности, тем не менее, занимает много времени. Зависимости и необходимые программы для базового образа уже были перечислены в разделе 3. Кроме того, необходимо было подготовить образ к настройке собственных параметров в среде Openstack, для этого существуют уже настроенные образы дистрибутива CentOS (хотя автор делал эти шаги самостоятельно, чтобы исключить дополнительные источники возможных ошибок). Заранее подготовленные образы для различных версий Apache Spark можно найти по адресу http://spark.at.ispras.ru/ , список образов пополняется по мере выхода новых версий дистрибутива. В подготовленные образы также включен настроенный дистрибутив хранилища ключей/значений Redis [21].

Портирование скриптов было произведено при помощи библиотек, предоставляемых проектом Openstack. Исходные коды публично доступны по адресам https://github.com/ispras/incubator-spark и https://github.com/ispras/spark-

openstack под лицензией Apache 2.0. Инструкции по использованию находятся там же.

Разработанные скрипты также обладают дополнительной функциональностью по сравнению с изначальным вариантом для Amazon EC2:

- Явное указание времени ожидания старта виртуальных машин теперь не используется: вместо этого реализован механизм опроса состояния виртуальных машин, поэтому ожидание длится ровно столько, сколько требуется машинам для запуска.
- Добавлен слой гибкой конфигурации виртуальных сетей, используемых виртуальными машинами.

5. Результаты

В рамках данной работы было проведено исследование методов развертывания и масштабирования программных средств для обработки сверхбольших объемов данных в облачной среде Openstack, и было разработано открытое и свободное программное решение для автоматизации создания виртуальных кластеров Apache Spark и Apache Hadoop/HDFS в облачной среде Openstack. Побочно был реализован механизм настройки Apache Hive и Apache Shark, но эти инструменты протестированы не были, и были исключены из репозитория из-за недостаточного тестирования.

Данное решение уже активно используется в текущих исследованиях, в скором времени будет опубликована научная работа, посвященная генерации случайных социальных графов [21], использующая в своей основе Apache Spark и результаты данной работы. Использовались кластеры размером в 49 виртуальный узел по 8ГБ оперативной памяти и по 2 вычислительных ядра на каждом из них.

Список литературы

[1]. Страница проекта Apache Hadoop — http://hadoop.apache.org/
[2]. Страница проекта Cloudera CDH Apache Hadoop — http://www.cloudera.com/content/cloudera/en/products-and-services/cdh.html
[3]. Страница проекта Infinispan — http://infinispan.org/
[4]. Страница проекта Basho Riak — http://basho.com/riak/
[5]. Страница проекта Apache Spark — http://spark.apache.org/
[6]. M. Chowdhury, M. Zaharia, I. Stoica. Performance and Scalability of Broadcast in Spark. 2010.
[7]. Gu, Lei, and Huan Li. Memory or Time: Performance Evaluation for Iterative Operation on Hadoop and Spark. High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC), 2013 IEEE 10th International Conference on. IEEE, 2013.
[8]. Страница проекта VMWare Serengeti — http://www.vmware.com/hadoop/serengeti
[9]. Страница проекта Cloudera Manager — http://www.cloudera.com/content/cloudera/en/products-and-services/cloudera-enterprise/cloudera-manager.html

[10]. Страница проекта Openstack Sahara, план очередности разработки — https://wiki.openstack.org/wiki/Sahara/Roadmap
[11]. Foley, Matt. High Availability HDFS. 28th IEEE Conference on Massive Data Storage, MSST. Vol. 12. 2012.
[12]. Hunt, Patrick, et al. ZooKeeper: Wait-free Coordination for Internet-scale Systems. USENIX Annual Technical Conference. Vol. 8. 2010.
[13]. Massie, Matthew, B. Chun, and D. Culler. The ganglia distributed monitoring system: design, implementation, and experience. Parallel Computing 30.7 (2004): 817-840.
[14]. Страница сервиса Amazon Elastic Compute Cloud (EC2) — http://aws.amazon.com/ec2/
[15]. Creeger, Mache. Cloud Computing: An Overview. ACM Queue 7.5 2009.
[16]. Страница проекта Openstack Heat — https://wiki.openstack.org/wiki/Heat
[17]. Yokoyama, Shigetoshi, and Nobukazu Yoshioka. Cluster as a Service for self-deployable cloud applications. Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on. IEEE, 2012.
[18]. Страница проекта Chef http://www.getchef.com/
[19]. Страница проекта Salt http://www.saltstack.com/
[20]. Страница проекта Ansible http://www.ansible.com/home
[21]. *Ожидает публикации*. К. Чихрадзе, А. Коршунов, Н. Бузун, Н. Кузюрин. Использование модели социальной сети с сообществами пользователей для распределённой генерации случайных социальных графов. 10-я Международная конференция «Интеллектуализация обработки информации» 2014.

Automating cluster creation and management for Apache Spark in Openstack cloud

O. Borisenko <al@somestuff.ru>
D. Turdakov <turdakov@ispras.ru>
S. Kuznetsov <kuzloc@ispras.ru>
ISP RAS, 25 Alexander Solzhenitsyn Str., Moscow, 109004, Russian Federation

Abstract. This article is dedicated to automation of cluster creation and management for Apache Spark MapReduce implementation in Openstack environments. As a result of this project open-source (Apache 2.0 license) implementation of toolchain for virtual cluster on-demand creation in Openstack environments was presented. The article contains an overview of existing solutions for clustering automation in cloud environments by the start of 2014 year. The article provides a shallow overview of issues and problems in Openstack Heat project that provides a compatibility layer for Amazon EC2 API. The final implementation provided in the article is almost strainforward port of existing toolchain for cluster creation automation for Apache Spark in Amazon EC2 environment with some improvements. Also prepared base system virtual machine image for Openstack is provided. Plans for further work are connected with Ansible project. Using Ansible for observed problem will make possible to implement versatile environment-agnostic solution that is able to work using any cloud computing services provider, set of Docker containers or bare-metal clusters without any dependencies for prepared operating system image. Current article doesn't use Ansible due to the lack of key features at the moment of the project start. The solution provided in this article has been already tested in production environment for graph theory research arcticle.

Keywords: Apache Spark, Openstack, Amazon EC2, Map-Reduce, HDFS, virtual cluster, cloud computing, Big Data.

References

[1]. Apache Hadoop project web page — http://hadoop.apache.org/
[2]. Cloudera CDH Apache Hadoop project web page — http://www.cloudera.com/content/cloudera/en/products-and-services/cdh.html
[3]. Infinispan project web page — http://infinispan.org/
[4]. Basho Riak project web page — http://basho.com/riak/
[5]. Apache Spark project web page — http://spark.apache.org/
[6]. M. Chowdhury, M. Zaharia, I. Stoica. Performance and Scalability of Broadcast in Spark. 2010.
[7]. Gu, Lei, and Huan Li. Memory or Time: Performance Evaluation for Iterative Operation on Hadoop and Spark. High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC), 2013 IEEE 10th International Conference on. IEEE, 2013.

[8]. VMWare Serengeti project web page — http://www.vmware.com/hadoop/serengeti
[9]. Cloudera Manager project web page — http://www.cloudera.com/content/cloudera/en/products-and-services/cloudera-enterprise/cloudera-manager.html
[10]. Openstack Sahara project web page, roadmap — https://wiki.openstack.org/wiki/Sahara/Roadmap
[11]. Foley, Matt. High Availability HDFS. 28th IEEE Conference on Massive Data Storage, MSST. Vol. 12. 2012.
[12]. Hunt, Patrick, et al. ZooKeeper: Wait-free Coordination for Internet-scale Systems. USENIX Annual Technical Conference. Vol. 8. 2010.
[13]. Massie, Matthew, B. Chun, and D. Culler. The ganglia distributed monitoring system: design, implementation, and experience. Parallel Computing 30.7 (2004): 817-840.
[14]. Amazon Elastic Compute Cloud (EC2) service webpage — http://aws.amazon.com/ec2/
[15]. Creeger, Mache. Cloud Computing: An Overview. ACM Queue 7.5 2009.
[16]. Openstack Heat project web page — https://wiki.openstack.org/wiki/Heat
[17]. Yokoyama, Shigetoshi, and Nobukazu Yoshioka. Cluster as a Service for self-deployable cloud applications. Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on. IEEE, 2012.
[18]. Chef project web page — http://www.getchef.com/
[19]. Salt project web page — http://www.saltstack.com/
[20]. Ansible project web page — http://www.ansible.com/home
[21]. *In print*. K. Chikhradze, A. Korshunov, N. Buzun, N. Kuzyurin. Ispol'zovanie modeli sotsial'noj seti s soobshhestvami pol'zovatelej dlya raspredelyonnoj generatsii sluchajnykh sotsial'nykh grafov [On a model of social network with user communities for distributed generation of random social graphs]. 10-ya Mezhdunarodnaya konferentsiya «Intellektualizatsiya obrabotki informatsii» [10th International conference “Intelligent Information Processing”] 2014.

Разработка масштабируемой программной инфраструктуры для хранения и обработки данных в задачах вычислительной биологии

О. Д. Борисенко <al@somestuff.ru>
А. В. Лагута <laguta@ispras.ru>
Д. Ю. Турдаков <turdakov@ispras.ru>
С. Д. Кузнецов <kuzloc@ispras.ru>
ИСП РАН, 109004, Россия, г. Москва,
ул. А. Солженицына, дом 25

Аннотация. В работе кратко описывается масштабируемая программная инфраструктура для хранения и обработки данных в задачах вычислительной биологии. Обсуждаются использованные технологии, собственное программное решение для предсказания сайтов связывания транскрипционных факторов в геномах, реализация предоставления решения как части веб-лаборатории с REST API и веб-интерфейсом для исследователей.

Ключевые слова: ISPRAS API, OpenZFS, Swift, виртуальная лаборатория, облачные вычисления, Big Data.

1. Введение

Задачи молекулярной биологии и генетики являются очень востребованными и требуют все больших мощностей для обработки данных. Данные, необходимые для решения этих задач, собираются все быстрее с каждым годом, при этом мощностей индивидуальных компьютеров исследователей уже недостаточно для проведения расчетов. К примеру, референсный геном мыши (mm9) занимает на диске больше трех с половиной гигабайт, при этом даже открытые базы данных известных геномов насчитывают сотни живых организмов, и их число растет очень быстро [1], [2].

Также стоит учитывать процесс получения референсных геномов (это также является одной из возможных задач). Дело в том, что устройства для «оцифровки» геномов (секвенаторы) не получают геномы в собранном виде.

Все существующие на текущий момент секвенаторы выдают так называемые «сырые» данные, которые впоследствии необходимо обрабатывать для получения цифрового представления данных генома. Сырые данные представляют собой множество строк фиксированной длины (длина зависит от модели секвенатора), которые в дальнейшем обрабатываются специальными программами-сборщиками для получения генома. Объем сырых данных может в десятки раз превосходить объем собранного генома.

При этом публично доступных сервисов, предоставляющих открытое API для обработки биологических данных, фактически нет. На момент написания статьи из известных ресурсов подобного толка представлен лишь проект Google Genomics [3] в формате закрытого тестирования. На текущий момент сервис Google Genomics предоставляет REST API для получения данных геномов, обработка этих данных пока не представлена. Также стоит отметить, что это API уже год предоставляется лишь в формате закрытого бета-теста, так что для его использования необходимо предварительно подать заявку с описанием целей проекта в Google и дождаться подтверждения от представителей компании.

Таким образом, исследователи в областях молекулярной биологии и генетики вынуждены самостоятельно настраивать все необходимые им утилиты и рабочие инструменты для каждой задачи, причем большая часть этих инструментов работает только в ОС Linux и предоставляет только консольный способ управления.

В данной работе описывается расширяемая инфраструктура для предоставления публичного API и веб-сервиса для исследователей в областях молекулярной биологии и генетики. В рамках этой инфраструктуры предоставляется доступ к инструменту предсказания сайтов связывания транскрипционных факторов с использованием коллекции HOCOMOCO мотивов ДНК [4].

2. Архитектура решения

В этом разделе описывается устройство программной инфраструктуры для обработки и хранения данных для биологических задач в том виде, в котором решение работает сейчас.

2.1 Общая схема

На рис. 1 иллюстрируется архитектура описываемой системы:

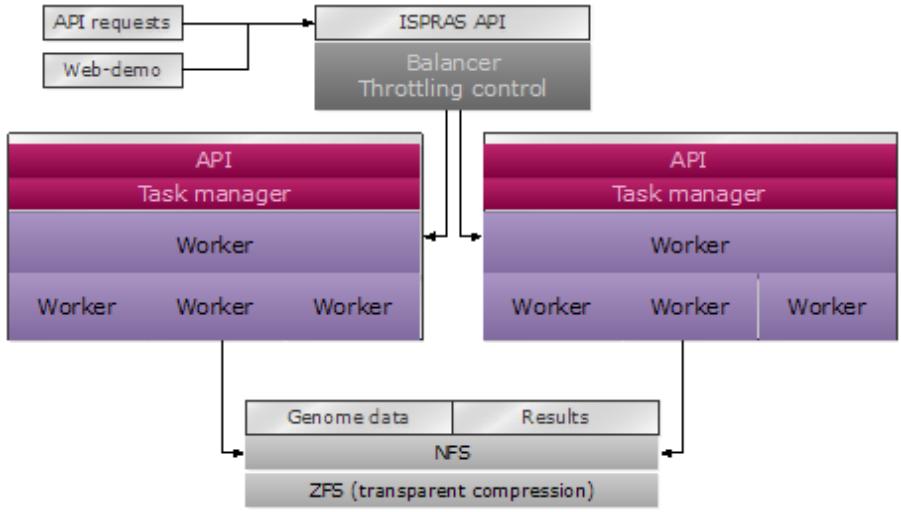

Рис. 1. Архитектура виртуальной лаборатории.

В следующих подразделах описываются составляющие этой схемы. Тем не менее, следует отметить, что такая архитектура системы используется сейчас в связи с малой нагрузкой на реализованный сервис.

Решение разработано таким образом, что в случае увеличения нагрузки мы сможем прозрачно и безболезненно перейти на более гибкую схему с точки зрения хранения и управления ресурсами, так как эта поддержка уже реализована в представленном проекте. На рис. 2 изображена архитектура системы, которую мы будем использовать в случае увеличения нагрузки на сервис.

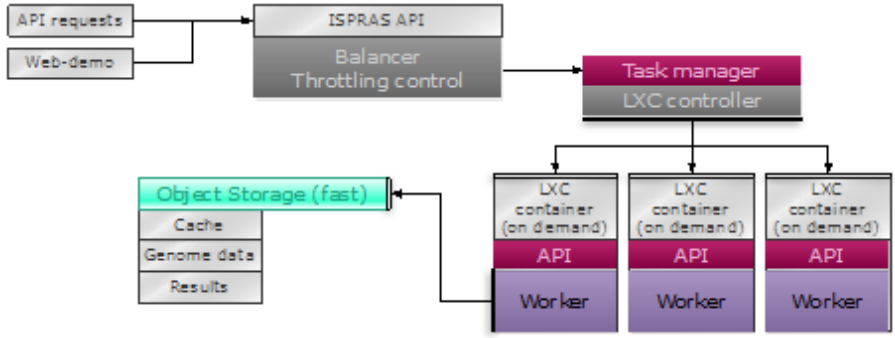

Рис. 2. Альтернативная архитектура виртуальной лаборатории.

Способ миграции на указанную схему работы будет также указан в ходе описания компонентов системы.

2.2 Уровень хранения

Хранение геномов, необходимых для обработки данных и результатов осуществляется на выделенном виртуальном сервере. В качестве файловой системы используется ZFS on Linux (подпроект OpenZFS [5]), в качестве механизма предоставления доступа к данным используется NFS [6]. ZFS on Linux предоставляет те же возможности, что и оригинальная файловая система из ОС Solaris, с том числе возможности прозрачного сжатия, и это очень актуально для задач генетики. Каждый считающий задания узел монтирует систему хранения в свою локальную систему и использует это пространство в качестве своей локальной папки. У этого подхода есть очевидные недостатки: узел хранения становится единой точкой возможного отказа, и потенциально могут возникнуть проблемы с доступом к хранилища из-за особенностей устройства NFS.

Поскольку эти недостатки было несложно предвидеть ещё на этапе проектирования, система была реализована с поддержкой хранения данных в объектном хранилище Openstack Swift [7]. Оно лишено описанных недостатков, предоставляет надежное хранение данных, распределение нагрузки на узлы хранения и позволяет получать доступ к ним при помощи REST API.

2.3 Уровень вычислений

В основе виртуальной лаборатории лежит идея разделения программной части, предоставляющей REST API, и программной части вычислительных сервисов. Вычислительные сервисы могут быть написаны на любом языке программирования и должны отвечать нескольким требованиям для того, чтобы можно было использовать их в рамках виртуальной лаборатории:

1. Программная часть вычислительных сервисов должна гарантировать возможность явного ограничения используемой памяти. Ограничение на память передается из уровня API.
2. Программная часть вычислительных сервисов должна предоставлять данные о статусе выполнения задания в ходе выполнения.
3. Программная часть вычислительных сервисов должна гарантировать возможность ограничения числа подпроцессов, которые могут быть запущены в ходе выполнения.

В качестве примера: введенный в эксплуатацию сервис реализован на языке C++, позволяет в явном виде задавать ограничение по использованию оперативной памяти и позволяет задавать степень многопоточности выполнения.

2.4 Уровень API

В ходе проекта была реализована служебная программа, которая отвечает за обработку и передачу запросов на вычисления нижележащим сервисам. Также

этот программный слой занимается поддержанием пула рабочих процессов. В нынешней схеме размещения ресурсов это означает то, что при инициализации программа оценивает количество ресурсов на локальном узле и выделяет пространство памяти и доступное количество процессоров под выполнение задач. При поступлении запроса на вычисления запрос преобразуется в тот вид, который может принимать на вход нижележащий вычислительный сервис, заданию присваивается идентификатор, и он выдается в качестве ответа на запрос. В дальнейшем при помощи этого идентификатора можно получить статус выполнения задания. Такая схема была реализована прежде всего по причине «тяжеловесности» вычислительных заданий. Для того, чтобы клиенту не нужно было поддерживать соединение, задания ставятся в очередь выполнения, и выполняются в соответствии с количеством свободных рабочих процессов в пуле. В момент, когда статус задания приобретает значение «выполнено», в ответе также указывается URL, по которому располагается файл с результатами вычсислений. Благодаря такой схеме работы возможно использовать как локальные, так и облачные системы хранения. Кроме того, это позволяет реализовать выполнение задач по цепочке (т.е следущий сервис принимает в качестве входных данных результат работы другого сервиса) в случае, если появится потребность в решении таких задач.

Кроме того, поддерживается режим работы, в котором служебная программа следит не за ресурсами локальной машины, а за предоставленными ограничениями на создание экземпляров контейнеров или виртуальных машин в облачной платформе Openstack. При такой схеме работы вычислительные сервисы запускаются не в локальных процессах, а в контейнерах или виртуальных машинах Openstack, и служебная программа следит за созданием контейнеров или виртуальных машин при помощи API Openstack Nova [8].

2.5 Уровни распределения ресурсов

В данной системе используется два уровня распределения ресурсов. Первый уровень реализован на уровне служебной программы, принимающий запросы (см. 2.4). Вторым уровнем распределения ресурсов является технология ISPRAS API, разработанная в Институте системного программирования РАН. ISPRAS API является round-robin балансировщиком нагрузки с единой точкой входа. Со стороны пользователя это выглядит так, что все запросы поступают на один URL (http://api.ispras.ru) со специальными параметрами, и ISPRAS API перенаправляет его на нужный вычислительный сервер. Таким образом обеспечивается прозрачное для пользователя масштабирование сервисов. Кроме того, при помощи ISPRAS API обеспечивается защита от злоупотреблений: каждый пользователь системы (авторизованный или анонимный) обладает заданными для него ограничениями на количество запросов в единицу времени. В случае превышения пользователем этих

ограничений, система сообщает ему об этом, и задание не передается на вычислительные узлы.

2.6 Описание веб-сервиса

Веб-сервис работает по адресу https://api.ispras.ru/demo/gen в демонстрационном режиме и будет дорабатываться в дальнейшем. Сервис реализован с использованием предоставляемого виртуальной лабораторией API и по сути является обыкновенным клиентским приложением к этому API.

Веб-сервис демонстрирует возможности только одного биологического сервиса и позволяет строить предсказания о сайтах связывания транскрипционных факторов на интересующих исследователя наборах участков генома с заданной точностью предсказания. Подробнее задача описывается в следующем разделе.

3. Предсказание сайтов связывания транскрипционных факторов

Поскольку тематика задачи пересекается с этой работой лишь частично, в данной работе приводится постановка задачи и описание возможностей виртуальной лаборатории для решения задач этого класса. Более подробно эта проблема описывается в работах [9], [10], [11]. Описанию решения этой задачи планируется посвятить отдельную статью.

Введем необходимые определения. Геномом называют цифровое представление ДНК организма. Существует множество форматов для хранения геномов для использования в компьютерных программах. В рамках этой задачи традиционно используют формат FASTA [12].

Фактором транскрипции (или фактором инициации транскрипции) называют белок, контроллирующий процесс синтеза матричной РНК при помощи связывания со специфичными для этого фактора участками ДНК. Фактор транскрипции связывается с одним и тем же участком ДНК лишь с какой-то вероятностью.

Участок ДНК (и генома), с которым связывается фактор транскрипции, называется сайтом связывания.

Существует специальный формат моделирования вероятности связывания с заданным участком генома для любого фактора транскрипции (PWM [13]).

Данные для описания модели получаются исследователями экспериментальным путем. Существует набор алгоритмов для вычисления вероятности связывания фактора транскрипции для любого заданного участка генома с использованием PWM модели фактора транскрипции.

Задача предсказания сайтов связывания транскрипционных факторов сводится к нахождению в геноме участков, для которых вероятность связывания с заданным транскрипционным фактором выше заданного исследователем порога.

В качестве входных данных для решения этой задачи пользователь лаборатории может указать следующие параметры:

1. интересующий пользователя геном;
2. транскрипционный фактор, для которого нужно строить предсказания; в системе используется коллекция PWM-моделей транскрипционных факторов HOCOMOCO [4], разработанная исследователями из Института Общей Генетики РАН;
3. степень достоверности предсказания, выраженная через p-value;
4. список отрезков генома, для которых необходимо строить предсказания; исследователей редко интересует весь геном целиком, обычно такой анализ проводится в областях, соответствующих известным генам.

Система обладает самодокументирующим API: при помощи HTTP GET-запросов можно получить оглавление доступных ресурсов (в данном случае списки доступных геномов и транскрипционных факторов с их описанием).

4. Результаты

Построена инфраструктура для решения задач молекулярной биологии и генетики с возможностью эффективного роста и работы в облачных средах.

Разработана система быстрого доступа к результатам обработки данных секвенирования нового поколения с возможностью их обработки на стороне виртуальной лаборатории.

Реализован первый инструмент для решения задач, связанных с предсказанием сайтов связывания транскрипционных факторов с геномом на основе коллекции мотивов ДНК HOCOMOCO, работающий полностью на стороне виртуальной лаборатории по запросу пользователя.

Реализованы REST API для использования лаборатории в собственных программах исследователей и веб-сервис, основанный на реализованном API.

Список литературы

[1]. Страница проекта «1000 Genomes project» — http://www.1000genomes.org/about
[2]. Страница проекта по предоставлению данных геномов и разметки известных генов от университета Санта-Круз — http://genome.ucsc.edu/
[3]. Страница проекта Google Genomics - https://cloud.google.com/genomics/
[4]. Ivan V. Kulakovskiy, Yulia A. Medvedeva, Ulf Schaefer, Artem S. Kasianov, Ilya E. Vorontsov, Vladimir B. Bajic and Vsevolod J. Makeev, HOCOMOCO: a comprehensive collection of human transcription factor binding sites models, Nucleic Acids Research 2012. doi: 10.1093/nar/gks1089
[5]. M. Ahrens, OpenZFS: a Community of Open Source ZFS Developers."AsiaBSDCon 2014, pp. 27-32.
[6]. Страница проекта NFS - http://nfs.sourceforge.net/
[7]. Duan, Zhi Ying, and Yi Zhen Cao, The Implementation of Cloud Storage System Based on OpenStack Swift, Applied Mechanics and Materials. Vol. 644. 2014, pp. 2981-2984.

[8]. Beernaert, L., Matos, M., Vilaça, R., & Oliveira, R., Automatic elasticity in Openstack, In Proceedings of the Workshop on Secure and Dependable Middleware for Cloud Monitoring and Management, ACM, p. 2.
[9]. Zhou, Qing, and Jun S. Liu., Modeling within-motif dependence for transcription factor binding site predictions., Bioinformatics 20.6, 2004, pp. 909-916.
[10]. Bintu, Lacramioara, et al., Transcriptional regulation by the numbers: applications. Current opinion in genetics & development 15.2, 2005: pp. 125-135.
[11]. Описание механики процесса связывания факторов транскрипции с геномом с точки зрения статистики - http://www.bio-physics.at/wiki/index.php?title=Statistical_Mechanics_of_Binding
[12]. Описание формата FASTA - http://genetics.bwh.harvard.edu/pph/FASTA.html
[13]. Zhang, Xiujun, Position Weight Matrices., Encyclopedia of Systems Biology. Springer New York, 2013, 1721-1722.

Developing scalable software infrastructure for data storage and processing for computational biology problems

O. Borisenko <al@somestuff.ru>
A. Laguta <laguta@ispras.ru>
D. Turdakov <turdakov@ispras.ru>
S. Kuznetsov <kuzloc@ispras.ru>
ISP RAS, 25 Alexander Solzhenitsyn Str., Moscow, 109004, Russian Federation

Abstract. This article is an overview of scalable infrastructure for storage and processing of genome data in genetics problems. The overview covers used technologies descriptions, the organization of unified access to genome processing API of different underlying services. The article also covers methods for scalable and cloud computing technologies support. The first service in virtual genome processing laboratory is provided and presented. The service solves transcription factors bindning sites prediction problem. The main principles of service construction are provided. Basic requirements for underlying comptutaion software in virtual laboratory environments are provided. Overview describes the implemented web-service (**https://api.ispras.ru/demo/gen**) for transcription factors binding site prediction. Provided solution is based on ISPRAS API project as an API gateway and load-balancer; the middle-ware task-manager software for pool of workers support and for communications with Openstack infrastructure; OpenZFS as an intermediate storage with transparent compression support. The described solution is easy to extend with new services fitting the basic requirements.

Keywords: ISPRAS API, OpenZFS, Swift, virtual laboratory, cloud computing, Big Data.

References

[1]. 1000 Genomes project web page — http://www.1000genomes.org/about
[2]. University of California, Santa Cruz genome project — http://genome.ucsc.edu/
[3]. Google Genomics web page — https://cloud.google.com/genomics/
[4]. Ivan V. Kulakovskiy, Yulia A. Medvedeva, Ulf Schaefer, Artem S. Kasianov, Ilya E. Vorontsov, Vladimir B. Bajic and Vsevolod J. Makeev, HOCOMOCO: a comprehensive collection of human transcription factor binding sites models, Nucleic Acids Research 2012. doi: 10.1093/nar/gks1089
[5]. M. Ahrens, OpenZFS: a Community of Open Source ZFS Developers."AsiaBSDCon 2014, pp. 27-32.
[6]. NFS project web page — http://nfs.sourceforge.net/
[7]. Duan, Zhi Ying, and Yi Zhen Cao, The Implementation of Cloud Storage System Based on OpenStack Swift, Applied Mechanics and Materials. Vol. 644. 2014, pp. 2981-2984.

[8]. Beernaert, L., Matos, M., Vilaça, R., & Oliveira, R., Automatic elasticity in Openstack, In Proceedings of the Workshop on Secure and Dependable Middleware for Cloud Monitoring and Management, ACM, p. 2.
[9]. Zhou, Qing, and Jun S. Liu., Modeling within-motif dependence for transcription factor binding site predictions., Bioinformatics 20.6, 2004, pp. 909-916.
[10]. Bintu, Lacramioara, et al., Transcriptional regulation by the numbers: applications. Current opinion in genetics & development 15.2, 2005: pp. 125-135.
[11]. Statistical mechanics of transctiption factors binding sites — http://www.biophysics.at/wiki/index.php?title=Statistical_Mechanics_of_Binding
[12]. FASTA format description — http://genetics.bwh.harvard.edu/pph/FASTA.html
[13]. Zhang, Xiujun, Position Weight Matrices., Encyclopedia of Systems Biology. Springer New York, 2013, 1721-1722.

Automatic Recognition of Domain-Specific Terms: an Experimental Evaluation

D. Fedorenko <fedorenko@ispras.ru>
N. Astrakhantsev <astrakhantsev@ispras.ru>
D. Turdakov <turdakov@ispras.ru>
ISP RAS, 25 Alexander Solzhenitsyn Str., Moscow, 109004, Russian Federation

Abstract. This paper presents an experimental evaluation of the state-of-the-art approaches for automatic term recognition based on multiple features: machine learning method and voting algorithm. We show that in most cases machine learning approach obtains the best results and needs little data for training; we also find the best subsets of all popular features.

Keywords: automatic term recognition, term extraction, machine learning, experimental evaluation, feature selection.

1. Introduction

Automatic term recognition (ATR) is an actual problem of text processing. The task is to recognize and extract terminological units from different domain-specific text collections. Resulting terms can be useful in more complex tasks such as semantic search, question-answering, ontology construction, word sense induction, etc.

There are a lot of studies of ATR. Most of them split the task into three common steps:

- **Extracting term candidates.** At this step special algorithm extracts words and word sequences admissible to be terms. In most cases researches use predefined or generated part-of-speech patterns to filter out word sequences that do not match such the patterns. The rest of word sequences becomes term candidates.
- **Extracting features of term candidates.** Feature is a measurable characteristic of a candidate that is used to recognize terms. There are a lot of statistical and linguistic features that can be useful for term recognition.
- **Extracting final terms from candidates.** This step varies depending upon the way in which researches use features to recognize terms. In some studies authors filter out non-terms by comparing feature values with thresholds: if feature values lies in specific ranges, then candidate is considered

to be a term. Others try to rank candidates and expect the top-N ones to be terms. At last, few studies apply supervised machine learning methods in order to combine features effectively.

There are several studies comparing different approaches for ATR. In [1] authors compare different single statistical features by their effectiveness for term candidates ranking. In [2] the same comparison is extended by voting algorithm that combines multiple features. Studies [3], [4] compare supervised machine learning method with the approach based on single feature again.

In turn, the present study experimentally evaluates the ranking methods combining multiple features: supervised machine learning approach and voting algorithm. We pay most of the attention to the supervised method in order to explore its applicability to ATR.

The purposes of the study are the following:

- To compare results of machine learning approach and voting algorithm;
- To compare different machine learning algorithms applied to ATR;
- To explore how much training data is needed to rank terms;
- To find the most valuable features for the methods;

This study is organized as follows. At the beginning we describe the approaches more detailed. Section 3 is devoted to the performed experiments: firstly, we describe evaluation methodology, then report the obtained results, and, finally, discuss them. In Section 4 we conclude the study and consider the further research.

2. Related Work

In this section we describe some of the approaches to ATR. Most of them have the same extracting algorithm but consider different feature sets, so the final results depend only on the used features. We also briefly describe features used in the task. For more detailed survey of ATR see [5], [6].

2.1 Extracting Term Candidates Overview

Strictly, all of the word sequences, or *n-grams*, occurring in text collections can be term candidates. But in most cases researchers consider only unigrams and bigrams [1]. Of course, only the little part of such the candidates are terms, because the candidates' list mainly consists of sequences like “a”, “the”, “some of”, “so the”, etc. Hence such the noise should be filtered out.

One of the first methods for such the filtering was described in [7]. The algorithm extracts term candidates by matching the text collection with predefined Part-of-Speech (PoS) patterns, such as:

- Noun
- Adjective Noun
- Adjective Noun Noun

As was reported in [7], such the patterns cut off much of the noise (word sequences that are not terms) but retain real terms, because in most cases terms are noun phrases [8]. Filtering of term candidates that do not satisfy some of the morphological properties of word sequences is known as *linguistic step* of ATR.

In work [3] the authors do not use predefined patterns appealing to the fact that PoS tagger can be not precise enough on some texts; they instead generate patterns for each text collection. In study [9] no linguistic step is used: the algorithm considers all n-grams from text collection.

2.2 Features overview

Having a lot of term candidates, it is necessary to recognize domain specific ones among them. It can be done by using the statistical features computed on the basis of the text collection or some another resource, for example general corpus [7], domain ontology [10] or Web [11]. This part of ATR algorithm is known as *statistical step.*

Term Frequency is a number of occurrences of the word sequence in the text collection. This feature is based on the assumption that if the word sequence is specific for some domain, then it often occurs in such domain texts. In some studies frequency is also used as an initial filter of term candidates [1 2]: if a candidate has a very low frequency, then it is filtered out. It helps to reduce much of the noise and improves precision of the results.

*TF*IDF* has high values for terms that often occur only in few documents: TF is a term frequency and IDF is an inversed number of documents, where the term occurs:

$$TF \cdot IDF(t) = TF(t) \cdot \log \frac{|Docs|}{|Doc: t \in Doc|}$$

To find domain-specific terms that are distributed on the whole text collection, in [7] IDF is considered as an inversed number of documents in *reference corpus*, where the term occurs. Reference corpus is a some general, i.e. not specific, text collection.

The described features shows how the word sequence is related to the text collection, or *termhood* of a candidate. There is another class of features that show inner strength of words cohesion, or *unithood* [5]. One of the first features of this class is T-test.

T-test [7] is a statistical test that was initially designed for bigrams and checks the hypothesis of independence of words constituting a term:

$$T - stat(t) = \frac{\frac{TF(t)}{N} - p}{\sqrt{\frac{p(1-p)}{N}}}$$

where p – hypothesis of independence, N – a number of bigrams in the corpus.

The assumption of this feature is that the text is a Bernoulli process, where meeting of bigram t is a "success", while meeting of other bigrams is a "failure".

Hypothesis of independence is usually expressed as follows: $p=P(w_1w_2)=P(w_1) \bullet P(w_2)$, where $P(w_1)$ - a probability to encounter the first word of the bigram, $P(w_2)$ - a probability to encounter the second one. This expression can be assessed by replacing the probabilities of words to their normalized frequencies within a text:

$$p=TF(w_1)/N \bullet TF(w_2)/N,$$

where N – an overall number of words in the text.

If words are independently distributed in text collection, then they do not form persistent *collocation.* It is assumed that any domain-specific term is a collocation, while not any collocation is a specific term. So considering features like T-test, we can increase the confidence in that candidate is a collocation, but not necessarily specific term.

There are much more features that are used in ATR.

C-Value [13] has higher values for candidates that are not parts of other word sequences:

$$C-Value(t) = \log_2|t| \cdot TF(t) - \frac{1}{|\{seq: t \in seq\}|} \cdot \sum_{t \in seq} TF(seq)$$

Domain Consensus [14] recognizes terms that are uniformly distributed on the whole dataset:

$$DC(t) = -\sum_{d \in Docs} \frac{TF_d(t)}{TF(t)} \cdot \log_2 \frac{TF_d(t)}{TF(t)}$$

Domain Relevance [15] compares frequencies of the term in two datasets – target and general:

$$DR(t) = \frac{TF_{target}(t)}{TF_{target}(t) + TF_{reference}(t)}$$

Lexical Cohesion [16] is the unithood feature that compares frequency of term and frequency of words from which it consists:

$$LC(t) = \frac{|t| \cdot TF(t) \cdot \log_{10} TF(t)}{\sum_{w \in t} TF(w)}$$

Loglikelihood [7] is the analogue of T-test but without assumption about how words in a text are distributed:

$$LogL(t) = \log \frac{b(c_{12}; c_1, p) \cdot b(c_2 - c_{12}; N - c_1, p)}{b(c_{12}; c_1, p_1) \cdot b(c_2 - c_{12}; N - c_1, p_2)}$$

where c_{12} – a frequency of bigram t, c_1 – a frequency of the bigram's first word, c_2 – a frequency of the second one, $p=c_2/N$, $p_1=c_{12}/c_1$, $p_2=(c_2-c_{12})/(N-c_1)$, $b(\bullet;\bullet,\bullet)$ – binomial distribution.

Relevance [17] is the more sophisticated analogue of Domain Relevance:

$$R(t) = 1 - \frac{1}{\log_2(2 + \frac{TF_{target}(t) \cdot DF_{target}(t)}{TF_{reference}(t)})}$$

Weirdness [18] compares frequencies in different collections but also takes into account their sizes:

$$W(t) = \frac{TF_{target}(t) \cdot |Corpus_{reference}|}{TF_{reference} \cdot |Corpus_{target}|}$$

The described feature list includes termhood, unithood and *hybrid* features. The termhood features are Domain Consensus, Domain Relevance, Relevance, and Weirdness. The unithood features are Lexical Cohesion and Loglikelihood. The hybrid feature, or feature that shows both termhood and unithood, is C-Value.

A lot of works still concentrate on feature engineering, trying to find more informative features. Nevertheless, recent trend is to combine all these features effectively.

2.3 Recognizing terms overview

Having feature values, final results can be produced. The studies [13], [7], [18] use ranking algorithm to provide the most probable terms, but this algorithm considers only one feature. The studies [15], [16] describe the simplest way of how multiple features can be considered: all values are simply reduced in a one weighted average value that then is used during ranking.

In work [19] authors introduce special rules based on thresholds for feature values. An example of such a rule is the following:

$$Rule_i(t) = F_i(t) > a \text{and} F_i(t) < b$$

where F_i is a *i-th* feature; *a, b* are thresholds for feature values

Note that the thresholds are selected manually or computed from the marked-up corpora, so this method can not be considered as purely automatic and unsupervised.

Effective way of combining multiple features was introduced in [2]. It combines the features in a voting manner using the following formula:

$$V(t) = \sum_{i}^{n} \frac{1}{rank(F_i(t))}$$

where n is a number of considered features, $rank(F_i(t))$ is a rank of the term t among values of other terms considering feature F_i.

Table 1: Results of cross-validation without frequency filter

Dataset	Algorithm	AvP
GENIA	Random Forest	0.54
GENIA	Logistic Regression	**0.55**
GENIA	Voting	0.53
Biol	Random Forest	0.35
Biol	Logistic Regression	**0.40**
Biol	Voting	0.23

Table 2: Results of cross-validation with frequency filter

Dataset	Algorithm	AvP
GENIA	Random Forest	0.66
GENIA	Logistic Regression	**0.70**
GENIA	Voting	0.65
Biol	Random Forest	0.52
Biol	Logistic Regression	**0.58**
Biol	Voting	0.31

In addition, study [2] shows that the described voting method in general outperforms most of the methods that consider only one feature or reduce them in a weighted average value. Another important advantage of the voting algorithm is that it does not require normalization of feature values.

There are several studies that apply supervised methods for term recognition. In [3] authors apply AdaBoost meta-classifier, while in [9] Ripper system is used. The study [20] describes hybrid approach including both unsupervised and supervised methods.

3. Evaluation

For our experiments we implemented two approaches for ATR. We used voting algorithm as the first one, while in supervised case we trained two classifiers: Random Forest and Logistic Regression from WEKA library[1]. These classifiers were chosen because of their effectiveness and good generalization ability of the resulting model. Furthermore, these classifiers are able to produce *classification confidence* – a numeric score that can be used to rank an example in overall test set. It is an important property of the selected algorithms that allows to compare their results with results produced by other ranking methods.

[1] Official website of the project: http://www.cs.waikato.ac.nz/ml/weka/

3.1 Evaluation methodology

The quality of the algorithms is usually assessed by two common metrics: precision and recall [21]. *Precision* is the fraction of retrieved instances that are relevant:

$$P = \frac{|\text{correct returned results}|}{|\text{all returned results}|}$$

Recall is the fraction of relevant instances that are retrieved:

$$R = \frac{|\text{correct returned results}|}{|\text{all correct results}|}$$

In addition to $precision$ and $recall$ scores, Average Precision (AvP) [7] is commonly used [2] to assess ranked results. It defines as:

$$AvP = \sum_{i=1}^{N} P(i)\Delta R(i)$$

where *P(i)* is the precision of top-i results, $\Delta R(i)$ change in recall from top-(i-1) to top-i results.

Obviously, this score tends to be higher for algorithms that print out correct terms on top positions of the result.

In our experiments we considered only the AvP score, while precision and recall are omitted. For voting algorithm it is no simple way to compute recall, because it is not obvious what number of top results should be considered as correct terms. Also in a general case the overall number of terms in dataset is unknown.

3.2 Features

For our experiments we implemented the following features: C-Value, Domain Consensus, Domain Relevance, Frequency, Lexical Cohesion, Loglikelihood, Relevance, TF*IDF, Weirdness and Words Count. Words Count is the simple feature that shows a number of words in a word sequence. This feature may be useful for the classifier since values of other features may have different meanings for single- and multi-word terms [6].

Most of these features are capable to recognize both single- and multi-word terms, except T-test and Loglikelihood that are designed to recognize only two-word terms (bigrams). We generalize them to the case of n-grams according to the study [22].

Some of the features consider information from the collection of general-domain texts (reference corpus), in our case these features are Domain Relevance, Relevance, Weirdness. For this purpose we use statistics from Corpus of Contemporary American English[2].

[2] Statistics available at http://www.ngrams.info

For extracting term candidates we implemented simple approach based on predefined part-of-speech patterns. For simplicity, we extracted only unigrams, bigrams and trigrams by using patterns such as:

- Noun
- Noun Noun
- Adjective Noun
- Noun Noun Noun
- Adjective Noun Noun
- Noun Adjective Noun

3.3 Datasets

Evaluation of the approaches was performed on two datasets of medical and biological domains consisting of short English texts with marked-up specific terms:

Corpus	Documents	Words	Terms
GENIA	2000	400000	35000
Bio1	100	20000	1200

The last one (Bio1) has common texts with the first (GENIA), so we filtered out the texts that occur in both the corpora. We left GENIA without any modifications, while 20 texts were removed from Bio1 as common texts of the corpora.

Table 3: Results of evaluation on separated train and test sets without frequency filter

Trainset	Testset	Algorithm	AvP
GENIA	Bio1	Random Forest	0.30
GENIA	Bio1	Logistic Regression	**0.35**
–	Bio1	Voting	0.25
Bio1	GENIA	Random Forest	0.44
Bio1	GENIA	Logistic Regression	0.42
–	GENIA	Voting	**0.55**

3.4 Experimental results

3.4.1 Machine learning method versus Voting algorithm. We considered two test scenarios in order to compare quality of the implemented algorithms. For each scenario we performed two kinds of tests: with and without filtering of rare term candidates.

In the following tests the whole feature set was considered and the overall ranked result was assessed.

Cross-validation. We performed 4-fold cross-validation of the algorithms on both the corpora. We extracted term candidates from the whole dataset and divided them

on train and test sets. In other words, we considered the case when having some marked-up examples (train set) we should recognize terms in the rest of data (test set) extracted from the same corpus. So in case of voting algorithm the training set was simply omitted.

The results of cross-validation are shown in the Tables 1, 2. The Table 2 presents results of cross-validation on term candidates that appears at least two times in the corpus.

As we can see, in both the cases machine learning approach outperformed voting algorithm. Moreover, in the case without rare terms a difference of scores is higher. It can be explained by the following: feature values of rare terms (especially Frequency, Domain Consensus) are useless for the classification and add a noise to the model. When such the terms are omitted, the model becomes more clear.

Also in most cases Logistic Regression algorithm outperformed Random Forest, so in most of further tests we used only the best one.

Separate train and test datasets. Having two datasets of the same field, the idea is to check how the model trained on the one can predict the data from the other. For this purpose we used GENIA as a training set and Bio1 as a test one, then visa versa.

The results are shown in the Tables 3, 4. In the case when Bio1 was used as a training set, voting algorithm outperformed trained classifier. It could happen due to the fact that the training data from Bio1 does not fully reflect properties of terms in GENIA.

3.4.2 Dependency of average precision from number of top results.

In previous tests we considered overall results produced by the algorithms. Descending from the top to the bottom of the ranked list, AvP score can significantly change, so one algorithm can outperform another one on top-100 results but lose on top-1000. In order to explore this dependency, we measured AvP for different slices of the top results.

The Figure 1 shows the dependency of AvP from number of top results given by 4-fold cross-validation.

We also considered a scenario when GENIA was used for training and Bio1 for testing. The results are presented on the Figure 2.

3.4.3 Dependency of classifier performance from training set size.

In order to explore dependency between the amount of data used for training and average precision, we considered three test scenarios.

At first, we trained the classifier on GENIA dataset and tested it on Bio1. At each step the amount of training data was being decreased, while the test data remained without any modifications. The results of the test are presented on the Figure 3.

Table 4: Results of evaluation on separated train and test sets with frequency filter

Trainset	Testset	Algorithm	AvP
GENIA	Bio1	Random Forest	0.34
GENIA	Bio1	Logistic Regression	**0.48**
–	Bio1	Voting	0.31
Bio1	GENIA	Random Forest	0.60
Bio1	GENIA	Logistic Regression	0.62
–	GENIA	Voting	**0.65**

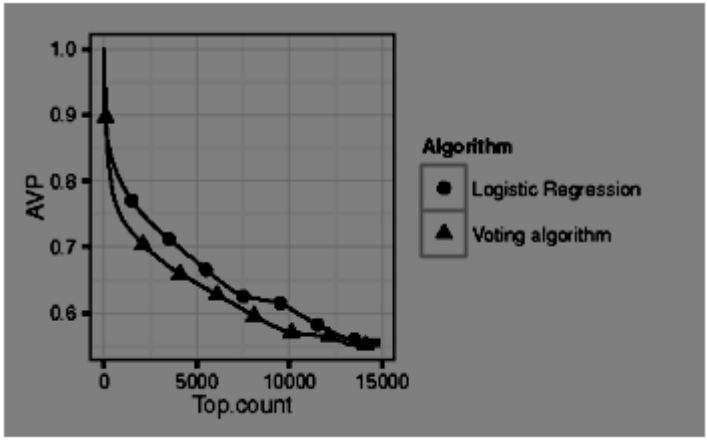

Figure 1: Dependency of AvP from top results given by cross-validation

Next, we started with 10-fold cross-validation on GENIA and at each step decreased the number of folds used for training of Logistic Regression and did not change the number of folds used for testing. The results are shown on the Figures 4-8.

The last test is the same as the previous one, except that the number of test folds was being increased at each step. So we started with nine folds used for training and one fold used for the test. At the next step we moved one fold from training set to the test set and evaluated again. The results are presented on the Figures 9-13. The interesting observation is that higher values of AvP correspond to the bigger sizes of the test set. It could happen because with increasing of the test set the number of high-confident terms is also growing: such the terms take most of the top positions of the list and improve AvP. In case of GENIA and Bio1 the top of the list mainly consists from the highly domain-specific terms that take high values for the features like Domain Relevance, Relevance, Weirdness: such the terms occur in the corpora frequently enough.

As we can see, in all of the cases the gain of AvP stopped quickly. So, in case of GENIA, it is enough to train on 10% of candidates to rank the rest 90% with the same performance. It could happen because of the relatively small number of fea-

tures are used and their specificity: most of them designed to have high magnitude for terms and low for non-terms. So, the data can be easily separated by the classifier having few training examples.

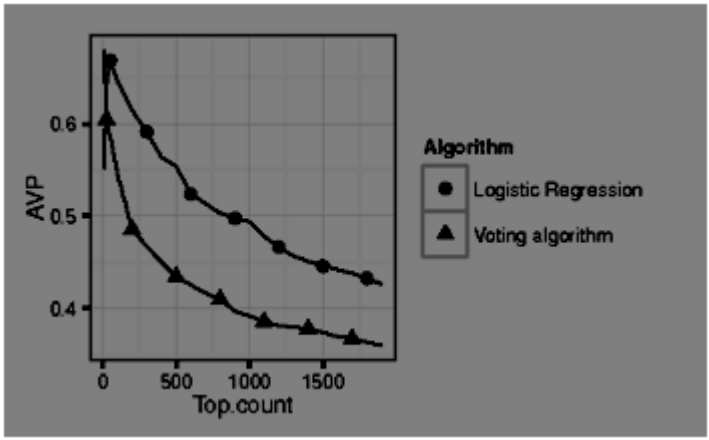

Figure 2: Dependency of AvP from top results on separated train and test sets

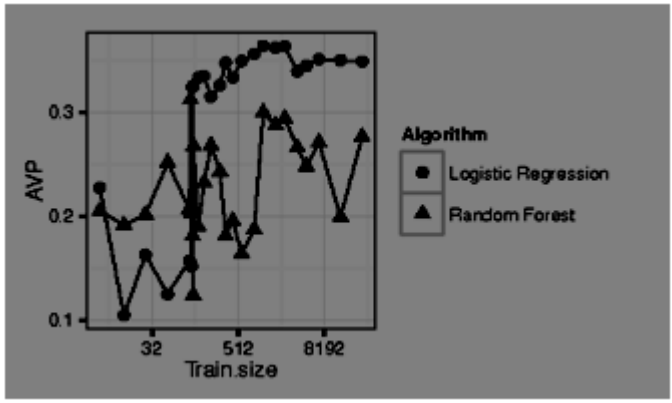

Figure 3: Dependency of AvP from train set size on separated train and test sets

3.5 Feature selection

Feature selection (FS) is the process of finding the most relevant features for the task. Having a lot of different features, the goal is to exclude redundant and irrelevant ones from the feature set. *Redundant* features provide no useful information as compared with the current feature set, while *irrelevant* features do not provide information in any context.

There are different algorithms of FS. Some of them rank separate features by relevance to the task, while others search subsets of features that get the best model for

the predictor [23]. Also the algorithms differ by their complexity. Because of big amount of features used in some tasks, it is not possible to do exhaustive search, so features are selected by greedy algorithms [24].

In our task we concentrated on searching the subsets of features that get the best results for the task. For such purpose we ran quality tests for all possible feature subsets, or, in other words, performed the exhaustive search. Having 10 features, we check 2^{10}-1 different combinations of them. In case of the machine learning method, we used 9 folds for test and one fold for train. The reason of such the configuration is that the classifier needs little data for training to rank terms with the same performance (see the previous section). For voting algorithm, we simply ranked candidates and then assessed overall list. All of the tests were performed on GENIA corpus and only the Logistic Regression was used as the machine learning algorithm.

The AvP score was computed for different slices of the top terms: 100, 1000, 5000, 10000, and 20000. The same slices are used in [2]. The best results for the algorithms are presented in the Tables 5, 6. This table shows that voting algorithm has better scores then machine learning method, but such the results are not fully comparable: FS for voting algorithm was performed on the whole dataset, while Logistic Regression was trained on 10% of term candidates. The average performance gain for voting algorithm is about 7%; while for machine learning it is only about 3%.

The best features for voting algorithm:

- **Top-100:** Relevance, TF*IDF
- **Top-1000**: Relevance, Weirdness, TF*IDF
- **Top-5000**: Weirdness
- **Top-10000**: Weirdness
- **Top-20000**: CValue, Frequency, Domain Relevance, Weirdness

The best features for the machine learning approach:

- **Top-100**: Words Count, Domain Consensus, Normalized Frequency, Domain Relevance, TF*IDF
- **Top-1000**: Words Count, Domain Relevance, Weirdness, TF*IDF
- **Top-5000**: Words Count, Frequency, Lexical Cohesion, Relevance, Weirdness
- **Top-10000**: Words Count, CValue, Domain Consensus, Frequency, Weirdness, TF*IDF
- **Top-20000**: Words Count, CValue, Domain Relevance, Weirdness, TF*IDF

As we can see, most of the subsets contain features based on a general domain. The reason can be that the target corpus has high specificity, so the most of terms do not occur in a general corpus.

The next observation is that in case of the machine learning algorithm, Words Count feature occurs in all of the subsets. This observation confirms an assumption that

this feature is useful for algorithms that recognize both the single- and multi-word terms.

Table 5: Results of FS for voting algorithm

Top count	All features	The best features
100	0.9256	0.9915
1000	0.8138	0.8761
5000	0.7128	0.7885
10000	0.667	0.7380
20000	0.6174	0.6804

Table 6: Results of FS for Logistic regression

Top count	All features	Supervised AvP
100	0.8997	0.9856
1000	0.8414	0.8757
5000	0.7694	0.7875
10000	0.7309	0.7329
20000	0.6623	0.6714

3.6 Discussion

Despite the fact that filtering of the candidates occurring only once in the corpus improves average precision of the methods, it is not always a good idea to exclude such the candidates. The reason is that a lot of specific terms can occur only once in a dataset: for example, in GENIA there are 50% of considered terms that occur only once. Of course, omitting such the terms extremely affects recall of the result. Thus such the cases should be considered for the ATR task.

One of the interesting observations is that the amount of training data is needed to rank terms without sufficient performance drop is extremely low. It leads to the idea of applying the bootstrapping approach for ATR:

- Having few marked-up examples, train the classifier
- Use the classifier to extract new terms
- Use the most confident terms as initial data at step 1.
- Iterate until all of confident terms will be extracted

This is a semi-supervised method, because only little marked-up data is needed to run the algorithm. Also the method can be transformed into fully unsupervised, if initial data will be extracted by some unsupervised approach (for example, by voting algorithm). The similar idea is implemented in study [20].

4. Conclusion and Future work

In this paper we have compared the performance of two approaches for ATR: machine learning method and voting algorithm. For this purpose we implemented the

set of features that include linguistic, statistical, termhood and unithood feature types. All of the algorithms produced ranked list of terms that then was assessed by average precision score.

In most tests machine learning method outperforms voting algorithm. Moreover it was explored that for the supervised method it is enough to have few marked-up examples, about 10% in case of GENIA dataset, to rank terms with good performance.

It leads to the idea of applying bootstrapping to ATR. Furthermore, initial data for bootstrapping can be obtained by voting algorithm because its top results are precise enough (see the Figure 1)

The best feature subsets for the task were also explored. Most of these features are based on a comparison between domain-specific documents collection and a reference general corpus. In case of the supervised approach, the feature Words Count occurs in all of the subsets, so this feature is useful for the classifier, because values of other features may have different meanings for single- and multi-word terms.

In cases when one dataset is used for training and another to test, we could not get stable performance gain using machine learning. Even the datasets are of the same field, a distribution of terms can be different. So it is still unclear if it is possible to recognize terms from unseen data of the same field having the once-trained classifier.

For our experiments we implemented the simple method of term candidates extraction: we filter out ngrams that do not match predefined part-of-speech patterns. This step of ATR can be performed in other ways, for example by shallow parsing, or chunking[3], generating patterns from the dataset [3] or recognizing term variants.

Another direction of further research is related to the evaluation of the algorithms on more datasets of different languages and researching the ability of cross-domain term recognition, i.e. using a dataset of one domain to recognize terms from others.

Also of particular interest is the implementation and evaluation of semi- and unsupervised methods that involve machine learning techniques.

References

[1]. Pazienza M., Pennacchiotti M., Zanzotto F. Terminology extraction: an analysis of linguistic and statistical approaches // Knowledge Mining. — 2005. — P. 255-279.

[2]. Zhang Z, Brewster C, Ciravegna F. A comparative evaluation of term recognition algorithms // Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC08), Marrakech, Morocco. — 2008.

[3]. Patry A., Langlais P. Corpus-based terminology extraction // Terminology and Content Development-Proceedings of 7th International Conference On Terminology and Knowledge Engineering, Litera, Copenhagen. — 2005.

[4]. Nokel M, Bolshakova E, Loukachevitch N. Combining multiple features for singleword term extraction. — 2012.

[3] Free chunker can be found in OpenNLP project: http://opennlp.apache.org

[5]. Kageura K., Umino B. Methods of automatic term recognition: A review // Terminology. — 1996. — V. 3, No 2. — P. 259-289.
[6]. Ahrenberg L. Term extraction: A review draft version 091221. — 2009.
[7]. Manning C, Schutze H. Foundations of statistical natural language processing. — MIT press, 1999.
[8]. Empirical observation of term variations and principles for their description / B. Daille, B. Habert, C. Jacquemin, J. Royaute // Terminology. — 1996.— V. 3, No 2. — P. 197-257.
[9]. Foo J. Term extraction using machine learning. — 2009.
[10]. Zhang W, Yoshida T., Tang X. Using ontology to improve precision of terminology extraction from documents // Expert Systems with Applications. — 2009. — V. 36, No 5. — P. 9333-9339.
[11]. Dobrov B., Loukachevitch N. Multiple evidence for term extraction in broad domains // Proceedings of the 8th Recent Advances in Natural Language Processing Conference (RANLP 2011). Hissar, Bulgaria. — 2011. — P. 710-715.
[12]. Church K., Hanks P. Word association norms, mutual information, and lexicography // Computational linguistics. — 1990. — V. 16, No 1. — P. 22-29.
[13]. Frantzi K., Ananiadou S. Extracting nested collocations // Proceedings of the 16th conference on Computational linguistics-Volume 1 / Association for Computational Linguistics. — 1996. — P. 41-46.
[14]. Navigli R., Velardi P. Semantic interpretation of terminological strings // Proc. 6th IntB^TMl Conf. Terminology and Knowledge Eng. — 2002. — P. 95-100.
[15]. Sclano F, Velardi P. Termextractor: a web application to learn the shared terminology of emergent web communities // Enterprise Interoperability II. — 2007. — P. 287-290.
[16]. Park Y, Byrd R., Boguraev B. Automatic glossary extraction: beyond terminology identification // Proceedings of the 19th international conference on Computational linguistics-Volume 1 / Association for Computational Linguistics. — 2002. — P. 1-7.
[17]. Corpus-based terminology extraction applied to information access / A. Penas, F. Verdejo, J. Gonzalo et al. // Proceedings of Corpus Linguistics / Citeseer. — V. 2001. — 2001.
[18]. University of surrey participation in trec8: Weirdness indexing for logical document extrapolation and retrieval (wilder) / K. Ahmad, L. Gillam, L. Tostevin et al. // The Eighth Text REtrieval Conference (TREC-8). — 1999.
[19]. Velardi P., Missikoff M., Basili R. Identification of relevant terms to support the construction of domain ontologies // Proceedings of the workshop on Human Language Technology and Knowledge Management-Volume 2001 / Association for Computational Linguistics. — 2001. — P. 5.
[20]. Fault-tolerant learning for term extraction / Y. Yang, H. Yu, Y. Meng et al. — 2011.
[21]. Manning C, Raghavan P. Introduction to information retrieval. — V. 1.
[22]. Daille B. Study and implementation of combined techniques for automatic extraction of terminology // The balancing act: Combining symbolic and statistical approaches to language. — 1996. — V. 1. — P. 49-66.
[23]. Guyon I., Elisseeff A. An introduction to variable and feature selection // The Journal of Machine Learning Research. — 2003. — V. 3. — P. 1157-1182.
[24]. Molina L., Belanche L., Nebot A. Feature selection algorithms: A survey and experimental evaluation // Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on / IEEE. — 2002. — P. 306-313.

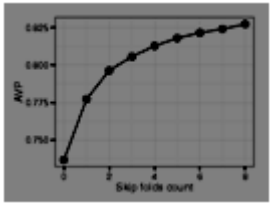

Figure 4: Dependency of AvP from number of excluded folds with fixed testset size: 10-fold cross-validation with 1 test fold and 9 to 1 train folds: Top-100terms

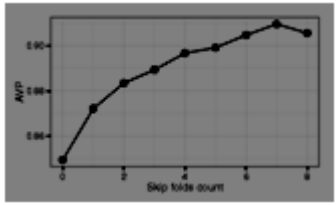

Figure 9: Dependency of AvP from number of excluded folds with changing testset size: 10-fold cross-validation with 1 to 9 test folds and 9 to 1 train folds:Top-100 terms

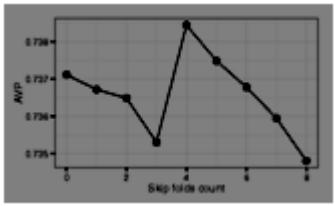

Figure 5: Dependency of AvP from number of excluded folds with fixed testset size: Top-1000 terms

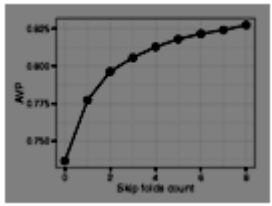

Figure 10: Dependency of AvP from number of excluded folds with changing testset size: Top-1000 terms

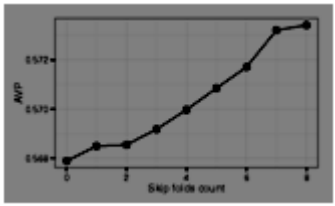

Figure 6: Dependency of AvP from number of excluded folds with fixed testset size: Top-5000 terms

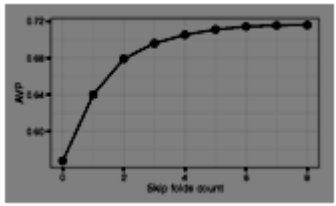

Figure 11: Dependency of AvP from number of excluded folds with changing testset size: Top-5000 terms

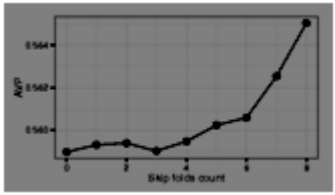

Figure 7: Dependency of AvP from number of excluded folds with fixed testset size: Top-10000 terms

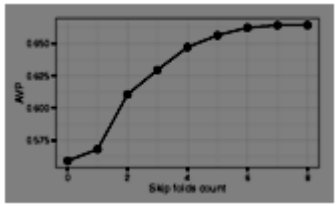

Figure 12: Dependency of AvP from number of excluded folds with changing testset size: Top-10000 terms

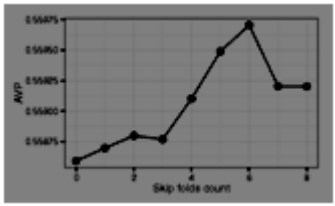

Figure 8: Dependency of AvP from number of excluded folds with fixed testset size: Top-20000 terms

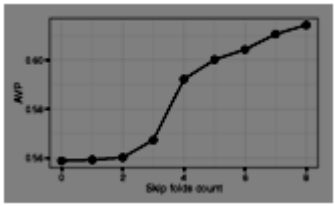

Figure 13: Dependency of AvP from number of excluded folds with changing testset size: Top-20000 terms

Автоматическое распознавание предметно-специфичных терминов: экспериментальная проверка

Д.Г. Федоренко <fedorenko@ispras.ru >
Н.А. Астраханцев <astrakhantsev@ispras.ru>
Д.Ю. Турдаков <turdakov@ispras.ru>
ИСП РАН, 109004, Россия, г. Москва, ул. А. Солженицына, дом 25.

Аннотация. В статье приводятся результаты экспериментальной проверки современных подходов распознавания предметно-специфичных терминов: подхода на основе машинного обучения и подхода на основе алгоритма голосования. Показывается, что в большинстве случаев подход на основе машинного обучения показывает лучшие результаты и требует мало данных для обучения; также для обоих методов производится поиск наиболее информативных признаков.

Ключевые слова: автоматическое распознавание терминов, извлечение терминов, машинное обучение, экспериментальная проверка, поиск информативных признаков.

To sort or not to sort: the evaluation of R-Tree and B+-Tree in transactional environment with ordered result requirement[1]

P.V. Fedotovsky <pavel.v.fedotovsky@gmail.com>
G.A. Erokhin <george.erokhin@gmail.com>
K.E. Cherednik <kirill.cherednik@math.spbu.ru>
K.K. Smirnov <kirill.k.smirnov@gmail.com>
G.A. Chernishev <chernishev@gmail.com>
SPbU, 11 Universitetskaya nab., St. Petersburg, 199034, Russian Federation

Abstract: In this paper we consider multidimensional indexing with the additional constraint of lexicographical ordering. In order to deal with this problem we discuss two well-known tree data structures: R-tree and B-tree. We study the problem in the transactional environment with read committed isolation level. To evaluate these approaches we had implemented these structures (modified GiST ensures concurrency) and provide extensive experiments.

Keywords: R-Tree; B-Tree; Indexing; Multidimensional Indexing; Lexicographical Order; Experimentation; PostgreSQL.

1. Introduction

In this paper we consider the problem of multidimensional indexing with one additional constraint — the lexicographical ordering of the result set. Effective multidimensional indexing is rather old and well-explored topic, however, one can't say that the problem is solved. New approaches continue to emerge. The addition of the ordering requirement further drives this problem into the domain of research activity.

Effective solutions for the problem of multidimensional indexing are needed for geospatial data, CAD systems, multimedia data and also of use for OLAP data.

There are two main approaches for multidimensional indexing: tree-based and hash-based. The former are R-Tree, KDB tree, Octree, X-Tree and many others. The latter are mainly used for nearest neighbor and similarity query evaluation.

[1] This work is partially supported by Russian Foundation for Basic Research grant 12-07-31050.

We are mainly interested in R-Tree because of its popularity in commercial DBMS systems [1]: PostgreSQL, Oracle, Informix, SQLite and MySQL use this approach. This interest proves, that despite being rather old (more than 25 years), R-Tree still may be called industrial-strength technology. Moreover, until recently R-Tree was the only one method of multidimensional indexing in PostgreSQL2.

This work was inspired by participation in ACM SIGMOD Contest 2012. This problem was provided by the contest organizers, as well as benchmarks and example Berkeley DB-based implementation. Our team participated in this contest and was ranked 5th on public tests3.

The problem is formulated as follows: given an n-dimensional space and queries in transactional environment, what kind of data structure should we use for optimal performance?

In order to solve this problem we implemented a prototype of multidimensional transactional index. This index works within read committed isolation level. Our prototype contains both B^+-Tree and R-Tree built around GiST model.

The contribution of this paper is following:

- The validation of our prototypes by comparison with industrial-strength databases: Berkeley DB and PostgreSQL.
- Experimental study of influence of workload parameters on performance of these two structures. These workload parameters include query window size and others.

The rest of this paper is organized as follows. In the next section we provide detailed specification of the task, describe queries and data. Then, in the section 3 we describe two alternative approaches and survey related works. Section 4 contains overview of our system. In the section 5 we provide evaluations and comparisons with PostgreSQL and Berkeley DB.

2. The Task

The task offered at the contest was to build a multidimensional high-throughput in-memory indexing system. The index should support concurrent access by many threads and work within read committed isolation level. This level is chosen because of its popularity in commercial database systems, for example it is the default isolation level in PostgreSQL. The reason of its popularity is that it offers a good trade-off between correctness and performance. According to the study [2] it can offer 2.5 – 3 times better performance than serializable isolation level. Our index resides in-memory and no crash-recovery component is required.

[2] http://www.postgresql.org/docs/9.2/static/spgist.html

[3] http://web.archive.org/web/20120424201336/http://wwwdb.inf.tu-dresden.de/sigmod2012contest/leaderboard/ accessed: 26/04/2014

2.1. Qu

Ol.eries.

There are several possible types of queries:

- Point queries: insert, update, delete and select.
- Range queries — they select a subset of data and the result should be sorted. This type of query is defined by a conjunction of attribute predicates. The individual predicates may be not only be intervals or points, but also a wildcards.

The distribution of query types is described in the specification.

Another important aspect to consider is the admissible amount of operations per transaction. It is specified, that there are no more than few hundred retrieved points per transaction. In particular, the original task states that no more than 200 points are touched by any transaction. This number is justified by the fact that OLTP transactions are very light-weight. For example, the heaviest transaction in TPC-C reads about 200 records [3].

2.2. Data and Workloads

The task statement specifies several data types: INT(4), INT(8) and VARCHAR(512). However, in this work, we had to drop VARCHAR (see section 5 for details). The key consists of several attributes of these data types. The payload is represented by a sequence of bytes.

The data may come in one of several types of distributions: normal, uniform and Zipf's law (each is applied to coordinate independently). In our tests we used only uniform one.

Duplicate keys are allowed, we refer the reader to the web site for the detailed handling description.

In our experiments we heavily rely upon workloads and benchmark driver provided by organizers. These workloads are essentially synthetic datasets. We don't reuse workloads used during the contest, instead we use the provided framework to define our own.

Thorough task specification can be found here[4].

3. Related Work and Architectural Alternatives

In order to solve this problem two architectural approaches may be used. The first one is to use B^+-Tree and concatenate the values of individual coordinates into the composite key. The B^+-Tree [4] is the balanced data structure, which contains values in the leaf nodes while inner nodes contain pointers and intervals. These intervals define the unique path to the leaf.

[4] http://wwwdb.inf.tu-dresden.de/sigmod2012contest/

The strong points of this approach are:

- The overall simplicity of this data structure and general easiness for implementation.
- The abundance of concurrency control mechanisms for this kind of tree [4].
- It is possible to tune one, a lot of cache-conscious modifications exist.
- No need to sort, because keys are already stored in the right order.

Let's review the last item. Suppose that we have a three dimensional index and a query: (1,2,*). In order to evaluate it, we have to find the first entry with prefix "1|2|" and then sequentially scan the tree until prefix mismatch.

However one can name weak points:

- We have to pack and unpack the keys with each comparison.
- Queries containing interval predicates are harder to process.
- This tree may perform poorly with wildcard queries.

The first one is the minor drawback, its cost may be negligible. However, the second and the third are more formidable ones.

The intervals inside attributes can be processed in the same manner as above, but additional checks are needed. This results in additional complexity of the implementation.

Regarding the third item, consider query (1,*,3). In order to evaluate it, we have to find the key starting with a prefix "1", then we have to iterate through all values which have it. It will require a lot more of comparisons, and what is more important, we will be forced to discard a lot of values in the middle. Consider the following leaf level:

$$1|2|3|, 1|2|4|, 1|2|4|, ..., 1|2|4|, 1|3|3|.$$

In this situation we will need only two values: 1|2|3| and 1|3|3|. But we would be forced to iterate through all these values and discard them. The situation becomes grave when we have wildcard condition in the first attribute: (*,2,3). In this case we have to scan the whole index.

One of the most popular data structures used for indexing multidimensional data is R-tree [5]. According to [6] R-tree is a tree data structure, defined by a pair (m, M) with the following properties:

- Each leaf node (unless it is the root) can host up to M entries, whereas the minimum allowed number of entries is $m \leq M/2$. Each entry is of the form (mbr, oid), such that mbr is the MBR that spatially contains the object and oid is the object identifier.
- The number of entries that each internal node can store is again between $m \leq M/2$ and M. Each entry is of the form (mbr, p), where p is a pointer to a child of the node and mbr is the MBR that spatially contains the MBRs contained in this child.

- The minimum allowed number of entries in the root node is 2, unless it is a leaf (in this case, it may contain zero or a single entry).
- All leaves of the R-tree are at the same level.

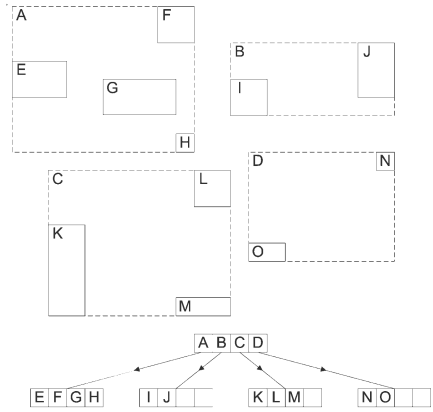

Fig. 1: R-Tree example

Example data and corresponding R-Tree are presented on figure 1. The data consists of boxes marked E-O and boxes A-D denote the MBRs.

R-Tree was proposed first by Antonin Guttman in [7]. This study prompted a wave of research papers and one can say that it gave birth to the new area of research. This research related to development of the new R-Tree variants [1,8,9], niche approaches [6,9], split techniques [10–12], concurrency techniques [13,14] etc. The study [6] states that there are several dozens of R-Tree variants.

One may think of an R-Tree as a generalization of B^+-Tree:

- Data are also kept in the leaves.
- This data structure is also balanced, all leaf nodes are at the same height.
- Inner nodes keep bounding boxes, which can be considered as a generalization of intervals which guide the search in B^+-Tree.

However, there are several differences:

- It might be necessary to explore more than one path from the root to a leaf in order to locate a key. This is the result of the MBR intersection allowance, which leads to more complex search and traversal algorithms than those of B^+-Tree.
- A node split is ambiguous, determining the optimal node split is a computationally hard problem. A “bad” split may lead to a serious degradation of the performance.

- R-Tree does not contain links to sibling leaves for an easy range query execution.

GiST (Generalized Search Tree) [14] is a "template" index structure which supports extensible set of queries and datatypes. This index can be parameterized by a variety of data structures.

Unlike B^+-Tree based one, this approach would require sorting of the results. This is a significant drawback which may negatively impact performance. The goal of this paper is to evaluate, which of these approaches is better. Intuitively one can say that the outcome should depend on the query selectivity.

4. System Overview

Our system follows classical design guidelines and contains several components:

- A tree data structure. Currently implemented as B^+-Tree and R-Tree. R-Tree is based upon GiST [13], a popular template index structure including concurrency control techniques. This model allows to extend with the means of concurrent access almost any tree conforming to certain requirements. This is a widespread approach and it is used, for example, in PostgreSQL.
- Concurrency control. We used mechanism adapted from [14] with locks, latches and Node Sequence Numbers. Also we provided deadlock resolution mechanism. Eventually, we ensure the read committed isolation level. However currently our prototype lacks logging and recovery features.
- Memory manager. It is a well-known fact that a standard memory manager can't provide optimal performance for the whole range of applications and sometimes it is desirable to find or implement a specifically-tailored one. Our memory manager is essentially a wrapper which intercepts new and delete calls to make use a pool of free blocks.
- Sorting of the results. In order to solve the problem one must present lexicographically sorted results. While B^+-Tree provides already ordered results, R-Tree does not. Our R-Tree implementation sorts the results via merge-sort (we keep sorted data inside boxes).
- Deletion of records. In our implementation we don't delete records, instead, we mark them as "deleted" and take this into account during the processing.

5. Validation and Experiments

5.1. Validation

We validated our implementation in two ways. First, we used public unit-tests supplied by the contest organizers. These unit-tests ensured correctness of an isolation level (read committed) implementation and several other implementation

issues. We also extended basic set of test cases with new ones. Then, our implementation participated in the contest [15].

5.2. PostgreSQL validation and tuning

We also compared our implementation with PostgreSQL v9.1 database system. This step was needed to check the relative level of achieved performance and general transferability of results. We implemented a simple wrapper application which directed queries to PostgreSQL. PostgreSQL uses a disk-based GiST index, while our prototype is an in-memory one. Also, our prototype lacks a logging and recovery component. Thus, in order to conduct fair tests we had to simulate in-memory index in PostgreSQL.

To completely eliminate slow disk-related operations we placed database cluster on tmpfs. This way we can be sure that every operation PostgreSQL performs (logging, committing, buffers flushing, etc.) does not involve interactions with a hard drive.

Other important implementation aspects included:

- Wrapper connection pooling. We used a pool of connections inside our wrapper to eliminate the cost of connection creation every time a transaction is executed.
- We parameterized GiST with cube data structure.
- To eliminate overheads related to durability we turned off: fsync, full page writes and synchronous commit. Checkpoint segments setting was left intact.
- We were forced to abandon string datatype due to PostgreSQL cube restrictions (only float parameters supported).
- PostgreSQL runs in read committed isolation level by default.

Unfortunately, due to several reasons, we were not able to completely approach the performance of our system. First, unlike BDB, PostgreSQL needs to maintain not only the index, but also a table. Second, calls to PostgreSQL via connections are less effective than the direct function calls. The last issue is the security checks which were also left intact.

5.3. Hardware and software setup

For the first group of experiments (comparison with PostgreSQL and Berkeley DB) used the following hardware and software setup:

- Intel Core i7-2630QM, 2.00 GHz, Hyper-Threading Enabled, L1 Cache 64KB, L2 Cache 256 (per core), L3 Cache 6MB, 6GB RAM
- x86_64 GNU/Linux, kernel 3.5.0-21, gcc 4.7.2
- PostgreSQL 9.1.7

The second group used the more performing one:

- Hardware: 2 x Intel Xeon CPU E5-2660 0 @ 2.20GHz, 64GB RAM, MB S2600GZ
- Software: Linux Ubuntu 3.2.0-29-generic x86_64, GCC 4.6.3

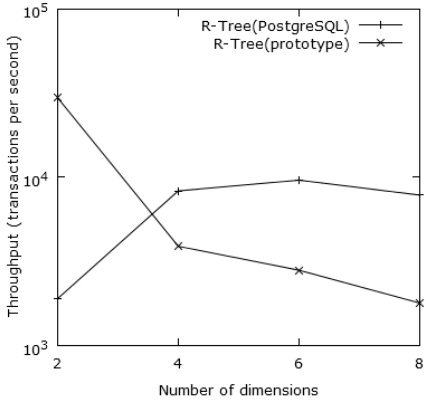

Fig. 2: Performance of PostgreSQL and our prototype (R-Tree).

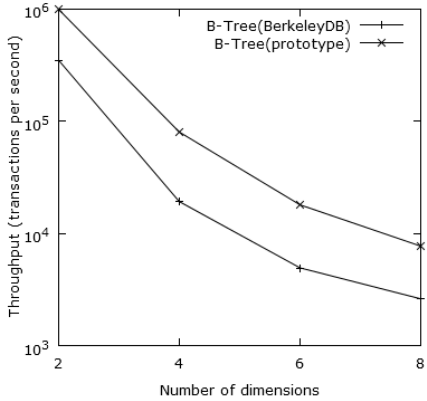

Fig. 3: Performance of Berkeley DB and our prototype (B+-Tree).

5.4. Comparison with PostgreSQL and Berkeley DB

In this section we provide a comparison of our prototypes with industrial strength systems. The wrapper for Berkeley DB was provided by the organizers, PostgreSQL wrapper was developed by the authors (its architecture was described earlier). We compare the performance varying the number of dimension and use single 64MB index. The query type distribution is the same as in the original contest task, uniformly distributed data was used. The results are presented on figures 2 and 3.

We can see:

- Our prototypes are comparable to industrial ones in terms of overall performance.
- The solution which uses R-Tree significantly differs from B^+-Tree in terms of performance. This difference has prompted us into further investigation, which resulted in this paper.

5.5. Experimental Evaluation

The goal of this paper is to evaluate, what is better: to use R-Tree and to sort or not to sort with B^+-Tree, but risk excess comparisons.

Table 1: Measured exponent value for R-Tree and B+-Tree, second hardware setup

Tree type	2	4	6	8
R-Tree (64MB)	-0.389 ± 0.052	-0.227 ± 0.011	-0.117 ± 0.012	0.026 ± 0.028
R-Tree (512MB)	-0.547 ± 0.053	-0.250 ± 0.013	-0.135 ± 0.017	-0.071 ± 0.007
B^+-Tree (64MB)	0.482 ± 0.011	0.632 ± 0.025	0.465 ± 0.051	0.333 ± 0.028
B^+-Tree (512MB)	0.476 ± 0.020	0.626 ± 0.038	0.470 ± 0.039	0.321 ± 0.031

In order to solve this problem we had conducted a series of experiments. In these experiments we evaluate the performance of two systems, while varying the query selectivity. We separately consider the following dimensions: 2, 4, 6, 8. We had considered indexes of two sizes: 64 and 512 MB, uniform data distribution. We concentrate on the most interesting query type, which present in the original contest workload: a range query without wildcard predicates. These experiments were conducted using our prototypes, which we had described in the previous section. The reason of this switch is the time it takes to construct an index by PostgreSQL DBMS and also the query plan problem. The plans which are generated by the optimizer are essentially the following: at first, perform index scan (e.g. read all R-Tree boxes), then sort the results. It is impossible to push down sorting in PostgreSQL because its GiST selection method doesn't use merge-sort. This is a critical drawback, because in our task we select at most 200 entries. Thus, our prototype can read only a part of the data and don't sort all the content of the touched boxes. The query plan problem is not an issue in BDB, because of the simplicity of BDB and the fact that B^+-Tree is already sorted.

The graphs describing the experimental results are presented on Figures 4-11.

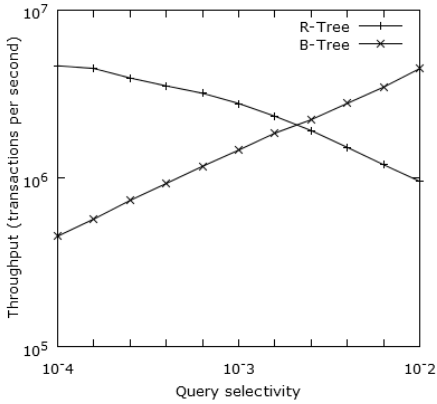

Fig. 4: Performance of R-Tree and B-tree indexes, 64MB, 2 dimensions.

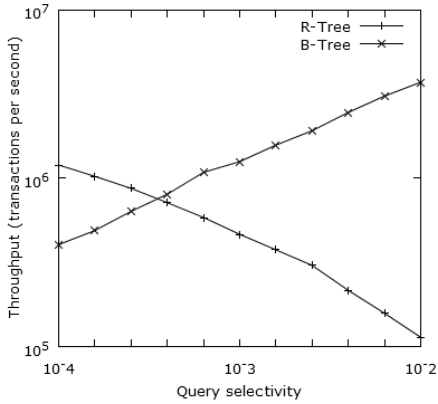

Fig. 5: Performance of R-Tree and B-tree indexes, 512MB, 2 dimensions.

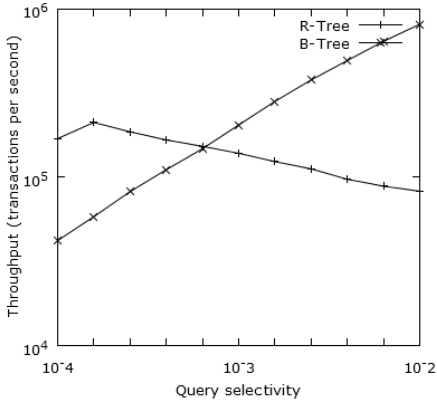

Fig. 6: Performance of R-Tree and B-tree indexes, 64MB, 4 dimensions.

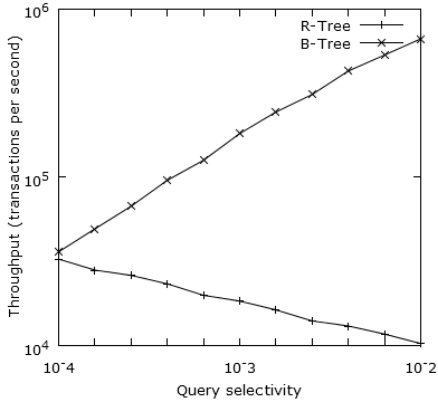

Fig. 7: Performance of R-Tree and B-tree indexes, 512MB, 4 dimensions.

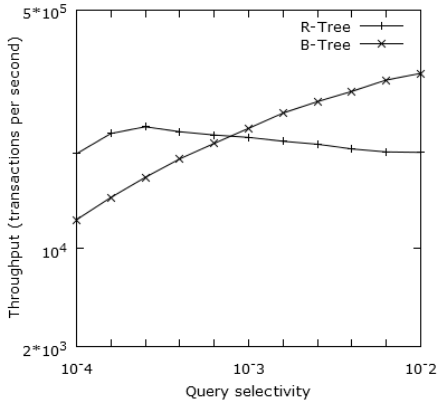

Fig. 8: Performance of R-Tree and B-tree indexes, 64MB, 6 dimensions.

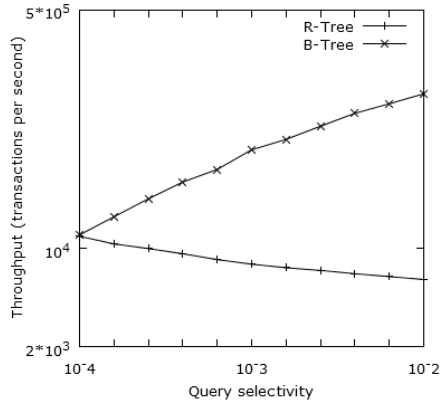

Fig. 9: Performance of R-Tree and B-tree indexes, 512MB, 6 dimensions.

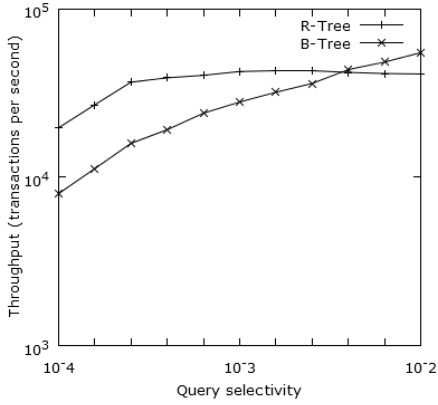

Fig. 10: Performance of R-Tree and B-tree indexes, 64MB, 8 dimensions.

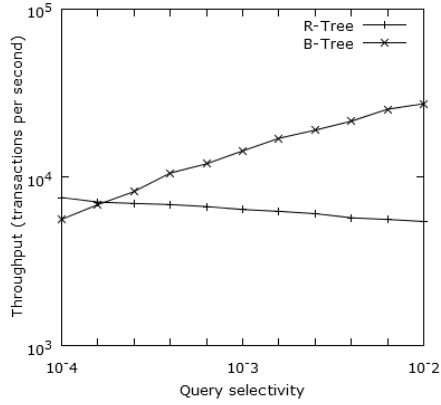

Fig. 11: Performance of R-Tree and B-tree indexes, 512MB, 8 dimensions.

Note the double logarithmic scales, which we used in order to illustrate our finds. They are the following:

- The throughput of the system depends on a query selectivity. This dependence can be described by the power law:
- $$P = a * S^b,$$
- where P denotes the throughput, S — query selectivity, a and b are parameters. The graphs show this kind of dependency by the straight line. This approximately linear dependency persists in all considered dimension sizes.
- The considered query type affects the performance of the systems in the following way: the performance of R-Tree degrades as the value of query selectivity decreases, while at the same time B^+-Tree performance increases.
- As the number of dimensions increases, the exponent b changes in the way shown in the Table 1. Increasing the dimensionality leads to b decrease in case of R-Tree, i.e. having more dimensions lowers impact of query selectivity. There is no manifested trend in B^+-Tree behaviour.
- There is no simple way to determine intersection point of R-Tree and B-Tree, it depends on number of dimensions and index size.

Table 2: Measured exponent value for R-Tree and B+-Tree, 64MB only, both hardware setups

Tree type	2	4	6	8
R-Tree (1st setup)	-0.369 ± 0.041	-0.220 ± 0.003	-0.115 ± 0.008	0.026 ± 0.032
R-Tree (2nd setup)	-0.389 ± 0.052	-0.227 ± 0.011	-0.117 ± 0.012	0.026 ± 0.028
B^+-Tree (1st setup)	0.478 ± 0.006	0.635 ± 0.034	0.467 ± 0.032	0.332 ± 0.033
B^+-Tree (2nd setup)	0.482 ± 0.011	0.632 ± 0.025	0.465 ± 0.051	0.333 ± 0.028

Despite that the primary goal of this paper is not fully reached, we have obtained a very promising result — the power-law dependency. One of the possible ways of the advancement of this research is to verify, whether the calculated exponent values depend on a hardware setup or not. If we could prove that these value are hardware-independent, then they are fundamental properties of the data structures (at least, implementations) and the sets of queries.

We took the first step the first step of this way and re-evaluated exponent value on different hardware platforms. We had used the first setup (see section 5.3) as this different platform. Unfortunately we could only re-evaluate 64MB workload due to memory limit. The measured values can be found in Table 2 and the corresponding graphs are presented on Figures 12-19. For the easiness of analysis they had been provided with their well-performing counterparts.

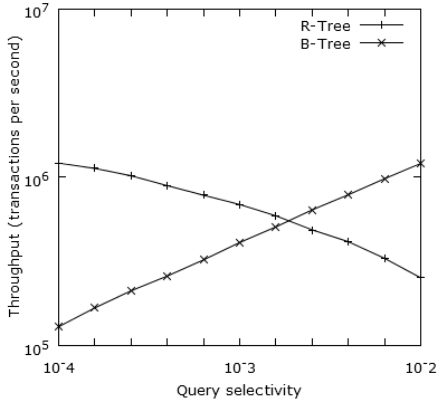

Fig. 12: Performance of R-Tree and B-tree indexes, 64MB, 2 dimensions, 1st configuration.

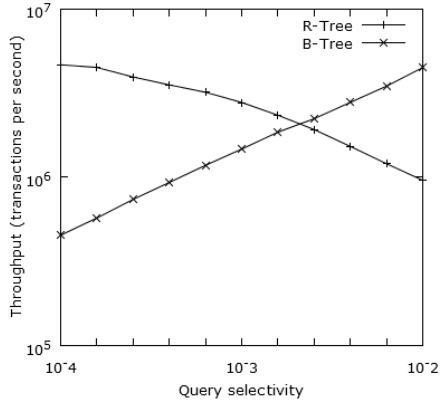

Fig. 13: Performance of R-Tree and B-tree indexes, 64MB, 2 dimensions, 2nd configuration.

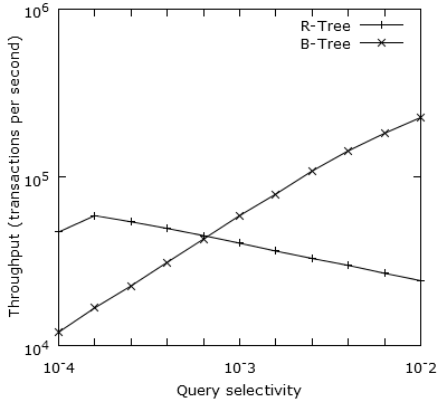

Fig. 14: Performance of R-Tree and B-tree indexes, 64MB, 4 dimensions, 1st configuration.

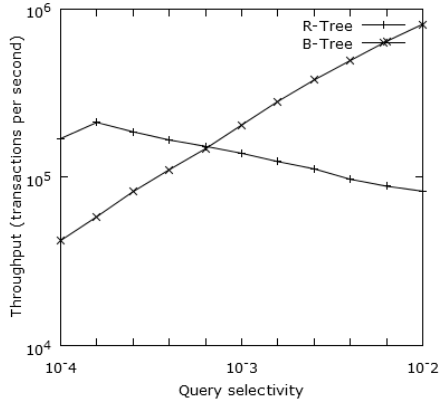

Fig. 15: Performance of R-Tree and B-tree indexes, 64MB, 4 dimensions, 2nd configuration.

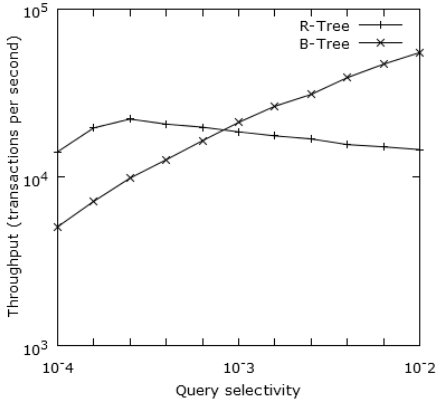

Fig. 16: Performance of R-Tree and B-tree indexes, 64MB, 6 dimensions, 1st configuration.

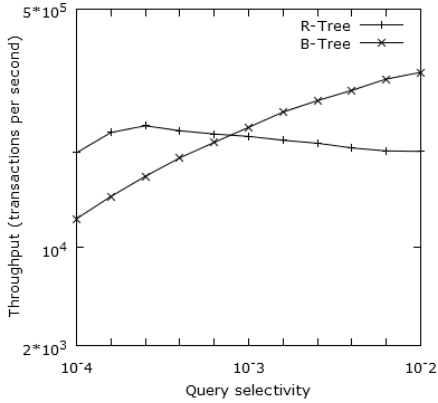

Fig. 17: Performance of R-Tree and B-tree indexes, 64MB, 6 dimensions, 2nd configuration.

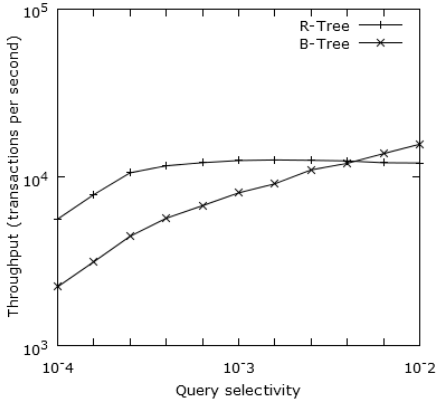

Fig. 18: Performance of R-Tree and B-tree indexes, 64MB, 8 dimensions, 1st configuration.

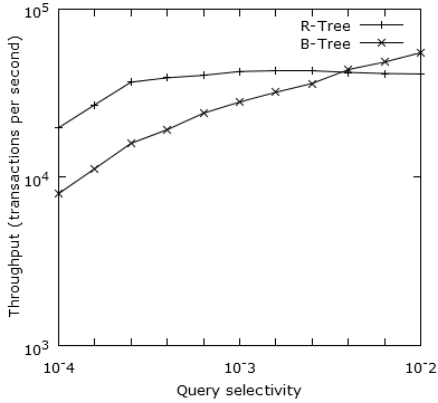

Fig. 19: Performance of R-Tree and B-tree indexes, 64MB, 8 dimensions, 2nd configuration.

As we can see, the values obtained on different hardware setups match each other, so it is very likely that hypothesis of hardware independence holds true and should be carefully examined.

6. Conclusions

In this paper we have considered the problem of multidimensional point indexing under additional restriction: ordering of the result set. We have experimentally evaluated two data structures — R-Tree and B^+-Tree on uniformly distributed data. The experiments allowed us to establish the impact of the query selectivity on system performance as power function. Also we examined the dependency of power-law parameters on dimension cardinality. As a future work we will provide more empirical evidence to the hypothesis of independence of power-law exponent on index size. Recommendation for B-Tree and R-Tree user: unfortunately, we were not able to find an easy way to calculate intersection point, so workloads should be evaluated ad hoc.

7. Acknowledgements

We would like to thank organizers of ACM SIGMOD Programming Contest'12 for providing a benchmark, data generator, unit tests and Berkeley DB wrapper implementation. This work is partially supported by Russian Foundation for Basic Research grant 12-07-31050.

References

[1]. Beckmann N., Seeger B. A revised R*-tree in comparison with related index structures // Proc. of the 2009 ACM SIGMOD International Conference on Management of data, SIGMOD '09. ACM, New York, NY, USA, 2009. P. 799–812.

[2]. Bober P.M., Carey M.J. On mixing queries and transactions via multiversion locking // Proc. of the Eighth International Conference on Data Engineering. IEEE Computer Society, Washington, DC, USA, 1992. P. 535–545.

[3]. Stonebraker M., Madden S., Abadi D.J., Harizopoulos S., Hachem N., Helland P. The end of an architectural era: (it's time for a complete rewrite) // Proc. of the 33rd international conference on Very large databases, VLDB'07. VLDB Endowment, 2007. P. 1150–1160.

[4]. Weikum G., Vossen G. Transactional information systems: theory, algorithms, and the practice of concurrency control and recovery. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001. 852 P.

[5]. Papadopoulos A.N., Corral A., Nanopoulos A., Theodoridis Y. R-Tree (and Family). In LING LIU and M. TAMER J OZSU, eds. Encyclopedia of Database Systems. Springer US, 2009. P. 2453–2459. doi: 10.1007/978-0-387-39940-9_300.

[6]. Manolopoulos Y., Nanopoulos A., Papadopoulos A.N., Theodoridis Y. R-Trees: Theory and Applications (Advanced Information and Knowledge Processing). Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[7]. Guttman A. R-trees: a dynamic index structure for spatial searching // SIGMOD Rec. - June 1984. – No. 14(2). P 47–57.

[8]. Kamel I., Faloutsos C. Hilbert Rtree: An Improved R-tree using Fractals // Proc. of the 20th International Conference on Very Large Data Bases, VLDB'94. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1994. P. 500–509.

[9]. Sellis T.K., Roussopoulos N., Faloutsos C. The R+-Tree: A Dynamic Index for Multi-Dimensional Objects // Proc. of the 13th International Conference on Very Large Data

Bases, VLDB'87. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 1987. P. 507–518.
[10]. Al-Badarneh A.F., Yaseen Q., Hmeidi I. A new enhancement to the R-tree node splitting // J. Inf. Sci. –feb 2010. –No 36(1). – P 3–18.
[11]. Ang C., Tan T. New linear node splitting algorithm for R-trees // Advances in Spatial Databases Michel Scholl and Agnes Voisard eds vol. 1262 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 1997. P. 337–349. doi 10.1007/3-540-63238-7_38.
[12]. Brakatsoulas S., Pfoser D., Theodoridis Y. Revisiting R-Tree Construction Principles // Proc. of the 6th East European Conference on Advances in Databases and Information Systems, ADBIS'02. London, UK, 2002. Springer-Verlag. P 149–162.
[13]. Hellerstein J.M., Naughton J.F., Pfeffer A. Generalized Search Trees for Database Systems // Proc. of the 21th International Conference on Very Large Data Bases, VLDB'95. Morgan Kaufmann Publishers Inc, San Francisco, CA, USA, 1995. P. 562–573.
[14]. Kornacker M., Mohan C., Hellerstein J.M. Concurrency and recovery in generalized search trees // SIGMOD Rec. – June 1997. –No. 26(2). – P 62–72.
[15]. ACM SIGMOD Programming Contest'12. http://wwwdb.inf.tu-dresden.de/sigmod2012contest/. Accessed: 11/09/2012.

Сортировать или нет: экспериментальное сравнение R-Tree и B^+-Tree в транзакционной системе для упорядоченной выдачи

Федотовский П.В. <pavel.v.fedotovsky@gmail.com>
Ерохин Г.А. <george.erokhin@gmail.com>
Чередник К. Е. <kirill.cherednik@math.spbu.ru>
Смирнов К.К. <kirill.k.smirnov@gmail.com>
Чернышев Г.А. <chernishev@gmail.com>
Санкт-Петербургский государственный университет, г. Санкт-Петербург, Россия

Аннотация: В данной работе мы изучаем задачу многомерного индексирования с учетом дополнительного требования – лексикографической упорядоченности результатов запроса. Для решения этой задачи мы рассматриваем две хорошо известные структуры данных – R-дерево и **B+**-дерево, которые используются в транзакционной системе с использованием уровня изоляции read committed. Для сравнения подходов мы реализовали эти структуры (параллельный доступ обеспечивается с помощью GiST) и провели с их помощью ряд экспериментов, результаты которых и представлены в статье.

Ключевые слова: R-дерево; B+-дерево; Индексирование; Многомерное индексирование; Лексикографический порядок; Эксперименты; PostgreSQL.

Comparison of partial orders clustering techniques

A. Raskin <a.a.raskin@gmail.com>
National research nuclear university «MEPhI»,
115409, Russia, Moscow, Kashirskoe shosse, 31.

Abstract. In this paper, we compare three approaches of clustering partial ordered subsets of a set of items. First approach was k-medoids clustering algorithm with distance function based on Levenshtein distance. The second approach was k-means algorithm with cosine distance as distance function after vectorization of partial orders. And the third one was k-medoids algorithm with Kendall's tau as a distance function. We use Adjusted Rand Index as a measure of quality of clustering and find out that clustering with all three methods get stable results when variance of number of items ranked is high. Vectorization of partial orders get best results if number of items ranked is low.

Keywords: Levenshtein distance; partial orders; clustering; distance measure: Kendall's tau distance

1. Introduction and Motivation

This investigation is a part of big project of developing clustering module for weighted sequences. As an example of such data we can suggest log of WEB site pages user opens with time, number clicks etc as characteristics of each state. Another data example (less obvious, but it is a real data we use) is set of medical treatments, provided in hospitals and polyclinics: sequence of medical treatments, which were provided to patient with a diagnosis during some fixed period of time. The main problem we try to solve is a development of system, which help specialists to analyze such sequences. One of the tools we need to implement is clustering module.

The main problem of research is a distance function between such complex-structured data. We need to take into account:

- a set of objects (e.g. medical treatments);
- parameters of objects;
- order of objects;

We start to making our own distance based on Levenshtein (it can be easily modified for our purpose), but decide to test new distance on each step to make sure, that our new distance is good enough in comparison with other distances. This

paper consider first step of our research: comparison Levenshtein distance with another distances for partial orders. Partial order is simplest example of weighted sequences: there are no repeated objects and no weights.

So this paper considers the problem of clustering partial orders as a part of problem mentioned above. Since the problem of clustering orders does not differ much from the problem of clustering any set of objects we focused on distance function between objects of clustering. Comparison of partial orders obviously is quite difficult problem because if we compare two of them we need to take into account not only set of elements, but in addition an order of them. Despite complexity and interest of this theme it has surprisingly little work has been done.

We decide to compare Levenshtein distance as a function of similarity between partial orders and compare it with a recently presented approach proposed in [1] and well-known Kendall tau rank distance [5] to find out their performance in different circumstance.

2. Definitions and Problem Statement

According to [1] chain is a "totally ordered subsets of a set of items, meaning that for all items that belong to a chain we know the order, and for items not belonging to the chain the order is unknown". Hence every chain can't include one object more than one time. As an example of such data we can suggest a rating of some objects (films, music compositions etc). More precisely, when we talk about clustering chains we assume, that full data set of chains was generated from some total orders. We want to make such clusters, where all chains in one clusters were generated by one total order.

For our analysis we use Lloyd's algorithm, also known as k-means, which is one of the most common clustering algorithms and the k-medoids algorithm, which is a medoidshift clustering algorithm related to the k-means. Both the k-means and k-medoids algorithms are partional (breaking the dataset up into groups) and both attempt to minimize the distance between points labeled to be in a cluster and a point designated as the center of that cluster. In contrast to the k-means algorithm, k-medoids chooses datapoints as centers (medoids or exemplars) and works with an arbitrary matrix of distances between datapoints [2]. We use two different algorithms in depend on distance function and ability to calculate mean value.

3. Distance Algorithms

As we mentioned above clustering algorithms themselves does not differ much for different objects, but the distance function highly depends on data we want to analyze. So we focused on distance function between partial orders and implement Levenshtein distance function to calculate distance between them. We also try to compare three distance functions: vectorizing algorithm presented in [1] (Ukkonen distance), Kendall's tau rank distance and our implementation of Levenshtein distance [3].

3.1. Ukkonen Distance

There were a number of different distances between partial orders in [1]. For analysis we choose planted partion model, which is very interesting first of all because it help to vectorize partial orders. It doesn't compare two orders directly, but firstly vectorize them and then use ordinary mathematical distances (Cosine, Euclidian or any other). Additionally it is very simple from computational point of view: it needs just O(nm) to compute vectors for *n* partial orders, when size of total order is m.

The main idea of planted partion model is next. A function *f* that maps total orders to R^m as follows: let τ be a total order on M, and let $\tau(u)$ denote the position of $u \in M$ in τ. Consider the vector f_τ where

$$f_\tau(u) = -\frac{m+1}{2} + \tau(u)$$

If partial orders are shorter than total order we need to take into account cases, when element from total order not exist in partial order (is not ranked). So if π is a partial order and u - one of the elements of M:

$$f_\pi(u) = \begin{cases} -\frac{|\pi+1|}{2} + \pi(u) & \text{iff } u \in \pi \\ 0 & \text{iff } u \notin \pi \end{cases}$$

And after normalization of function we get:

$$f(n) = {f_\pi}/{\|f_\pi\|}$$

After this vectorization procedure we can use any of classical distances between objects, for example, cosine distance which we use in this work. Using this distance we can use k-means algorithm, because we can easily calculate mean value of number of partial orders.

3.2 Levenshtein Distance

In information theory and computer science, the Levenshtein distance is a string metric for measuring the difference between two sequences. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (insertion, deletion, substitution) required to change one word into the other.

If we think about total orders as an alphabet, partial orders as a words and elements of order as a letter we can draw full analogy from distance between partial orders to distance between words:

$$\mathrm{Lev}_{\pi,\pi'}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0 \\ \min \begin{cases} Lev_{\pi,\pi'}(i,j-1)+1 \\ Lev_{\pi,\pi'}(i-1,j)+1 \\ Lev_{\pi,\pi'}(i-1,j-1)+2[\pi(i) \neq \pi'(j)] \end{cases} & , else \end{cases}$$

In this case we cannot use k-means algorithm, because mean value of partial orders is not defined, so we need to use k-medoids clustering algorithm.

3.3 Kendall's Tau Rank Distance

The Kendall tau rank distance is a metric that counts the number of pairwise disagreements between two ranking lists. The larger the distance, the more dissimilar the two lists are. The main problem is that if the chains π_1 and π_2 have no items in common, we have to use a fixed distance between π_1 and π_2. For example it was made for Spearmen's rho by [4]. We can use the same approach also with the Kendall distance by defining the distance between the chains π_1 and π_2 as the (normalized) Kendall distance between the permutations that are induced by the common items in π_1 and π_2. If there are no common items we set the distance to 0.5.

The Kendall tau ranking distance between two lists L_1 and L_2 is

$$K(\tau_1,\tau_2) = \left|\left\{(i,j): i<j, \left(\tau_1(i)<\tau_1(j) \wedge \tau_2(i)>\tau_2(j)\right) \vee \left(\tau_1(i)>\tau_1(j) \wedge \tau_2(i)<\tau_2(j)\right)\right\}\right|$$

where τ_1 and τ_2 are the rankings of the elements in L_1 and L_2.

4. Experiments and Results

For testing these distance functions we produce a number of clusterizations and evaluate results of clustering. We assume that quality of clusters is strongly correlated to quality of distance functions. Data we use for clustering was artificial: we generate a number of partial orders from three total orders. So we have an opportunity to use Adjusted Rand Index as a measure of quality of clustering [6,7]. For testing we make Python program in which implement K-means clustering algorithm with Ukkonen distance function, K-medoids algorithm with Kendall's tau distance and K-medoids algorithm with Levenshtein distance.

First thing we want to test is how the quality of clustering depends on fraction of items ranked. It was predictable that the bigger fraction is the easier it is to distinguish them from each other, so we produce a number of test with different fraction of items ranked. We assume that all partial orders are the same length. Results of multiple clustering tests with different number of items ranked and different number of items in total order are in fig.1

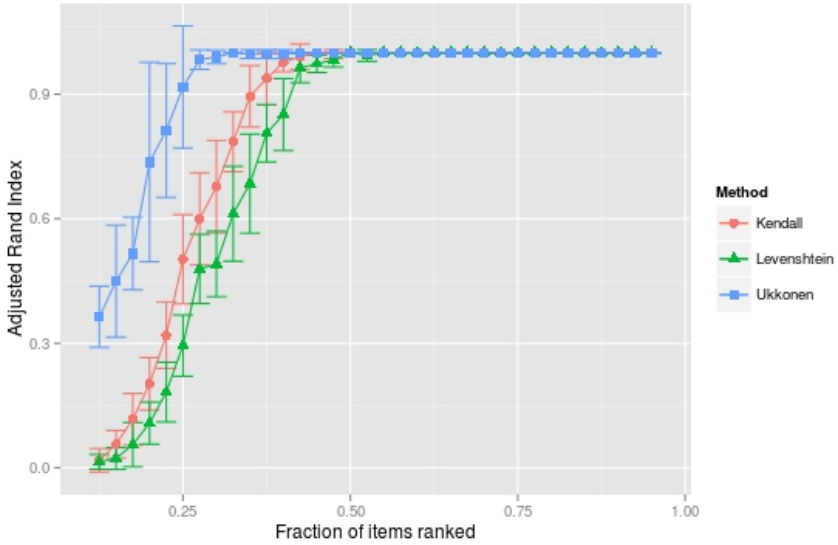

Fig. 1. Quality of clustering in depending on fraction of items ranked (all partial orders has the same length)

We can see, that if number of items ranked is equal to number of elements in total order (in other words, all elements of total order are in partial order) all three algorithms are quite good, but when partial orders are very little all of them cannot perform well.

In previous test we assume that all partial orders are of equal length. Next test helps us to define quality of distance functions in case of comparison of partial orders with different length. We want to understand if distance function can correctly compare partial orders with different number of elements. So the idea of experiment was the next one. We assume that length of chain is a random value generated by normal distribution with some mean value and some variance. The mean value is not so important in this test, because the main idea is to understand dependency of clustering quality on variance of partial orders length, so it was fixed for all experiments. Accordingly to this assumption we generate partial orders with different lengths (from normal distributions with same mean value and different variance). For each variance we evaluate Adjusted Rand Index. Results are in fig.2.

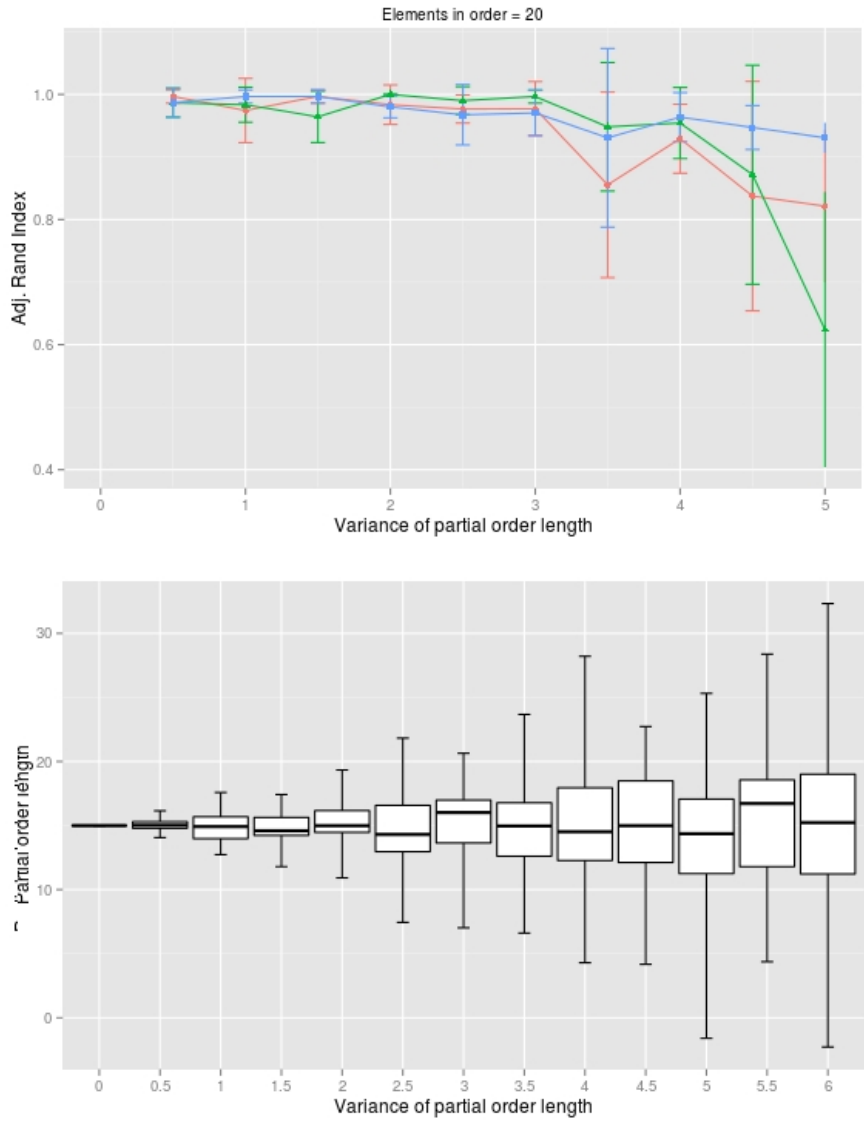

Fig. 2. Quality of clustering in depending on variance of number of items ranked.

All algorithms decreased their quality with increasing variance of number of items ranked, but we want to emphasize, that variance of clustering quality with Kendall distance significantly increase in comparison with Levenshtein and Ukkonen distances.

5. Conclusion

We find out that using Ukkonen distance help to achieve more stable results with higher quality than Levenshtein and Kendall distances. Levenshtein distance is relatively good when we take into account partial orders with the same number of elements in them. But quality of clustering process decreased with increasing variance of number of items ranked.

Kendall's tau distance get stable result with quality close to Levenshtein distance, but there is no reasonable way to modify this distance to compare weighted sequences.

We do not consider that fact in paper, but we cannot to ignore that fact that Ukkonen distance showing great promise property: we can vectorize (and in some cases vizualize) partial orders using this algorithm while Levenshtein distance is applied directly to partial orders and all problems of vizualization. Another good property is the computational complexity of the algorithm: we can vectorize n objects in O(nm), when the size of the total order is m and use after that simple functions to get distances. The main problem of such approach is necessity to know size of full order, while other distance functions has no need in such information.

References

[1]. A. Ukkonen. Clustering algorithms for chains. Journal of Machine Learning Research, 12:1389–1423, 2011.

[2]. L. Kaufman, P. Rousseeuw. Clustering by means of medoids. In Dodge, Y. (Ed.) Statistical Data Analysis based on the L1 Norm. Elsevier/North Holland, Amsterdam, 1987, pp. 405-416.

[3]. V.I. Levenshein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. Soviet Physics Doklady. Vol. 10, No. 8, 1966, pp. 707-710

[4]. T. Kamishima, J. Fujiki. Clustering orders. Proceedings of the 6th International Conference on Discovery Science, 2003, pp.194-207

[5]. M. Kendall, J.D. Gibbons. Rank Correlation Methods. A Charles Griffin Title, 1990. 272 p.

[6]. W.M. Rand. Objective criteria for the evaluation of clustering methods. Journal of the American Statistical association. Vol. 66, No. 336, 1971, pp. 846-850

[7]. L. Hubert, A. Phipps. Comparing partitions. Journal of Classification. Vol. 2, No. 1, pp.193-218

Сравнение методик кластеризации частично упорядоченных множеств

А.А. Раскин <a.a.raskin@gmail.com>
Национальный исследовательский ядерный университет «МИФИ», Каширское ш., 31, Москва, 115409

Аннотация. В статье предлагается сравнение трех подходов к кластеризации частично упорядоченных множеств. Первый подход заключается в применение алгоритма кластеризации k-medoids с использованием расстояния Левенштейна. В качестве второго подхода рассматривается векторизация частично упорядоченных множеств с дальнейшей кластеризацией с помощью алгоритма k-means и косинусного расстояния в качестве функции расстояния между объектами. Последним рассматриваемым подходом является кластеризация с помощью алгоритма k-medoids и коэффициента ранговой корреляции Кендалла в качестве функции расстояния. Для оценки качества кластеризации мы использовали Adjusted Rand Index и определили, что кластеризация с использованием всех трех подходов дает стабильный результат даже в тех случаях, когда количество элементов в кластеризуемых множествах существенно различается. В случаях, когда доля ранжированных элементов мала, наилучшие результаты показывает метод векторизации частично упорядоченных множеств.

Ключевые слова: Расстояние Левенштейна; частично упорядоченные множества; кластеризация; меры близости; коэффициент корреляции Кендалла.

Time invariant hand gesture recognition for human-computer interaction

D. Kostyrev <dmitry.kostyrev@gmail.com>
S. Anischenko <sergey.anishenko@gmail.com>
M. Petrushan <drn@bk.ru>
A.B. Kogan Research Institute for Neurocybernetics,
Stachki av. 194/1, Rostov-on-Don, Russian Federation.

Abstract. Hand motion driven human-computer interface based on novel time-invariant gesture description is proposed. Description is represented as a sequence of overthreshold motion distribution histograms. Such description utilizes information about gesture spatial configuration and motion dynamics. K-nearest-neighbour classifier was trained on six gesture types. Application for remote slideshow control was developed based on the proposed algorithm.

Keywords: human-computer interfaces, hand motion tracking, dynamic pattern recognition

1. Introduction

Popularity of natural interfaces for desktop and mobile computer control has been rapidly growing within last decade. Nowadays common human-computer interfaces (like keyboard) are gradually replaced by natural control interfaces based on gesture-driven, voice-driven, finger or full body motion driven control. These new methods are widely used in entertainment applications or in such fields, where a contact between human and input device is impossible or unwanted because of sterility requirement or in case, when a device have to be controlled by a group of people simultaneously.

Hand motion recognition task is concerned with several fundamental computer vision problems, in particular with the problems of dynamic patterns detection and recognition. The standard pipeline for single image analysis is represented as the sequence of procedures: preprocessing - segmentation - classification. This pipeline is admissible for video analysis only if the task is to detect and classify the objects, which movements are not of value. Otherwise, additional information about object movement or transformation has to be considered. In contrast to single image, video contains such additional information, that has to be utilized for object detection and recognition.

The main goal of our project is to develop the robust descriptor for the dynamic objects, invariant to object deformations and perspective transformations during a movement. In order to develop such descriptor the modifications of the standard single image analysis pipeline are proposed. The new dynamic gesture recognition method is described below. It utilizes information about duration, direction and amplitude of a motion along with spatial and intensity-based feature descriptions of images in video sequence. This method was used to develop the human-computer interface and application for presentation remote control.

2. Research background

Gesture-based human-computer interfaces can utilize hand stationary configuration (configuration is relevant), like "open palm" or "thumb up", hand motion (dynamics is relevant), like "from palm to fist" motion, "hands up" motion, etc. A variety of gesture recognition algorithms was developed, that can be divided into two groups according to configuration or motion relevance:

- single image analysis algorithms, that detect and recognize hand configuration in each frame of video;
- image sequences analysis algorithms, that detects hand configuration changing pattern for whole gesture video sequence.

Single image analysis pipeline for gesture recognition is similar to commonly used analysis procedures: preprocessing, segmentation, classification. Following steps for gesture analysis were proposed [1]: hand contour extraction, tracking and recognition based on selected features. The image sequence analysis pipeline differs from single image approach and contains following steps: background subtraction, description of a gesture and classification. In spite of the fact that every gesture detection and recognition method is unique, combining different approaches and algorithms, there are some common steps used in most methods, such as background subtraction, features extraction and classification of a gesture.

Gesture detection and recognition methods use different background subtraction algorithms from very simple like frame difference [2 - 4] to more complex methods such as Adaptive Mixture of Gaussians [5, 6] and frame difference enhanced with Gaussian filter [7].

Various feature extraction methods for gesture recognition have been described in [3, 7, 8, 9]. Methods based on calculation of histograms of oriented gradients (HOG) are used in [9] and [3]. In [3] it is used for features extraction from a motion history image which is created from several frames in sequence. Fourier analysis based methods are used for different kind of movements in [7]. Hand shapes within single image analysis approach can be described by shape context descriptor [8]. Gesture classification is performed by different algorithms based on fuzzy logic [8], SVM for large feature vector (3780 elements) [9], Euclidian distances [3], and spectrum analysis [7].

Each above-mentioned gesture analysis method has its own advantages and disadvantages. For instance, single image based analysis provides precise estimation of hand location in image but it is limited with fixed hand configuration because of non-rigid nature of a human hand. On the other hand image sequence based methods are invariant to hand configuration in common but are dependent on gesture duration and completeness. Thus, gesture analysis methods have its limitations, for example, most of the methods based on skin color segmentation [10 - 11] or background subtraction methods [2-4, 7, 9] are highly dependent on scene light conditions, quality of camera sensor, etc. Along with recognition quality the method processing speed is one of the key factors in a sense of end user experience, so it has to be taken into account when comparing gesture recognition methods.

3. Gesture detection and recognition

The new method of a hand gestures detection and recognition is presented. Following requirements were set during problem formulation. According to them the method must be invariant to gesture duration, hand initial position and be able to detect and recognize transformable hand configurations.

According to the requirements and research background this method should describe a gesture in terms of integral motion characteristics. Algorithm workflow scheme is presented in fig. 1.

According to this scheme two main components can be highlighted in the workflow of this algorithm: background subtraction and features extraction.

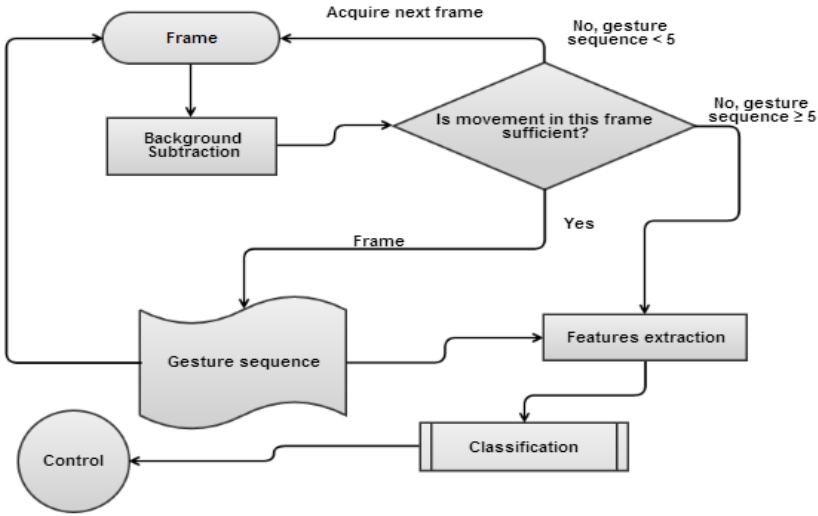

Fig. 1. The workflow of the algorithm.

3.1 Background subtraction

Background subtraction is used for gesture duration estimation and as a preprocessing step for gesture description in our approach. Following requirements for background subtraction method were formulated:

- object contours estimation;
- no contour traces;
- real time performance.

Several popular background subtraction methods were reviewed within the research. Each of them was evaluated according to the following parameters: performance, contours continuity, length of contour traces. Background subtraction quality was evaluated qualitatively and algorithm performance was estimated according to the video processing frame rate. Algorithm performance is considered "realtime" with $framerate \geqslant 30$. All algorithms (except ViBe) were evaluated according to the current implementation in BGSLibrary [12] on 3th generation Intel Core i5 processor powered PC. Results of performance and quality estimations are presented in the table 1.

Table. 1 Overview of background subtraction methods.

Algrorithm	Performance	Contours continuity	Length of traces
Frame difference	realtime	low	no
Moving mean	realtime	low	no
Adaptive Mixture of Gaussians	realtime	low	short
Gaussian Average	realtime	high	short
Multi - Layer BGS	offline	low	no
Fuzzy Gaussian	realtime	high	long
Fussy Adaptive Som	offline	high	long
ViBe (serial)	offline	high	no

Reviewed background subtraction methods are disbalanced in a sense of sufficient quality, performance and traces absence. Thus, a new background subtraction algorithm is presented which should combine performance and contours continuity. It is based on the computation of time-dependent intensity variance map between N frames as follows:

$$D(x,y) = \frac{\sum_i^N p_i\,(x,y)^2}{N} - \left(\frac{\sum_i^N p_i\,(x,y)}{N}\right)^2 \quad (1)$$

$$B(x,y) = \begin{cases} 255, if D(x,y) \geq Threshold^2 \\ 0, otherwise \end{cases} \quad (2)$$

where $D(x,y)$ - resulting variance map, p_i - i_{th} frame, N - number of frames in sequence, x, y - pixel coordinates, $Threshold$ - binarization threshold.

Presented background subtraction algorithm demonstrates realtime performance, sufficient quality of contour separation without long traces. However, this method is depended on light conditions. Parameter *N* regulates background model update speed. Binarized variance maps of two adjacent frames with different *N* is presented in the fig. 2.

Binarized variance map *B* is prepared by median filtering for noise reduction and nearby contours merging.

Current frame motion rate is estimated by counting all non-zero elements in the binarized variance map. Motion is considered to be a gesture candidate when the following condition is met:

$$\sum_x^{\omega} \sum_y^{h} B\,(x,y) \geq w * h * MotionThreshold \quad (3)$$

where x, y - pixel coordinates in binarized variance map B, w, h - width and height of variance map accordingly, $MotionThreshold$ - threshold for estimating a gesture candidate in the frame.

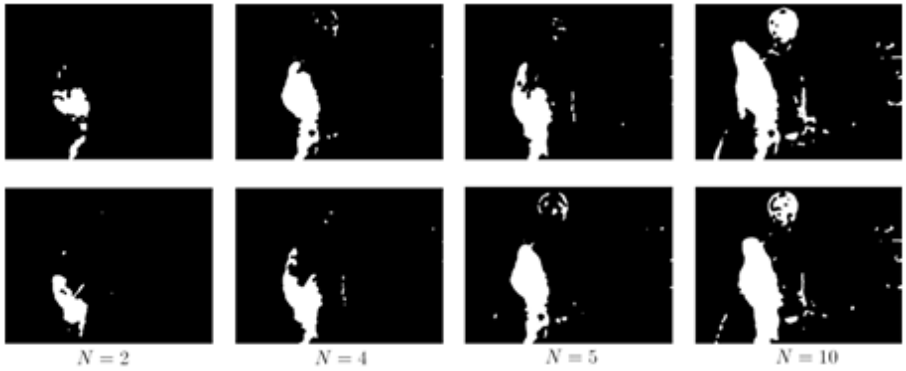

Fig. 2. Binarized variance maps of two adjacent frames with different N.

If the current frame satisfies the above-mentioned condition it is being added to the current gesture sequence. Current gesture sequence is considered complete when number of frames in sequence $\geq N_{min}$ and the condition (eq. 3) was not met in the $N_{min} + 1$ frame. As soon as the gesture sequence is considered complete it is being described and classified.

3.2 Gesture description and classification

Any gesture sequence, containing more than N_{min} frames, can be described by a feature vector. Motion Distribution Histogram is proposed to be the integral description of the frame in gesture sequence. It describes a distribution of overthreshold variance (eq. 3) over frame. Each motion distribution histogram is calculated according to the mean center of masses for all variance maps in the sequence, thus, we can achieve initial hand position invariance. Variance maps are being divided into 16 sectors in which all non-zero elements are counted and stored in corresponding element of motion distribution histogram. Sectors are numbered clockwise from horizontal axis orientation. Each element of motion distribution histogram is normalized according to the area of binarized variance map of the frame. Examples of variance maps and corresponding motion distribution histograms are shown in the fig. 3.

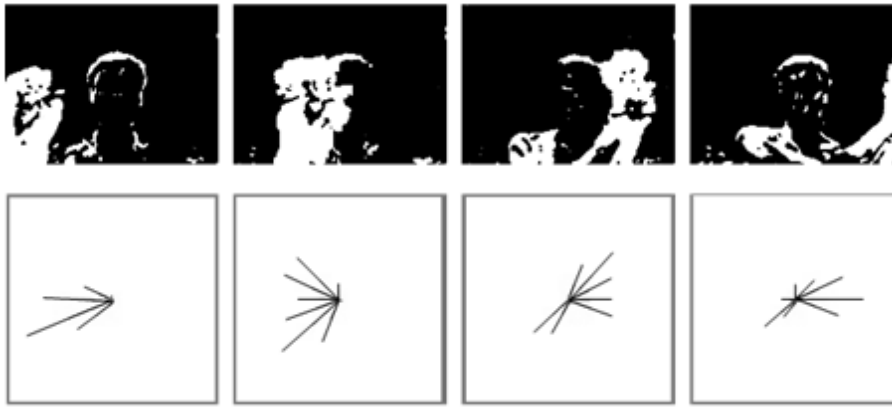

Fig. 3. Binarized variance maps for different frames of "from left to right" gesture sequence and corresponding motion distribution histograms.

Variance map and motion distribution histogram calculated for all frames in the sequence. Motion distribution histograms sequence containing N histograms are being divided into 4 subsequences according to the following intervals I:

$$I_1 = \left[1..\frac{N}{4}\right], I_2 = \left[\frac{N}{4}..\frac{N}{2}\right], I_3 = \left[\frac{N}{2}..\frac{3N}{4}\right], I_4 = \left[\frac{3N}{4}..N\right] \qquad (4)$$

where $N$is the number of histograms in the sequence.

4. Experimental results

Proposed method has advantages and disadvantages comparing to the methods that are based on object detection and recognition in each frame of video. The main advantages of this approach are invariance to hand configuration transformations during the movement, invariance to the speed of the gesture and hand initial position in the camera field of view. However, this method is highly depended on background movement (which can be eliminated with depth map), distance between human and camera and lighting conditions.

In our approach we used $N_{min} = 5$, $MotionThreshold = 0.08$ (eq. 3) for motion detection in close range from the camera (approximately 50 cm), $N = 2$(eq. 2) and $Threshold = 2$ (eq. 2).

Six gesture types were selected for detection and recognition: hand movement from left to right, right to left, hand up, hand down, both hands from the center of the screen and both hands to the center of the screen. Training set containing samples of each gesture type was collected. The set containing seven examples of 2 classes (hands down and hands up) are displayed in the fig. 4, where "hand down" movement is visualized with black colour, and "hands up" movement is visualized with gray colour.

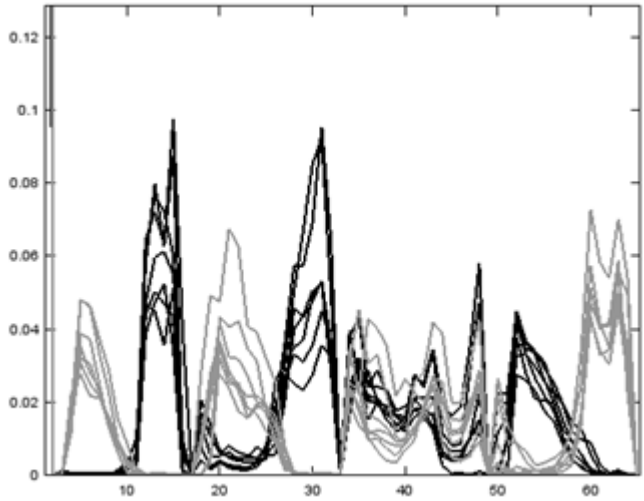

Fig. 4. Plot of 14 histograms containing samples from two classes: "hand up" and "hand down". Black colour corresponds to "hands down" movement, gray colour corresponds to "hans up" movement. Feature vector element numbers displayed on horizontal axis and values on vertical axis.

Classification is based on k-nearest neighbours method with Euclidian distance. Cross-class similarity estimation was performed by calculating mean Euclidian distances and standard deviation between descriptions of each class compared to another classes in the fig. 5.

5. Application

The new human-computer interface was implemented based on the gesture recognition method described above. The application for hand driven slideshow control was developed as an example of this interface.

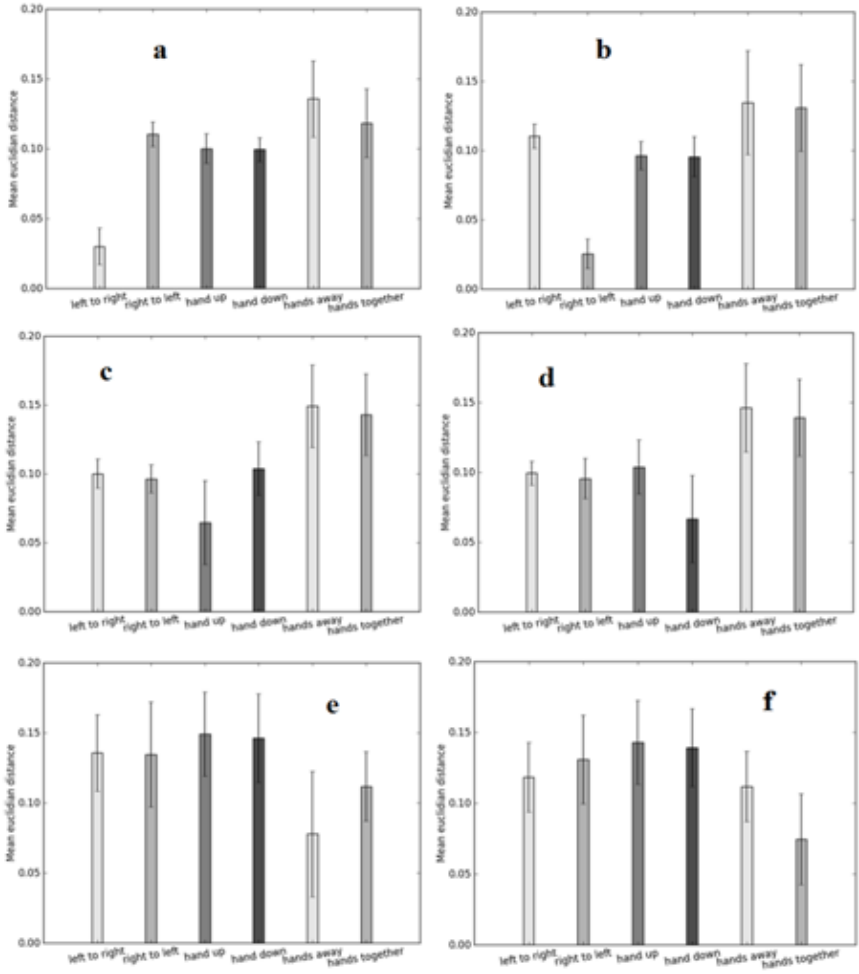

Fig. 5. Mean Euclidian distances and standard deviation (error bars) between feature vectors of each class compared to another classes: a - "left to right" gesture, b - "right to left", c - "hand up", d - "hand down", e - "hands away", f - "hands together".

The client – server application architecture was proposed to achieve remote slideshow control. The architecture of the demonstration application is presented in the fig. 6.

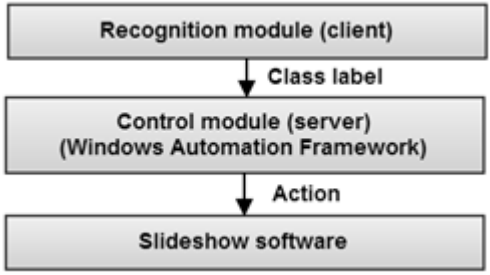

Fig. 6. The architecture of the application.

Conclusion

Hand motion guided human-computer interface based on the new dynamic patterns descriptor is presented. The distinctiveness of proposed gesture description was demonstrated by cross-class Euclidian distance measurement of training samples. Hand motion is described by the sequence of motion distribution histograms. This method demonstrates sufficient processing speed in terms of end user experience and classification accuracy for gesture sequences to be used for remote slideshow control. Further research within proposed approach aims to support different gestures types and non-relevant objects motion filtering using skin color map, depth map and motion map.

Acknowledgments

The work is supported by SFedu project № 213.01-2014/001, Russian Foundation for Basic Research, grants 12-01-31226 mol_a and 12-01-31266 mol_a.

References

[1]. Rautaray S.S., Agrawal A. A real time hand tracking system for interactive applications. *International Journal of Computer Applications*. 2011, vol. 18, no 6, pp. 28-33.

[2]. Shan C., Tan T., Wei Y. Real – time hand tracking using a mean shift embedded particle filter. *Pattern Recognition.* 2007, vol. 40, no 7, pp. 1958-1970. doi: 10.1016/j.patcog.2006.12.012

[3]. Davis J. W. Recognizing Movement using Motion Histograms. *Technial Report 487, MIT Media Lab.* 1999. vol. 1, no 487. doi: 10.1.1.46.6887

[4]. Torres G. Gesture recognition using motion detection. *University of Kansas.* 2009.

[5]. Banerjee P., Sengupta S. Human motion detection and tracking for video surveillance. *Proceedings of the national Conference of Communications*, IIT Bombay, Mumbai. 2008, pp. 88-92

[6]. Stauffer C., Grimson W.E.L. Adaptive background mixture models for real-time tracking. *Computer Vision and Pattern Recognition. 1999. IEEE Computer Society Conference on.* IEEE. 1999, vol. 2. doi: 10.1109/CVPR.1999.784637

[7]. Cutler R., Davis L. Robust Real – time periodic motion detection, analysis, and applications. *Pattern Analysis and Machine Intelligence, IEEE Transactions on.* 2000, vol. 22, no. 8. pp. 781-796. doi: 10.1.1.112.8904

[8]. Mori G., Belongie S., Malik J. Efficient shape matching using shape contexts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on.* 2005, vol.. 27, no. 11. pp. 1832-1837. doi: 10.1109/TPAMI.2005.220

[9]. Fogelton A. Real-time Hand Tracking using Modificated Flocks of Features Algorithm. *Information Sciences and Technologies Bulletin of the ACM Slovakia, Special Section on Student Research in Informatics and Information Technologies.* 2011, vol. 3, no 2, pp. 37-41. doi: 10.1.1.295.2305

[10]. Manresa C., Varona J., Mas R. Perales F. J. Real – time hand tracking and gesture recognition for human – computer interaction. *Electronic Letters on Computer Vision and Image Analysis.* 2005, vol.. 5, no. 3, pp. 96-104.

[11]. Deng L.Y., Hung J.C., Keh H., Lin K., Liu Y., Huang N. Real – time hand gesture recognition by shape context based matching and cost matrix. *Journal of networks*. 2011, vol. 6, no 5, pp. 697-704. doi:10.4304/jnw.6.5.697-704

[12]. Sobral, Andrews. BGSLibrary: An OpenCV C++ Background Subtraction Library. *Proceedings of IX Workshop de Visao Computacional (WVC'2013).* 2013.

Метод инвариантного распознавания жестов для реализации человеко-компьютерного интерфейса

Д. В. Костырев <dmitry.kostyrev@gmail.com>
С. И. Анищенко <sergey.anishenko@gmail.com>
М. В. Петрушан <drn@bk.ru>
НИИ нейрокибернетики им. А.Б.Когана Академии биологии и биотехнологии Южного федерального университета

Аннотация. В данной статье представлен способ человеко-компьютерного взаимодействия с помощью жестов рук, основанный на новом способе описания жестов, инвариантном относительно длительности жеста. Описание представлено в виде последовательности сверхпороговых гистограмм распределения областей движения в поле зрения видеокамеры. Такой способ описания учитывает информацию о пространственной конфигурации жеста и динамики движения. В качестве классификатора использован метод k ближайших соседей. Для обучения классификатора были выбраны шесть типов жестов. На основе предложенного алгоритма было разработано демонстрационное приложение для удаленного управления показом презентаций.

Ключевые слова: человеко-компьютерное взаимодействие, распознавание жестов, распознавание динамических паттернов

References

[1]. Rautaray S.S., Agrawal A. A real time hand tracking system for interactive applications. *International Journal of Computer Applications*. 2011, vol. 18, no 6, pp. 28-33.
[2]. Shan C., Tan T., Wei Y. Real – time hand tracking using a mean shift embedded particle filter. *Pattern Recognition*. 2007, vol. 40, no 7, pp. 1958-1970. doi: 10.1016/j.patcog.2006.12.012
[3]. Davis J. W. Recognizing Movement using Motion Histograms. *Technial Report 487, MIT Media Lab*. 1999. vol. 1, no 487. doi: 10.1.1.46.6887
[4]. Torres G. Gesture recognition using motion detection. *University of Kansas*. 2009.
[5]. Banerjee P., Sengupta S. Human motion detection and tracking for video surveillance. *Proceedings of the national Conference of Communications*, IIT Bombay, Mumbai. 2008, pp. 88-92

[6]. Stauffer C., Grimson W.E.L. Adaptive background mixture models for real-time tracking. *Computer Vision and Pattern Recognition. 1999. IEEE Computer Society Conference on.* IEEE. 1999, vol. 2. doi: 10.1109/CVPR.1999.784637

[7]. Cutler R., Davis L. Robust Real – time periodic motion detection, analysis, and applications. *Pattern Analysis and Machine Intelligence, IEEE Transactions on.* 2000, vol. 22, no. 8. pp. 781-796. doi: 10.1.1.112.8904

[8]. Mori G., Belongie S., Malik J. Efficient shape matching using shape contexts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on.* 2005, vol.. 27, no. 11. pp. 1832-1837. doi: 10.1109/TPAMI.2005.220

[9]. Fogelton A. Real-time Hand Tracking using Modificated Flocks of Features Algorithm. *Information Sciences and Technologies Bulletin of the ACM Slovakia, Special Section on Student Research in Informatics and Information Technologies.* 2011, vol. 3, no 2, pp. 37-41. doi: 10.1.1.295.2305

[10]. Manresa C., Varona J., Mas R. Perales F. J. Real – time hand tracking and gesture recognition for human – computer interaction. *Electronic Letters on Computer Vision and Image Analysis.* 2005, vol.. 5, no. 3, pp. 96-104.

[11]. Deng L.Y., Hung J.C., Keh H., Lin K., Liu Y., Huang N. Real – time hand gesture recognition by shape context based matching and cost matrix. *Journal of networks*. 2011, vol. 6, no 5, pp. 697-704. doi:10.4304/jnw.6.5.697-704

[12]. Sobral, Andrews. BGSLibrary: An OpenCV C++ Background Subtraction Library. *Proceedings of IX Workshop de Visao Computacional (WVC'2013).* 2013.

Linguistic Approach to Suicide Detection

[1 2] *L. Ermakova <liana.ermakova.87@gmail,.com>*
[2] *S. Ermakov <choufler@gmail.com>*
[1] *Perm State National Research University,*
15 Bukireva Str., Perm, 614990, Russian Federation
[2] *Institut de Recherche en Informatique de Toulouse,*
118 Route de Narbonne , F-31062 Toulouse Cedex 9, France

Abstract. Suicide is a major, preventable public health problem. Particularly the problem is critical for young people. In Russia every year thousands of teenagers commit suicide. In most of the cases it can be prevented if a risky state is detected. Nowadays internet becomes a major way of communication, mainly in the text form. Therefore we suggest a method to detect a tendency to suicide based on text messages. Our main approach is to study indicators of such condition and based on it use machine learning approach to build a classifier that could determine, whether the person is about to commit a suicide. Our experiments are based on the analysis of texts of Russian writers for past 100 years that committed suicide.

Keywords: sentiment analysis, machine learning, suicide.

1. Introduction

Suicide poses a serious public health problem worldwide. Every year around 1 million deaths and 10 million attempts occurs due to that [1, 2]. In Russia this problem is particularly urgent among young people. According to official number provided by investigating committee, it was more than one thousand deaths in 2012 [3]. Around 10 thousands of teenagers attempt to commit suicide. This mainly happens due to their social disadaptation. For the past 50 years this number has grown for around 60% [4]. This problem also is present in the army, police and jails, and other places where people experience strong stresses. Suicide can be prevented and most people who feel suicidal demonstrate warning signs. The only problem is that these signs should be noticed. However, ordinary person could easily miss it. Thus, for the major part it is a duty of psychologists to recognize that dangerous behavior. However, the number of this kind of staff in schools is constantly reducing, e.g. in 2011-2012 school year, 70% of schools did not have psychologists at all (this is more 8% to the previous year). In the police and army there is one specialist for several thousand soldiers / policemen. There is an extremely urgent need in the tools that could help to analyze the emotional condition of big groups of people.

Internet communication and self-expression are becoming more and more popular. Almost every young person uses social networks, blogs and forums. The mean of

expression is mainly text. That gives us an idea that these text messages could give us a valuable source for automatic analysis in order to help solving suicide problem.

2. Techniques used to detect and prevent suicides

First step to prevent suicide is to detect the signs that indicate it. There are numerous studies on that topic [2, 4–7]. Most of the researches agree on the fact, that suicide is related to many emotions: hostility, despair, shame, guilt, dependency and hopelessness. The general psychological state commonly assumed to be associated with suicide is a state of intolerable emotion or unbearable despair.

The common indictors are:

- sadness;
- loss of interest or pleasure;
- disturbance in sleep and appetite;
- difficulty in concentrating;
- thoughts and talks of suicide;
- isolation;
- looking for a way to kill oneself;
- talking about being trapped or in unbearable pain;
- talking about being a burden to the others;
- talking about having no reason to live or feeling hopeless;
- displaying extreme mood swings;
- extreme behavior;
- exposure to suicidal behavior of others, such as family members or peers [4, 6].

The risk factors are:

- history of previous suicide attempts;
- family history of suicide;
- family violence;
- history of depression or other mental illness;
- alcohol or drug abuse;
- stressful life event or loss;
- exposure to the suicidal behavior of others;
- incarceration [4, 6, 8].

Person, who has set up his mind to commit suicide usually talks about this. Suicide is not always impulsive: people tend to prepare themselves to the suicide, according to the researches it could take around one year. During this period of time they give

some type of verbal or behavioral message about their ideations of intent to hurt themselves [4].

Thus, we could attempt to determine using text analysis:

- emotional state of the person;
- specific verbal suicidal signs.

3. Analysis based on machine learning

First part of the analysis is to analyze emotions expressed in the text. This is one of the applications of sentiment analysis task [9]. There are various approaches proposed by researchers in this field [9–12]. One of the traditional sentiment analysis methods is using machine-learning techniques to perform classification: Naïve Bayes, SVM, neural networks, etc. Most of classifiers use words, word combinations and their characteristics as features. In our research we decided to start with a most simple two-way classification: does the text express potential suicide or not. To perform classifications we used WEKA toolkit [13]. As a classifier, SVM was chosen as it proved to be one of the most efficient in sentiment classification [14]. We started from using word N-grams (unigrams, bigrams and trigrams) as features. Then in order to improve the efficiency of determining emotional state we proposed to use following advanced features:

- share of all punctuation signs in the size of the text;
- number of specific "mood" signs used: three dots, exclamation marks, question marks;
- average and maximum length of the words;
- share of each part of speech (obtained from MyStem annotation);
- various phonetic characteristics (e.g. individual letters, bigrams, trigrams and quantitative features like word length, maximum sequential consonant length in a word, etc.) [15].

The second part of analysis is about specific verbal signs that could indicate suicide. This part was also partly covered by words and word combinations as features, such as burden, death, pain, despair, etc.

4. Experiments

We classified individual texts into two classes: texts written by a person committed suicide and not.

All experiments were based on the manually retrieved corpus of Russian texts. Information about people who has committed suicide is not public. Therefore we had to choose people, who are well-known. The most accessible texts are those produced by writers, poets, critics, etc. The list of them was extracted from research

[16]. To make an experiment clear we chose only native speakers. We retrieved the list of 49 persons, excluding those whose suicide is doubtful. All texts were written in XIX – XXI c. As per study described above, we used texts, written at the time most close to death (last year of life). This corpus contained 469 texts (around 10 texts per person). We added the same number of texts, taken from the writers, who have not committed suicide, including texts about death. Hence, full corpus for experiments contained 938 texts. The examples of each category of texts are given below:

Example 1. *A text written by an author committed suicide (V. Mayakovsky).*

Как говорят - "инцидент исперчен", любовная лодка
разбилась о быт. Я с жизнью в расчете и не к чему
перечень взаимных болей, бед и обид. Счастливо
оставаться.
Владимир Маяковский.

Example 2. *A text written by a random author who did not committed suicide (A. Pushkin).*

Я памятник себе воздвиг нерукотворный,
К нему не зарастет народная тропа,
Вознесся выше он главою непокорной
Александрийского столпа.

Нет, весь я не умру — душа в заветной лире
Мой прах переживет и тленья убежит —
И славен буду я, доколь в подлунном мире
Жив будет хоть один пиит.

Example 3. *A text about death written by an author who did not committed suicide (A. Akunin).*

Ну все, пора, меня уже зовут.
Увидимся позднее – не мешайте:
Мне надо что-то вспомнить напоследок.
Но что? Но что?
Ума не приложу.
Все спуталися мысли. Все, пора.
Что будет за последним окоемом,
Спешу узнать.

Вперед!
Царевич Смерть,
Приди в кроваво-красном облаченьи,
Подай мне руку, выведи на свет.
Где буду я стоять, простерши руки,
Как ангел, как судьба, как отраженье
Себя самой.
Другого – не дано.

For a baseline we chose word unigrams as features.

Before classification all the words in the texts were stemmed using MyStem application [17].

There are several methods used to evaluate the models. The basic ones are

- precision,
- recall,

where $precision = \frac{true\ positive}{true\ positive + false\ positive}$ $recall = \frac{true\ positive}{true\ positive + false\ negative}$

F-measure, which is a harmonic mean between precision and recall

$$F1 = 2 * \frac{precision * recall}{precison + recall}$$

Receiver Operating Characteristic (ROC-Area) [18], which is an area under the curve of tradeoff between true positive and false positive. We used F1 and Roc-Area to compare results of the runs.

Our experiments consisted of build several models with different feature set and perform 10 folds cross-validation using WEKA package.

We evaluated the following models:

- only unigrams (baseline);
- combination of unigrams and bigrams;
- combination of unigrams, bigrams, trigrams;
- combination of unigrams, bigrams, trigrams and set of quantitative features;
- full model (combination of unigrams, bigrams, trigrams and set of quantitative features and phonetic features).

The last model showed the best results, all the results are presented in the Table 1

Table 1. *Two-way classification results for Russian (SVM, 10 folds cross-validation)*

Feature set	Class	Precision	Recall	F-measure	ROC Area
Unigrams	Positive	0.633	0.671	0.651	0.607
	Negative	0.641	0.677	0.659	0.607
Unigrams + bigrams	Positive	0.642	0.691	0.666	0.621
	Negative	0.663	0.699	0.680	0.621
N-grams	Positive	0.653	0.710	0.680	0.693
	Negative	0.671	0.722	0.696	0.693
N-grams and quantitative features	Positive	0.779	0.792	0.785	0.789
	Negative	0.783	0.804	0.793	0.789
Full combination of features	Positive	0.805	0.821	0.813	0.819
	Negative	0.817	0.834	0.825	0.819

Compared to baseline results (unigrams) we increased our F-measure average value from 0.65 to 0.82, ROC-area increased from 0.6 to 0.819. Though our task was not purely sentiment classification, we compared our results with recent researches in two-way sentiment classification, which is about 0.8-0.85 [19]. We can conclude that our best effort showed satisfactory results.

5. Conclusion and Future Work

In this paper, we presented an approach to suicide detection in texts using linguistic characteristics and machine learning. With the model we built we obtained F-measure 0.82 with the ROC-area 0.819. We observed that using proposed feature set we were able to improve almost for 20% compared to the baseline, and 13% compared to the N-grams model. This shows, that in our area not only words are important, but also the punctuation signs, word lengths, and phonetic characteristics. We used only two-way classification, which means that if text is detected as positive, this directly indicates, that person is in trouble. However, psychologists determine several levels of threat, and the lowest of them could contain a lot of false-positives, meaning that low level of signs could be normal for the most of the people. In the future work we would like to split two classes to 5, but this would require an assessment of texts made by professional psychologists, to determine a

real level of danger. We also will try our approach on English texts, which will require manual collection like it was done for Russian.

As for the Russian language we try to cooperate with the medical organization to retrieve more texts for training our model, and also possibly to determine other features for the model, that are not obvious to a non-professional psychologists.

Our final goal is to build software that will help to monitor emotional state of people in order to prevent tragedies.

References

[1]. Wasserman, D., Cheng, Q., Jiang, G.X.: Global suicide rates among young people aged 15-19. World psychiatry. 4, pp. 114–120 (2005).

[2]. Paci, W.R.O.W. Towards Evidence-Based Suicide Prevention Programmes. World Health Organization, Western Pacific Region (2010).

[3]. SK: V Rossii rezko vozroslo chislo detskikh samoubijstv [Committee of inquiry: The number of children suicide dramatically increased in Russia], http//www.rg.ru/2012/10/24/deti.html. (in Russian)

[4]. World Health Organization: Preventing suicide : a resource for counsellors.

[5]. Gomez, D.D., Blasco-Fontecilla, H., Sukno, F., Ramos-Plasencia, M.S., Baca-Garcia, E. Suicide attempters classification: Toward predictive models of suicidal behavior. Neurocomputing. 92, pp. 3–8 (2012).

[6]. Early detection of depression and prevention of suicide, http://www.philstar.com/science-and-technology/147031/early-detection-depression-and-prevention-suicide.

[7]. Hjelmeland, H. Cultural Context Is Crucial in Suicide Research and Prevention, (2011).

[8]. Kostin, R.A., Shamkova, S.V. Suitsidy v molodezhnoj srede Sankt-Peterburga: sotsiologicheskij analiz [Suicide among the young people of Saint Petersburg: sociological analysis]. Sankt-Peterburgskij gos. universitet servisa i ehkonomiki [Saint-Petersburg state university of service and economy] (2007). (in Russian)

[9]. Pang, B., Lee, L. Opinion Mining and Sentiment Analysis. Foundations and Trends in Information Retrieval. 2, pp. 1–135 (2008).

[10]. Jia, L., Yu, C.T., Meng, W. Improve the effectiveness of the opinion retrieval and opinion polarity classification. In: Shanahan, J.G., Amer-Yahia, S., Manolescu, I., Zhang, Y., Evans, D.A., Kolcz, A., Choi, K.-S., and Chowdhury, A. (eds.) CIKM. pp. 1415–1416. ACM (2008).

[11]. Eguchi, K., Lavrenko, V. Sentiment Retrieval using Generative Models. In: Jurafsky, D. and Gaussier, É. (eds.) EMNLP. pp. 345–354. ACL (2006).

[12]. Mukherjee, S., Bhattacharyya, P. Feature Specific Sentiment Analysis for Product Reviews. CoRR. abs/1209.2352, (2012).

[13]. Witten, I.H., Frank, E., Hall, M.A. Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, Amsterdam (2011).

[14]. Chetviorkin, I.I., Loukachevitch, N.V. Sentiment analysis track at ROMI P 2012, (2013).

[15]. Ermakov, S., Ermakova, L. Sentiment Classification Based on Phonetic Characteristics. In: Serdyukov, P., Braslavski, P., Kuznetsov, S., Kamps, J., Rüger, S., Agichtein, E., Segalovich, I., and Yilmaz, E. (eds.) Advances in Information Retrieval. pp. 706–709. Springer Berlin Heidelberg (2013).

[16]. Chkhartishvili, G. Pisatel' i samoubijstvo.[Writers and suicide]. (2007). (in Russian)

[17]. Segalovich, I. A fast morphological algorithm with unknown word guessing induced by a dictionary for a web search engine. Proceedings of MLMTA. (2003).

[18]. Powers, D.M.W. Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation. School of Informatics and Engineering, Flinders University, Adelaide, Australia (2007).

[19]. Piao, S., Tsuruoka, Y., Ananiadou, S. Sentiment Analysis with Knowledge Resource and NLP Tools. The International Journal of Interdisciplinary Social Sciences. 4, pp. 17–28 (2012).

Лингвистический подход к определению суицида

[1 2] *Л. М. Ермакова <liana.ermakova.87@gmail,.com>*
[2] *С. А.Ермаков <choufler@gmail.com>*
[1] *Пермский государственный национальный исследовательский университет, ул. Букирева, 15, Пермь, 614990, Россия*
[2] *Institut de Recherche en Informatique de Toulouse, 118 Route de Narbonne , F-31062 Toulouse Cedex 9, France*

Аннотация. Суицид является одной из главных предотвращаемых проблем в здравоохранении. В особенности остро проблема стоит для молодых людей: в России каждый год тысячи подростков совершают самоубийство. В большинстве случаев беду можно предотвратить, если вовремя выявить опасное психологическое состояние. Интернет в наши дни становится основным способом общения, которое по большей части происходит в текстовой форме, поэтому в статье мы предлагаем метод определения склонности к суициду, основанный на анализе текстовых сообщений. Наш подход главным образом заключается в изучении показателей такого состояния и использования их для построения классификатора, который может определить насколько вероятно, что человек собирается совершить суицид. Для экспериментов мы использовали тексты русских писателей, которые совершили самоубийство за последние 100 лет.

Ключевые слова: анализ эмоциональной окраски текста, машинное обучение, суицид.

Ranking in keyphrase extraction problem: is it suitable to use statistics of words occurrences?

[1,2] *S. V. Popova <svp@list.ru>*
[2] *I. A. Khodyrev <kivan.mih@gmail.com>*
[1] *Saint-Petersburg State University,*
7-9, Universitetskaya nab., St. Petersburg, 199034, Russia
[2] *ITMO University,*
49, Kronverksky Ave., St.Petersburg, 197101, Russia

Abstract. The paper deals with keyphrase extraction problem for single documents, e.g. scientific abstracts. Keyphrase extraction task is important and its results could be used in a variety of applications: data indexing, clustering and classification of documents, meta-information extraction, automatic ontologies creation etc. In the paper we discuss an approach to keyphrase extraction, its' first step is building of candidate phrases which are then ranked and the best are selected as keyphrases. The paper is focused on the evaluation of weighting approaches to candidate phrases in the unsupervised ex-traction methods. A number of in-phrase word weighting procedures is evaluated. Unsuitable approaches to weighting are identified. Testing of some approaches shows their equivalence as applied to keyphrase extraction. A feature, which allows to increase the quality of extracted keyphrases and shows better results in comparison to the state of the art, is proposed. Experiments are based on Inspec dataset.

Keywords: keyphrase extraction; keyphrase ranking; statistical features for keyphrase ranking; information extraction; scientific abstracts processing.

1. Introduction

The paper deals with the keyphrase extraction problem for single documents. We define keyphrase as a word or a group of words, which reflects the domain-specific of the text Keyphrase extraction could be used further in different natural language processing applications such as data indexing [1], clustering documents [2-4], automatic ontology creation etc. We are using results of this paper in an academic search system [4], we are mainly interested in a keyphrase extraction task from abstracts of scientific papers, because most abstracts are freely available and texts of papers are usually not. We focus on analysis of approaches to keyphrase selection from a set of candidates, built for a document [5-8]. The weighted approach is used to evaluate

quality of a particular candidate, then after the ranking procedure, the best candidates are selected as keyphrases. In the paper we use only statistical information related to the word frequency in single documents and in a document collection. It is also shown that a number of measures is not adequate and some other measures are almost equivalent. We have shown that usage of some measure estimated by researchers as suitable, in reality leads to the situation where measured phrases are selected almost randomly and thus such measures could be considered equivalent for the annotation task. The novel feature which is proposed in the paper, is based on the exclusion of one-word phrases from candidates,that increases significantly the an-notation quality. The reminder of the paper is organized as follows. Section 2 is dedicated to the state of the art. In Section 3 experiment is described and description of test collection is provided. In section 4 the experiment's results are presented and discussed. In Section 5 additional experiment and its results are presented and dis-cussed. Section 6 contains conclusions.

2. State-of-the-Art

There are two main approaches to solve the keyphrase extraction task. The first is based on single word ranking, best words selection and concatenation of best words following each other in the text [9-12]. The dominating approach [5-8, 12-16] consists of two stages: a selection stage, when candidate phrases are selected, and a classifying or ranking stage. On the selection stage a number of procedures is used to extract candidate phrases: n-gram extraction, noun phrase extraction, word sequence extraction or their combinations, which satisfy some limitations. The examples of limitations are following: length limit of a phrase (usually not more than 4-5 words per phrase), parts of speech limits, etc. It has been shown that keyphrases should consist from nouns and adjectives to achieve the best results and this result is actively used. In [14] the author proposes to use part of speech information in classification process. In pioneer systems on the second stage supervised methods were used to decide for each candidate whether it is keyphrase. In [15] a Naive Bayes classifier is used. In [16] a keyphrase extraction process is based on a number of threshold values of some variables which are optimized using genetic algorithm. These methods [14-16] could be used for the case, when there is a set of documents with keyphrases already extracted by the expert. On the ranking stage all candidate phrases are weighted and ranked. Then k-best candidate phrases are selected as keyphrases. Ranking methods are usually based on phrase weight measurement [5-7, 12, 13]. In this case, statistical measurements are often used for phrases and phrase words as well as information about the first position of a phrase in text and the size of a phrase with its frequency. However, researchers do not address and analyze possibilities of different variants of phrases' weight evaluation based on in-phrase word's weights. In this paper, we fill the gap. We evaluate several approaches to phrase weighting and use a number of statistical measures for this task. Experiments have shown that selected statistical measures do not allow identifying correct keyphrases among other phrases. It seems that simple exclusion

of some set of candidates is more efficient, that is the set where most keyphrases are not correct apriori. In presented paper we have shown that the set of one-word candidate phrases is a set of this kind and its exclusion leads to relatively good results. As a result of cur-rent research we make a statement about possible reasons, why information about the length of a phrase influences the result of keyphrase extraction.

3. Experiment Description

3.1 Candidate Phrase Ranking

One of the goals of the presented paper is to analyze a number of approaches to phrase weight measurement. We deal only with weight measurement based on in-phrase words evaluation. We are using the following notations. The phrase with n words is denoted as (w_1, w_2,…,w_n), where w is a single word. Phrase weight is denoted as $weight(w_1, w_2,\ldots,w_n)$ and the weight of a word as $weight(w)$. We measure weights of phrases as:

1. Average weight among in-phrase words:

$$weight(w_1, w_2, \ldots, w_n) = \frac{\sum_{i=1}^{n} weight(w_i)}{n}. \quad (1)$$

2. Geometric mean of word weights in phrase:

$$weight(w_1, w_2, \ldots, w_n) = \sqrt[n]{\prod_{i-1}^{n} weight(w_i)}. \quad (2)$$

3. Degree of relationship between words in a phrase and a main word in a phrase.

For the case 3 (degree of relationship between words in a phrase and a main word in a phrase) six measuring approaches described below were used to determine a main word in a phrase. Word w is determined as w^{main} for the phrase if its weight is the best weight in a phrase compared to the weights of other words in a phrase. When the main word has been chosen the relationship value between each other word w in a phrase and main word w^{main} is calculated. In our research Two measures were used to calculate words relation:

- Pointwise mutual information, calculated between the main word w^{main} and every other word w in a phrase:

$$mi(w^{main}, w) = \frac{p(w^{main}, w)}{p(w^{main}) * p(w)}, \quad (3)$$

  where $p(w^{main},w)$ is a probability to meet word w^{main} next to every other word in-phrase w (in window 3), $p(w^{main})$ and $p(w)$ are probabilities of meeting words w^{main} and w. A phrase weight is defined as an average among the obtained values:

$$weight(w_1, w_2, \dots, w_n) = \frac{\sum_{i=1}^{n-1} mi(w^{main}, w_i)}{n-1}. \quad (4)$$

- Word w^{main} and word w relationship:

$$rel(w^{main}, w) = \max\{p(w^{main}|w), p(w|w^{main})\}, \quad (5)$$

$$p(w_1|w_2) = \frac{\sum_{d \in D_{w_2}} tf^d(d, w_1)}{\sum_{d \in D_{w_2} w\prime} tf^d(d, w\prime)}, \quad (6)$$

where D_{w_2} – set of all documents that contains w_2, $tf^d(d,w_1)$ – the number of occurrence of the word w_1 in the document d, w' belong to words in D_{w_2}. An average of obtained values is defined as a weight of a phrase as in (4) but $rel(w^{main}|w)$ is used instead $mi(w^{main,}, w)$.

To evaluate the weight of a word *weight*(*w*) in a text *d* for (1) and (2) or for a selection of main word in phrase, we use the following six values:

- Number of documents where the word *w* occurs at least once (*df*).
- Within collection word *w* frequency (*tf*).
- Within document *d* word *w* frequency (tf^d);
- Ratio: *tf*/*df*
- *tf-idf* [17]:

$$weight(w) = \mathrm{tf}^{\mathrm{d}}(\mathrm{w}) \cdot log \frac{N}{df(w)}, \quad (7)$$

where *N* is the number of documents in the collection.

- The evaluation of word's *w* context narrowness (word context).

Concept of narrow context is borrowed from [18]. Words with narrow context are domain-specific. For example, "motherboard" is the word with narrow context. If a document contains this word we can conclude with high probability that this document is about computer hardware. The word "computer" has wide context. If a document contains such word it is difficult to define the content of this document. It can be about hardware, art, health, e.t.c. with almost the same probabilities. Simplifying the method of detection words with narrow context [18], we define for each word *w* its context $p(\mathrm{Y}|w)$ by using $p(y|w)$ (6), where *y* belongs to collection's vocabulary. Then entropy *H* is calculated for every obtained context. Based on assumption that the context of word with narrow context has low entropy, we use word's context entropy to evaluate words:

$$weight(w) = H(\mathrm{Y}, w) = -\sum_y p(y|w) log p(y|w). \quad (8)$$

The best word's weight in a phrase for *df*, *tf*, tf^d, *tf*/*df*, *tf-idf* is the highest weight and for word context is the lowest weight.

3.2 Data Preprocessing and Candidate Phrase Extraction

Presented paper is focused on the problem of ranking of candidate phrases. Thus, we used basic algorithm for candidate extraction described as follows. The POS-

tagged text is fed to the input of the algorithm (we used Stanford POS-tagging tool [19]). The sequences of nouns and adjectives are extracted from the text. Stop words, punctuation and other parts of speech, excluding nouns and adjectives, are used on this stage as delimiters. The size of obtained sequences is limited to 5. All extracted sequences are considered as candidate phrases.

3.3 Dataset

We have used Inspec dataset collection for our research, because in presented paper we are focusing on keyphrase extraction from abstracts of scientific articles. Inspec contains annotations to scientific articles in English (from disciplines "Computers and Control", and "Information technology"). Inspec collection contains three sub-collections: training dataset (1000 documents), evaluation dataset (500 documents) and testing dataset (500 documents). Each text has a gold standard, which contains phrases, extracted by an expert. Gold standard includes two types of annotations: contr set and uncontr set. As in most other papers [9, 12, 14, 20, 21] test dataset and uncontr gold standard set are used for this paper. A detailed collection description is presented in [14].

3.4 Evaluation

To measure the quality of extracted keyphrases we use the traditional approach based on F-score, which is a combination of Precision and Recall [17], and is one of the most popular quality measures in keyphrase extraction domain:

$$F - score = \frac{2 * Precision * Recall}{Precision + Recall},$$

$$Precision = \frac{|G \cap C|}{G}, \quad Recall = \frac{|G \cap C|}{C},$$

where *G* is the number of automatically extracted kephrases from all documents and *C* is the number of all keyphrases extracted by expert (number of phrase in the gold standard). In the case when a number of extracted keyphrases is less than given in the gold standard Precision is used instead F-score as it depends on the number of correct phrases among the extracted keyphrases. Otherwise, F-score declines with decrease of the number of extracted phrases because Recall also declines. When the number of extracted keyphrases is the same as in the gold standard, F-score and Precision are identical because *G* equals *C*.

3.5 Experiment

On the first stage, candidate phrases were extracted for each text using approach proposed in section 3.2. For each phrase in a document its weight is calculated. Weight calculation is done using strategies described in 3.1 as average weight of all

words in a phrase (1), as a geometric mean of all words in a phrase (2), as an average weight of relation between main word and other words in a phrase (3-6). One of six measures presented in 3.1 was used for a word's weight evaluation: *tf-idf* (7), *df*, *tf*, tf^d, *tf*/*df*, word context (8).

It is important to say that if a phrase contains only one word, then (3) and (6) are not usable, because they need at least two words to be calculated. For these cases one-word phrases were excluded. To compare this weight evaluation approach with the other available approaches, we have conducted experiment, where for each approach mentioned above one-word phrases were filtered. This experiment has shown interesting results which are presented further. After weight evaluation, phrases were ranked according to their weights and *k*-best were selected as keyphrases. We have examined a number of cases to determine *k*:

- *k* was taken according to the number of phrases mentioned in the gold standard [12];
- *k* equals to 7;
- all candidate phrases are selected as keyphrases (no ranking was performed).

4. Experimental Results and Discussion

4.1 Experiment Results

Results of keyphrase extraction experiment are presented in Tables 1 and 2, the weight of a phrase was calculated as an average of word weights, contained in a phrase (1). To calculate the word's weight six approaches were tested (3.1) and appropriate results are presented in columns. The number of phrases to select was defined as follows: the same number as in the gold standard and 7 (this information is presented by rows). Table 1 presents results, when no phrase was filtered. Table 2 presents results, when all one-word phrases were filtered. Experiments, which results are presented in Table 3 and Table 4, differ to the experiments in Tables 1 and 2 only in the change of keyphrase weight function, for these experiments geometric mean was used (2). Table 5 presents results of experiments, where the phrase weight was calculated using main word, which was chosen among the words in the phrase and then pointwise mutual information (3) was calculated for each pair, where the first word was the main word and second word - every other word in the phrase. One-word phrases were filtered. The main word was selected as a word with the best weight in the phrase. To evaluate word weights measures, described in 3.1, were used: *tf-idf* (7), *df*, *tf*, tf^d, *tf*/*df*, word context (8). In Table 6 results of a similar experiments are shown for the case, when relationship of each word with the main word was calculated (5). Table 7 contains results of extracted keyphrases for the case, when the candidate phrases were not ranked and all of them were selected as keyphrases. Table 8 contains results for the case when keyphrases were selected randomly from the set of candidate phrases and the number of extracted keyphrases was equals the keyphrase number in the gold standard.

Table 1. Results: keyphrase weight was calculated as an average of weights among words in phrase weights

The number of extracted kephrases	Evaluation measure	tf-idf	df	tf	tf^d	tf/df	word context
The same number as in gold standard	F-score	0.31	0.20	0.23	0.29	0.31	0.28
7	Precision	0.29	0.18	0.20	0.27	0.30	0.25

Table 2. Results: keyphrase weight was calculated as an average of weights among words in-phrase weights, when one-word phrases were filtered

The number of extracted kephrases	Evaluation measure	tf-idf	df	tf	tf^d	tf/df	word context
The same number as in gold standard	F-score	**0.40**	0.36	0.37	**0.39**	**0.40**	**0.39**
7	Precision	**0.39**	0.34	0.36	**0.39**	**0.40**	**0.40**

Table 3. Results: keyphrase weight was calculated as a geometry mean of weights among words in-phrase weights

The number of extracted kephrases	Evaluation measure	tf-idf	df	tf	tf^d	tf/df	word context
The same number as in gold standard	F-score	0.31	0.18	0.19	0.29	0.31	0.29
7	Precision	0.28	0.14	0.16	0.26	0.29	0.25

Table 4. Results: keyphrase weight was calculated as a geometry mean of weights among words in-phrase weights, when one-word phrases were filtered

The number of extracted kephrases	Evaluation measure	tf-idf	df	tf	tf^d	tf/df	word context
The same number as in gold standard	F-score	**0.40**	0.36	0.37	**0.39**	**0.41**	**0.40**
7	Precision	**0.40**	0.34	0.35	**0.39**	**0.42**	**0.40**

Table 5. Results: main word was selected, then pointwise mutual information was calculated between main word and other words in-phrase and average values was calculated as a score of a phrase

The number of extracted kephrases	Evaluation measure	tf-idf	df	tf	tf^d	tf/df	word context
The same number as in gold standard	F-score	**0.39**	**0.39**	**0.39**	**0.40**	**0.40**	**0.40**
7	Precision	**0.40**	**0.40**	**0.40**	**0.40**	**0.40**	**0.40**

Table 6. Results: main word was selected, words relationship was calculated betwee main word and other words in-phrase (5), average values was calculated as a score of a phrase

The number of extracted keywords	Evaluation measure	tf-idf	Df	tf	tf^d	tf/df	word context
The same number as in gold standard	F-score	**0.40**	**0.40**	**0.40**	**0.40**	**0.40**	**0.40**
7	Precision	**0.41**	**0.41**	**0.41**	**0.41**	**0.41**	**0.41**

Table 7. Results: all candidate phrases were selected as keyphrases

Including/Excluding one-word phrase candidates	F-score
Without one-word phrase filtering	0.30
One-word phrase filtering was used	**0.40**

Table 8. Results: keyphrases were selected randomly

Including/Excluding one-word phrase candidates	F-score
Without one-word phrase filtering	0.23
One-word phrase filtering was used	0.38

4.2 Discussion

Results presented in Table 1 and Table 3 show that usage of *tf* (within collection term frequency) and *df* (within collection document frequency) measures to evaluate words weight decreases the quality of extracted keyphrases even in comparison with arbitrary selection (Table 8). Other measure's results do not differ much regardless the way how phrase weight is calculated and these measures we will discuss below.
Experiments show that results in Tables 2, 4, 5, 6 are very similar. Thus we can conclude that all methods give near the same results in respect to one-word phrases

filtering, regardless of a way to weight words and regardless of the number of extracted keyphrases. Slighter better result is achieved when keyphrase weight is calculated as a geometry mean and *tf/df* is used.

Another interesting observation is the fact that filtering one-word phrases significantly increases quality of remained keyphrases and improves results of the state of the art [9, 12, 14]. It is interesting that if we only filter out all the one-word keyphrases without performing resulting ranking at all, we will get F-score=0.40, the same result as with ranking. So it seems that ranking doesn't improve quality of keyphrases.

In fact experiments show that filtering of one-word keyphrases makes significantly greater impact than phrase weighting, based on statistics mentioned above. We have made an assumption as well, that all ranking approaches, mentioned above, essentially select keyphrases randomly and thus the results of different approaches are very close. To prove it an additional experiment was conducted, which goal was to show that the ratio between correct and incorrect keyphrases before and after ranking remains almost the same.

5. Additional Experiment

5.1 Experiment Description

The goal of proposed additional experiment is to show that all phrase-ranking approaches, used to select keyphrases in this paper, essentially select keyphrases randomly. Input data to the experiment is a set of pre-ranked phrase candidates. For this set for each phrase-length a number of phrases is set, and also known the number of correct and incorrect phrases. The ranking algorithm forms the output data, which is a set of selected keyphrases with the information about the number of selected phrases for each phrase length, including information about correctness of such se-lection. Number of selected keyphrases is the same as in the gold standard. The goal is to evaluate the ratio between all phrases and correct phrases before and after keyphrase selection step.

5.2 Experiment Results and Discussion

Because experiments in section 4 give almost the same results for a number of measures, here we are using only one of them – tf^d (within document frequency) measure. Experimental results are described in Table 9. In first column phrase length is presented and also the information about one-word phrases inclusion during experiment: are they filtered or not. In other columns additional information is presented: number of candidate phrases, how many of them are correct, ratio between the number of candidates and the number of correct among them and the same information for the case when ranking is performed.

Table 9. Results of additional experiment

INSPEC Phrase's length	The number of extracted candidate phrases	The number of correctly extracted candidate phrases	Ratio between the number of candidates to the number of correct among them	The number of extracted keyphrases after ranking	The number of correctly extracted keyphrases after ranking	Ratio between the number of keyphrases to the number of correct among them (after ranking)
With filtering one-word phrases						
2	4349	1552	2.80	2873	1233	2.33
3	1577	625	2.53	1195	513	2.33
4	370	128	2.89	299	109	2.74
5	130	34	3.82	116	31	3.74
Without filtering one-word phrases						
1	3056	392	**7.80**	1450	244	**5.90**
2	4349	1552	2.80	1698	798	2.13
3	1577	625	2.53	780	351	2.22
4	370	128	2.89	203	84	2.42
5	130	34	3.82	81	24	3.38

For keyphrases of 2-4 words length ratio between the number of phrases to the number of correct keyphrases lies inside range 2-3 (before and after ranking) and for one-word phrases this ratio is close to 8 on input data and is close to 6 on output data. It means that the set of one-word keyphrases contains much more incorrect keyphrases than correct ones. Notice that the number of one-word phrases in input data is the third part of all phrases. Thus it becomes obvious why filtering one-word phrases yields much better results. When we filter one-word phrases and arbitrary select the number of keyphrases as in the gold standard the F-score = 0.38 which is better than state of the art results for Inspec, which use complex ranking techniques [9][12][14]. Analysis of experimental results in Table 9 shows that the ratio between all keyhrases and correct keyphrases after ranking slightly improves the result before ranking. Taking this fact and results from Section 4 (in which it was shown, that using one-word phrase filtering, results of all methods are nearly the same) into account we can conclude that the results of all methods, which were investigated in this paper (excluding *tf* and *df*) are quite close to results of random pick of phrases from initial set. This result also shows that methods that weight phrases using information about phrase length should work good on Inspec dataset (longer phrases

usually evaluate with more weight than short phrases and so one-word phrases become filtered). Remind that one-word phrase consists of alone noun/adjective and separated from other nouns and adjectives by punctuation, stop-words and other words excluding nouns and adjectives.

6. Conclusion

The results of presented research show that investigated approaches to phrase weighting (excluding *tf* and *df*) show almost equal results and only slightly increase random phrase selection from phrase candidates. They differ mostly in the way how they rank one-word phrases. If one-word phrases are excluded, all methods would give rather similar results. Exclusion of one-word candidate phrases increases extraction quality, because in one-word phrases ratio between correct keyphrases and all phrases is significantly bigger comparing to the phrases of other lengths.

Experiments were based on Inspec dataset, which is popular for the task of keyphrase extraction from scientific abstracts. Experiments prove that for this collection good results will be given by algorithms which filter one-word phrases, even if other phrases are ranked randomly. This result should be considered when working with Inspec collection and further evaluating approaches, investigated in this paper.

Acknowledgments: This work was partially supported by federal target program research project 16.740.11.0751 and by research work 414648 ITMO University.

References

[1]. Gutwina, C., Paynterb, G., Wittenb, I., Nevill-Manningc C., Frankb E.: Improving browsing in digital libraries with keyphrase indexes. Journal of Decision Support Systems, 27(1-2), pp. 81–104 (1999)

[2]. Zhang, D. and Dong, Y.: Semantic, Hierarchical, Online Clustering of Web Search Results. In: 6th Asia-Pacific Web Conference. Hangzhou, China (2004)

[3]. Zeng, H.J., He, Q.C., Chen, Z., Ma, W.Y., Ma, J.: Learning to cluster web search results. In: the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 210-217 (2004)

[4]. Popova, S., Khodyrev, I., Egorov, A., Logvin, S., Gulyaev, S., Karpova, M. and Mouromtsev, D. Sci-Search: Academic Search and Analysis System Based on Keyphrases. In: KESW 2013, Communications in Computer and Information Science, CCIS, vol. 394, pp 281-288, Springer Berlin Heidelberg

[5]. Pudota, N., Dattolo, A., Baruzzo, A., Ferrara, F., Tasso, C.: Automatic keyphrase extraction and ontology mining for content-based tag recommendation. International Journal of Intelligent Systems, vol 25, pp. 1158-1186, 2010

[6]. You, W., Fontaine, D., Barhes, J.-P.: An automatic keyphrase extraction system for scientific documents. In: Knowl Inf Syst 34, pp. 691-724, 2013

[7]. El-Beltagy, S. R., and Rafea, A.,: KP-Miner: A keyphrase extraction system for english and arabic documents. In: Information Systems, 34, pp. 132-144, 2009

[8]. Popova, S., Khodyrev, I.: Izvlechenie i ranzirovanie klyuchevix fraz v zadache annotirovaniya [Keyphrase extraction and ranking in annotation problem]. Journal Nauchno-Texnicheskiy Vestnik Informatsionnix technologiy mechaniki i optiki

[Scientific and Technical Journal of Information Technologies, Mechanics and Optocs], Vol. 1, 2013
[9]. Mihalcea, R., Tarau, P.: TextRank: Bringing order into texts. In: Conference on Empirical Methods in Natural Language Processing, pp. 404–411, 2004
[10]. Xiaojun, W. and Xiao, J.: Single document keyphrase extraction using neighborhood knowledge. In: Proceedings of the 23rd AAAI Conferenceon Artificial Intelligence, pp. 855–860, 2008
[11]. Xiaojun W., Xiao J.: Exploiting Neighborhood Knowledge for Single Document Summarization and Keyphrase Extraction ACM Transactions on Information Systems, 28(2), Article 8, 2010
[12]. Zesch, T., Gurevych, I.: Approximate Matching for Evaluating Keyphrase Extraction. In: International Conference RANLP 2009. pp. 484–489, Borovets, Bulgaria, 2009
[13]. Kim, S.N., Medelyan, O., Yen, M.: Automatic keyphrase extraction from scientific articles. Language Resources and Evaluation, Springer Kan & Timothy Baldwin, 2012
[14]. Hulth A.: Improved automatic keyword extraction given more linguistic knowledge. In: Conference on Empirical Methods in Natural Language Processing, pp. 216–223, 2003
[15]. Frank, E., Paynter, G.W., Witten, I.H., Gutwin, C., Nevill-Manning, C.G.: Domain-specific keyphrase extraction. In: Proc. of IJCAI. pp. 688–673,1999
[16]. Turney, P.: Learning to Extract Keyphrases from Text. In: NRC/ERB-1057, pp. 17– 43, 1999
[17]. Manning, C., Raghavan, P., Schutz,e H.: Introduction to Information Retrieval. Cambridge University Press, 2009
[18]. Dobrynin, V., Patterson, D., Rooney, N.: Contextual Document Clustering. In Advances in Information Retrieval. Lecture Notes in Computer Science. 2997, pp.167-180, 2004
[19]. Standford POS tagging tool DOI: http://nlp.stanford.edu/software/tagger.shtml (09.11.2012).
[20]. Tsatsaronis, G., Varlamis, I., Norvag, K.: SemanticRank: Ranking Keywords and Sentences Using Semantic Graphs. In: Proc. of the 23rd International Conference on Computational Linguistics, pp. 1074–1082, 2010
[21]. Hasan, K. S., Ng, V.: Conundrums in Unsupervised Keyphrase Extraction: Making Sense of the State-of-the-Art. In: Coling, Poster Volume, Beijing, pp. 365–373, 2010

Ранжирование в задаче извлечения ключевых фраз: есть ли польза от использования частотных характеристик слов фраз-претендентов?

[1,2] *С. В. Попова <svp@list.ru>*
[2] *И. А. Ходырев <kivan.mih@gmail.com>*
[1] *Санкт-Петербургский Государственный Университет,*
Россия, 199034, Санкт-Петербург, Университетская наб. д.7-9
[2] *Университет ИТМО,*
Россия, 197101, г. Санкт-Петербург, Кронверкский проспект, д.49

Аннотация. В работе исследуется задача извлечения ключевых фраз из отдельных текстов, таких как аннотации к научным публикациям. Проблема извлечения ключевых фраз имеет высокую практическую ценности, фразы могут быть использованы в задачах индексирования данных поисковыми системами, для кластеризации/классификации данных, для пополнения онтологий и извлечения метаинформации. Работа основана на подходе в рамках которого для извлечения ключевых фраз сперва из текста извлекаются фразы-претенденты, которые затем ранжируются и фразы с лучшим рангом отбираются как ключевые. Исследуются способы ранжирования фраз-претендентов на основе статистических характеристик слов, входящих во фразы-претенденты. Определены статистические характеристики слов, которые плохо подходят для ранжирования фраз-претендентов, показано что большая часть рассмотренных способов ранжирования фраз-претендентов в действительности работают аналогично рандомному ранжированию и отличаются только способами ранжирования однословных фраз. Предложен подход, основанный на удаление однословных фраз, позволяющий значительно повысить качество отбираемых ключевых фраз.

Ключевые слова: извлечение ключевых фраз; ранжирование ключевых фраз; статистические характеристики в задаче извлечения ключевых фраз; извлечение информации; обработка аннотаций к научным публикациям.