ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Издается с сентября 2010 г.

Журнал издается при поддержке Отделения математических наук РАН, Отделения нанотехнологий и информационных технологий РАН, МГУ имени М.В. Ломоносова, МГТУ имени Н.Э. Баумана, ОАО "Концерн "Сириус".

СОДЕРЖАНИЕ

Журнал зарегистрирован в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций. Свидетельство о регистрации ПИ № ФС77-38590 от 24 декабря 2009 г.

Журнал распространяется по подписке, которую можно оформить в любом почтовом отделении (индексы: по каталогу агентства "Роспечать" — **22765**, по Объединенному каталогу "Пресса России" — **39795**) или непосредственно в редакции.
Тел.: (499) 269-53-97. Факс: (499) 269-55-10.
Http://novtex.ru E-mail: prin@novtex.ru

Журнал включен в систему Российского индекса научного цитирования.
Журнал входит в Перечень научных журналов, в которых по рекомендации ВАК РФ должны быть опубликованы научные результаты диссертаций на соискание ученой степени доктора и кандидата наук.

УДК 004.5

Б. Б. Величковский, канд. психол. наук, доц., Факультет психологии МГУ им. М. В. Ломоносова, нач. лаборатории, НБИКС-Центр НИЦ "Курчатовский институт", e-mail: velitchk@mail.ru
А. И. Данилова, проектировщик интерфейсов, UIDesign Group, г. Москва

Влияние нагрузки на рабочую память пользователя на эффективность навигации в меню мобильного устройства

Описаны эксперименты, в которых показано, что увеличение нагрузки на рабочую память пользователя (подсистему памяти человека для оперативного хранения информации) снижает скорость и точность его навигации в меню симулированного мобильного устройства. Обсуждается возможность использования понятия рабочей памяти при проектировании меню реальных мобильных устройств.

Ключевые слова: *рабочая память, навигация в меню, мобильные устройства*

Разнообразные мобильные вычислительные устройства — мобильные телефоны, смартфоны, карманные персональные и планшетные компьютеры — в настоящее время широко используются в личной жизни и профессиональной деятельности людей. Начиная с 2007 г. в мире ежегодно продается свыше 1 млрд мобильных устройств [1]. Важной производной от этих процессов тенденцией является совершающийся во многих отраслях экономики переход к "мобильной работе". Мобильная работа предполагает, что работник имеет доступ ко всем необходимым ему информационным и вычислительным ресурсам вне стационарного рабочего места. Таким образом, важной практической задачей сегодня становится выявление принципов проектирования удобных в использовании интерфейсов мобильных устройств. Полноценное решение этой задачи невозможно без изучения факторов, влияющих на эффективность взаимодействия пользователя с мобильным устройством.

Для организации взаимодействия с пользователем в интерфейсах стационарных и мобильных устройств часто используют различные виды меню. Меню предоставляет доступ к функциям вычислительного устройства посредством выбора элементов меню. Выбор элемента приводит либо к необходимости выбора другого элемента, либо к выполнению одной из функций устройства. Таким образом, взаимодействие с помощью меню сводится к выполнению пользователем целенаправленного "перемещения" из текущего положения в структуре меню к "целевой" функции путем выбора элементов меню. В связи с этим можно говорить о "навигации" в меню.

Эффективность навигации в меню можно оценивать на основе различных критериев. Например, можно измерять скорость навигации — время, затрачиваемое пользователем на выбор заданного элемента меню. Можно также измерять точность навигации — число переходов между элементами меню, "лишних" по сравнению с минимально необходимым числом. Как показывают исследования, на эффективность навигации влияют различные факторы: "ширина" и "глубина" меню; его симметричность; критерии распределения элементов внутри меню и многие другие [2, 3]. Важным психологическим фактором эффективности

навигации являются индивидуальные возможности рабочей памяти пользователя.

Под рабочей памятью человека понимается система когнитивных процессов, обеспечивающих оперативное хранение и изменение информации в целях выполнения актуальной задачи [4]. Рабочая память имеет два важных ограничения. Во-первых, объем рабочей памяти — количество одновременно удерживаемой и изменяемой информации — резко ограничен. У "обычного человека" объем рабочей памяти не превышает 3—4 независимых единицы информации. Во-вторых, информация, хранящаяся в рабочей памяти, нестабильна и достаточно быстро распадается. Вербальная информация, например, "стирается" в течение примерно 2 с.

Рабочая память играет важную роль в осуществлении человеком интеллектуальной деятельности. Показатели объема рабочей памяти коррелируют с академической успеваемостью, эффективностью изучения иностранных языков и языков программирования, способностью к управлению сложными техническими системами и т. д. [4, 5]. В современных психологических теориях рабочая память человека рассматривается как "глобальное рабочее пространство", в котором разворачиваются процессы мышления и принятия решений [6]. Мыслительные задачи различаются объемами хранения информации и интенсивностью ее обработки, т. е. возникающей в ходе их решения нагрузкой на рабочую память.

При навигации в меню рабочая память может выполнять различные функции. В частности, ресурсы рабочей памяти необходимы для удержания структуры ("карты") меню и для определения оптимального "маршрута движения" с учетом исходного положения и цели движения. Кроме того, в ходе самой навигации рабочая память необходима для оперативного удержания и обновления "текущего положения" в меню для отслеживания успешности продвижения к цели. Поэтому ограничения рабочей памяти могут влиять и на эффективность навигации. Например, если глубина (число уровней) меню превышает объем рабочей памяти пользователя, то высока вероятность потери пользователем ориентации в структуре меню. Следствием такого состояния является совершение пользователем избыточных переходов между элементами меню, т. е. "ошибок навигации".

Особенности мобильных устройств и контекста их использования сами по себе могут увеличивать нагрузки на рабочую память. Как следствие, это может приводить к еще большему снижению эффективности навигации в меню. Например, небольшие размеры экранов мобильных устройств заставляют проектировщиков использовать достаточно глубокие меню. Ограничения размеров экранов мобильных устройств также приводят к тому, что пользователи должны удерживать большее количество информации в рабочей памяти, что еще больше увеличивает нагрузку на нее. Кроме того, взаимодействие с мобильными устройствами обычно осуществляется одновременно с выполнением других, возможно, нетривиальных действий. Для выполнения каждого из них также может использоваться рабочая память. В результате навигация в меню мобильного устройства может значительно усложняться.

Результаты эмпирических исследований подтверждают сделанные выше предположения. Навигация в меню мобильных устройств обычно менее эффективна, чем навигация в функционально подобных меню стационарных устройств [7]. Такое снижение эффективности обусловлено меньшими размерами экранов мобильных устройств, что приводит к увеличению глубины меню и усложнению их структуры. На эффективность навигации в меню мобильных устройств также влияют индивидуальные особенности пользователей, прямо или косвенно связанные с доступностью ресурсов рабочей памяти — ее объем, имеющийся у пользователей опыт взаимодействия с мобильными интерфейсами, возраст пользователей [8, 9]. Эти результаты позволяют предположить, что уровень нагрузки на рабочую память пользователя при навигации в меню мобильного устройства оказывает большое влияние на ее эффективность. Целью работы, описанной в статье, была непосредственная экспериментальная проверка этого предположения.

Методика

Стимульный материал. На базе лаборатории психологии труда факультета психологии МГУ им. М. В. Ломоносова авторами была создана интерактивная модель современного смартфона, исполняемая на персональном компьютере (ПК). Программирование модели было осуществлено в среде для создания поведенческих экспериментов E-Prime 2.0. На экране ПК с сохранением размеров условно воспроизводился внешний вид моделируемого устройства. На экране имитации находились интерактивные кнопки, соответствующие пунктам меню. Выбор определенного пункта меню осуществлялся с помощью мыши. Выбор пункта меню сопровождался зрительной (подсветка) и звуковой ("щелчок") обратной связью.

В модели были реализованы три меню ("телефон", "календарь", "контакты"). Для каждого меню был составлен список заданий, соответствующих типичным сценариям использования смартфонов. Для меню "календарь" было составлено 15 заданий ("Создайте новое событие", "Удалите событие из списка", "Установите ежемесячный повтор события" и подобные им). Для меню "контакты" было составлено 23 задания ("Создайте новый контакт", "Добавьте контакт в избранное", "Добавьте номер телефона к контакту" и ряд других). Для меню "телефон" было составлено 11

заданий ("Наберите номер телефона", "Просмотрите пропущенные вызовы", "Удалите последний вызов" и др.). Чтобы выполнить задание, испытуемый должен был перейти к соответствующему пункту меню. Выбор испытуемым нужного пункта сопровождался звуковым сигналом.

Для манипуляции уровнем нагрузки на рабочую память испытуемого использовалась методика дополнительной задачи [4]. Методика предполагает сравнение эффективности работы испытуемого в двух экспериментальных условиях. При первом условии испытуемый выполняет только одну — основную — задачу, для которой необходимо доказать использование ресурсов рабочей памяти. Во втором случае (условии) испытуемый одновременно выполняет две задачи — основную и дополнительную. Дополнительная задача формируется так, что ее выполнение создает дополнительную нагрузку на рабочую память испытуемого. Заметное ухудшение эффективности работы испытуемого в условии с дополнительной задачей трактуется как доказательство вовлечения рабочей памяти в выполнение основной задачи.

В рассматриваемом эксперименте дополнительная задача заключалась в необходимости отслеживать звуковые сигналы ("телефонные звонки") и реагировать нажатием определенной клавиши при каждом третьем "звонке". В случае ошибки (т. е. при нажатии клавиши при недостаточном количестве прозвучавших сигналов или при пропуске трех сигналов) испытуемому давалась звуковая обратная связь. После каждой верной и каждой ошибочной реакции отсчет "звонков" начинался заново.

Процедура эксперимента. Испытуемый знакомился с письменной инструкцией, после чего экспериментатор на примерах пояснял работу модели устройства и отвечал на возможные вопросы. Испытуемые выполняли три блока экспериментальных заданий, по одному блоку для каждого меню. Использовался фиксированный порядок блоков ("контакты" — "телефон" — "календарь"). Задания в блоке предъявлялись в случайном порядке, каждое задание предъявлялось 2 раза. Текст задания отображался в верхней части экрана в течение всего времени выполнения одной пробы. В ходе эксперимента испытуемые выполняли две серии блоков — серию без выполнения нагрузочной задачи, а также серию с выполнением нагрузочной задачи. Порядок выполнения серий был сбалансирован между испытуемыми: половина испытуемых сначала выполняла серию без нагрузочной задачи, а половина — серию с нагрузочной задачей. При выполнении каждой пробы регистрировались следующие показатели эффективности навигации (все временные показатели измерялись в миллисекундах):

- общее время, потраченное на переход к целевой команде;
- среднее время перехода от одного пункта меню к другому;
- время первого перехода после предъявления нового задания;
- число переходов, совершаемых испытуемым для достижения целевой команды.

Испытуемые. В исследовании приняли участие восемь человек, студенты пятого курса факультета психологии МГУ им. М. В. Ломоносова, в возрасте от 21 до 22 лет.

Результаты

В целях очистки данных для каждого испытуемого были исключены значения, отклоняющиеся от индивидуального среднего более чем на три индивидуальных стандартных отклонения. Средние и стандартные отклонения очищенных показателей эффективности навигации в условии с дополнительной нагрузкой на рабочую память и без нее приведены в табл. 1. Перед анализом данных значения всех показателей были подвергнуты логарифмической трансформации в целях нормализации их распределений.

Оценка статистической значимости влияния фактора нагрузки на эффективность навигации проводилась методом дисперсионного анализа [10]. Под влиянием дополнительной нагрузки на рабочую память статистически значимо увеличивается общее время навигации. Значение F-критерия[1] Фишера равенства дисперсий составило $F(1, 1511) = 21{,}9$; вероятность ошибки первого рода p была меньше 0,001. Также статистически значимо увеличилось среднее время одного перехода ($F(1, 1511) = 41{,}9$; $p < 0{,}001$) и время, необходимое для выполнения первого перехода ($F(1, 1511) = 9{,}1$; $p < 0{,}01$). Дополнительная нагрузка на рабочую память не приводит к значимому изменению числа совершаемых испытуемыми переходов ($F(1, 1511) = 0{,}1$; $p > 0{,}05$).

Таблица 1

Средние и стандартные (в скобках) отклонения показателей эффективности навигации в зависимости от нагрузки на рабочую память

Нагрузка	Эффективность навигации			
	Общее время, мс	Среднее время, мс	Время первого перехода, мс	Число переходов
Нет	4468 (3402)	1703 (896)	2314 (1424)	2,65 (1,16)
Есть	5110 (3552)	1965 (959)	2502 (1394)	2,63 (1,14)

[1] Значения в скобках после F — это степени свободы (первое значение равно числу сравниваемых экспериментальных условий — 1, а второе связано с числом измерений).

Таблица 2

Средние и стандартные (в скобках) отклонения показателей эффективности навигации в зависимости от нагрузки на рабочую память и необходимого числа переходов

Нагрузка	Необходимое число переходов	Эффективность навигации			
		Общее время, мс	Среднее время, мс	Время первого перехода, мс	Число переходов
Нет	1	2436 (1781)	2099 (1176)	2188 (1226)	1,19 (0,67)
	2	3786 (2653)	1717 (860)	2289 (1289)	2,18 (0,64)
	3	5142 (3075)	1585 (819)	2347 (1574)	3,21 (0,71)
	4	7709 (5830)	1594 (793)	2470 (1637)	4,54 (1,27)
Есть	1	2636 (1556)	2386 (1093)	2398 (1103)	1,12 (0,51)
	2	4111 (2286)	1937 (1023)	2553 (1556)	2,13 (0,52)
	3	5919 (3066)	1833 (799)	2389 (1193)	3,20 (0,65)
	4	10 125 (6282)	2106 (923)	2933 (1684)	4,70 (1,28)

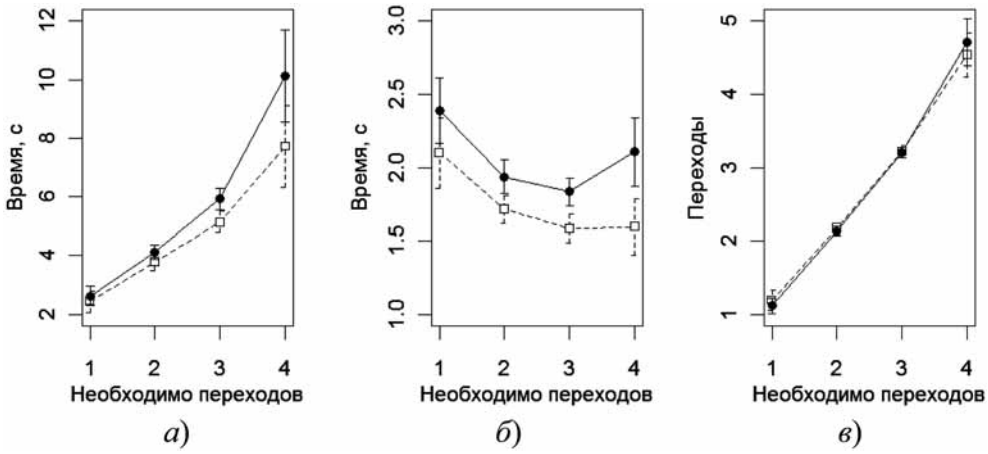
В ходе исследований был также проведен анализ влияния дополнительной нагрузки на рабочую память, при котором учитывалось необходимое число переходов. Под необходимым числом переходов понимается минимальное число переходов, которое позволяло достигнуть заданного пункта меню из того положения в структуре меню, в котором испытуемый оказался после выполнения предыдущего задания. Необходимое число переходов, по сути, задает "дистанцию", которую необходимо "преодолеть", чтобы достигнуть целевого пункта меню. Очевидно, что при увеличении расстояния до целевого объекта навигация может затрудняться, и роль рабочей памяти в этом случае может проявиться особенно отчетливо.

На основе экспериментальных протоколов для каждого задания было определено необходимое число переходов (диапазон значений от 1 до 4). Средние и стандартные отклонения показателей эффективности навигации для разных уровней нагрузки и разного необходимого числа переходов приведены в табл. 2. Для оценки статистической значимости влияния необходимого числа переходов и его взаимодействия с нагрузкой на рабочую память был проведен двухфакторный дисперсионный анализ. Увеличение необходимого числа переходов естественным образом приводит к значимому увеличению общего времени навигации ($F(1, 1509) = 637{,}6$; $p < 0{,}001$). Оно также приводит к значимому снижению среднего времени одного перехода ($F(1, 1509) = 21{,}6$; $p < 0{,}001$). Таким образом, при увеличении "расстояния", которое необходимо преодолеть в меню, испытуемые увеличивают скорость выполнения переходов.

Наибольший интерес для исследования представляло существование статистического взаимодействия факторов дополнительной нагрузки и необходимого числа переходов. Это взаимодействие является значимым для общего времени навигации ($F(1, 1509) = 4{,}2$; $p < 0{,}05$). Оно обусловлено тем, что в условии с нагрузкой общее время навигации при большом необходимом числе переходов (три и — особенно отчетливо — четыре, см. рисунок, *а*) увеличивается больше, чем в условии без нагрузки.

Взаимодействие нагрузки и необходимого числа переходов для среднего времени перехода не значимо, однако оно приближается к значимому на уровне тенденции ($F(1, 1509) = 2{,}3$; $p = 0{,}13$). Примечательно, что при дополнительной нагрузке на рабочую память характерное увеличение скорости навигации при большом (четыре) необходимом числе переходов прекращается. В условии без нагрузки такого эффекта не наблюдается (см. рисунок, *б*). Для времени первого перехода взаимодействие нагрузки и необходимого числа переходов не является незначимым и не приближается к значимому на уровне тенденции ($F(1, 1509) = 0{,}02$; $p > 0{,}2$).

Взаимодействие нагрузки и необходимого числа переходов для количества реально выполненных переходов незначимо, однако оно приближается к значимому на уровне тенденции ($F(1, 1509) = 2{,}53$; $p = 0{,}11$). Наблюдающаяся тенденция к взаимодействию факторов обусловлена тем обстоятельством, что при четырех необходимых переходах в условии с нагрузкой наблюдается увеличение числа выполняемых переходов по сравнению с условием без нагрузки (см. рисунок, *в*).



Зависимость эффективности навигации от нагрузки на рабочую память испытуемого и необходимого числа переходов:
а — общее время навигации; *б* — среднее время одного перехода; *в* — число выполненных переходов. Условие с нагрузкой — сплошная линия; без нагрузки — штриховая линия. Вертикальные линии соответствуют доверительному интервалу (уровень доверия 95 %)

Обсуждение

Полученные результаты свидетельствуют о наличии влияния дополнительной задачи на скорость навигации в меню симулированного мобильного устройства. Необходимость выполнять дополнительную задачу приводит к выраженному увеличению общего времени выполнения задания. Увеличивается количество времени, которое испытуемые тратят на выполнение каждого перехода в среднем. Это может быть свидетельством того, что испытуемые вынуждены осуществлять навигацию медленнее, чаще сталкиваясь с необходимостью сознательного припоминания того, какой пункт меню должен быть выбран следующим. Увеличивается также количество времени, которое необходимо испытуемым для осуществления первого перехода. Это может быть свидетельством трудностей, которые испытуемые ощущают с припоминанием структуры меню и формированием корректного маршрута к целевой команде. Обе функции требуют активного использования ресурсов рабочей памяти, и полученные результаты подкрепляют предположение о важной роли рабочей памяти в осуществлении навигации в мобильном меню.

Дополнительная задача не оказывает статистически значимого влияния на другой важный показатель эффективности навигации — общее число совершенных переходов. Увеличение числа реально совершаемых испытуемыми переходов свидетельствует об "ошибках навигации". При выполнении многих видов задач типичной реакцией на усложнение условий работы или повышение трудности задачи является снижение скорости работы в целях сохранения субъективно приемлемого уровня ошибок. Представляется, что полученные результаты связаны со стремлением испытуемых поддерживать высокую точность работы за счет большего напряжения и низкой скорости.

Обнаруженное взаимодействие между нагрузкой на рабочую память и числом переходов, минимально необходимых для достижения целевой команды, представляется еще одним свидетельством влияния ограничений рабочей памяти на эффективность навигации в меню мобильного устройства. Этот эффект может быть связан как с усложнением планирования навигации, так и с усложнением ее непосредственного выполнения. Если ресурсы рабочей памяти используют при выполнении навигации, и если дополнительная задача уменьшает доступность ресурсов рабочей памяти, то дополнительная задача должна оказывать более выраженное негативное влияние на эффективность навигации в случае навигации на "большие расстояния". Обнаруженное для общего времени навигации статистически значимое взаимодействие нагрузки на рабочую память и необходимого числа переходов полностью соответствует этому предположению. С ним также согласуется и наблюдающаяся тенденция к существованию аналогичного взаимодействия для среднего времени одного перехода и для числа реально выполненных переходов.

Полученные результаты свидетельствуют о важной роли рабочей памяти пользователя в успешном выполнении навигации в меню мобильного устройства. Такая интерпретация поддерживается и результатами указанных выше зарубежных исследований. Она имеет значение для анализа взаимодействия пользователей с мобильными устройствами в реальных условиях, а также для разработки удобных в использовании мобильных устройств.

При реальном использовании мобильных устройств степень распределения ресурсов рабочей памяти может быть очень высокой. Это связано с самим контекстом использования подобных устройств. Оно часто осуществляется в динамичной, потенциально опасной среде, требующей постоянного мониторинга изменений ситуации и быстрых реакций на них. Использование мобильного устройства также часто происходит на фоне выполнения другой деятельности. Эта деятельность сама

по себе может создавать нагрузку на рабочую память. Поэтому в реальных условиях при навигации в меню (и взаимодействии с мобильным устройством в целом) нагрузка на рабочую память пользователя может быть значительно выше, чем в относительно стабильных условиях лабораторного тестирования.

Такое положение дел усугубляется тем, что мобильные устройства не предполагают систематического обучения пользователя. Они также не предполагают постоянного долговременного использования в неизменном контексте. Это препятствует формированию автоматизированных навыков использования, которые минимизируют потребность в сознательном контроле действий и в привлечении ресурсов рабочей памяти. Другим негативным фактором являются выраженные индивидуальные и возрастные различия в возможностях рабочей памяти. Если навигация в меню мобильного устройства требует привлечения значительных ресурсов рабочей памяти, то люди с низким объемом рабочей памяти, а также пожилые люди будут испытывать значительные трудности при выполнении этой задачи.

Вместе с тем высокая нагрузка на рабочую память при навигации в меню мобильного устройства может оказать негативное влияние на выполнение одновременно осуществляемой деятельности. В частности, это может затруднять мониторинг пользователем своего окружения. "Перегрузка" рабочей памяти пользователя вследствие плохо организованной навигации в меню мобильного устройства может привести к возникновению опасных для пользователя ситуаций. Поэтому учет ограничений рабочей памяти пользователей при разработке меню мобильных устройств повышает не только удобство, но и безопасность их использования.

Учитывать ограничения рабочей памяти при создании меню мобильного устройства можно несколькими способами. На этапе проектирования меню мобильного устройства следует обращать внимание на факторы, заведомо приводящие к увеличению нагрузки на рабочую память человека. Одним таким фактором является соотношение "глубины" и "ширины" меню (т. е. числа уровней меню и числа элементов на одном уровне меню). Учитывая типичные ограничения объема рабочей памяти и особенности контекста использования мобильных устройств, не следует использовать глубину меню более 3—4 уровней. Важным фактором также является симметричность структуры различных разделов и ветвей меню. Сходство организации значительно упрощает задачу ориентации в структуре, исключая необходимость сознательного припоминания особенностей конкретного подраздела меню. Также следует использовать очевидные для пользователей критерии распределения элементов по разделам меню и обозначения пунктов меню.

При пользовательском тестировании эффективности использования (прототипа) мобильного устройства целесообразно моделировать условия распределения ресурсов рабочей памяти, например через дополнительную нагрузочную задачу. При отборе пользователей для участия в тестировании можно измерять значения объема рабочей памяти для целенаправленного отбора лиц с высоким и низким объемами. Дополнительно может осуществляться регистрация психофизиологических показателей, коррелирующих с уровнем нагрузки на рабочую память. Например, увеличение нагрузки на рабочую память может проявляться в увеличении размера зрачка и специфических изменениях электрической активности мозга [6]. Сочетание снижения возможной нагрузки на рабочую память будущего пользователя на этапе проектирования меню мобильного устройства с оценкой эффективности навигации в условиях, моделирующих реальную ситуацию использования мобильного устройства, позволит оценить резерв ресурсов рабочей памяти, который будет доступен пользователю при работе с мобильным устройством.

Выводы

Навигация в меню мобильных вычислительных устройств предъявляет высокие требования к рабочей памяти пользователя. Рабочая память также привлекается к решению других задач, возникающих в "мобильном контексте использования". Повышенная нагрузка на рабочую память может приводить к снижению эффективности навигации в меню мобильных устройств, а также к увеличению риска попадания пользователя в опасные ситуации. Уровень нагрузки на рабочую память пользователей является важным критерием качества интерфейса при проектировании и оценке меню мобильных устройств.

Работа выполнена при поддержке РФФИ, грант № 11-06-00343а.

Список литературы

1. **Gartner** Says Worldwide Mobile Phone Sales Increased 16 Per Cent in 2007. URL: http://www.gartner.com/newsroom/id/612207

2. **Norman K.** The psychology of menu selection: designing cognitive control at the human/computer interface. Norwood, N. J.: Ablex Publishing Corporation, 1991.

3. **Paap K., Cooke N.** Design of menus//Handbook of HCI / M. Helander, T. K. Landauer eds. Amsterdam: Elsevier Science, 1997. P. 533—572.

4. **Баддли А., Айзенк М., Андерсон М.** Память. СПб.: Питер, 2011.

5. **Клингберг Т.** Перегруженный мозг. Информационный поток и пределы рабочей памяти. М.: Ломоносов, 2010.

6. **Величковский Б. Б., Козловский С. А.** Рабочая память человека: фундаментальные исследования и практические приложения // Интеграл. 2012. Т. 68, № 6. С. 14—16.

7. **Parush A., Yuviler-Gavish N.** Web navigation structures in cellular phones: the depth/breadth trade-off issue // International Journal of Human-Computer Studies. 2004. V. 60. P. 753—770.

8. **Sanchez A., Branaghan R.** Turning to learn: Screen orientation and reasoning with small devices // Computers in Human Behavior. 2011. № 27. P. 793—797.

9. **Ziefle M., Bay S.** Mental models of Cellular Phones Menu. Comparing older and younger novice users // Mobile Human Computer Interaction / S. Brewster, M. Dunlop eds. Heidelberg: Springer, 2004. P. 25—37.

10. **Шеффе Г.** Дисперсионный анализ. М.: Физматгиз, 1963.

УДК 004.82; 004.912; 81.322.2

Д. Е. Пальчунов, д-р физ.-мат. наук, доц., вед. науч. сотр., Институт математики СО РАН им. С. Л. Соболева, Новосибирский государственный университет, e-mail: palch@math.nsc.ru,
П. А. Степанов, аспирант, Новосибирский государственный университет,
e-mail: stefan.nsk@gmail.com

Применение теоретико-модельных методов извлечения онтологических знаний в предметной области информационной безопасности[1]

Работа посвящена методам извлечения онтологической информации из текстов естественного языка. Приведена теоретико-модельная формализация таких отношений между понятиями синонимия, "общее — частное", "одно понятие используется для определения другого понятия". Разработаны методы извлечения этих отношений, а также определений понятий из текстов естественного языка. Для анализа текстов использован язык описания лингвистических шаблонов. Разработанная программная система показала достаточно высокую точность и полноту извлекаемых знаний. Она применяется при разработках экспертной системы по информационной безопасности и системы управления рисками при обеспечении информационной безопасности.

Ключевые слова: *онтология предметной области, информационная безопасность, теоретико-модельные методы, логический анализ естественного языка, теория автоматов, недетерминированные автоматы, обработка текстов, извлечение знаний, извлечение определений терминов*

Введение

В 2001—2003 гг. перспективной казалась идея семантической паутины (*Semantic Web*) [1—3]. Это было очень похоже на то, как в середине 1980-х гг. в огромные надежды возлагались на ЭВМ пятого поколения, основанные на технологиях логического программирования [4—6]. Однако к середине 2000-х гг. надежды на серьезный прорыв в семантических технологиях стали таять [7], а сейчас на смену этим надеждам пришел достаточно большой скепсис по поводу применения семантических технологий в сети Интернет. Акцент переместился на обработку больших данных, основанную, в первую очередь, на технологиях анализа данных (*Data Mining*).

Технологии семантической паутины тесно связаны с применением онтологий [8—12]. Для целей разработки и представления онтологий был разработан язык OWL (*Web Ontology Language*) [13—15]. По мнению авторов, определенный неуспех семантической паутины связан с недостаточно детальной проработкой методов извлечения онтологических знаний, причем в первую очередь из текстов естественного языка. Не был решен вопрос о различии между онтологическим знанием и произвольным знанием о данной предметной области. По существу, этот вопрос не

[1] Исследование выполнено при поддержке Министерства образования и науки Российской Федерации, соглашение 14.B37.21.0400 "Методы извлечения и порождения знаний для обеспечения информационной безопасности" и гранта Сибирского отделения РАН "Принципы построения онтологии на основе концептуализаций средствами логических дескриптивных языков", № 3.

только не был решен, но и не был строго и формально поставлен.

С точки зрения авторов, главное отличие онтологического знания от произвольного, онтологии предметной области от теории предметной области состоит в следующем. Онтология содержит знание о смысле ключевых понятий, отражающих специфику данной предметной области. Теория содержит знание о предметной области, излагаемое в терминах этих ключевых понятиях. Путаница между знаниями о смысле терминов и собственно знаниями о предметной области, по мнению авторов, является одной из основных проблем онтологического подхода в инженерии знаний и, соответственно, одной из главных причин неуспехов онтологического подхода на настоящее время.

Данная статья посвящена вопросам формального описания онтологического знания и разработке методов извлечения онтологических знаний из текстов естественного языка.

При попытке автоматизации процесса извлечения информации из текстов естественного языка исследователь неизбежно сталкивается с рядом сложно разрешимых задач. Они в основном связаны со сложностями анализа структур естественного языка, не допускающих точной формализации. Часто тексты содержат большое число различного типа ошибок — от опечаток до ошибок в построении структуры предложений. На каждом этапе анализа возникает значительное число альтернативных вариантов его реализации. Все это усложняет алгоритмы анализа, которые применяют для автоматизации обработки текстов. Тем не менее именно тексты естественного языка содержат большой объем полезной и относительно доступной информации. Подавляющее большинство знаний, как онтологических, так и теоретических, опубликовано в виде текста. Именно по этой причине задачи автоматизации обработки текстов и извлечения из них знаний очень актуальны.

Отметим, что разрабатываемые методы применяют для извлечения определений ключевых понятий в предметной области информационной безопасности [16]. Такие методы необходимы, например, для решения задач извлечения знаний из текстов на естественном языке, описывающих возможные компьютерные угрозы. На следующем этапе исследований авторы планируют разработку программной системы, поддерживающей обеспечение информационной безопасности предприятия. В результате извлечения знаний по информационной безопасности из различных текстов естественного языка, представленных в сети Интернет (ленты RSS-новостей и другие источники), строится класс алгебраических систем, формализующих извлеченные из текстов знания. На основе этих алгебраических систем разрабатывается экспертная система по информационной безопасности, являющаяся составной частью системы управления рисками при обеспечении информационной безопасности предприятия.

Теоретико-модельный подход к формализации онтологического знания

Необходимые теоретико-модельные определения и утверждения можно найти в работах [17, 18].

Без ограничения общности, в качестве математической формализации понятий в данной работе будем рассматривать только предикаты, поскольку функции и константы могут быть выражены через предикаты. Поэтому сигнатурой далее будем называть набор $\sigma = \langle P_1, ..., P_n \rangle$, где $P_1, ..., P_n$ — символы предикатов, местность (арность) которых может быть различна. Через $S(\sigma)$ обозначим множество всех предложений (т. е. формул без свободных переменных) сигнатуры σ. Через $\sigma(\varphi)$ обозначим сигнатуру предложения φ, т. е. множество всех сигнатурных символов, входящих в φ; $\sigma(\Gamma)$ обозначает сигнатуру множества предложений Γ. Обозначим $\Gamma \vdash \varphi$, если из множества предложений Γ выводимо предложение φ.

Напомним, что теорией называется дедуктивно замкнутое множество предложений, т. е. множество предложений, в котором для каждого предложения $\varphi \in S(\sigma(T))$, если $T \vdash \varphi$, то $\varphi \in T$. Для множества предложений Γ обозначим $Th(\Gamma) = \{\varphi \in S(\sigma(\Gamma)) | \Gamma \vdash \varphi\}$ — теория, аксиоматизируемая множеством предложений Γ (в сигнатуре множества предложений Γ).

В работах [19, 20] авторами было дано теоретико-модельное определение онтологии предметной области.

Определение 1. Формальной онтологией предметной области назовем пару $O = \langle \Gamma, \sigma \rangle$, где σ — множество ключевых понятий предметной области, а Γ — множество аналитических предложений, описывающих смысл ключевых понятий. Множество T предложений, которые являются верными в каждом экземпляре предметной области, назовем теорией предметной области.

Здесь понятие аналитического предложения соответствует классификации Р. Карнапа истинности высказываний [21]. Предложение называется логически истинным, если его значение истинности зависит только от его логической формы (например, $\varphi \vee \neg \varphi$ или $\varphi \rightarrow \varphi$). Предложение является аналитическим, если его значение истинности полностью определяется смыслом понятий, входящих в это предложение. Оставшиеся предложения, для определения истинности которых необходимо обращаться к реальному миру, называют синтетическими.

В настоящей работе в определении онтологии будем считать, что $\sigma = \sigma(\Gamma)$. Поэтому для простоты онтологией здесь будем называть множество предложений Γ, определяющих смысл понятий из $\sigma(\Gamma)$.

При помощи $\boldsymbol{x}$ будем обозначать кортеж переменных $\langle x_1, ..., x_n \rangle$. Для предложения φ через $[\varphi]^Q_P$ обозначим предложение, полученное из φ заменой всех вхождений предикатного символа Q на предикатный

символ P. Для множества предложений Γ обозначим $[\Gamma]_P^Q = \left\{ [\varphi]_P^Q \mid \varphi \in \Gamma \right\}$.

Определим в теоретико-модельных терминах отношения синонимии и "общее-частное" на множестве понятий.

Определение 2. Рассмотрим онтологию Γ и два понятия P, $Q \in \sigma(\Gamma)$. Будем считать, что понятия P и Q являются *абсолютными синонимами* в онтологии Γ, если выполнено

$$\Gamma \vdash \forall \boldsymbol{x}(P(\boldsymbol{x}) \leftrightarrow Q(\boldsymbol{x})).$$

Будем считать, что понятия P и Q являются *относительными синонимами* в онтологии Γ, если выполнено $[Th(\Gamma) \cap S(\sigma(\Gamma)\backslash\{P\})]_P^Q = Th(\Gamma) \cap S(\sigma(\Gamma)\backslash\{Q\})$.

Замечание 1. Пусть P, $Q \in \sigma(\Gamma)$. Выполнено $[Th(\Gamma) \cap S(\sigma(\Gamma)\backslash\{P\})]_P^Q = Th(\Gamma) \cap S(\sigma(\Gamma)\backslash\{Q\})$ тогда и только тогда, когда выполнено $[Th(\Gamma) \cap S(\sigma(\Gamma)\backslash\{Q\})]_P^Q = Th(\Gamma) \cap S(\sigma(\Gamma)\backslash\{P\})$.

Замечание 1 показывает, что в Определение 2 понятия P и Q входят симметрично.

Два понятия являются абсолютными синонимами в онтологии Γ, если в теории $Th(\Gamma)$ они логически эквивалентны. Два понятия являются относительными синонимами в онтологии Γ, если понятие P входит в те же самые предложения, истинные в теории $Th(\Gamma)$, что и понятие Q. Это означает, что употребление понятий P и Q в онтологии Γ неразличимо.

Если понятия P и Q являются абсолютными синонимами, то это отношение, во-первых, не изменяется с течением времени (в частности, при добавлении новых знаний) и, во-вторых, оно сохраняется при погружении онтологии Γ в большую онтологию, содержащую больший набор понятий. Примерами абсолютных синонимов являются бегемот и гиппопотам, тетраэдр и треугольная пирамида, псевдобулева алгебра и гейтингова алгебра. В отличие от абсолютных, относительные синонимы могут перестать быть синонимами как при добавлении новой информации о понятиях, содержащихся в данной онтологии, так и при расширении онтологии до большей, содержащей новые понятия. Такие синонимы встречаются более часто, чем абсолютные. Большое количество их представляют пары: русский термин и латинизм. К их числу относятся, например, следующие пары: явный — эксплицитный; неявный — имплицитный; болезнь — патология; предикат — отношение; трехместный — тетрарный. Отношение синонимии таких терминов может не сохраняться, поскольку при расширении онтологии изменяется контекст рассмотрения, а вместе с ним расширяется и смысл понятий, добавляются возможные способы их употребления. Вместо "это явная глупость" никто не скажет "это эксплицитная глупость". Предикат является двухместным тогда и только тогда, когда он является бинарным, однако вместо "трехместный мотоцикл" нельзя сказать "тетрарный мотоцикл". Таким образом, при расширении онтологии тождество $[Th(\Gamma) \cap S(\sigma(\Gamma)\backslash\{P\})]_P^Q = Th(\Gamma) \cap S(\sigma(\Gamma)\backslash\{Q\})$ может перестать быть истинным.

Вместе с тем истинность утверждения $\Gamma \vdash \forall \boldsymbol{x}(P(\boldsymbol{x}) \leftrightarrow Q(\boldsymbol{x}))$ сохраняется при расширении онтологии Γ до более богатой онтологии $\Gamma' \supseteq \Gamma$: в этом случае из $\Gamma \vdash \forall \boldsymbol{x}(P(\boldsymbol{x}) \leftrightarrow Q(\boldsymbol{x}))$ вытекает $\Gamma' \vdash \forall \boldsymbol{x}(P(\boldsymbol{x}) \leftrightarrow Q(\boldsymbol{x}))$. Отсюда следует, что если в исходную онтологию Γ включить утверждение $\forall \boldsymbol{x}(P(\boldsymbol{x}) \leftrightarrow Q(\boldsymbol{x}))$, то онтология Γ будет несовместна (противоречива) с любой онтологией, в которой это утверждение уже не является верным. В частности, в этом случае нельзя погрузить онтологию Γ в более широкую онтологию, в которой понятия P и Q не являются полными синонимами. Принимая во внимание изложенное выше, во многих случаях более естественно требовать, чтобы понятия были не абсолютными, а относительными синонимами.

Наряду с синонимией, среди отношений между понятиями особое значение имеет отношение "общее—частное". В частности, это отношение является центральным в методологии объектно-ориентированного программирования. Дадим теоретико-модельную формализацию этого отношения.

Определение 3. Рассмотрим онтологию Γ и два понятия P, $Q \in \sigma(\Gamma)$. Будем считать, что понятие P является *абсолютным частным* для понятия Q в онтологии Γ, если выполнено

$$\Gamma \vdash \forall \boldsymbol{x}(P(\boldsymbol{x}) \rightarrow Q(\boldsymbol{x})).$$

Будем считать, что понятие P является *относительным частным* для понятия Q в онтологии Γ, если выполнено $[Th(\Gamma) \cap S(\sigma(\Gamma)\backslash\{P\})]_P^Q \subseteq Th(\Gamma) \cap S(\sigma(\Gamma)\backslash\{Q\})$.

Для абсолютного частного и относительного частного имеет место та же ситуация, как и для абсолютных синонимов и относительных синонимов. Если понятие P является абсолютным частным для понятия Q в онтологии Γ, то это отношение сохранится при расширении этой онтологии. Парами "абсолютное частное — общее" являются параллелограмм — четырехугольник, четырехугольник — многоугольник, куропатка — птица, вольфрам — металл. Вместе с тем понятия могут перестать находиться в отношении "частное — общее" при изменении контекста, в частности, при расширении онтологии. Такими парами понятий являются вирус — вредоносная программа, червь — вредоносная программа (информационная безопасность), поле — кольцо (алгебра). Очевидно, что в широкой онтологии, включающей в себя агрономию, не каждое поле имеет форму кольца, а черви не являются вредоносными программами.

Рассмотрим онтологию Γ и понятия P, $Q \in \sigma(\Gamma)$.

Замечание 2. Понятия P и Q являются абсолютными (относительными) синонимами в онтологии Γ тогда и только тогда, когда в онтологии Γ понятие P является абсолютным (соответственно, относительным) частным для понятия Q, и понятие Q является абсолютным (соответственно, относительным) частным для понятия P.

Замечание 3. Если понятие P является абсолютным частным для понятия Q в онтологии Г, то в онтологии Г понятие P является относительным частным для понятия Q.

Следствие 1. В онтологии Г, если понятия P и Q являются абсолютными синонимами, то P и Q являются относительными синонимами.

Рассмотрим более подробно отношение "общее — частное". Как было отмечено выше, это отношение зависит от контекста, в частности, от онтологии, в рамках которой оно рассматривается. Зафиксируем некоторую онтологию и соответствующую ее теорию.

Заметим, что для некоторых понятий есть более частные понятия, а для других более частных понятий нет. Например, если рассмотреть понятие "функция", то у него есть более частное понятие — "непрерывная функция", у которого, в свою очередь, есть более частное понятие — "дифференцируемая функция", далее "непрерывно дифференцируемая функция", "дважды дифференцируемая функция" и т. д. Вместе с тем у понятия "функция $\sin x$" более частного понятия нет. Возникает вопрос: для каких понятий не существует более частного понятия, т. е. какие понятия являются финальными относительно отношения "общее —частное"? Очевидно, что решение этого вопроса зависит от того, рассматриваем абсолютное или относительное отношение "общее—частное".

Определение 4. Рассмотрим онтологию Г. Пусть $Q \in \sigma(\Gamma)$. Понятие Q назовем финальным для абсолютного (относительного) отношения "общее — частное", если для любой онтологии $\Gamma' \supseteq \Gamma$ и любого понятия $P \in \sigma(\Gamma')$ таких, что $Th(\Gamma') \cap S(\sigma(\Gamma)) = Th(\Gamma)$ и понятие P является абсолютным (соответственно, относительным) частным для понятия Q в онтологии Γ', понятие Q является абсолютным (соответственно, относительным) частным для P в онтологии Γ', т. е. в онтологии Γ' понятия P и Q являются абсолютными (соответственно, относительными) синонимами.

Требование $Th(\Gamma') \cap S(\sigma(\Gamma)) = Th(\Gamma)$ означает, что онтология Γ' консервативно расширяет онтологию Г, т. е. Γ' не содержит новой информации о понятиях из Г, не переопределяет их смысл. Это требование необходимо, поскольку без него определение "более частного" понятия P в онтологии Γ' может содержать дополнительную информацию, не имеющую никакого отношения к смыслу понятия Q. Из последнего вывода вытекает, в частности, что без требования консервативности расширения онтологии необходимым условием существования финальных понятий была бы полнота теории $Th(\Gamma)$, что никогда не имеет места на практике.

Предложение 1. Для любой онтологии Г и любого понятия $Q \in \sigma(\Gamma)$ существует онтология $\Gamma' \supseteq \Gamma$ и понятие $P \in \sigma(\Gamma')$ такие, что $Th(\Gamma') \cap S(\sigma(\Gamma)) = Th(\Gamma)$, понятие P является абсолютным частным для понятия Q в онтологии Γ', а понятие Q не является абсолютным частным для P в онтологии Γ'.

Доказательство. Пусть $\Gamma' = \Gamma \cup \{\forall \boldsymbol{x}(P(\boldsymbol{x}) \rightarrow Q(\boldsymbol{x}))\}$. Непосредственно проверяется выполнение всех условий Предложения 1.

Из Предложения 1 вытекает, что для любой онтологии Г никакое понятие $Q \in \sigma(\Gamma)$ не является финальным для абсолютного отношения "общее — частное". Таким образом, для абсолютного отношения "общее — частное" финальных понятий не существует. Причем доказательство Предложения 1 показывает, что это связано не с природой отношений между понятиями, а с определенной неестественностью абсолютного отношения "общее — частное" (хотя, на первый взгляд, именно такая формализация отношения "общее — частное" может показаться наиболее естественной). Теперь рассмотрим относительный вариант отношения "общее — частное".

Теорема 1. Для любой онтологии Г и любого понятия $Q \in \sigma(\Gamma)$ существуют онтология $\Gamma' \supseteq \Gamma$ и понятие $P \in \sigma(\Gamma')$ такие, что $Th(\Gamma') \cap S(\sigma(\Gamma)) = Th(\Gamma)$, и выполнены следующие условия:

- понятие P является относительным частным для понятия Q в онтологии Γ';
- понятие P является финальным для относительного отношения "общее — частное".

Доказательство проводится по схеме, подобной доказательству того, что любая непротиворечивая теория вкладывается в полную непротиворечивую теорию или того, что любое множество формул от фиксированного набора переменных, совместное с данной теорией, может быть дополнено до реализуемого типа. Необходимо только выполнить требование сохранения консервативности.

Заметим, что в результате построения, проводимого в доказательстве Теоремы 1, получается множество формул, которое является в определенном смысле полным. Де-факто строится максимальный из фильтров, консервативно расширяющих теорию $Th(\Gamma)$. Понятно, что в общем случае этот фильтр не обязательно будет главным, т. е. построенная теория не обязательно будет конечно аксиоматизируемой. Этот факт означает, что в общем случае полученное финальное понятие может не иметь явного определения. По существу, это обстоятельство является побудительным мотивом для формализации и порождения новых понятий, набор которых породит новую онтологию. Эта расширенная онтология будет являться более мощным инструментарием исследования и описания реальности. В ситуации, когда имеет место бесконечный набор свойств, вводится новое понятие, позволяющее единым образом указывать на этот бесконечный набор. Введение таких понятий позволяет при помощи конечной записи манипулировать бесконечными наборами утверждений.

Таким образом, в отличие от абсолютного, для относительного отношения "общее — частное" любое понятие имеет финальный частный случай. Этот факт является очень важным как для решения проблемы построения онтологий предметных областей, так и

для приложений, таких как задачи объектно-ориентированного моделирования.

С точки зрения авторов, наибольший интерес представляет рассмотрение именно относительных синонимов и относительного отношения "общее — частное". Во-первых, такие отношения между понятиями встречаются значительно более часто. Во-вторых, при расширении и интеграции онтологий очень важно учитывать тот факт, что синонимы исходной онтологии могут перестать быть синонимами, точно так же, как и имеющееся отношение "общее — частное" может не сохраняться. В-третьих, как показывает Теорема 1, относительные отношения являются наиболее естественной формализацией взаимосвязи между понятиями. С учетом изложенного выше, далее в первую очередь будем рассматривать относительные синонимы и относительные частные понятий. При рассмотрении и формализации других отношений между понятиями будем придерживаться такого же подхода.

Наряду с очень популярным отношением между понятиями "общее — частное" с точки зрения разработки онтологий большой интерес представляет также другое отношение: "понятие P используется для определения понятия Q". Рассмотрим это отношение между понятиями более подробно.

Первый вопрос, который подлежит изучению: будет ли это отношение антисимметричным? Другими словами, верно ли, что если понятие P используется для определения отличного от него понятия Q, то понятие Q не может быть использовано для определения понятия P?

Сначала введем несколько необходимых определений.

Определение 5. Рассмотрим онтологию Г, сигнатуру $\sigma_1 \subseteq \sigma(\Gamma)$ и понятие $P \in \sigma(\Gamma)$. Будем считать, что понятие P определимо в сигнатуре σ_1, если существует предложение $\varphi \in S(\sigma_1 \cup \{P\})$ такое, что $\{\psi \in S(\sigma_1 \cup \{P\}) | \{\varphi\} \cup (Th(\Gamma) \cap S(\sigma_1)) \vdash \psi\} = Th(\Gamma) \cap S(\sigma_1 \cup \{P\})$.

Определимость понятия P в сигнатуре σ_1 означает следующее: используя только знания о смысле понятий из σ_1, представленные в онтологии Г (это в точности $Th(\Gamma) \cap S(\sigma_1)$), при помощи одного предложения φ можно полностью описать смысл понятия P, а также связь понятия P с понятиями, содержащимися в сигнатуре σ_1.

Замечание 4. Пусть $\sigma' \subseteq \sigma(\Gamma)$ и $P \in \sigma'$. Если теория $Th(\Gamma) \cap S(\sigma')$ конечно аксиоматизируема, то понятие P определимо в сигнатуре $\sigma' \backslash \{P\}$.

Далее потребуется обобщение Определения 5 на случай набора из нескольких определяемых понятий.

Определение 6. Рассмотрим онтологию Г и сигнатуры σ_1, $\sigma_2 \subseteq \sigma(\Gamma)$. Будем считать, что понятия из σ_2 определимы в сигнатуре σ_1, если существует предложение $\varphi \in S(\sigma_1 \cup \sigma_2)$ такое, что $\{\psi \in S(\sigma_1 \cup \sigma_2) \mid \{\varphi\} \cup (Th(\Gamma) \cap \cap S(\sigma_1)) \vdash \psi\} = Th(\Gamma) \cap S(\sigma_1 \cup \sigma_2)$.

Предложение ψ является совместным определением всего набора понятий из σ_2 (и их взаимосвязи с понятиями из σ_1) на основе знания о смысле понятий из σ_1, представленного в онтологии Г.

В работе [22] было введено понятие формального глоссария.

Определение 7. Пусть σ — сигнатура. Последовательность предложений $\langle \varphi_1, ..., \varphi_n \rangle$ назовем *формальным глоссарием* (*определяющим понятия из* σ), если выполнены следующие условия:

- $\sigma(\{\varphi_1\}) \subset \sigma(\{\varphi_1, \varphi_2\}) \subset ... \subset \sigma(\{\varphi_1, ..., \varphi_n\}) = \sigma$;
- добавление каждого нового предложения φ_{k+1} консервативно расширяет предыдущий набор предложений $\varphi_1, ..., \varphi_k$, т. е.

$$Th(\{\varphi_1, ..., \varphi_n\}) \cap S(\sigma(\{\varphi_1, ..., \varphi_k\})) = Th(\{\varphi_1, ..., \varphi_k\}).$$

Считаем, что формальный глоссарий $\langle \varphi_1, ..., \varphi_n \rangle$ определяет понятия из онтологии Г, если $Th(\Gamma) = Th(\{\varphi_1, ..., \varphi_n\})$.

В работах [22, 23] был исследован вопрос: всегда ли можно построить определение новых понятий по одному, т. е. таким образом, чтобы каждое определение из глоссария содержало бы ровно одно новое понятие?

Определение 8. Будем считать, что формальный глоссарий $\langle \varphi_1, ..., \varphi_n \rangle$ определяет понятия *по одному*, если для любого k множество $\sigma(\{\varphi_1, ..., \varphi_{k+1}\}) \backslash \sigma(\{\varphi_1, ..., \varphi_k\})$ является одноэлементным.

В работах [22, 23] был дан отрицательный ответ на отмеченный выше вопрос.

Теорема из работ [22, 23]. Существуют конечная онтология Г с двухэлементной сигнатурой $\sigma(\Gamma)$, для которой нет формального глоссария, определяющего понятия по одному.

Далее усилим этот результат и одновременно покажем, что отношение "понятие P используется для определения понятия Q" не является антисимметричным.

Теорема 2. Для любых n и k существует конечная онтология Г с $\sigma(\Gamma) = \langle P_1, ..., P_n, Q_1, ..., Q_k \rangle$, где $P_1, ..., P_n$ — символы одноместных предикатов, а $Q_1, ..., Q_k$ — символы двухместных предикатов, такая, что выполняются следующие условия:

- не существует глоссария, определяющего понятия "по одному", который бы представлял онтологию Г;
- ни одно из понятий из сигнатуры $\sigma(\Gamma)$ не определимо в пустой сигнатуре;
- любая пара понятий $\{P_i, Q_j\}$ определима в пустой сигнатуре;
- любое понятие из $\sigma(\Gamma)$ определимо в сигнатуре $\{P_i, Q_j\}$;
- для любой сигнатуры $\sigma_1 \subseteq \sigma(\Gamma)$ если $\sigma_1 \cap \{P_1, ..., P_n\} \neq \varnothing$ и $\sigma_1 \cap \{Q_1, ..., Q_k\} \neq \varnothing$, то в сигнатуре σ_1 определимо любое понятие из $\sigma(\Gamma)$.

Следствие 2. Существуют онтологии Г со сколь угодно большими наборами понятий $\sigma(\Gamma)$ такие, что для любых двух понятий из $\sigma(\Gamma)$ найдутся два глоссария, определяющих понятия из онтологии Г, таких, что в одном из них первое понятие определяется через второе, а в другом — второе понятие через первое.

Таким образом, отношение "понятие P используется для определения понятия Q" не является антисимметричным.

Определение 9. Формальный глоссарий $\langle\varphi_1, ..., \varphi_n\rangle$ назовем строгим, если для любого k множество $\sigma(\{\varphi_1, ..., \varphi_{k+1}\})\backslash\sigma(\{\varphi_1, ..., \varphi_k\})$ является непустым.

Строгий глоссарий отличается тем, что каждая его статья (предложение φ_k) определяет хотя бы одно новое понятие.

Выше было показано, что не всегда понятия, представленные в онтологии, можно определять по одному. Возникает вопрос: по сколько понятий минимально можно определять за один шаг? Ответ на этот вопрос дает следующая теорема.

Теорема 3. Для любого n существует конечная онтология Γ_n, сигнатура $\sigma(\Gamma_n) = \langle Q, P_1, ..., P_n\rangle$ которой состоит из символов трехместного предиката Q и двухместных предикатов $P_1, ..., P_n$, такая, что для любой сигнатуры $\sigma_1 \subset \sigma(\Gamma_n)$ понятия из σ_1 не могут быть определены конечным набором предложений. В частности, не существует формального глоссария, определяющего понятия из σ_1.

Таким образом, понятия из $\sigma(\Gamma_n)$ могут быть определены только все сразу. И в онтологии таких понятий может быть сколь угодно много. Поэтому нельзя ограничить сверху минимальное число понятий, которые определяются на каждом шаге (в каждой статье) глоссария для данной онтологии.

Математическая модель описания алгоритмов анализа текстов

Для описания математических моделей алгоритмов анализа и распознавания текстов на естественном языке в данной статье будут использоваться понятия, определенные в работе [24]. Эти понятия тесно связаны с теорией автоматов, изложенной в работе [25].

Использование недетерминированных автоматов позволяет решить ряд задач и устранить неудобства, которые неизбежно возникают в системах автоматической обработки текста, подобных системам, описанным в работах [26, 27].

На основе полученных результатов были разработаны и программно реализованы алгоритмы извлечения онтологической информации из текстов естественного языка. Описание программной реализации алгоритмов извлечения онтологической информации приведено в следующих двух разделах.

Приведем определения используемых в статье понятий, касающиеся недетерминированных автоматов, а также сформулируем ряд утверждений, часть которых была доказана в работе [24].

Определение 10. Недетерминированным автоматом называется кортеж $\langle S, X, Y, S_0, \delta\rangle$, где:

S — непустое множество состояний автомата;

X — непустое множество входных сигналов (входной алфавит);

Y — непустое множество выходных сигналов (выходной алфавит);

$S_0 \subseteq S$ — множество начальных состояний;

$\delta: S \times X \to P(S \times Y)$ — функция переходов.

Определение 11. Рассмотрим недетерминированный автомат $A = \langle S, X, Y, S_0, \delta\rangle, x_1, ..., x_n$, где $x_i \in X$ — последовательность символов входного алфавита, $y_1, ..., y_n$, где $y_i \in Y$ — последовательность символов выходного алфавита. Будем считать, что автомат преобразует входную последовательность $x_1, ..., x_n$ в выходную последовательность $y_1, ..., y_n$, если $\exists s_0, s_1, ..., s_k, s_i \in S$: $s_0 \in S_0$ и $\forall i (s_i, y_i) \in \delta(s_{i-1}, x_i)$. Последовательность $y_1, ..., y_n$ будем называть результатом работы автомата A на входной последовательности $x_1, ..., x_n$. Множество всех возможных последовательностей $y_1, ..., y_n$, являющихся результатом работы автомата A на входной последовательности $x_1, ..., x_n$, будем называть множеством результатов работы автомата A на входной последовательности $x_1, ..., x_n$.

Определение 12. Рассмотрим недетерминированный автомат $A = \langle S, X, Y, s_0, \delta\rangle$ и последовательность $x_1, ..., x_n$, где $x_i \in X$. Множеством интерпретаций по $x_1, ..., x_n$ автоматом A будем называть множество $I = \{(k, s, t, y) \in N \times S \times S \times Y\}$ такое, что $\forall (k, s, t, y) \in$ $\in I\ \exists s_0, s_1, ..., s_k \in S\ \exists y_1, ..., y_k \in Y$ для которых выполнено $(k, s, t, y) = (k, s_{k-1}, s_k, y_k)$, $s_0 \in S_0$, $\forall i: (s_i, y_i) \in$ $\in \delta(s_{i-1}, x_i)$; N — множество натуральных чисел.

Определение 13. Рассмотрим недетерминированный автомат $A = \langle S, X, Y, S_0, \delta\rangle$ и последовательность $x_1, ..., x_n$, где $x_i \in X$, и $I = \{(k, s, t, y) \in N \times S \times S \times Y\}$ — множество интерпретаций последовательности $x_1, ..., x_n$ автоматом A. Будем называть множество всех последовательностей вида $\{z_1, ..., z_n \vee \forall s_i \in S: \forall i: (i, s_{i-1}, s_i, z_i) \in I\}$ восстановленным по I множеством результатов работы автомата A на входной последовательности $x_1, ..., x_n$.

Предложение 2. Рассмотрим недетерминированный автомат $A = \langle S, X, Y, S_0, \delta\rangle$, последовательность $x_1, ..., x_n$, где $x_i \in X$, и $I = \{(k, s, t, y) \in N \times S \times S \times Y\}$ — множество интерпретаций последовательности $x_1, ..., x_n$ автоматом A. Тогда восстановленное множество результатов совпадает с множеством результатов работы автомата A на входной последовательности $x_1, ..., x_n$.

Определение 14. Рассмотрим $A = \langle S, X, Y, S_{A0}, \delta_A\rangle$ и $B = \langle S_B, Y, Z, S_{B0}, \delta_B\rangle$ — недетерминированные автоматы, $x_1, ..., x_n$ — последовательность символов, где $x_i \in X$. Обозначим $I_A = \{(k, s_A, t_A, y) \in N \times S_A \times S_A \times Y\}$ — множество интерпретаций автоматом A последовательности $x_1, ..., x_n$. Множество $I_B = \{(k, \bar{s}, \bar{t}, z) \in$ $\in N \times S_{AB} \times S_{AB} \times Z\}$, где $S_{AB} = S_A \times S_B$ будем называть множеством интерпретаций автоматом B множества интерпретаций I_A, для которых выполнено $(k, \bar{s}, \bar{t}, z) = (k, (t^A_{k-1}, t^B_{k-1}), (t^A_k, t^B_k), z_k)$, $t^A_0 \in S_{A0}$, $t^B_0 \in S_{B0}$, $\forall i: (t^B_i, z_i) \in \delta(t^B_{i-1}, y_i)$ и $\forall i: (i, t^A_{i-1}, t^A_i, y_i) \in$ $\in I_A$.

Предложение 3. Рассмотрим $A = \langle S, X, Y, S_{A0}, \delta_A\rangle$ и $B = \langle S_B, Y, Z, S_{B0}, \delta_B\rangle$ — недетерминированные автоматы, $x_1, ..., x_n$ — последовательность символов, где $x_i \in X$, $I_A = \{(k, s_A, t_A, y) \in N \times S_A \times S_A \times Y\}$ — множество

интерпретаций автоматом A последовательности $x_1, ..., x_n$, $I_B = \{(k, s, t, z) \in N \times S_{AB} \times S_{AB} \times Z\}$ — множество интерпретаций автоматом B множества интерпретаций I_A. В таком случае восстановленное по I_B множество результатов автомата B совпадает с объединением всех множеств результатов работы автомата B на последовательностях, принадлежащих множеству результатов работы автомата A на входной последовательности $x_1, ..., x_n$.

Выше был определен класс недетерминированных автоматов, а также введена новая структура данных — множество интерпретаций, позволяющая хранить результаты работы подобных автоматов. Предложения 2 и 3 показывают, что введенные множества интерпретаций могут быть использованы не только в качестве хранилища результатов, но также и в качестве хранилища исходных данных для недетерминированных автоматов из цепочки анализа.

Более подробно приведенные понятия и их свойства были рассмотрены в работе [24].

Извлечение явных определений из текстов естественного языка

В рамках программной реализации описанного выше подхода алгоритмы анализа текста были представлены в виде недетерминированных автоматов, обрабатывающих в качестве входных данных некоторое множество интерпретаций и генерирующих другое множество интерпретаций в качестве результата.

Целью разработанной программной системы является поиск явных определений понятий в текстах естественного языка, а также извлечение отношений между этими понятиями. Множество извлеченных терминов и отношений между ними необходимо для создания онтологий предметных областей. Сложности создания таких онтологий описаны в работах [22—24].

Процесс поиска терминов разбит на несколько этапов. На каждом из этих этапов анализа обработка данных осуществляется при помощи недетерминированного автомата, преобразующего описанным выше способом множество интерпретаций, являющееся результатом предыдущего этапа анализа.

Процесс поиска и извлечения определений из текстов естественного языка состоит из следующих этапов.

1. Извлечение текста из HTML-документа. На этом этапе входной HTML-документ преобразуется в поток интерпретаций, в котором все служебные строки и символы интерпретируются как пробельные символы.

2. Разбиение текста на слова, предложения. На этом этапе входной текст преобразуется в множество интерпретаций, соответствующих найденным в тексте словам. Другими словами, данный автомат ищет и отмечает найденные слова в тексте. Кроме того, отмечаются некоторые знаки препинания.

3. Выделение в тексте незначащих конструкций. Найденные в тексте незначащие конструкции интерпретируются как пробельные символы, что порождает в дальнейшем альтернативные ветки анализа, в которых эти конструкции проигнорированы.

4. Морфологический анализ слов текста. Найденные в тексте слова дополняются морфологическими характеристиками.

5. Синтаксический анализ. Во входном множестве интерпретаций ведется поиск грамматических конструкций языка, соответствующих явным определениям терминов.

Разработанная система использует морфологический анализатор, основанный на словоизменительном словаре русского языка [28]. Работа грамматического модуля основана на материалах книги [29], где описаны принципы, лежащие в основе грамматики русского языка.

На вход цепочке автоматов поступает текст естественного языка, представленный в виде HTML-документа. В результате работы цепочка возвращает множество интерпретаций, содержащее в качестве элементов участки текста, которые были распознаны как явные определения некоторых терминов.

В качестве критериев для поиска грамматических конструкций, соответствующих явным определениям, используют выражения, записанные на языке описания лингвистических шаблонов. Эти выражения позволяют выразить условия на словарный состав, а также потребовать соблюдения различных грамматических связей внутри искомой конструкции. Грамматике этого языка, а также особенностям его применения для поиска грамматических конструкций в тексте, посвящена работа [30].

Для извлечения явных определений используется шаблон "объект", представленный далее. Этот шаблон отвечает за поиск всех словосочетаний в тексте.

```
// базовые шаблоны
ADJ(число, род, падеж)   : = < *:"ПРИЛ"/"ПРИЧ":$число:$род:$падеж > ;
NOUN(число, род, падеж)  : = < *:"СУЩ":$число:$род:$падеж > ;
ADVERD()                 : = < *:"НАРЕЧ" > ;

// связь согласования
AGREEMENT(число, род, падеж) : =
   ADVERD() * ADJ($число, $род, $падеж) * NOUN($число, $род, $падеж);

// шаблон "объект", основанный связью подчинения
OBJECT(число, род, падеж) : =
   AGREEMENT($число, $род, $падеж) AGREEMENT(*, *, "рд.п.") *;
```

На основе шаблона "объект" строятся более сложные шаблоны, отвечающие за извлечение явных определений. Ниже приведены примеры некоторых из них.

```
// что-то это что-то
DEFINITION_IS_1(число, род, падеж) : =
   OBJECT(*, *, "им.п.") < "это" > OBJECT(*, *, "им.п.") *;

// что-то является чем-то
DEFINITION_IS_2(число, род, падеж) : =
   OBJECT(*, *, "им.п.") < "являться":"ГЛАГ":"Зл." > OBJECT(*, *, "тв.п.") *;

// будем называть чем-то что-то
DEFINITION_WILL_CALL(число, род, падеж) : =
   < "быть":"ГЛАГ":"буд.вр." > < "называть":"инф." >
   OBJECT(*, *, "тв.п.") OBJECT(*, *, "им.п.") *;
```

Полученная система применялась для поиска определений терминов из области информационной безопасности. После выполнения анализа были найдены 90 % явных определений от общего количества явных определений, найденных человеком. Доля ошибок составила 15 %. В текстах встречались также неявные определения, задача извлечения которых решается в рамках развития разработанной системы.

Приведем некоторые примеры извлеченных определений терминов.

- "Компьютерный вирус — разновидность компьютерных программ или вредоносный код ..."
- "Файловые вирусы — это вирусы-паразиты, которые при распространении своих копий ..."
- "Загрузочные вирусы — это компьютерные вирусы, записывающиеся в первый сектор ..."
- "Так, переменная в подпрограмме PERVADE ..., называлась VIRUS".
- "Черви — вид вирусов, которые проникают на компьютер-жертву без участия пользователя".
- "Уязвимости — это ошибки и недоработки в программном обеспечении, которые ...".
- "... полиморфизм — модификацию последовательности команд, не изменяющую выполняемых действий".

Извлечение отношений "часть — целое", "общее — частное" и "одно понятие используется для определения другого понятия" из текстов естественного языка

Разработанная программная система предназначена также для поиска отношений "часть — целое", "общее — частное" и "понятие *P* используется для определения понятия *Q*". Извлечение этих отношений проводилось при помощи лингвистических шаблонов, описывающих грамматические конструкции, соответствующие искомым отношениям. Приведем примеры извлеченных отношений.

- Что-то — это часть чего-то: "Геометрия — это **часть математики**, изучающая отношения и формы ...";
- Что-то является типом/частью чего-то: "Код является специальным **типом шифра**".

В результате анализа словаря по информационной безопасности при помощи разработанной системы были найдены 60 % всех отношений, найденных человеком. Выдача системы содержала 20 % ошибок. При анализе математического словаря были получены гораздо лучшие результаты: 80 % найденных отношений и 10 % ошибок в выдаче. Такое различие результатов обусловлено более формальным стилем изложения материала в математическом словаре.

Кроме отношений "часть — целое" и "общее — частное" оказалось полезным извлечение отношения порядка определения терминов, в котором состоят определяемый термин и термины, использующиеся для его определения, т. е. отношения: "понятие *P* используется для определения понятия *Q*". Для извлечения отношений данного типа используется лингвистический шаблон "объект", описанный выше. В процессе извлечения отношений словарь разбивается на словарные статьи, в которых затем проводится поиск терминов с использованием шаблона "объект". Найденные в результате термины расцениваются системой как использующиеся для определения термина данной словарной статьи.

Например, описанный алгоритм устанавливает, что для определения термина "теория вероятностей" использованы термины "математика", "случайное событие", "случайная величина" после обработки следующего определения:

"Теория вероятностей — раздел математики, изучающий закономерности случайных явлений: случайные события, случайные величины, их свойства и операции над ними".

Отметим, что хотя и при поиске участков текста, соответствующих лингвистическому шаблону "объект" будут найдены также "случайные явления", "свойства" и "операции", они не будут расцениваться как термины, так как в тексте отсутствуют их явные определения.

Заключение

В работе описаны методы извлечения онтологической информации из текстов естественного языка. Приведена теоретико-модельная формализация наиболее важных и интересных отношений между поня-

тиями: синонимия, "общее — частное", "одно понятие используется для определения другого понятия". Точная математическая формализация позволила исследовать достаточно тонкие свойства этих отношений. Разработаны методы извлечения явных определений терминов из текстов естественного языка, а также отношений "общее — частное", "часть — целое" и отношение порядка определения между этими терминами. Для анализа текста использовался язык описания лингвистических шаблонов, позволяющий описывать условия на грамматические конструкции. Кроме этого, была предложена специальная структура данных — множество интерпретаций, позволяющая хранить промежуточные результаты анализа в удобном виде, а также управлять альтернативными ветками анализа.

Предложенные подходы к анализу текстов были апробированы на математических текстах и текстах естественного языка, относящихся к предметной области информационной безопасности. Программная система показала достаточно высокую точность и полноту извлекаемых знаний.

Описанная в статье программная система, предназначенная для извлечения онтологической информации из текстов естественного языка, в настоящий момент применяется при разработке экспертной системы по информационной безопасности, а также при разработке новой версии системы управления рисками при обеспечении информационной безопасности. Модель угрозы информационной безопасности предприятия пополняется за счет анализа текстов естественного языка, представленных в сети Интернет, в автоматизированном (полуавтоматическом) режиме. Исходя из модели информационной системы предприятия и моделей компьютерных угроз, обрабатываются получаемые из Интернет сведения по информационной безопасности для пополнения формальной модели угрозы информационной безопасности данного предприятия. Программная система имеет функциональные возможности, позволяющие пользователю формулировать на естественном языке вопросы по информационной безопасности предприятия, включая возможные компьютерные угрозы.

Список литературы

1. **Berners-Lee T., Hendler J., Lassila O.** The Semantic Web // Scientific American. 2001. May. P. 34—43.

2. **Fensel D., Wahlster W., Lieberman H., Hendler J.** Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential. The MIT Press, 2002.

3. **Daconta M. C., Obrst L. J., Smith K. T.** The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management. Wiley Publishing, 2003.

4. **Fuchi K.** Revisiting Original Philosophy of Fifth Generation Computer Systems Project // Proceedings of the International Conference on Fifth Generation Computer Systems 1984. Tokyo, Japan, November 6—9, 1984. OHMSHA Ltd. Tokyo and North-Holland, 1984. P. 1—2.

5. **Moto-Oka T., Kitsuregawa M.** The Fifth Generation Computer: The Japanese Challenge. New York, 1985.

6. **Bishop P.** Fifth Generation Computers. New York, 1986.

7. **Shadbolt N., Hall W., Berners-Lee T.** The Semantic Web Revisited // IEEE Intelligent Systems. 2006. Vol. 21, N 3. P. 96—101.

8. **Fensel D. van Harmelen F., Horrocks I., McGuinness D. L., Patel-Schneider P. F.** OIL: An Ontology Infrastructure for the Semantic Web // IEEE Intelligent Systems 2001. Vol. 16, N 2. P. 38—45.

9. **Maedche A.** Ontology Learning for the Semantic Web. Kluwer Academic Publishers, 2002.

10. **Towards** the Semantic Web: Ontology-driven Knowledge Management / J. Davies, D. Fensel, F. van Harmelen (eds.). John Wiley & Sons, 2003.

11. **The Handbook** on Ontologies in Information Systems / S. Staab, R. Studer (eds.). Springer Verlag, 2003.

12. **Gбmez-Pérez A., Fernandez-Lopez M., Corcho O.** Ontological Engineering with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. Springer-Verlag, 2004.

13. **OWL** Web Ontology Language Overview/ D. McGuinness, F. Harmelen (eds.). URL: http://www.w3.org/TR/2003/WD-owl-features-20030331/

14. **Hendler J. A.** Frequently Asked Questions on W3C's Web Ontology Language (OWL). URL: http://www.w3.org/2003/08/owlfaq

15. **OWL** Web Ontology Language Semantics and Abstract Syntax / P. F. Patel-Schneider, P. Hayes, I. Horrocks (eds.). URL:http://www.w3.org/TR/owl-absyn/

16. **Васенин В. А.** К созданию международной системы мониторинга и анализа информационного пространства для предотвращения и прекращения военно-политических киберконфликтов // Информационные технологии. 2012. № 9. С. 2—10.

17. **Кейслер Г., Чэн Ч. Ч.** Теория моделей. М.: Мир, 1977.

18. **Ершов Ю. Л., Палютин Е. А.** Математическая логика. М.: Наука, 1979.

19. **Пальчунов Д. Е.** Моделирование мышления и формализация рефлексии. I. Теоретико-модельная формализация онтологии и рефлексии // Философия науки. 2006. № 4 (31). С. 86—114.

20. **Пальчунов Д. Е.** Решение задачи поиска информации на основе онтологий // Бизнес-информатика. 2008. № 1. С. 3—13.

21. **Carnap R.** Meaning and Necessity. A Study in Semantics and Modal Logic. Chicago, University of Chicago Press, 1956.

22. **Пальчунов Д. Е.** Моделирование мышления и формализация рефлексии. II. Онтологии и формализация понятий // Философия науки. 2008. № 2 (37). С. 62—99.

23. **Пальчунов Д. Е.** Определимость предложений языка булевых алгебр с выделенными идеалами // Вестник НГУ. Серия: Математика, механика, информатика. 2008. Т. 8, № 2. С. 62—75.

24. **Степанов П. А.** Автоматизация обработки текстов естественного языка // Вестник НГУ, серия: Информационные технологии. 2013. Т. 11, № 2. С. 109—115.

25. **Карпов Ю. Г.** Теория автоматов: Учебник для вузов. СПб.: Питер, 2003.

26. **Сокирко А. В.** Реализация первичного семантического анализа в системе Диалинг // Труды Международного семинара "Диалог'2000" по компьютерной лингвистике и ее приложениям. Протвино. 15 июня 2000 г. URL: http://www.aot.ru/docs/sokirko/dialog2000.htm

27. **Cunningham H., Maynard D., and Tablan V.** JAPE: a Java Annotation Patterns Engine (Second Edition). Technical report CS-00-10, University of Sheffild, Department of Conmuter Science, 2000.

28. **Зализняк А. А.** Грамматический словарь русского языка. Словоизменение. Около 100 000 слов. М.: Русский язык, 1997.

29. **Белошапкова В. А., Брызгунова Е. А., Земская Е. А.** и др. Современный русский язык: Учеб. для филол. спец. высших учебных заведений. 3-е изд., испр. и доп. М.: Азбуковник, 1997.

30. **Власов Д. Ю., Пальчунов Д. Е., Степанов П. А.** Извлечение отношений между понятиями из текстов на естественном языке // Вестник НГУ, серия: Информационные технологии. 2010. Т. 8, № 3. С. 23—33.

УДК 004.78:519.7

А. Н. Пустыгин[1], канд. техн. наук, доц., e-mail: p2007an@ya.ru,
Ю. К. Язов[2], д-р техн. наук, проф., глав. науч. сотр., e-mail: yazoff_1946@mail.ru,
О. А. Машин[2], стар. науч. сотр., e-mail: plazma-m@mail.ru,
М. В. Зубов[1], аспирант, e-mail: zubovmv@gmail.com,
[1] Челябинский государственный университет
[2] ФАУ "Государственный научно-исследовательский испытательный институт проблем технической защиты информации ФСТЭК России", г. Воронеж

К вопросу об автоматическом комментировании на естественном языке исходных текстов программ

Статья посвящена анализу подходов к созданию механизмов автоматического комментирования кодов программ. Отмечается, что формирование технических комментариев исходных текстов программ является не только объемной, но и трудоемкой процедурой. По результатам такого анализа представлены предложения по стандартизации средств автоматизации технического комментирования, направленные на формирование единых форматов промежуточных представлений исходного текста программ.

Ключевые слова: *программирование, технический комментарий, исходные тексты программ, автоматическое комментирование, анализ кода программ, формат представления комментария, стандарт*

В статье [1] было показано, что создание программных механизмов формирования технических комментариев исходных текстов программ является в настоящее время одним из востребованных практикой направлений развития современных сред программирования. Кроме того, было отмечено, что комментарии обеспечивают более точное понимание содержания программы, они позволяют решать ряд других важных задач. К их числу относятся задачи локализации потенциальных уязвимостей программного кода, обнаружения недокументированных возможностей программ, оценки качества кода программ и ряд других [1].

Вместе с тем формирование технических комментариев является не только объемной, но и достаточно трудоемкой процедурой, что обусловливает необходимость ее автоматизации. При анализе подходов к созданию таких средств автоматизации следует принимать во внимание, что придется решать очень сложную задачу. Ее сложность обусловлена тем обстоятельством, что синтаксическое дерево разбора кода программы является внутренним набором данных компилятора и не предназначено для целей комментирования. В связи с этим сначала необходимо получить промежуточное представление исходного текста, по сути эквивалентное дереву разбора и отличающееся от него нотацией.

Цель данной статьи состоит в раскрытии основных аспектов создания указанных средств автоматизации технического комментирования, связанных с построением промежуточных представлений комментариев исходных текстов программ и формированием единых форматов таких представлений.

При рассмотрении подходов к созданию целевых механизмов (средств) автоматического комментирования необходимо отметить следующее. Во-первых, ключевым моментом создания указанных средств является модульность пакета преобразования программного текста в текст технического комментария к нему. Технический комментарий при этом можно сгенерировать и из исходных текстов, и из загрузочных модулей. Во-вторых, форма представления технических комментариев может быть как текстовой, так и графической. Она определяется целями, которые преследуют при комментировании. В-третьих, в ходе создания комментария может быть охвачен широкий диапазон текстов — от текстов на структурированных специальных языках, таких как XML [2], до текстов на

Таблица 1

Пример сравнения синтаксических конструкций операторов цикла в языке Java и соответствующих им конструкций машинного комментария на русском языке

Оператор языка Java	Число символов в исходном тексте	Эквивалентное представление оператора в виде машинного комментария	Число символов в машинном комментарии	Увеличение числа символов в машинном комментарии по отношению к исходному тексту
while(){}	9	Выполнять тело цикла, пока истинно условие: Тело цикла: Конец тела цикла	62	В 6,8 раз
for () {}	7	Выполнять тело цикла для "параметра", пока истинно условие с "параметром": Тело цикла: Конец тела цикла Изменить значение "параметра"	126	В 18 раз
do {} while()	11	Тело цикла: Конец тела цикла Если истинно условие, то выполнять тело цикла	63	В 5,7 раз

естественных языках, например, на русском. При этом приходится формировать целый ряд промежуточных представлений исходного текста подлежащей анализу программы с использованием не только языка XML, но и других форматов структурированного представления текста, или языков разметки, таких как HTML, YAML, JSON и др.

Следует отметить, что применение стандартных языков разметки делает возможным последующую обработку посредством существующих анализаторов текстов на языке разметки — парсеров[1], таких как Expat, SAX, Xerces, DOM [3].

В случае применения единого формата промежуточного представления оказывается возможным построение универсальных инструментальных средств обработки промежуточного представления исходных текстов программ для получения единообразных по стилю технических комментариев, мало зависящих от языка написания исходного текста. Для уменьшения вычислительных ресурсов, необходимых для таких преобразований, можно сократить мощности выходного алфавита интерпретатора путем использования подмножества естественного языка. Следует отметить, что представление машинного комментария в форме текста на естественном языке позволяет преобразовывать комментарии на естественном языке в синтезированную речь.

Как и любое эквивалентное представление, текстовое представление на естественном языке пригодно для дальнейшего анализа. При использовании универсального формата эквивалентного представления кода программы ее текстовое эквивалентное представление будет одинаковым при использовании одних и тех же алгоритмов для написания программ на разных языках с точностью до названия переменных. Если переменные сделать одинаковыми, то хэш-функции совпадают, точно так же расстояния Хэмминга становятся нулевыми.

Электронные переводчики с русского на английский язык имеют примерно одинаковые словари на русском и английском языках. В отличие от них переводчик с машинного на русский язык должен иметь явно несимметричные словари. В отличие от машинного языка, который выразительно беден, словарь русского языка богат, что указывает на важные его преимущества. В качестве примера можно провести сравнение синтаксических конструкций операторов цикла в языке Java и соответствующих им языковых конструкций машинного комментария на русском языке (табл. 1).

Из представленного примера следует, что полученный машинный комментарий на русском языке характеризуется намного большим объемом словаря. Например, для цикла for имеет место 18-кратное увеличение числа символов.

Однако вложенные в скобки конструкции алгоритмического языка "схлопываются", так что смысловое содержание автоматического комментария для цикла становится "матрешкой", в которой каждое высказывание "обертывает" одно или несколько других, вложенных. Каждое из "обернутых" высказываний может быть извлечено из "схлопнутого" состояния по запросу. По этой причине на понятийном уровне представление текста на языке программирования упрощается за счет перехода от словаря "короткого языка" к словарю, близкому к естественному языку. При таком

[1] Парсер — синтаксический анализатор, т. е. программа или часть программы, выполняющая синтаксический анализ, результатом которого является синтаксическая структура предложения, представленная либо в виде дерева зависимостей, либо в виде дерева составляющих, либо в виде некоторой комбинации первого и второго способов представления.

преобразовании не происходит потерь информации, а знания разработчика преобразуются в эквивалентную форму.

По механизмам функционирования система автоматического комментирования на естественном языке является кодогенератором, потому что такая система строит выходной набор данных из модулей, поставленных в соответствие синтаксическим конструкциям исходного языка вручную разработчиком.

Выбор языка для построения автоматического комментария определяется языковой средой разработчика. При этом предполагается существование пакетов словарей национальных языков, которые могут подключаться к автоматическому комментатору. Использование словарей приводит к необходимости стандартизации их форматов и интерфейсов для доступа к ним процедур комментирования.

Автоматическая генерация машинных комментариев позволяет строить эквивалентные представления разных уровней, основываясь на введенных стандартных форматах данных, начиная от текста, эквивалентного исходному. Далее уровень обобщения построенного комментария может увеличиваться за счет исключения части информации внутри законченных синтаксических конструкций. Возможность такой обработки текстового комментария на естественном языке основана на метрике Левенштейна и алгоритмах редакционного предписания (Вагнера-Фишера, Хишберга, нечеткого поиска и др.) [4, 5].

Следующим уровнем обобщения стандартов автоматического комментирования может быть построение словарей типовых алгоритмов в формате, основанном на стандартизованных форматах нижнего уровня.

Рассмотрим пример построения машинного комментария на естественном языке по исходному тексту на языке Java (листинг 1).

Листинг 1

Исходный текст программы на JAVA

```
package com.example.program;
public class Program {
public static void main(String[] args) {
int a = 0;
for (int i = 0; i < 10; i++){
a += 1;
}
}
}
```

В приведенном исходном тексте выделим часть текста — цикл for (листинг 2).

Листинг 2

```
for (int i = 0; i < 10; i++){
a += 1;
}
```

С помощью компилятора Java [6] сначала формируется промежуточное представление исходного текста листинга 1 в текстовом XML-формате. Фрагмент текстового файла, соответствующий выделенному циклу, представлен в листинге 3.

Листинг 3

```
< for_loop line = "7" col = "3" >
< nodename.initialize_section >
< variable line = "7" col = "8" name = "i" >
< modifiers line = "-1" col = "-1" >
< /modifiers >
< vartype >
< primitive_type line = "7" col = "8" name = "int" >
< /primitive_type >
< /vartype >
< init >
< int_literal line = "7" col = "16" type = "int" value = "0" >
< /int_literal >
< /init >
< /variable >
< /nodename.initialize_section >
< less_than line = "7" col = "19" >
< identifier line = "7" col = "19" name = "i" >
< /identifier >
< int_literal line = "7" col = "23" type = "int" value = "10" >
< /int_literal >
< /less_than >
< nodename.step_section >
< expression_statement line = "7" col = "27" >
< postfix_increment line = "7" col = "27" >
< identifier line = "7" col = "27" name = "i" >
```

```
< /identifier >
< /postfix_increment >
< /expression_statement >
< /nodename.step_section >
< block line = "7" col = "31" >
< nodename.statements >
< expression_statement line = "8" col = "4" >
< plus_assignment line = "8" col = "4" >
< identifier line = "8" col = "4" name = "a" >
< /identifier >
< int_literal line = "8" col = "9" type = "int" value = "1" >
< /int_literal >
< /plus_assignment >
< /expression_statement >
< /nodename.statements >
< /block >
< /for_loop >
```

Затем это промежуточное представление цикла в XML-формате преобразуется в промежуточное представление с русским комментарием, соответствующим XML-тегам (листинг 4).

Листинг 4

```
          < ?xml version = "1.0" ? >
< for_loop line = "7" col = "3" > < !-- Объявлен цикл for в строке 7, позиция 3. -- >
< nodename.initialize_section > < !-- Цикл инициализируется -- >
< variable line = "7" col = "8" name = "i" > < !— переменной i -- >
< modifiers line = "-1" col = "-1" >
< /modifiers >
< vartype > < !-- типа -- >
< primitive_type line = "7" col = "8" name = "int" > < !-- int -- >
< /primitive_type >
< /vartype >
< init > < !-- значением -- >
< int_literal line = "7" col = "16" type = "int" value = "0" > < !-- 0. -- >
< /int_literal >
< /init >
< /variable >
< /nodename.initialize_section >
< less_than line = "7" col = "19" > < !-- Проверка условия окончания: "меньше" для -- >
< identifier line = "7" col = "19" name = "i" > < !-- идентификатора i -- >
< /identifier >
< int_literal line = "7" col = "23" type = "int" value = "10" > < !-- и значения 10. -- >
< /int_literal >
< /less_than >
< nodename.step_section > < !-- Шаг цикла: -- >
< expression_statement line = "7" col = "27" > < !— выражение -- >
< postfix_increment line = "7" col = "27" > < !-- постфиксный инкремент -- >
< identifier line = "7" col = "27" name = "i" > < !-- идентификатора i. -- >
< /identifier >
< /postfix_increment >
< /expression_statement >
< /nodename.step_section >
< block line = "7" col = "31" > < !-- \nТело цикла:\n -- >
< nodename.statements >
< expression_statement line = "8" col = "4" > < !-- # Выражение в строке 8 позиция 4 -- >
< plus_assignment line = "8" col = "4" > < !-- "сокращенная форма бинарной операции сложения" -- >
< identifier line = "8" col = "4" name = "a" > < !-- идентификатора a -- >
< /identifier >
< int_literal line = "8" col = "9" type = "int" value = "1" > < !-- и константы "1" типа int. -- >
< /int_literal >
/plus_assignment >
< /expression_statement >
< /nodename.statements >
< /block >
< /for_loop >
```

На третьем этапе обработки исходного текста промежуточное представление с русским комментарием преобразуется в комментарий на русском языке в виде повествовательного текста (листинг 5).

Листинг 5

Объявлен цикл for в строке 7, позиция 3. Цикл инициализируется переменной i типа int значением 0. Проверка условия окончания: "меньше" для идентификатора i и значения 10. Шаг цикла: выражение постфиксный инкремент идентификатора i.

Тело цикла:

Выражение в строке 8 позиция 4 "сокращенная форма бинарной операции сложения" идентификатора a и константы "1" типа int.

Из изложенного выше достаточно хорошо видно, каким образом поэтапно формируется технический комментарий. Построение средств создания таких комментариев является одним из направлений внедрения методов искусственного интеллекта в сфере программирования. Фактически это направление связано с индустриализацией извлечения знаний из исходного текста исходной программы, в данном случае — извлечение с его представлением в форме машинного комментария. Однако построение указанных средств в настоящее время сдерживается в силу отсутствия стандартов, регламентирующих стадии (этапы), типы представления машинного комментария, форматы документов и ряда других причин. Предложения по составу стандартов, которые, по мнению авторов, целесообразно разрабатывать в первую очередь, приведены в табл. 2.

Таблица 2

Предложения по составу первоочередных подлежащих разработке стандартов в интересах создания программных средств формирования технических комментариев*

Наименование проекта стандарта	Назначение стандарта
ГОСТ Р "Информационные технологии. Общие требования к программным средствам формирования технических комментариев исходных кодов программ"	Определяет состав общих требований к программным средствам формирования технических комментариев, касающихся назначения средства, состава программных конструкций, подлежащих комментированию, порядка формирования промежуточных эквивалентных представлений, объема комментария и др.
ГОСТ Р "Информационные технологии. Основные этапы формирования машинных комментариев исходных кодов программ, формы машинных комментариев, способы получения"	Определяет состав требований к процессу последовательного формирования технических комментариев, касающихся этапов преобразования исходного текста, наборов эквивалентных представлений исходного текста, подлежащих комментированию, обеспечения навигации как между эквивалентными представлениями, так и между эквивалентными представлениями и исходным текстом и др.
ГОСТ Р "Информационные технологии. Форматы представления машинных комментариев исходных кодов программ"	Регламентирует форматы промежуточных и конечных эквивалентных представлений технических комментариев, обеспечивая единство эквивалентного представления независимо от языка исходного текста
ГОСТ Р "Информационные технологии. Общие требования к мультиязыковой поддержке средств формирования технических комментариев исходных кодов программ"	Определяет состав требований к языковой среде формирования технических комментариев, состав словарей национальных языков, которые могут подключаться к автоматическому комментатору, требования к интерфейсам и др.
* **Примечание.** Указанные наименования проектов стандартов являются предварительными, их состав, содержание и наименования должны быть уточнены в ходе дальнейших исследований, однако даже в таком изложении указанные предложения указывают на важность данного направления стандартизации.	

Внедрение предлагаемой системы стандартов даст импульс развитию технологий программирования и программной инженерии в целом. Оно будет способствовать ускорению решения многих зависимых от этих технологий прикладных вопросов и, в частности, вопросов автоматического исследования программных кодов в интересах оценки их качества, выявления уязвимостей и недекларированных возможностей программ, решения задач рефакторинга [1].

Список литературы

1. **Пустыгин А. Н., Иванов А. И., Язов Ю. К., Соловьев С. В.** Автоматический синтез комментариев к программным кодам: перспективы развития и применения // Программная инженерия. 2012. № 3. С. 30—33.

2. **Расширяемый** язык разметки XML. URL: http://ru.wikipedia.org/wiki/XML

3. **Тидвелл Д.** Основы программирования парсеров. Сообщество developerWorks. URL: http://www.ibm.com/developerworks/ru/edu/xmljava/section2.html

4. **Левенштейн В. И.** Двоичные коды с исправлением выпадений, вставок и замещений символов. Доклады Академии Наук СССР, 1965.

5. **Романовский И. В.** Дискретный анализ. 3-е изд. СПб.: БХВ-Петербург, 2003.

6. **The Java Programming** Language Compiler — javac. URL: http://download.oracle.com/javase/6 /docs/technotes/guides/javac/index.html

УДК 004.85; 004.91

Н. А. Сергиевский, стар. науч. сотр., Государственный институт информационных и телекоммуникационных технологий "Информика", г. Москва,
А. А. Харламов, д-р техн. наук, стар. науч. сотр., e-mail: kharlamov@analyst.ru,
Институт высшей нервной деятельности и нейрофизиологии РАН, г. Москва

Семантический анализ как основа для выявления дублирующих фрагментов текста

Представлены методы и средства поиска нечетких дубликатов фрагментов текста на основе анализа семантической сети текста. В их основе — выявление смыслового портрета текстов в виде их семантических сетей, которые строятся с помощью технологии автоматического смыслового анализа текстов TextAnalyst, с последующим их использованием для сравнения смыслов текстов. Предлагаемый подход подразумевает несколько уровней "просеивания" текстов для быстрого и точного поиска дубликатов: быстрое сравнение семантических сетей, поиск нечетких копий фрагментов текста.

Ключевые слова: *нечеткие дубликаты текста, семантическая сеть, сравнение смыслов, поиск нечетких копий*

Введение

На настоящее время проблема поиска дубликатов в тексте частично решена [1—3]. Однако до сих пор актуально создание эффективных и быстрых систем поиска нечетких дубликатов, а также систем, удовлетворяющих более жестким требованиям по размерам сравниваемых фрагментов и по изменениям в тексте.

В сети Интернет доступно большое число методических указаний, курсов лекций, учебников и других учебных пособий [4]. Кроме того, появились огромные коллекции рефератов, готовых лабораторных работ, курсовых и дипломных проектов и даже диссертаций. Использование компьютерной техники значительно облегчило задачу поиска и копирования подобной информации. Если раньше для написания реферата или контрольной работы информацию нужно было, по крайней мере, найти в книгах и переписать (вручную, перепечатать или ввести в компьютер с помощью сканера и программ распознавания текстов), то теперь достаточно ввести название темы в поисковую систему и скопировать найденные материалы. Широко распространен метод написания работ, получивший название "Copy & Paste", который заключается в простом копировании информации из одного или нескольких источников с минимальным редактированием получающегося таким образом текста. Аналогичная ситуация наблюдается с творческими работами студентов учебных заведений. В связи с тем, что большое число пояснительных записок по курсовым и дипломным проектам выполняется с использованием компьютеров, происходит их дублирование и повторное использование. Ситуация усугубляется тем обстоятельством, что студенты, копирующие материалы, зачастую не читают того, что написано в собственных работах. Это приводит к плагиату.

По определению плагиат — это умышленное присвоение авторства чужого произведения науки или искусства, чужих идей или изобретений. Как понятно из этого определения, можно отнести подобные заимствованные работы к разряду плагиата. Как следствие, задача обнаружения недобросовестного использования заимствованных текстов (фактов плагиата) приобретает высокую актуальность.

1. Обзор алгоритмов и подходов

Борьба с плагиатом в последние годы является одной из приоритетных задач для образовательного и научного сообщества. По результатам исследований, направленных на ее решение, опубликовано много работ. К результатам российских исследователей можно отнести, например, работы [4—6].

Одними из первых исследований в области нахождения нечетких дубликатов являются работы U. Manber [7] и N. Heintze [8]. U. Manber искал похожие файлы (утилита *sif*), а N. Heintze — нечеткие дубликаты документов (система *Koala*). В этих работах дескриптор текста (представление текста для дальнейшей обработки и анализа) строится как последовательность соседних букв текста заданной длины. Дескриптор файла, или документа включает все текстовые подстроки фиксированной длины. Численное значение дескриптора текста вычисляется с помощью алгоритма случайных полиномов Карпа-Рабина [9].

Алгоритм Карпа-Рабина осуществляет поиск подстроки в строке, используя хеширование. Алгоритм эффективен тем, что дает возможность осуществлять сравнение с большим числом шаблонов (подстрок). Для каждой подстроки вычисляется специальная хеш-функция:

$$hash(s_{0...R}) = s_0 + ps_1 + ... + p^{R-1}s_{R-1} + p^R s_R,$$

где p — некоторое заданное число; R — длина строки; s_i — символ строки.

Хеш-функцию можно также представить следующим образом:

$$hash(s_{0...R}) = (s_0 + ps_1 + ... + p^{R-1}s_{R-1} + p^R s_R) \bmod q,$$

где q — некоторое большое целое положительное число, а mod — операция целочисленного деления.

Например,

$$hash(\text{‘mama’}) = 109 \cdot 11^0 + 97 \cdot 11^1 + 109 \cdot 11^2 + 97 \cdot 11^3 = 143\,472.$$

Существует быстрый вариант пересчета хэш-функции для вычисления подстроки фиксированной длины. Решение этой задачи необходимо для дальнейшего вычисления "шинглов". В этом варианте сканирующим окном пробегается строка и вычисляется каждое значение хеш-функции внутри сканирующего окна. Для этого хеш-функцию для следующей подстроки фиксированной длины выразим через соотношение

$$hash(s_{J...R}) = (1/p^J)(hash(s_{0...R}) - hash(s_{0...J-1})),$$

где J — индекс, с которого начинается подстрока.

В качестве меры сходства двух документов используется отношение числа общих подстрок к размеру файла или документа. Работы этого направления сводятся к созданию методов, направленных на снижение вычислительной сложности алгоритмов.

В 1997 г. A. Broder с соавторами [10, 11] предложили новый, "синтаксический" метод оценки сходства между документами, основанный на представлении документа в виде множества всевозможных последовательностей фиксированной длины k, состоящих из соседних слов. Такие последовательности были названы "шинглами". Два документа считались похожими, если их множества шинглов существенно пересекались. Поскольку число шинглов примерно равно длине документа в словах, т. е. является достаточно большим, авторами были предложены два метода сэмплирования для получения репрезентативных подмножеств.

В первом методе использовались только шинглы, дескрипторы которых, вычисляемые по алгоритму Карпа-Рабина [9], делились без остатка на некоторое число m. Основной недостаток метода — зависимость выборки от длины документа. Документы небольшого размера (в словах) представлялись или очень короткими выборками, или вообще не имели таковых. Второй метод отбирал только фиксированное число шинглов s с наименьшими значениями дескрипторов или оставлял все шинглы, если их общее число не превышало s.

Дальнейшим развитием концепций A. Broder и соавторов являются исследования, представленные в работах [10, 12]. Для каждой цепочки вычисляются 84 дескриптора по алгоритму Карпа-Рабина [9] с помощью взаимно-однозначных и независимых функций, использующих случайные наборы ("min-wise independent") простых полиномов. В результате каждый документ представлялся 84 шинглами, минимизирующими значение соответствующей функции. Затем 84 шингла разбивают на 6 групп по 14 (независимых) шинглов в каждой. Эти группы называют "супершинглами".

Пример. Рассмотрим наивное построение одного супершингла. Возьмем предложение "Плагиат — умышленное присвоение авторства чужого произведения науки или искусства, чужих идей или изобретений".

Для построения супершингла выберем три шингла длиной по четыре слова.

"Плагиат умышленное присвоение авторства", "умышленное присвоение авторства чужого", "искусства чужих идей изобретений"

Хеш для шинглов будет иметь следующий вид:

$$sh\_hash_1 = (hash(\text{"Плагиат"}) \cdot 101^0 + hash(\text{"умышленное"}) \cdot 101^1 + hash(\text{"присвоение"}) \cdot 101^2 + hash(\text{"авторства"}) \cdot 101^3) \bmod 1000 = 586;$$

$$sh\_hash_2 = (hash(\text{"умышленное"}) \cdot 101^0 + hash(\text{"присвоение"}) \cdot 101^1 + hash(\text{"авторства"}) \cdot 101^2 + hash(\text{"чужого"}) \cdot 101^3) \bmod 1000 = 278;$$

$$sh\_hash_3 = (hash(\text{"искусства"}) \cdot 101^0 + hash(\text{"чужих"}) \cdot 101^1 + hash(\text{"идей"}) \cdot 101^2 + hash(\text{"изобретений"}) \cdot 101^3) \bmod 1000 = 151.$$

Хеш для наивного супершингла будет иметь следующий вид:

$$super\_sh\_hash = sh\_hash_1 \cdot 1001^0 + sh\_hash_2 \cdot 1001^1 + sh\_hash_3 \cdot 1001^2 = 151\ 581\ 015.$$

Если два документа имеют сходство, например, $P \approx 0{,}95$ (95 %), то два соответствующих супершингла в них совпадают с вероятностью $P^{14} \approx 0{,}95^{14} \approx 0{,}49$ (49 %).

Поскольку каждый документ представляется шестью супершинглами, то вероятность того, что у двух документов совпадут не менее двух супершинглов равна $1 - (1 - 0{,}49)^6 - 6 \cdot 0{,}49(1 - 0{,}49)^5 \approx 0{,}90$ (90 %). Для сравнения: если два документа имеют сходство только $P \approx 0{,}80$ (80 %), то вероятность совпадения не менее двух супершинглов составляет всего 0,026 (2,6 %).

Таким образом, для эффективной проверки совпадения не менее двух супершинглов (и, следовательно, подтверждения гипотезы о сходстве содержания) каждый документ представляется всевозможными попарными сочетаниями из шести супершинглов, которые называют "мегашинглами". Число таких мегашинглов равно 15 (число сочетаний из 6 по 2). Два документа сходны по содержанию, если у них совпадает хотя бы один мегашингл.

Ключевое преимущество данного алгоритма по сравнению с алгоритмом A. Broder с соавторами состоит в том, что, во-первых, любой документ (в том числе и очень маленький) всегда представляется вектором фиксированной длины, и, во-вторых, сходство определяется простым сравнением координат вектора и не требует (как у A. Broder) выполнения теоретико-множественных операций.

Еще один сигнатурный подход, основанный уже не на синтаксических, а на лексических принципах, был предложен A. Chowdhury с соавторами в 2002 г. и усовершенствован в 2004 г. [13, 14]. Его основная идея состоит в вычислении так называемой *I-Match*-сигнатуры для представления содержания документов. С этой целью сначала для исходной коллекции документов строится словарь L, который включает слова со средними значениями *IDF* (inverse document frequency — обратная частота документа, те слова, которые встречаются примерно в половине документов из корпуса текстов), поскольку такие слова обеспечивают более точные результаты при обнаружении нечетких дубликатов. Нечеткие дубликаты представляют собой копию документа с небольшими изменениями, вызванными перестановкой или добавлением нового текста значительно меньшего объема, чем обычные. Слова с большими и маленькими значениями *IDF* отбрасывают (так как редкие и общеупотребимые слова зашумляют частотный портрет текста (или сигнатуру текста)).

Затем для каждого документа формируется множество U различных слов, входящих в него, и определяется пересечение U и словаря L. Если размер этого пересечения больше некоторого минимального порога (определяемого экспериментально), то список слов, входящих в пересечение упорядочивается, и для него вычисляется *I-Match*-сигнатура. После проецирования документа U на словарь получается бинарный вектор (с единицами в тех местах, где слово из U содержится в словаре L), длина которого равна числу элементов в словаре L. После чего вычисляется хеш-код от этого бинарного вектора (сигнатуры).

Два документа считаются похожими, если у них совпадают *I-Match*-сигнатуры или имеет место коллизия хеш-кодов. Хеш-код от совершенно разных слов может совпадать, например, $hash(\text{"vision"}) = (118 \cdot 10^0 + 105 \cdot 10^1 + 115 \cdot 10^2 + 105 \cdot 10^3 + 111 \cdot 10^4 + 110 \cdot 10^5) \bmod 100 = 68$ и $hash(\text{"bumerang"}) = (98 \cdot 10^0 + 117 \cdot 10^1 + 109 \cdot 10^2 + 101 \cdot 10^3 + 114 \cdot 10^4 + 97 \cdot 10^5 + 110 \cdot 10^6 + 103 \cdot 10^7) \bmod 100 = 68$.

Алгоритм имеет высокую вычислительную эффективность, превосходящую показатели алгоритма A. Broder. Другим преимуществом алгоритма (также по сравнению с алгоритмом A. Broder) является его высокая эффективность при сравнении небольших по размеру документов. Основной недостаток — неустойчивость к небольшим изменениям содержания документа.

Для преодоления указанного недостатка исходный алгоритм был модифицирован его авторами, и в него была введена возможность многократного случайного перемешивания основного словаря. Суть новых усовершенствований состоит в следующем. Дополнительно к основному словарю L создают K различных словарей $L_1 - L_K$, получаемых путем случайного удаления из исходного словаря некоторой небольшой фиксированной части слов d, составляющей порядка 30...35 % от исходного объема L. Для каждого документа вместо одной вычисляется $(K + 1)$ *I-Match*-сигнатура по алгоритму, описанному выше. Этот факт означает, что документ представляется в виде вектора размера $K + 1$, и два документа считают дубликатами, если у них совпадает хотя бы одна из координат. Если документ подвергается небольшим изменениям (порядка n слов), то вероятность того, что по крайней мере одна из K дополнительных сигнатур останется неизменной, будет равна

$$1 - (1 - d^n)^K. \quad (1)$$

Действительно, вероятность того, что изменения не затронут какого-либо одного словаря, равна d^n, т. е. вероятности того, что все изменения попадут в удаленную часть исходного словаря. Тогда $(1 - d^n)$ — вероятность, что сигнатура изменится, а $(1 - d^n)^K$ — вероятность, что все сигнатуры изменятся (поскольку дополнительные словари формируются независимо), и, следовательно, (1) — и есть искомая вероятность.

Алгоритм показал высокие результаты при использовании в различных приложениях веб-поиска и фильтрации спама. Рекомендуемыми значениями параметров, хорошо проявившими себя на практике, являются $d = 0{,}33$ и $K = 10$.

Схожий подход описан в патенте US Patent W. Pugh [15]. Автор предлагает полный словарь документа разбить на фиксированное число списков слов с помощью какой-либо функции хеширования (например, с помощью остатка от деления хеш-кода на число списков). Затем для каждого списка вычисляется дескриптор, и два документа считаются подобными, если они имеют хотя бы один общий дескриптор. Автор приводит оценки устойчивости алгоритма к небольшим изменениям содержания исходного документа. В работе отсутствуют какие-либо сведения об эффективности практического применения по соображениям конфиденциальности.

Еще одним сигнатурным подходом, также основанным на использовании словаря, является метод опорных слов, предложенный в работе [16]. Суть алгоритма заключается в следующем. Сначала из индекса по некоторому правилу выбирается множество из N слов (N определяется экспериментально), называемых опорными. Затем каждый документ представляется N-мерным двоичным вектором, где i-я координата равна 1, если i-е опорное слово имеет в документе относительную частоту выше определенного порога (устанавливаемого отдельно для каждого опорного слова), и равна 0 в противном случае. Этот двоичный вектор называется сигнатурой документа. Два документа считаются похожими, если у них совпадают сигнатуры.

Общие соображения для построения множества опорных слов таковы:

- множество слов должно охватывать максимально возможное число документов;
- качество опорного слова должно быть наивысшим;
- число слов в наборе должно быть минимальным.

Алгоритм построения множества слов и выбора пороговых частот представлен ниже. Пусть частота это нормированная внутридокументная частота слова TF (term frequency), лежащая в диапазоне 0...1, где 1 — частота самого часто встречаемого слова в документе. Для каждого слова (однократно) строится распределение документов по такой внутридокументной частоте.

Проводим несколько итераций, каждая из которых состоит из двух фаз. На первой фазе максимизируется покрытие документов в индексе при фиксированной (ограниченной снизу) точности; на второй фазе максимизируется точность при фиксированном покрытии.

Точность слова тем выше, чем меньше встречаемость слова в дельта-окрестности данного значения относительной частоты (т. е. чем меньше документов с TF, равным $TF_{threshold} \pm \Delta$). Частота с наивысшей точностью называется пороговой и запоминается для дальнейшего использования в приведенном алгоритме.

После каждой итерации отбрасываем самые плохие слова. После последней итерации остается достаточно слов для хорошего покрытия. В результате из выборки в сотни тысяч слов, остается набор в 3—5 тысяч, расчет сигнатур по которому с применением полнотекстового индекса осуществляется на миллиардном индексе несколько минут на поисковом кластере Яндекса [17].

2. Анализ функциональных возможностей

Рассмотрим достоинства и недостатки нескольких программных систем поиска дубликатов [4]. Одной из таких систем является система "Антиплагиат" [1], которая осуществляет поиск по большому числу коллекций рефератов, контрольных работ и учебников, хранящихся в ее собственной базе. Наиболее существенным недостатком этой системы является то, что она не осуществляет поиск по всем документам, доступным в сети Интернет. Особенно это касается тематических сайтов и новостных порталов, принимая во внимание, что большое число заимствований осуществляется именно из таких источников. Таким образом, даже при полном дублировании подобной информации, система "Антиплагиат" соответствий не обнаружит. Также в системе существует ограничение на размер проверяемого текста (не более 5000 символов). Наконец, ограничен просмотр документов, частично соответствующих проверяемому тексту.

Программа Advego Plagiatus осуществляет проверку с использованием поисковых систем [3]. Она использует разные поисковые системы и проверяет их доступность. В отличие от аналогичных систем, Advego Plagiatus не использует Яндекс.XML (программа не использует специальный интерфейс для поисковых роботов, а получает информацию из стандартной поисковой выдачи). Качество обнаружения плагиата высокое. Программа выдает процент совпадения текста и выводит найденные источники. Недостатками являются отсутствие преобразования букв (студенты часто заменяют кириллические буквы на латинские, которые выглядят одинаково, например, ‘о’, ‘у’ и т. п.), а также отсутствие поддержки поиска по собственной базе. В силу особенностей работы программы возникают ситуации, когда результаты проверок отличаются от одной к другой.

Сервис на сайте www.miratools.ru позволяет осуществлять on-line проверку текста на плагиат [4] при использовании результатов выдачи поисковых систем. К достоинствам можно отнести возможность замены английских букв на русские. Существует возможность изменять длину и шаг шинглов, используемых для проверки. По результатам проверки выдается процент совпадений и найденные источники. Система не работает с собственной базой. Присутствуют ограничения на длину текста в 3000 символов и на число про-

верок в течение суток, что связано с быстродействием алгоритмов.

Сервис www.istio.com осуществляет проверку текста на наличие заимствованного контента с использованием поисковых систем [4]. Для этих целей используются Яндекс.XML и Yahoo.com. Возможности сервиса несколько слабее по сравнению с miratools. По результатам проверки выдается сообщение о том, является ли текст уникальным или нет, и выдается список подобных сайтов. Преобразование букв и поддержка поиска по собственной базе отсутствуют. Сервис предоставляет дополнительные средства для анализа текстов, например, проверку орфографии, анализ слов, которые чаще всего встречаются в тексте.

Программа Praide Unique Content Analyser II [18] имеет широкие возможности по проверке текстов с использованием поисковых систем. Существует возможность выбора используемых поисковых систем, содержатся средства добавления новых поисковых систем. Проверка осуществляется пассажами и шинглами, длину которых можно изменять. Можно задавать количества слов перекрытия шинглов. Выводится подробный отчет по проверке в каждой поисковой системе. К недостаткам можно отнести отсутствие замены букв и обработки стоп-слов. Нет поддержки работы с собственной базой.

Система Plagiatinform, по заверениям авторов, имеет наиболее широкие функциональные возможности [19]. Она умеет проверять документы на наличие заимствований как в локальной базе, так и в сети Интернет. Система умеет обрабатывать документы, скомпонованные из перемешанных кусков текста нескольких источников. Проверка может осуществляться с использованием быстрого или углубленного поиска. Результаты проверки выдаются в виде наглядного отчета. Авторы не предоставляют возможности свободного использования или тестирования системы, и оценить качество ее работы сложно.

3. Подход с использованием технологии TextAnalyst

Описанные выше программные системы подходят для выявления плагиата в виде сравнительно мало измененных текстов. Это примитивный, но наиболее распространенный случай плагиата. Подходы не работают, если текст изменен значительно. В этом случае формальные соответствия отсутствуют и необходимо искать смысловое подобие текстов. Или же требуется перебор такого числа измененных вариантов, которое не под силу современной вычислительной технике. Из представленного ранее анализа видны "жесткие" ограничения на число переборов, существующие у имеющихся на рынке программных систем. В настоящее время не существует систем антиплагиата, анализирующих смысловое подобие текстов, и не разработаны подходы к такому, более глубокому анализу. Однако существует технология, которая позволяет оценить смысловое соответствие текстов. Это технология автоматического смыслового анализа текстов TextAnalyst. Ее применение позволяет выявить тексты, по смыслу близкие к анализируемому. После предварительного отбора возможно применение отмеченных выше систем, но к уже небольшому числу источников.

Целью работы, результаты которой представлены в настоящей статье, является анализ возможностей использования методов и программных средств автоматического смыслового анализа текстов для предварительного отбора документов как потенциальных источников при выявлении дубликатов. После существенного сокращения числа подлежащих анализу источников предоставляется возможность широко варьировать тексты документов при формальном сравнении с исходным текстом при поиске дубликатов. Возможно, в наибольшей степени подобный подход годится для автоматизированного выявления заимствований. Этот вопрос в разработке систем антиплагиата к настоящему времени должным образом не исследован.

3.1. Описание технологии смыслового сравнения текстов TextAnalyst

Технология TextAnalyst позволяет автоматически построить смысловой портрет текста — его семантическую сеть — где вершины сети это множество ключевых понятий текста, а дуги — связи ключевых понятий в тексте [20]. И вершины, и дуги имеют числовые характеристики — смысловые веса. В такой постановке можно считать, что пересечение смысловых портретов текстов (их семантических сетей) характеризует степень смыслового подобия этих текстов.

3.1.1. Пересечение семантических сетей

Опишем более формально ассоциативную семантическую сеть N, а также рассмотрим ее как некоторое подмножество метрического пространства.

Определение 1. Под семантической сетью N понимается совокупность несимметричных пар понятий (представленных словами) $\langle\langle c_i c_j\rangle\rangle$:

$$N \cong \langle\langle c_i c_j\rangle\rangle.$$

В общем случае отношение понятий в паре несимметрично ("всякая селедка — рыба, но не всякая рыба — селедка" [Носов]), что предполагает несимметричность двух пар: $\langle c_i c_j\rangle \neq \langle c_j c_i\rangle$.

Определение 2. Семантическая сеть, описанная таким образом, может быть переописана как множество так называемых звездочек: $\langle\langle c_i\langle c_j\rangle\rangle\rangle$:

$$N \cong \langle\langle c_i\langle c_j\rangle\rangle\rangle.$$

Определение 3. Под звездочкой $S = \langle c_i\langle c_j\rangle\rangle$ понимается конструкция, включающая главное понятие c_i, связанное с множеством понятий-ассоциантов $\langle c_j\rangle$, которые являются семантическими признаками глав-

ного понятия, отстоящими от главного понятия на одну связь. В силу несимметричности отношений между понятиями связи направлены от главного понятия к понятиям-ассоциантам.

В одной из звездочек семантической сети главным понятием может быть одно из понятий-ассоциантов другой звездочки, и наоборот. Однако связи всегда направлены от главного понятия к понятию ассоцианту.

Определение 4. Звездочка с единичными значениями весов понятий-ассоциантов называется единичной звездочкой (звездочкой-ортом).

Определение 5. Звездочкой-подпространством называется звездочка, полученная на единичной звездочке введением семантических весов понятий (w_i, w_j — семантические веса понятий):

$$S \cong \langle\langle w_i c_i \langle w_j c_j \rangle\rangle\rangle.$$

Семантическая сеть в терминах этих определений представляет собой декартово произведение подпространств, порождаемых всеми звездочками, входящими в семантическую сеть, полученными на единичных звездочках за счет введения весовых характеристик понятий-ассоциантов:

$$N = S_1 \times S_2 \times \ldots \times S_i.$$

Для определения на семантической сети операции пересечения разобьем звездочки на пары понятий $\langle c_i c_j \rangle$. Здесь c_i — главное понятие звездочки, а $c_j \in \langle c_j \rangle$ — множество понятий-ассоциантов понятия c_i. Связь направлена от c_i к c_j.

Определение 6. Введем скалярное произведение на векторах $\bar{c}_i$ и $\bar{c}_j$, где угол между векторами понятий c_i, и c_j _ w_{ij} пропорционален весу связи от c_i к c_j: $w_{ij} \in (0...90°)$.

Из двух скалярных произведений, соответствующих двум парам одинаковых понятий обоих сетей, выбирается меньшее, которое и будет пересечением двух звездочек одинаковых понятий по одному из понятий-ассоциантов.

Определение 7. Под пересечением S_{12} двух звездочек S_1 и S_2 понимается сумма по всем понятиям-ассоциантам c_j, $j = 1...N$ главного понятия звездочки c_i пересечений двух звездочек с одинаковыми главными понятиями $c_{i_1} = c_{i_2}$ по одному из понятий-ассоциантов c_j. В случае если в одной из звездочек пары, для которой считается пересечение, не нашлось соответствующего понятия-ассоцианта, пересечение считается равным 0.

$$S_{12} = \langle c_{i_1} \langle c_{j_1} \rangle \rangle \cap c_{i_2} c_{j_2} \rangle\rangle = \sum_{j}^{\max(N_1 N_2)} (S_{j_1} S_{j_2});$$

N_1, N_2 — число ассоциантов в звездочках 1 и 2 соответственно.

Определение 8. Под пересечением N_{12} семантических сетей понимается сумма пересечений звездочек, включенных в эти сети (считая по главным понятиям):

$$N_{12} = \sum_{k=1}^{\max M_1 M_2} (S_{k_1} \cap S_{k_1});$$

M_1, M_2 — число звездочек, входящих в семантические сети 1 и 2 соответственно.

3.1.2. Оценка смысловой близости текстов

Для получения оценки смысловой близости двух текстов вычисляется степень их смыслового пересечения, для чего:

1) строятся семантические сети для двух сравниваемых текстов;

2) проводится выявление наличия в обеих семантических сетях значимых понятий;

3) проводится сравнение ассоциантов для совпадающих ключевых понятий; вычисляется степень их близости;

4) подсчитывается взвешенный (по объему текстов) показатель для сетей.

Возможно упрощение сравнения сетей за счет неучета веса связей между понятиями. Сравнение двух текстов идет на основании сравнения звездочек с совпадающими главными понятиями для наиболее значимых ключевых понятий. Для каждого текста А и Б выбирают понятия с весами выше порогового значения. Выписывают все понятия из двух текстов, превысившие порог. Затем по этому списку идет процесс попарного сравнения. Для пересекающихся понятий c_i, т. е. понятий с идентичными корневыми основами, идет сравнение ассоциантов c_j этих понятий (определение 7). Для каждого главного понятия идет поиск совпадающих ассоциантов и вычисляется пересечение для всей звездочки по следующей формуле:

$$K = \frac{\sum\limits_{c_i \in C_А} \sum\limits_{c_j C_Б} G\left(c_i, c_j\right)}{\sum\limits_{c_i \in C_А} w_i + \sum\limits_{c_j C_Б} w_j},$$

$$G(c_i, c_j) = \begin{cases} \min(w_i, w_j), \text{ если } LinkC_i = LinkC_j; \\ 0, \text{ иначе,} \end{cases}$$

где $LinkC_i \in Adj(c_i)$; через $Adj(c)$ обозначается множество всех вершин семантической сети, смежных с c_i; $C_А$ и $C_Б$ — множества всех вершин (семантических понятий) текстов А и Б соответственно; K — нормированный вес совпадающих семантических понятий у текстов.

На рис. 1 показан пример сравнения двух текстов.

Родитель	Частота	Вес		Родитель	Частота	Вес
Университетский устав	9	100		университет	71	100
университет	115	100		научных исследованиях	4	100
школа	22	99		государств	4	100
профессора	23	99		финансовые	3	99
государство	23	99		России	5	99
автор	6	99		преподавание	8	99
училище	19	99		наук	6	99
наук	23	99		Европе	5	99
латинская	10	99		дисциплин	7	99
лекционный	5	99		государственных универси	2	99
академия	21	99		финансовые	3	95
уставу	27	99		СССР	3	95
России	24	99		академические	3	95
Московский университет	8	99		контролирует	3	95
кадетского корпуса	8	99		Европе	6	95
преподавались	24	99		экономические	3	94
арифметика	4	99		преподавание	4	91

Рис. 1. Сравнение основных понятий (семантических понятий, имеющих наибольший вес) на примере сравнения текстов: "Университет" (www.wikipedia.ru) и "История развития высшего образования в России" (www.analyst.ru)

3.2. Описание подхода

Рассмотрим алгоритм поиска нечетких дубликатов в тексте. Алгоритм включает следующие два этапа.

1. Предварительная подготовка коллекции документов для сравнения.
   1.1. Построение семантических сетей текстов документов коллекции.
   1.2. Построение рефератов текстов.
2. Сравнение текста с коллекцией документов.
   2.1. Быстрое сравнение семантических сетей.
   2.2. Детальное сравнение семантических сетей.
   2.3. Сравнение предложений реферата.
   2.4. Поиск заимствованных фрагментов.

На первом этапе работы алгоритма осуществляется предобработка с помощью технологии TextAnalyst, которая позволяет сформировать смысловой портрет текста в виде семантической сети и построить рефераты текстов.

Одной из главных функций TextAnalyst SDK (software development kit — комплект средств разработки) является построение реферата текста. Реферат представляет собой последовательность из наиболее значимых предложений текста. Смысловой вес (рейтинг) предложения вычисляется как сумма входящих в него семантических весов понятий. Далее предложения ранжируются по весу и в реферате остается *ind* самых значимых предложения, где *ind* вычисляется из заданного процентного соотношения текст/реферат (т. е. коэффициент сжатия текста $k = [0...1]$). Для определения числа предложений используют следующие формулы:

$$Ref = \bigcup_{is=1}^{ind} sentence_{is},$$

где *is* — индекс в отсортированном списке предложений (2); *ind* — индекс предложения в отсортированном списке предложений, для которого выполнено условие:

$$\lim = cumSymb_i > ksymb\_count(T),$$

где *cumSymb* — накопленная сумма отсортированных предложений (т. е. предложения находятся в списке по убыванию веса предложения); $symb\_count(T)$ — число символов в тексте T:

$$cumSymb = cumsum(symb\_count(sentence_i)),$$

где *symb_count* — функция, определяющая число символов в куске текста, *cumsum* — накопленная сумма (суммирование элементов массива с накоплением).

$$S = sort(Ws), \qquad (2)$$

где *sort* — операция сортировки (от большего к меньшему); *Ws* — вес предложения,

$$Ws_i = \sum_{j=1}^{D_i} W_j,$$

где D — число семантически значимых понятий в предложении; W — вес семантически значимого по-

нятия, $D_i = count(concept \in sentence_i)$, где *concept* — семантически значимое понятие; *sentence* — предложение рассматриваемого текста; *count* — функция подсчета элементов.

Коллекция документов представляет собой набор текстовых данных (в форматах .pdf, .txt, .doc, .html) и по размерам может меняться от библиотеки текстов до одного документа — кандидата на плагиат. В результате обработки получается набор семантических сетей, которые можно разместить в специальной базе данных в формате .xml. Следующий шаг, необходимый для ускорения процесса сравнения текстов, это создание индекса предложений (реферата) текста по включению в них понятий из семантической сети. Объем реферата можно менять по числу символов (слов), либо в процентном соотношении.

На первом шаге алгоритм сравнения использует семантическую сеть текста — кандидата на плагиат как индекс текста. Эта сеть сравнивается с базой сетей текстов документов коллекции, и вычисляет тексты с наибольшим пересечением сетей. Для дальнейшего анализа выбирается заданное количество текстов с наибольшей мерой пересечения.

Далее для каждого текста вычисляется семантическая мера сходства:

$$sD = \sum_{i \in Nodes} \sum_{j \in LinksN1_i} \sum_{k \in LinksN2_i} Nod_{ij} \cap Nod_{ik}, \quad (3)$$

где *Nodes* — это множество пересекающихся понятий (из сравниваемых текста 1 и текста 2, это пересечение вычислено на предыдущем этапе); $LinksN1_i$ — множество связей понятия под номером i с другими понятиями текста 1; $LinksN2_i$ — множество связей понятия под номером i с другими понятиями текста 2; $Nod_{ij} \cap Nod_{ik}$ — функция пересечения связанных понятий (связанных с общим i-м понятием) текстов 1 и 2.

После применения формулы (3) тексты ранжируют по весомости пересечения с исходным текстом и сокращают число текстов-источников. После этого сравнивают рефераты, соответствующие пересечениям, по предложениям. Для сравнения можно использовать стандартный алгоритм шинглов [10, 11] по близости опорных длинных предложений в тексте. Для каждого предложения строятся шинглы длиной K слов, содержащие ключевые понятия, затем шинглы хешируют. Сравнение рефратов осуществляется как сравнение хеш-кодов полученных шинглов.

Эксперименты показали, что существует более эффективный способ сравнения (см. далее рис. 3). Для нахождения одинаковых предложений будем сравнивать предложения, содержащие как минимум два одинаковых понятия. Затем для найденной пары понятий будем сравнивать частотные гистограммы всех слов в предложении (либо шинглы вокруг опорных слов).

Для нахождения фрагмента текста рассмотрим область (допустим, абзац) вокруг совпавшего предложения из реферата, и по схожему критерию из предыдущего пункта осуществим поиск совпавших предложений.

4. Эксперименты

В качестве тестовой выборки использовался набор статей по теме "Компьютерное зрение". Для каждого текста выборки были сгенерированы еще M зашумленных отмеченными далее способами текстов. Зашумленные тексты сравнивались с исходным текстом и другими текстами из базы. Использовались следующие типы зашумления:

1) удаление фрагмента текста;

2) добавление фрагментов случайных текстов.

Для случая зашумления по типу 2 рассматривался текст T_1 длины L_1 и к этому тексту добавлялись фрагменты исследуемого текста T_2^i длины L_2: $Ts^i = = T_1 \cup T_2^i$. В качестве регулируемого параметра выступает соотношение $pL = L_2/(L_1 + L_2)$. Интерес в данном случае представляло, сколько процентов исследуемого текста T_2^i содержится в зашумленном тексте Ts^i.

Для сравнения алгоритмов использовалась F-мера $F = 2PrR/(Pr + R)$, где Pr — это точность, R — полнота.

С помощью технологии TextAnalyst строились семантические сети для каждого текста и для всех зашумленных текстов. Далее последовательно осуществлялось сравнение понятий из семантических сетей зашумленного текста с базой текстов (рис. 2).

В качестве теста использовался зашумленный текст по типу 2 (т. е. из исходного текста был вырезан фрагмент и добавлен фиксированный дополнительный текст).

Рассмотрим набор текстов, зашумленных по типам 1 и 2 способов зашумления (как и в предыдущем тесте). Для всех текстов строится реферат Ref, и таким образом сокращается область перебора с помощью сравнения семантически значимых понятий (как опи-

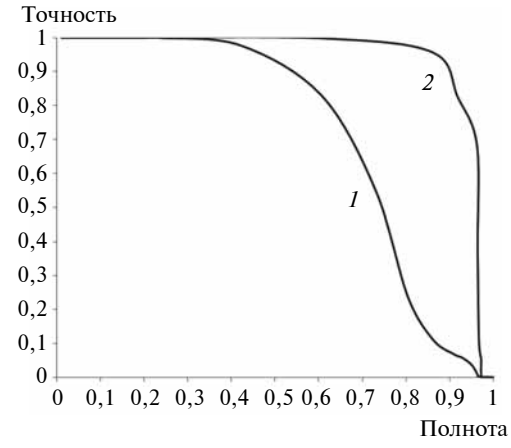


Рис. 2. Кривая точность—полнота для исследуемого текста, составляющего 25 % от общей длины текста:

кривая *1* — метод на основе частотного портрета текста; кривая *2* — метод сравнения на основе понятий семантической сети

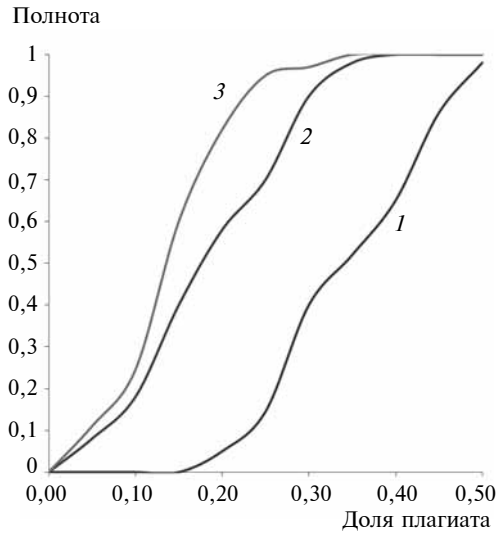


Рис. 3. Зависимости полноты определения плагиата от отношения длины исследуемого текста к длине полученного зашумленного текста

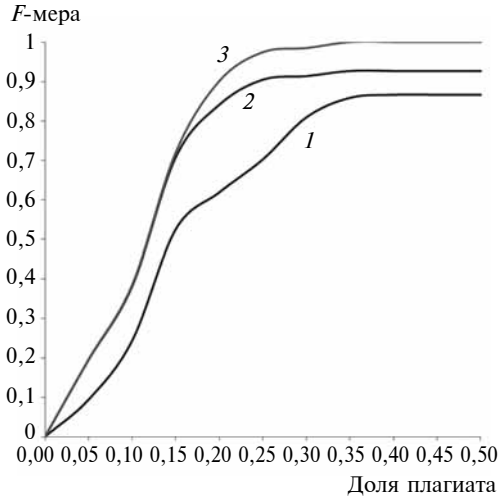


Рис. 4. Качество работы алгоритмов:
кривая *1* — стандартный подход на основе частотного портрета; кривая *2* — метод сравнения на основе понятий семантической сети; кривая *3* — многоэтапный подход проверки документа

сано выше). Для оставшихся текстов выполняется сравнение (рис. 3):

- простых шинглов;
- ранжированных шинглов, построенных вокруг семантически значимых понятий;
- супершинглов предложений.

Шинглы построены в пределах предложения с окном $n = 5$.

Предложения в реферате на тестовом наборе данных в среднем имеют 2,6 семантически значимых понятия. Окно вокруг семантически значимых понятий содержит 3,9 уникальных значения. Введем ограничение на число шинглов внутри предложения $ns = 15$. Для этого ранжируем все шинглы, как сумму семантически значимых понятий внутри них. Оставим в наборе шинглы с наибольшей суммой и, по крайней мере, по два шингла для каждого семантически значимого понятия (только в том случае, если семантически значимое понятие находится на первом и на последнем местах).

Полнота нахождения была вычислена при фиксированной точности в 95 %. На рис. 3: *1* — график сравнения рефератов по стандартному алгоритму сравнения шинглов; *2* — текст, в котором каждое предложение представлено супершинглом; *3* — график сравнения рефератов с помощью ранжированных шинглов, построенных вокруг семантически значимых понятий.

Как видно на графиках, подход с использованием сравнения семантических сетей дает прирост по отношению к подходу с использованием частотных портретов. Это видно даже при сравнении неполной модели семантической сети, в которой используется только индекс понятий семантической сети (см. рис. 2). Тестирование на исследуемой выборке показало, что метод эффективно работает при заимствовании фрагментов текста более чем на четверть (рис. 4).

Заключение

В работе рассмотрен подход к выявлению дублированных фрагментов текста, сочетающий традиционные методы, учитывающие синтагматику текста, и статистический подход к семантическому анализу текста, который позволяет ранжировать фрагменты текста по степени их важности в общем тексте.

При использовании стандартного подхода к выявлению дублированных фрагментов на основе анализа частотного портрета текста возникают трудности, связанные с необходимостью подавления несущественных слов. Для этого прибегают к технологии tf-idf (метод оценки важности слова в тексте на основе частотного портрета текста) и ее модификациям, однако подобные перевзвешивания не полностью решают возникающую задачу. Главной особенностью предлагаемого в настоящей статье подхода является использование семантического анализа целого текста. Такой подход позволяет исследовать не весь текст целиком, а только его наиболее значимые фрагменты и наиболее важные понятия (слова и словосочетания), встречающиеся в нем.

По результатам проведенных исследований удалось реализовать двухуровневый подход к выявлению элементов плагиата. На первом его этапе находятся ключевые понятия текста, и на их основе выбирают фрагменты текста, наиболее близкие к исследуемому тексту. На втором этапе сравнивают шинглы построенных на рефератах текстовой выборки и исследуемого текста. Результатом сравнения является числовая характеристика подобия анализируемого текста и индекс наиболее похожего текста из корпуса текстов. Данный подход обеспечивает достаточно высокую точность выявления элементов плагиата и имеет большие перспективы для работы с большими объемами

данных. Его использование позволяет получить выигрыш в точности выявления дублированных фрагментов при наличии заимствования в тексте более чем на четверть.

Статья написана в ходе выполнения работ по проекту "Исследование и разработка программного обеспечения понимания неструктурированной текстовой информации на русском и английском языках на базе создания методов компьютерного полного лингвистического анализа", грант Минобрнауки 2012-1.4-07-514-0018.

Список литературы

1. **Система** "Анти-Плагиат". URL: http://www.antiplagiat.ru/

2. **Система** "Плагиат-Информ". URL: http://www.plagiatinform.ru/

3. **Advego Plagiatus.** URL: http://advego.ru/plagiatus/

4. **Шарапов Р. В., Шарапова Е. В.** Система проверки текстов на заимствования из других источников // Труды II Всероссийской научной конференции "Электронные библиотеки: перспективные методы и технологии, электронные коллекции. RCDL' 2011". Воронеж, 19—22 октября 2011 г. Воронеж: Полиграфический центр Воронежского гос. ун-та, 2011. С. 233—238.

5. **Зеленков Ю. Г., Сегалович И. В.** Сравнительный анализ методов определения нечетких дубликатов для WEB-документов // Труды IX Всероссийской научной конференции "Электронные библиотеки: перспективные методы и технологии, электронные коллекции RCDL' 2007". Сб. работ участников конкурса. Переславль-Залесский, 2007. Т. 1. С. 166—174.

6. **Неелова Н. В.** Исследование лексического метода вычисления схожести строк веб-документа с учетом предварительной обработки // XXXV Гагаринские чтения: Научные труды Международной молодежной научной конференции в 8 томах. М.: МАТИ, 2009. Т. 4. С. 31—32.

7. **Manber U.** Finding Similar Files in a Large File System // Proc. of Winter USENIX Technical Conference, 1994. P. 1—10. URL: http://www.cs.princeton.edu/courses/archive/spr05/cos598E/bib/manber94finding.pdf

8. **Heintze N.** Scalable document fingerprinting // Proc. of the Second USENIX Workshop on Electronic Commerce, Oakland, California. November 18—21, 1996. P. 191—200.

9. **Гасфилд Д.** Строки, деревья и последовательности в алгоритмах. СПб.: Невский диалект, 2003.

10. **Broder A. Z., Charikar M., Frieze A. M.** et al. Min-wise independent permutations // Proc. of the thirtieth annual ACM symposium on Theory of computing, 1998. New York, NY, USA. ACM-Press, 1998. P. 327—336.

11. **Broder A., Glassman S., Manasse M. and Zweig G.** Syntactic clustering of the Web // Proc. of the 6th International World Wide Web Conference. Santa Clara, CA, USA. April 7—11, 1997. P. 391—404.

12. **Fetterly D., Manasse M., Najork M. A.** Large-Scale Study of the Evolution of Web Pages // In the proceedings of the 12th World Wide Web Conference (WWW2003). Budapest, Hungary. May 20—24, 2003. ACM Press, 2003. P. 669—678.

13. **Chowdhury A., Frieder O., Grossman D., McCabe M.** Collection statistics for fast duplicate document detection // ACM Transactions on Information Systems (TOIS). 2002. V. 20, Is. 2. P. 171—191.

14. **Kolcz A., Chowdhury A., Alspector J.** Improved Robustness of Signature-Based Near-Replica Detection via Lexicon Randomization // Proc. of KDD-2004. Seattle, Washington. August 22—25 2004. New York: ACM, 2004. P. 605—610.

15. **Pugh W., Henzinger M.** Detecting duplicate and near — duplicate files. US Patent 6658423. Google, Inc., 2001.

16. **Ilyinsky S., Kuzmin M., Melkov A., Segalovich I.** An efficient method to detect duplicates of Web documents with the use of inverted index // In Proc. of The Eleventh International World Wide Web Conference (WWW Conference 2002). Honolulu, Hawaii, USA. 7—11 May, 2002. URL: http://www2002.org/CDROM/poster/187/

17. http://www.yandex.ru

18. **Программа** Praide Unique Content Analyser II. URL: http://www.nado.su/downloads.html

19. **Ширяев М. А., Мустакимов В.** Plagiatinform избавит от плагиата в научных работах // Образовательные технологии и общество. 2008. Т. 11, № 1. С. 367—374.

20. **Харламов А. А.** Нейросетевая технология представления и обработки информации (естественное представление знаний). М.: Радиотехника, 2006. 89 с.

ИНФОРМАЦИЯ

УДК 004.912

А. В. Бахтин, аспирант, вед. программист, НИИ механики МГУ имени М. В. Ломоносова,
e-mail: anton@bakhtin.ru

К созданию программного комплекса для извлечения метаинформации о научно-технических конференциях

Объектом исследований, результаты которых представлены в настоящей статье, являются способы извлечения данных из сообщений о научных конференциях. Оперативное использование и анализ результатов научно-технических исследований в интересах отдельных ученых и структур управления наукой возможны только после их опубликования. В отличие от периодики, книг, патентов и т. п., более оперативно подобные результаты публикуются в трудах конференций. Однако еще раньше появляются объявления о предстоящих конференциях. Содержащаяся в них информация представляет интерес не только для специалистов, работающих по тематике соответствующих конференций. Она позволяет также получать оперативную информацию об активности исследований в тех или иных областях знаний, о внимании к ним со стороны научного сообщества и о ряде других факторов, что определяет актуальность настоящей работы.

Ключевые слова: *извлечение информации, автоматическая обработка текста*

Введение

Для современного состояния научных исследований характерен высокий уровень динамизма. Появляются новые области знаний и соответствующие им направления научного поиска, исследования в некоторых областях, напротив, по тем или иным причинам замедляются. Анализ и выявление этих причин — важная задача для научного сообщества и структур управления наукой на разных уровнях. Однако определение как наиболее актуальных, так и зарождающихся направлений научных исследований — непростая задача. Использование информации о публикующихся статьях в журналах и книгах не позволяет добиться достаточной оперативности. Более того, журналы в большинстве своем консервативны в выборе тематики, а книги зачастую аккумулируют знания, полученные в предшествующие их публикации годы. Информация, полученная путем анализа сообщений о конференциях, лишена этого недостатка. Этот факт обусловлен тем, что временной интервал от окончания рассмотрения заявок до начала конференции составляет всего несколько месяцев. Кроме того, хотя конференции, как и журналы, посвящены некоторому общему направлению научного поиска, конкретный список ключевых тем (так называемые topics of interest) меняется от года к году и представляет непосредственный интерес.

Информация о прошедших и планируемых конференциях существенна при решении широкого круга задач. Наличие базы знаний о датах, тематике и месте проведения конференций позволит создать программную систему с удобными механизмами поиска и навигации по предстоящим событиям, которая будет учитывать месторасположение пользователя и область его интересов. Данные о программном комитете кон-

ференции в совокупности с внешними данными об индексе цитируемости его участников могут служить основой для автоматического построения рейтинга конференции. Такая информация, в свою очередь, может быть использована при оценке научной работы сотрудников специализированными аналитическими системами, такими как система ИСТИНА [1].

В сети Интернет можно найти большое число ресурсов, собирающих сообщения о предстоящих конференциях[1]. Они предоставляют разную степень детализации имеющихся данных: на некоторых ресурсах известно только название и текст объявления о конференции (так называемые CFP, *Call For Papers*), другие ресурсы предоставляют дополнительную информацию о времени проведения и важных датах. Однако все подобные ресурсы обладают одним существенным недостатком — пополнение базы конференций осуществляется вручную пользователями или модераторами. Вследствие этого каждый ресурс имеет ограниченный и узкоспециализированный охват.

В работе [2] была предпринята попытка автоматизации сбора и извлечения данных из сообщений о конференциях. В качестве источника информации использовались некоторые неуказанные электронные рассылки с текстами CFP. Из них извлекались четыре поля: название конференции, конечный срок подачи статей, список основных тем и место, страна и город проведения. Хотя алгоритм строился на регулярных выражениях, гео-словаре и простейших эвристиках, его точность оказалась достаточно высокой: 82 % для названия конференции, 93 % для конечного срока и 94 % для места проведения. Однако в работе не была использована информация об используемых графических и структурных элементах, которая может существенно повысить качество распознавания, и извлекалось ограниченное число полей.

Методы выделения структурных элементов долгое время изучались в области оптического распознавания символов (*optical character recognition*, OCR). Так, в работе [3] предложен метод выделения таблиц как задающих разметку, так и содержащих информацию, а в работе [4] представлены алгоритмы выделения таких структурных элементов, как списки, заголовки сноски.

В рамках задачи извлечения информации из CFP распознавание структурных элементов, таких как список ключевых тем, также играет важную роль. Однако в отличие от оптического распознавания, единицы измерения не растровые, а символьные, и многие графические элементы эмулируются с помощью символьной псевдографики. Более того, некоторые типы источников (подробный анализ источников приведен в следующем разделе) могут содержать HTML-разметку с той же семантической нагрузкой (таблицы, списки).

Для анализа сообщений о конференциях необходимо представить информацию в виде структурированных данных. Понятие структуры можно рассматривать как на логическом, так и на физическом уровнях. На логическом уровне простейшей структурой для представления информации о конференции является фрейм. Фрейм представляет отображение из пары (объект, поле) в значение. Число полей и их названия при этом задают заранее. Значением каждого поля может быть как число или текстовый литерал, так и фрейм. Таким образом, фрейм позволяет представлять простейшие древовидные онтологии. Несмотря на то что такое представление не содержит некоторых интересующих в контексте целей текущей работы отношений, оно предоставляет объем информации, достаточный для автоматического построения таких отношений.

В данной работе предложена модель представления исходного текста, упрощающая его анализ и позволяющая помимо текста хранить его семантическую разметку, в которую может быть отображена имеющаяся HTML-разметка текста. Процесс анализа текста представляется в виде последовательности правил, каждое из которых выполняет определенную операцию, такую как распознавание списков. Информация о найденных правилом элементах сохраняется в разметке и может быть использована следующими правилами. Разметка, получившаяся после применения всех правил, может быть использована для заполнения фрейма о конференции.

Типы возможных источников

Основным источником данных о планируемых и состоявшихся конференциях являются сообщения о конференциях, размещенные в сети Интернет. Такие данные могут находится в следующих форматах.

- **Неструктурированный текст.** Под неструктурированными понимают тексты без машинно-читаемой разметки. Однако это не подразумевает отсутствия визуальной разметки, такой как списки с элементами, выделенными маркером "звездочка" или подобными, или заголовки, выделенные строками из нескольких знаков равенства. Поскольку подобные тексты обычно пишет человек, то можно считать, что они содержат полную в его представлении информацию о событии и ничего кроме нее. В качестве примера такого источника можно рассматривать оповещения о кон-

[1] http://www.wikicfp.com/cfp/; http://call-for-papers.sas.upenn.edu/; http://grid.hust.edu.cn/call/index.jsp; http://conference.researchbib.com/; http://www.h-net.org/announce/group.cgi?type = CFPs; http://eventseer.net/

ференциях в виде рассылок электронных писем (например, DBWorld[2]) .

- **HTML-документ или сайт.** Подавляющее большинство конференций имеют свои сайты. На многих из них есть страница с основной информацией, содержащая все необходимые потенциально заинтересованному пользователю сведения о конференции[3]. В этом случае перед разработчиком алгоритма, призванного анализировать информацию, стоит задача определить, является ли страница искомой, и выделить из нее подлежащую анализу информацию. В более сложном случае, либо неизвестно на какой странице сайта записана вся информация, либо такой страницы вообще не существует. В этой ситуации алгоритм дополнительно должен уметь отличать объекты, относящиеся к данной конференции, от случайных упоминаний о других конференциях, событиях, местах и датах.
- **Глобальная сеть.** Если пользователю (анализатору) доступна лишь частичная информация о конференции, например, ее название, то можно воспользоваться поиском по сети Интернет для извлечения недостающих данных.

Каждому типу данных соответствует своя задача извлечения информации. Несложно понять, что задача последнего типа сводится ко второй. Однако в данном случае анализирующий алгоритм должен также иметь возможность выдать отказ от распознавания в ситуации, если найденный ресурс не связан с конференциями. Кроме того, для поиска релевантных документов необходима поисковая система. В качестве таковой можно как использовать внешнюю поисковую систему общего назначения, так и реализовать полнотекстовый поиск по определенному срезу страниц сети.

В свою очередь, вторая задача сводится к первой с помощью средств рендеринга HTML-страниц при условии, что либо можно обнаружить страницу с полной информацией, либо создать суперстраницу из страниц с частичной информацией.

Таким образом, задача выделения данных из неструктурированных текстов является ключевой и рассмотрению методов ее решения посвящены следующие разделы.

Построение первичной модели CFP

Неструктурированные тексты содержат большое количество графической информации, задающей структуру и упрощающей навигацию по сообщению для человека. Также для удобства чтения абзацы текста могут быть разбиты на короткие строки (около 80 символов) одинаковой длины. Такая разметка легко воспринимается человеком, однако для автоматической обработки она лишь усложняет применение существующих методов выделения сущностей, основанных на анализе контекста. Чтобы преодолеть эту трудность можно воспользоваться обсуждаемыми далее алгоритмами распознавания структуры документа, которые заменяют в тексте графические элементы на разметку, описывающую положение структурных элементов в тексте. Обработка проводится в несколько шагов, решающих отдельные представленные далее задачи.

Формальное представление текста

Входной текст является непустой последовательностью символов в некотором алфавите. Предполагается, что текст является осмысленным для человека и написан на некотором естественном языке. Вследствие того, что обработка проводится в несколько этапов, необходимо определить формальную модель текста, которая будет хранить текст и результаты применения правил его анализа. Данная модель должна позволять хранить, изменять и просматривать текст, а также помечать отдельные интересующие объекты в тексте.

Модель хранения определяется атомарным элементом, над которым осуществляют преобразования. Он должен, с одной стороны, позволять осуществлять необходимые операции, а с другой стороны, максимально упрощать и стандартизировать такие операции. Существует четыре естественных варианта для атомарной единицы, а именно символ, слово, предложение и абзац. В первом случае каждое правило имеет в своем распоряжении всю информацию, и может выполнять любые операции. Однако при таком представлении каждое правило вынуждено заново выполнять простейшие подготовительные операции, например, разбиение на строки. В случае если в качестве атомарных элементов выступают предложения или абзацы, многие вспомогательные операции уже заложены в модель. Однако при таком подходе невозможно описать объекты меньшие, чем предложение, такие как названия стран и университетов, не нарушая принципов атомарности. По этой причине в роли атомарного элемента модели выступают слова.

Поскольку для дальнейших правил представляют интерес не только слова, но и прочие символы, будем считать, что текст представляется в виде последовательности токенов. Каждый токен содержит некоторый фрагмент текста и может иметь один из четырех типов, а именно токен-слово (*word*), токен-пунктуация (*punct*), токен-пробел (*space*) и токен-перенос строки (*nl*). Токен-слово содержит максимальную непрерывную подпоследовательность алфавитно-

[2] http://research.cs.wisc.edu/dbworld/browse.html
[3] Например, http://www.fdg2010.org/Call_for_Papers.html

цифровых символов в тексте. Токен-пунктуация содержит ровно один символ — пунктуационный знак. Токен-пробел соответствует последовательности пробелов, неразрывных пробелов, табуляций или других пробельных символов, но объект токена всегда содержит ровно один пробел. Вследствие этого все токены-пробелы равны между собой. И последний тип токенов, токен-перенос строки, возникает при обработке символов переноса строки в тексте.

Для выделения заголовков, выровненных по центру с использованием пробельных символов, необходимо иметь возможность вычислять расстояние от начала строки. Заметим, что для этого достаточно хранить позицию начала токена в тексте. Действительно, если данный токен первый, то его позиция и есть число предшествующих пробельных символов. Если предыдущий токен есть токен-перенос строки, то разность их позиций есть искомое значение. В противном случае, токен не обрамлен пробельными символами.

Таким образом, после токенизации текст представляет собой отображение Tokeinize: $\{1, 2, ..., n\} \to T$, где n — длина текста в токенах, а T — множество всех возможных токенов. Каждый токен суть тройка (Σ^+, *type*, *offset*), где Σ — используемый алфавит текста, $type \in \{word, punct, space, nl\}$ — тип токена, а *offset* — позиция токена в тексте.

Кроме токенизации, которая позволяет хранить текст в удобном для дальнейшей обработки виде, необходимо иметь возможность помечать определенные структурные (списки, заголовки) и семантические (названия городов, университетов) элементы в этом тексте, выраженные непрерывной последовательностью токенов. Для этого в модели предусмотрена структура данных, которую в дальнейшем будем называть *плиткой*.

Каждая плитка имеет тип, атрибуты, номер начального токена и номер последнего токена. Текст плитки определяется как конкатенация текстов токенов между первым и последним накрываемыми токенами. Атрибуты плитки представляют собой простой словарь "ключ-значения", и их множество зависит от типа плитки. Например, для плитки типа *A*, соответствующей гиперссылке, является обязательным атрибут с ключом *href*, а ее текст есть ссылочный текст, который содержался между открывающимся и закрывающимся тегами ⟨*a*⟩ в исходном тексте.

Таким образом, плитка — суть четверка ($position_{start}$, $position_{end}$, *type*, *attrs*), где $position_{start}$ и $position_{end}$ — позиции соответственно первого и последнего накрываемых токенов, $type \in \mathfrak{E}$ — тип плитки из конечного множества типов $\mathfrak{E}$, $attrs \in (L^+ \to \Sigma^+)$ — словарь атрибутов (L — латинский алфавит, Σ — алфавит текста). Разметка суть конечное множество плиток $M = \{Tile_1, Tile_2, ..., Tile_n\}$. Модель текста есть пара (T, M), состоящая из списка токенов и разметки.

Построение модели документа. Обрабатываемый документ может содержать HTML-разметку или не содержать. В первом случае список токенов для модели получается с помощью простой токенизации текста, а разметка представляет собой пустое множество.

При построении модели документа, содержащего HTML-разметку, ее необходимо удалить, а имеющуюся информацию закодировать с помощью плиток. Следует заметить, что HTML-разметка покрывает символы, а разметка с помощью плиток покрывает токены, однако текст плитки должен совпадать с текстом исходного HTML-тега. Для этого осуществляется грубая токенизация HTML-документа, и именно он представляется в виде последовательности элементов одного из трех типов: текст, открывающийся тег, закрывающийся тег. В случае если тег закрывает себя сам, то он игнорируется. Далее выполняется следующий алгоритм.

- Вход: список l, полученный после грубой токенизации документа.
- Выход: модель документа $\mathfrak{M} = (T, M)$.
- Подготовка: создать стек S текущих открытых тегов, функция *name*(*tag*), возвращающая тип тега *tag*, функция *attributes*(*tag*), возвращающая словарь параметров тега *tag*.
- Взять следующий элемент e с индексом i из l и выполнить:
  - Если e — открывающийся тег, то
    - положить в стек S пару $(e, |T|)$.
  - Если e — закрывающийся тег, то
    - выбрать из стека последний элемент (e_{start}, $position_{start}$);
    - проверить, что e является закрывающимся тегом для e_{start};
    - положить в разметку M элемент ($position_{start}$, $|T| - 1$, $name(e_{start})$, $attributes(e_{start})$).
  - Если e — текст, то
    - построить список токенов T' с помощью простой токенизации;
    - если список T не пуст, то взять последний токен t_{last} из T' и ко всем токенам t из T' применить: $t.offset \leftarrow t.offset + t_{last}.offset + length(t_{last}.word)$, где $length(\cdot)$ — длина слова;
    - приписать T' в конец T.

Работа с моделью

Описанная первичная модель представления текста практически не содержит никакой дополнительной информации кроме той, которая содержалась в тексте в явном виде. Для выделения дополнительной информации применяется последовательность правил, модифицирующих модель и добавляющих различные данные. Данные из разметки в дальнейшем могут быть использованы для заполнения фрейма о конференции.

В качестве примера работы с моделью далее описано извлечение одного поля фрейма — списка ключевых тем. Для этого используются три правила преобразования модели, а именно — правило, исправляющее искусственные переносы строк, правило, выделяющее списки, и правило, выделяющее ключевые темы.

Исправление искусственных переносов строк

Многие сообщения, поступающие в текстовом виде, содержат принудительные переносы строк. Такие тексты затрудняют восстановление абзацев, что, в свою очередь, ведет к семантически несвязанным блокам текста. Простейший подход, объединяющий все строки некоторой фиксированной длины с предыдущей, дает плохие результаты, так как перенос длинных строк проводится по словам, а не по символам. Кроме того, даже если известна длина строки, существует ненулевая вероятность встретить строку подобной длины вне абзаца, например, в списке. Ниже приведено типичное описание списка ключевых тем конференции.

```
participation. We seek to invite papers that address various aspects
of e-government projects from a theoretical, conceptual, or empirical
perspective to set the stage for future research direction in citizen
centric e-government efforts. Both quantitative as well as qualitative
studies on e-government from developed and developing countries will
be encouraged.
Contributed papers may deal with, but are not limited to:
* Theories and conceptual models informing citizens adoption of e-government
* Theories and models of citizens' satisfaction with e-government
implementations
* Citizens participation in e-democracy and e-governance
* Cross country comparison of citizens perspectives on e-government
```

Для решения задачи исправления искусственных переносов строк автором была разработана модель, которая для каждого переноса строки определяет, является ли он искусственным или нет. Модель использует реализацию алгоритма машинного обучения Случайный лес в библиотеке[4] языка R.

Пространство факторов. Факторы строились для групп подряд идущих символов переноса строки. Для каждой вычислялись следующие факторы:

1) *max_line_length* — длина максимальной строки в тексте;

2) *line_length* — длина текущей строки[5];

3) *length_difference* — разница между средней длиной строки в тексте и длиной текущей строки (средняя длина вычисляется как среднее арифметическое длин непустых строк);

4) *line_no* — минимум из номера строки и числа 10;

5) *line_no_from_end* — минимум из номера строки от конца и числа 10;

6) *nl_group_size* — число токенов в данной группе переносов строк;

7) *avg_nl_group_size* — среднее число токенов в группах переносов строк в документе;

8) *ends_with_stop_punct* — флаг того, что строка заканчивается на токен-пунктуацию, соответствующий восклицательному или вопросительному знакам или точке;

[4] http://cran.r-project.org/web/packages/randomForest/index.html

[5] Здесь и далее под текущей строкой понимается последовательность токенов от группы переносов строк, для которых строятся факторы, до предыдущей группы переносов строк.

9) *ends_with_dangling_word* — флаг того, что текст последнего непробельного токена текущей строки лежит в множестве *of*, *on*, *the*, *a*, *comma sign*, *an*, *dash sign*, *and*;

10) *last_tok_ freq_overall* — число раз, которое последний непробельный токен текущей строки был последним токеном абзаца в корпусе текстов;

11) *last_tok_ratio* — отношение *last_tok_freq_overall* к общему числу раз, которое последний непробельный токен текущей строки встретился в корпусе текстов;

12) *next_is_title* — флаг того, что текст первого непробельного токена следующей строки является прописным словом;

13) *starts_with_ punct* — флаг того, что первый непробельный токен текущей строки есть токен-пунктуация;

14) *common_ form_prev_ prefix_len* — число непробельных начальных токенов, совпадающих по форме с начальными непробельными токенами предыдущей строки;

15) *common_ form_next_prefix_len* — число непробельных начальных токенов, совпадающих по форме с начальными непробельными токенами следующей строки.

Используемые факторы можно поделить на три группы, а именно — статистические (1—7), лексические (8—11) и контекстные (12—15).

Статистические факторы не используют информацию внутри строк, а только информацию об их длинах и порядковых номерах. Фактор 1 должен помочь машинному обучению определить, требуется ли в данном тексте исправление строк или нет. Это предположение основано на замечании, что в текстах с обрезанными строками максимальная длина строки невелика. Факторы 2 и 3 определяют, насколько похожа длина текущей строки и потенциальная длина полей. Факторы 4 и 5 позволяют машинному обучению особым образом обрабатывать начальные и конечные строки, так как в них могут содержаться декоративные графические элементы, выходящие за поля. В некоторых документах для разделения абзацев используют два символа переноса строки подряд, и один — для принудительного переноса, а в других — два символа переноса используют во всех случаях. Данное замечание учтено в факторах 6 и 7.

Лексические факторы используют наблюдения, связанные с тем, что некоторые токены часто стоят в конце предложений, а некоторые, напротив, почти никогда. Факторы 8 и 9 используют фиксированные списки токенов подобного рода. Факторы 10 и 11 следуют той же идее, но строятся динамически по внешнему корпусу, в котором принудительные переносы заведомо отсутствуют. В данной работе в качестве такого корпуса использовались тексты из Википедии.

Контекстные факторы извлекают сигнал из окружающих строк. Фактор 12 позволяет осуществить проверку, что следующая строка начинается с прописного слова, что говорит в пользу естественности данного переноса строки. Факторы 13, 14 и 15 обнаруживают случаи, когда короткая строка в данном месте не должна объединяться со следующей, например, потому что данная строка является элементом списка или перечисления.

Результаты обучения. Для обучающей выборки было выбрано 30 текстовых сообщений о конференциях. Для каждого документа экспертом была построена версия с исправленными принудительными переносами строк. Далее сравнивались две версии документа, и находились естественные переносы строк, т. е. те, которые есть в обоих документах, и искусственные, содержащиеся только в оригинальном документе. Первое множество формировало отрицательные примеры, а второе — положительные. Такой выбор меток обусловлен тем фактом, что удаленные переносы строк требуют от алгоритма модификации текста и вследствие этого более важны. Описанным образом была сформирована обучающая выборка в более чем 2000 примеров. Классификатор для каждого конкретного переноса строки решает, стоит его удалить или нет. Для измерения качества работы классификатора использовался метод контроля по отдельным объектам. Поскольку часть документов из обучающей выборки не требовала исправления переносов строк, а следовательно, не содержала положительных примеров, то для таких документов и документов с некорректными переносами строк применялись разные метрики.

Для документов, не содержащих некорректных переносов строк, ключевой метрикой качества является точность. Таких документов содержалось в обучающей выборке 10, что порождало 371 обучающий пример. Из них в 370 (99,7 %) классификатор принял правильное решение, отказавшись от объединения строк.

Для документов с переносами строк важен баланс точности и полноты. Поэтому в качестве целевой метрики была выбрана F_1-мера[6]. Двадцать документов этого множества породили 1665 обучающих примеров. На этом множестве была достигнута средняя F_1-мера 88,9 %.

Выделение списков

Список является одним из важнейших структурных элементов документа. С помощью списка могут быть представлены такие данные, как ключевые темы, программный комитет, и др. Описанный ниже

[6] F_1-мера учитывает и полноту, и точность, и вычисляется по формуле: $F_1 = 2\frac{precision \cdot recall}{precision + recall}$, где *precision* — точность, а *recall* — полнота.

алгоритм выделяет ненумерованные списки и предполагает, что принудительные переносы строк удалены.

Алгоритм выделения списков состоит из трех этапов. На первом этапе строятся кластеры из строк токенов с одинаковым префиксом. Префиксом является начальная последовательность токенов строки, не содержащая токенов-слов, с исключением токенов-пробелов.

На следующем этапе происходит объединение идущих подряд кластеров с похожими префиксами. Префиксы считают похожими, если один из них является префиксом для другого. Данный этап призван сгладить небольшие вариации маркера списка, которые могли быть допущены при наборе.

На заключительном этапе кластеры переводятся в разметку. Названия типов плиток выбрано из соображения согласованности с восстановленной разметкой HTML. Поэтому токены от первого токена первой строки кластера до последнего токена последней строки кластера покрываются плиткой с типом *UL*. А токены каждой строки кластера за исключением префикса — плиткой с типом *LI*.

Выделение списка ключевых тем

Выполнение описанных выше преобразований делает выделение ключевых тем тривиальной задачей. А именно — выберем все плитки типа *UL* и для каждой выберем *N* (параметр алгоритма) предшествующих токенов. Добавим плитке атрибут *is_topic* со значением *true*, если в предшествующих токенах встречается одна из фраз маркеров, таких как:

- *possible topics include*;
- *interests include*;
- *topics of interest*;
- *but not limited to, the following areas.*

Заметим, что на данном шаге не имеет значения, были ли построены списки на предыдущем шаге, или они были перенесены из HTML при построении первичной модели.

Заключение

В работе представлены методы и средства решения задачи извлечения данных из сообщений о научно-технических конференциях. Анализ результатов исследований на этом направлении показал, что большинство существующих методов и программных решений этой задачи основаны на ручном вводе и разборе данных, а исследования по автоматическому разбору носят фрагментарный характер. В данной работе предложена модель данных, позволяющая упростить и стандартизировать процесс выделения информации из CFP, представив его в виде последовательности правил преобразования. На данном этапе работы реализовано извлечение одного из полей, а именно списка ключевых тем. В дальнейшем планируется построить аналогичные правила для заполнения остальных полей фрейма, а также провести общее тестирование качества работы системы.

Список литературы

1. **Васенин В. А., Афонин С. А., Козицын А. С., Голомазов Д. Д., Бахтин А. В., Ганкин Г. М.** Интеллектуальная система тематического исследования научно-технической информации // Обозрение прикладной и промышленной математики. 2012. Т. 19, № 2. С. 239—240.

2. **Correia F. L., Amaro R. F. S., Sarmento L., Rossetti R. J. F.** Allcall: An automated call for paper information extractor // Information Systems and Technologies (CISTI), 2010. 5th Iberian Conference on. 16—19 June. 2010. IEEE, 2010. P. 1—4.

3. **Klink S., Dengel A., Kieninger T.** Document structure analysis based on layout and textual features // In Proc. of International Workshop on Document Analysis Systems, DAS2000. Citeseer, 2000. P. 99—111.

4. **Shafait F., Smith R.** Table detection in heterogeneous documents // In Proc. of the 9th IAPR International Workshop on Document Analysis System. 09—11 June, 2010. Boston, Massachusetts, USA. ACM, 2010. P. 65—72.

УДК 004.912

А. Ю. Чугунов, аспирант, Механико-математический факультет МГУ им. Ломоносова,
e-mail: arcady.chugunov@gmail.com

Автоматическое контекстно-зависимое аннотирование текстовых документов

Рассматривается актуальная задача контекстно-зависимого аннотирования текстовых документов с учетом поискового запроса пользователя.

Приводится краткий обзор существующих работ по данной тематике, анализ их достоинств и недостатков. Рассматривается математическая модель аннотирования документов. Особое внимание уделяется методу спектрального оценивания лексических единиц текста, приводится соответствующий математический аппарат. Представлены результаты работы программной реализации алгоритма контекстно-зависимого аннотирования KGCDA на примере новостных публикаций. На основе полученных результатов сделаны выводы и сформулированы направления расширения функциональных возможностей выбранного для реализации алгоритма.

Ключевые слова: *автоматическое аннотирование, документ, контекстно-зависимый подход, динамическое аннотирование, спектральные характеристики лексем, алгоритм Кенни-Гудмана*

Введение

Автоматическое контекстно-зависимое аннотирование документов представляет собой практически значимую задачу. Предоставление кратких аннотаций к результатам выдачи де факто является стандартом современных систем информационного поиска. Подобная функция повышает качество и скорость поиска, позволяя пользователю оценить релевантность результатов еще до подробного знакомства с ними.

Основным объектом аннотирования в рамках данной статьи является документ. Под *документом* в данной работе понимается совокупность предложений естественного языка, объединенных единой предметной областью и составляющих семантически целостный текст. Под *автоматическим аннотированием* в данном случае понимается составление аннотации с использованием средств вычислительной техники без участия человека. Сочетание "контекстно-зависимое" указывает на зависимость генерируемой аннотации от контекста, который закладывает пользователь в постановку задачи, вводя поисковый запрос. Такой тип аннотирования называется *динамическим*. Его противоположностью является *статическое* аннотирование документа, или *реферирование*, при котором аннотация составляется без привлечения внешнего контекста, только на основе анализируемых документов. Следует отметить, что документ, для которого генерируется аннотация, как правило, рассматривается в рамках некоторой *коллекции* (*корпуса*) *документов* — т. е. набора документов, объединенных каким-либо общим критерием (тематически, типически, структурно).

Выделяют два основных подхода к решению задачи автоматического аннотирования (ЗАА), а именно *квазиреферирование* (метод составления выдержек, или экстракция) и *краткое изложение содержания* (абстракция). Большинство работ по данной тематике, часть из которых будет рассмотрена далее, базируются на первом подходе. Он требует существенно меньше вычислительных ресурсов, легко масштабируется на большие объемы данных, а соответствующие алгоритмы, как правило, довольно "прозрачны" по схеме их релизации. Следует однако отметить, что в настоящее время активно ведутся разработки в рамках второго подхода, предполагающего использование словарей-онтологий, синтаксический разбор текста и генерацию естественно-языковых конструкций.

Квазиреферирование

Метод составления выдержек сводится к выделению из исходного (подлежащего аннотированию) документа характерных фрагментов, в роли которых обычно выступают предложения или их части. Отобранные фрагменты компонуются в аннотацию. Значимость фрагментов оценивается с точки зрения релевантности запросу и частоты встречаемости входящих во фрагменты лемм. Как правило, для оценивания фрагментов используют *модель линейных весов*. Вес фрагмента U рассчитывается по общей формуле

$$Weight(U) = Location(U) + CuePhrase(U) + \\ + StatTerm(U) + AddTerm(U).$$

Весовой коэффициент расположения (*Location*) в этом случае определяется фактом того, где во всем тексте или в отдельно взятом параграфе появляется данный фрагмент — в начале, в середине или в конце, а также тем обстоятельством, используется ли он в ключевых разделах, например, в вводной части или в заключении.

Весовой коэффициент ключевой фразы (*CuePhrase*) зависит от того, встречаются ли в блоке лексические или фразовые резюмирующие конструкции. К их числу относятся такие как "в заключение", "в данной статье", "согласно результатам анализа" и т. д. или принятый в данной предметной области оценочный термин (например, "высокоэффективный" или "малозначащий").

Кроме перечисленных выше, при расчете весовых коэффициентов с использованием этой модели учитывается показатель статистической важности (*StatTerm*). Он демонстрирует, насколько весóм данный фрагмент в смысле наличия в нем высокочастотных терминов.

Наконец, коэффициент дополнительного наличия терминов (*AddTerm*) характеризует наличие в данном блоке терминов, которые также встречаются в заголовке, в колонтитуле, в первом параграфе, в запросе пользователя и т. д.

Основные подходы

Как было отмечено выше, рассматриваемый метод состоит в "интеллектуальном" семантическом обобщении исходного документа и вполне может содержать термины, которых в документе нет. Существуют два основных подхода к формированию краткого изложения содержания анализируемого документа.

- Использование традиционного лингвистического метода синтаксического разбора предложений. С его помощью на основе исходного документа строится *дерево разбора*. Затем построенное дерево упрощается путем сокращения ветвей на основании некоторых структурных критериев, таких как скобки или части сложносочиненных и сложноподчиненных предложений. После такой процедуры дерево разбора существенно упрощается, и на его основе строится аннотация.

- В отличие от первого подхода, который имеет дело с лексическими структурными блоками текста, второй подход оперирует концептуальными блоками и опирается на понимание естественного языка с использованием методов искусственного интеллекта. Он также предполагает синтаксический разбор одним из этапов, однако деревья разбора в этом случае не порождаются. Напротив, формируются концептуальные репрезентативные структуры исходного текста, которые аккумулируются в текстовой базе знаний. В качестве операций, связывающих структуры, могут быть использованы формулы логики предикатов или такие представления, как семантическая сеть.

В процессе преобразования концептуальное представление подлежащего анализу текста претерпевает несколько изменений. Избыточная и вторичная информация устраняется путем отсечения концептуальных подграфов. Затем информация подвергается дальнейшему агрегированию путем слияния графов или обобщения информации, например, при помощи таксономических иерархий отношений подклассов. Для выполнения этих преобразований предложены методологии на базе выводов. В их основе могут быть макроправила, которые манипулируют логическими предположениями, или операторы, которые выделяют определяющие шаблоны в текстовой базе знаний. В результате преобразования формируется концептуальная репрезентативная структура документа.

Краткий обзор существующих работ

Задача автоматического аннотирования является очень актуальной в свете стремительного (и с каждым годом кратно увеличивающегося) роста объемов информации в сети Интернет. Эта информация настолько разнородна, что без систем интеллектуального поиска найти нужные сведения не представляется возможным. Как следствие, поисковые системы постоянно совершенствуют (что и можно наблюдать на примере поисковых корпораций, таких как "Яндекс" и "Google"). Разработчики добавляют все новые средства, ускоряющие и упрощающие поиск. Механизм формирования кратких аннотаций является одним из таких средств.

Первые подходы к решению ЗАА были разработаны в конце 50-х годов XX века (H. P. Luhn, 1958 г.), и развиты в следующее десятилетие (G. J. Rath, 1961 г.; H. P. Edmundson, 1969 г.). Лун и Эдмундсон предложили простые методы генерации аннотации из фрагментов, характеристиками которых выступали частота встречаемости, наличие слов из заголовка и слов-маркеров, расположение предложения в тексте. Однако их подход никак не учитывал структурную составляющую текста.

Первая попытка исправить этот недостаток была сделана в работе Д. Марку [1]. Он предложил подход с использованием так называемых RST-деревьев. Они представляют собой деревья разбора, построенные на основе *риторического статуса* фрагментов, например "основной смысл", "обстоятельство", "причина" и по-

добные им. В таком дереве наиболее значимые с точки зрения смысла фрагменты расположены в листьях, и степень значимости этих фрагментов убывает с расстоянием до листа. Такой подход позволяет гибко настраивать параметры аннотации, например, делать ее более лаконичной или информативной, однако весьма затратен в смысле необходимых для его реализации вычислительных ресурсов. Для каждого нового документа необходимо построить свое RST-дерево, а это "дорогой" процесс.

Формальное деление фрагментов текста на значимые и незначимые сводится к решению задачи бинарной классификации машинного обучения. В работе [2] описан метод, использующий наивный байесовский классификатор. В качестве обучающей выборки в работе был использован корпус научных статей с аннотациями. В качестве определяющих параметров были использованы следующие:

- длина предложения;
- содержание определенных речевых конструкций;
- положение в параграфе (начало, середина, конец);
- содержание ключевых (высокочастотных) терминов;
- содержание акронимов.

В работе [3] рассматривался случай ЗАА применительно к коллекции научных статей. Специфика задачи была связана со структурно-смысловой неоднородностью статьи (по отношению, например, к новостному сообщению). Кроме того, научные статьи могут иметь довольно существенный объем, и сжатие ее до аннотации "стандартной" длины (10—20 предложений) неизбежно приведет к потере значимых фрагментов. В соответствии с отмеченными факторами, в статье [3] предложен усовершенствованный алгоритм аннотирования, учитывающий риторический статус фрагментов и использующий байесовский классификатор для определения значимости.

Еще более развитый подход к решению задачи описан в работе [4]. Алгоритм, названный *Grasshopper* (Кузнечик), базируется на *теории поглощающих случайных блужданий* на графе, разработанной авторами. Он призван взять лучшее от алгоритмов Page/Text/LemRank [5], принадлежащих классу машинного обучения без учителя (*unsupervised machine learning*), и свести к минимуму избыточность содержания аннотации. Авторы отметили, что стремятся к тому, чтобы выбранные предложения были и значимы, и разнообразны.

Последние разработки в области автоматического аннотирования ежегодно представляют на конференции Ассоциации Вычислительной Лингвистики (*Association for Computational Linguistics*). Так, в 2012 г. была представлена работа [6], где предложен алгоритм, все стандартные этапы которого (анализ, выделение фрагментов и генерация аннотации) используют чисто абстрактные методы и обработку естественного языка. Авторы используют так называемые *абстрактные схемы*, содержащие предзаданные лексико-синтаксические шаблоны и эвристики извлечения информации. Первые результаты тестирования демонстрируют хорошие перспективы описанного метода. Авторы убеждены, что путь "полной абстракции" наиболее эффективен для имитации аннотаций, составленных человеком. Причина в том, что такой подход позволяет моделировать "понимание" текста.

Российские разработки в области автоматического аннотирования

Также как и за рубежом, данное направление автоматического аннотирования развивается и в России, в том числе в последнее десятилетие. В работе [7], например, В. А. Яцко предложил метод симметричного реферирования. Его суть состоит в том, что вес предложения определяется числом связей между данным предложением и предложениями, находящимися слева и справа от него. Для этого в каждом предложении определяется список ключевых слов, входящих в предварительно составленный тематический словарь. Затем в предложениях, расположенных слева и справа, подсчитывается число найденных в них ключевых слов (связей) из определенного ранее списка. Сумма лево- и правосторонних связей определяет вес предложения.

В работе [8] авторами представлены три алгоритма аннотирования, перечисленные далее.

- **Базовый алгоритм.** Обусловлен "наивным" подходом к решению задачи. Данный алгоритм анализирует в документе только леммы запроса и выбирает фрагмент с их наибольшей плотностью. Из текста последовательно извлекают связные фрагменты длины, не превосходящей заданную, и оценивают при помощи весовой функции:

$$W = \sum W_i + Kn/L.$$

Здесь первое слагаемое представляет собой сумму весов лемм запроса, вошедших во фрагмент. Каждое слово учитывается только один раз. Вес каждого слова зависит от распределения слова в коллекции и он тем выше, чем более редкое это слово. K — некоторая числовая константа; n — число слов из запроса, которые встретились во фрагменте; L — расстояние между первым и последним словами запроса, встретившимся во фрагменте.

Таким образом, указанная формула оценивает выше фрагменты, в которых слова запроса располагались более "кучно", встречалось больше слов запроса, а также тех слов, которые реже встречались во всей коллекции документов. В аннотацию включается фрагмент с наибольшим весом. Если веса фрагментов совпадают, то выбирается тот, что ближе к началу текста.

Данный алгоритм обладает невысокой эффективностью в силу своей простоты, а также тех соображений, что в случае если запрос содержит единственный термин или же высокочастотное словосочетание, то в качестве аннотации будет выдан первый фрагмент, содержащий указанный термин.

- **Алгоритм Freq.** Данный алгоритм является расширением базового: кроме слов самого запроса учи-

тывается также встречаемость высокочастотных для данного документа терминов в анализируемом фрагменте. Такое усовершенствование обусловлено тем обстоятельством, что если в качестве запроса для базового алгоритма подано одно слово, либо высокочастотное словосочетание, то в качестве аннотации будет выдан первый фрагмент, содержащий слова запроса.

Из соображений быстродействия и возможной смысловой неоднородности текста, частоты слов насчитывают не по всему документу, а в рамках окна длиной в M слов, для которого анализируемый фрагмент является центром. Далее выбирают N слов, которые встречаются в данном фрагменте наиболее часто. Весовая функция рассчитывается по формуле

$$W_{freq} = W_b + \Sigma \log_2 F_k,$$

где W_b — вес слова, вычисленный базовым алгоритмом; F_k — частота k-го слова в рамках окна.

Данный алгоритм показывает лучшие результаты по сравнению с предыдущим. Однако он также далек от совершенства, поскольку на больших документах неприменим по причине их внутренней смысловой неоднородности.

- **Алгоритм LRU-K.** Оба описанных выше алгоритма используют при анализе лишь частоту встречаемости слова в документе, что никаким образом не характеризует его распределение в тексте. Кроме того, для определения частоты необходимо хранить все слова, содержащиеся в анализируемом фрагменте, а это повышает "вычислительную стоимость" алгоритма. Данный же алгоритм призван устранить указанные недостатки.

В работе [9] представлен алгоритм KGCDA (*Kenny-Goodman context-dependent annotation*). Он основан на построении многофакторной модели оценивания фрагментов текста и оптимизации ее параметров при помощи обучающей выборки документов. В качестве контекстно-зависимых критериев используют спектральные оценки лемм, о которых будет рассказано ниже. Этот алгоритм обладает наиболее высокими показателями качества и быстродействия (по результатам оценки на дорожке контекстно-зависимого аннотирования РОМИП-2009[1]). При этом он относительно прост в программной реализации по сравнению с приведенными выше алгоритмами аннотирования. По этой причине алгоритм KGCDA был положен в основу программного компонента для аннотирования документов, который разрабатывается автором. Результаты первого этапа исследований на этом направлении представлены в следующем разделе.

[1] РОМИП — Российский семинар по оценке методов информационного поиска. Дорожка — отдельная секция семинара, посвященная какой-либо одной конкретной задаче информационного поиска со строго определенными правилами оценки систем-участников.

Алгоритм контекстно-зависимого аннотирования KGCDA

Алгоритм контекстно-зависимого аннотирования KGCDA [9], который был представлен выше, относится к классу экстрактивных алгоритмов и работает по следующей схеме.

- *Фрагментация.* Используется метод скользящего окна: текст обрабатываемого документа разбивается на фрагменты, которые представлены либо предложениями, либо их связной частью, если длина предложения превосходит заданную длину аннотации. В первом случае фрагменты, очевидно, не пересекаются, во втором же случае возможны наложения, которые учитываются на завершающем этапе алгоритма.
- *Оценивание* фрагментов при помощи описанных ниже спектральных характеристик.
- *Ранжирование* фрагментов по убыванию весов.
- *Построение аннотации* путем последовательного извлечения элементов отсортированного списка фрагментов, пока выполняется ограничение на длину текста, с исключением добавления пересекающихся фрагментов.

Отметим, что полученная в результате работы алгоритма аннотация не обязательно является односвязным фрагментом документа. Она может состоять из нескольких предложений, разнесенных по смыслу, но в совокупности передающих содержание текста.

Математическая модель процесса аннотирования

Рассмотрим математическую модель, описывающую процесс контекстно-зависимого аннотирования [10].

Лексемой называется слово как абстрактная единица морфологического анализа. В одну лексему объединяются разные парадигматические формы (словоформы) одного слова. В словарях каждая лексема представлена одной из словоформ, которую называют исходной (нормальной) формой — или *леммой*, а сам процесс сведения словоформы к лемме называют *лемматизацией*.

В общем виде динамическая задача построения аннотации к документу может быть выражена следующим образом. Входные данные:

- документ, для которого строится аннотация, $D = \{d_i\}$ — множество лексем;
- запрос пользователя $Q = \{q_i\}$ — множество лексем;
- коллекция документов для обучения $Coll = \{D_j\}$.

На выходе получаем аннотацию (множество лексем) $A = A(D) = \{a_i\}$.

О спектральных характеристиках лемм. Пусть существует некоторая коллекция документов $Coll = \{D_j\}$, среди которых выделяем один документ $D = \{d_i\}$. Стандартные методы оценки лексических единиц (такие как TF.IDF) учитывают внутреннюю частоту встречаемости леммы лишь по данному документу. В работе [10] авторы предлагают метод оценивания лексем документа D с учетом их внутренней частоты встречаемости в каждом из документов коллекции

Coll. Далее перечислены основные вводимые ими характеристики, которые именуют *спектральными*.

- Внутренняя частота леммы *IF(L, D)* — число вхождений леммы L в документ D.
- Условные частоты

$$CLF(L, \nu) = Card(D|IF(L, D) = \nu)$$

— число документов D с внутренней частотой леммы L, равной ν

$$CLF2(L, \nu) = Card(D|IF(L, D) \geqslant \nu)$$

— число документов D с внутренней частотой леммы L, больше либо равной ν.

- Абсолютная документальная частота леммы $DF(L) = CLF2(L, 1)$.
- Обратная условная частота леммы

$$ICLF(L, \nu) = \frac{DF(L)}{CLF(L, \nu)}.$$

- Мера вхождения фрагмента F в запрос Q

$$IFQ(F, Q, D) = \sum_{q_j: q_j \in F} ICLF(q_j, IF(q_j, D))$$

и мера вхождения запроса Q во фрагмент F

$$IQF(F, Q, D) = \sum_{f_j: f_j \in Q} ICLF(f_j, IF(f_j, D)).$$

- По двум сформированным метрикам вводится свертка вида

$$FM = b_1 IFQ^{a_1} + b_2 IQF^{a_2}. \quad (1)$$

Здесь a_i и b_i, $i = 1, 2$ — постоянные коэффициенты.

О выборе констант в весовой функции

Константы a_i и b_i, входящие в формулы спектральных оценок, которые отмечались выше, были выбраны на основе исследований, описанных в работе [9]. Выбор значений констант проводился при помощи экспертной оценки:

$$a_1 = -1,\ b_1 = 1,\ a_2 = 1,\ b_2 = 10^{-18}.$$

Подставляя эти значения в формулу (1), получим следующую формулу для весовой функции:

$$FM1 = \frac{1}{IFQ} + 10^{-18} IQF. \quad (2)$$

Методики оценки аннотаций

Существуют два основных метода оценки качества автоматически сгенерированных аннотаций — внутренний (*intrinsic*) и внешний (*extrinsic*) [13].

Внутренние методы оценки. Этот подход предполагает оценку собственно текста аннотации по следующим критериям:

- являются ли предложения аннотации грамматически правильными;
- является ли текст аннотации связным;
- содержит ли аннотация все основные обсуждаемые темы исходного документа и др.

Для оценки автоматически сгенерированная аннотация сравнивается независимыми экспертами с аннотациями, сделанными другими людьми, участниками эксперимента. Эксперты выставляют оценки по пятибалльной шкале по каждому из критериев. Могут использоваться и более сложные методы экспертной оценки.

Сравнение автоматических аннотаций с теми, которые построены другими участниками эксперимента, может проводиться и в автоматическом режиме, без участия экспертов. Для этого применяют перечисленные далее метрики.

- Точность (*Sentence Recall*) определяет, насколько много общих предложений содержат сравниваемые аннотации. Эта метрика наиболее эффективна для экстрактивных методов.
- Ранг (*Sentence rank*) оценивает аннотации в терминах "значимости" предложений и отдельных слов. Эта метрика также используется в основном для экстрактивных методов.
- Полезность (*Utility-based*) является более точной формализацией свойства "информативности" аннотации. Эта метрика более универсальна, чем две предыдущих, и ближе к способу оценки человека.
- Содержательность (*Content-based*) сравнивает аннотации с точки зрения похожести извлеченных из них словарей. Преимуществом такой метрики является возможность не учитывать количества предложений в аннотациях. Наиболее эффективна эта метрика для экстрактивных или абстрактных методов, использующих большое число совпадений с исходным текстом.

При автоматическом оценивании новостных кластеров используют такие метрики, как ROUGE (*Recall Oriented Understudy for Gisting Evaluation*), которая подсчитывает число перекрытий (n-граммы слов) автоматической аннотации с "идеальными" аннотациями, составленными людьми [12].

Другой мерой оценки качества аннотаций является метод пирамид, который основан на ручном выделении экспертами "информационных единиц" из эталонных аннотаций — *Summary Content Units* (*SCUs*) и вычислении процентной доли этих единиц, упомянутых в автоматических аннотациях.

Кроме способов, перечисленных ранее, для оценки аннотацию разумно сравнить с текстом, из которого она получена. Особенно это относится к случаю абстрактных методов и составлению обзорных рефератов. Для этого применяют перечисленые далее подходы:

- *Семантический*. Заключается в том, что из документов коллекции выделяют ключевые и неключевые концепции. Аннотация оценивается с точки зре-

ния наличия в ней концепций, покрывающих содержание всех документов.

- *Поверхностный.* Вместо того чтобы выделять основные темы текста, данный подход предполагает выделение ключевых предложений текста с точки зрения соответствия тематике текста (которая предполагается заданной). Аннотация же в этом случае оценивается по критериям наличия в ней ответов на определенные вопросы, актуальные данной тематике. Эксперт выставляет оценку по шкале "имеется", "имеется частично" и "отсутствует".

Внешние методы оценки. В отличие от внутренних методов внешние предполагают оценивание аннотации с точки зрения решения задач, поставленных перед нею, т. е. косвенное оценивание. Критерии при этом могут быть весьма разнообразны. К их числу могут относиться:

- релевантность, которая характеризует, насколько аннотация помогает пользователю найти желаемое;
- эффективность с точки зрения системы, в которую встроен модуль аннотирования;
- понятность прочтения аннотации, которая характеризует возможность ответить на основные вопросы по тексту на основе полученной аннотации.

Выбор между внутренними и внешними методами оценки аннотации напрямую зависит от потребностей системных архитекторов и решаемых ими задач. Подробный обзор методов оценки аннотаций можно найти в работе [13].

Оценка качества работы программной реализации алгоритма KGCDA

Оценка качества работы программной реализации алгоритма KGCDA на первом этапе проводилась вручную, что обусловлено технической сложностью применения упомянутых выше метрик. Далее представлены некоторые результаты работы алгоритма на новостных публикациях (см. таблицу).

Новость	Запрос	Аннотация
Один человек погиб и пять пострадали в аварии, произошедшей в субботу вечером на Ленинградском шоссе Москвы, сообщил РИА Новости представитель пресс-службы столичного главка МВД РФ. "Четвертого мая в 18.27 напротив дома 39, строение 76 по Ленинградскому шоссе водитель автомобиля BMW не справился с управлением и выехал на встречную полосу, где столкнулся с автомобилем Daewoo Nexia. В результате аварии один из пассажиров последнего автомобиля скончался на месте. Остальные пассажиры иномарок и оба водителя были госпитализированы", — сказал собеседник агентства. По его словам, в настоящее время стражи порядка устанавливают все обстоятельства произошедшего.	москва новости	Один человек погиб и пять пострадали в аварии произошедшей в субботу вечером на ленинградском шоссе москвы сообщил риа новости представитель ... По его словам в настоящее время стражи порядка устанавливают все обстоятельства произошедшего ... Остальные пассажиры иномарок и оба водителя были госпитализированы сказал собеседник агентства ... Четвертого мая в напротив дома строение по ленинградскому шоссе водитель автомобиля не справился с управлением и выехал на встречную полосу ...
Премьер-министр РФ Дмитрий Медведев предложил упростить процедуру электронной записи к врачу через интернет. В понедельник премьер побывал в администрации Красногорска, где ознакомился с курсами повышения компьютерной грамотности для пенсионеров. В присутствии Медведева находившаяся на курсах женщина попыталась записаться к врачу в местную больницу через интернет. Премьер внимательно изучил все этапы этой процедуры и отметил, что требования об указании электронного адреса, а также верификации пользователя (необходимость ввести последовательность цифр, изображенных на картинке), являются излишними. По его словам, электронный адрес есть далеко не у всех. Что касается верификации, то Медведев заметил, что распознать цифры на картинке "даже для человека с острым зрением не всегда легко". Премьер-министру рассказали, что с 2009 по 2012 годы обучение на этих курсах прошли чуть более тысячи пенсионеров и инвалидов. "Здорово", — отметил Медведев. Также он осмотрел экспозицию военно-исторического клуба "Искатель", расположенную там же — в администрации Красногорска. Медведев отметил энтузиазм членов поискового клуба, а также заинтересовался найденной ими финской полевой печкой.	медведев	Премьер-министр рф дмитрий медведев предложил упростить процедуру электронной записи к врачу через интернет передает риа новости ... Что касается верификации то медведев заметил что распознавать цифры на картинке даже для человека с острым зрением не всегда легко ... кроме того медведев осмотрел фотовыставку поколение победителей состоящую из портретов ветеранов войны ... В присутствии медведева находившаяся на курсах женщина попыталась записаться к врачу в местную больницу через интернет ...

Продолжение таблицы

Новость	Запрос	Аннотация
Премьер похвалил и фотографии поисковиков, представленные на выставке. "Прямо как из кинофильмов фотографии", — заметил глава кабинета министров. Кроме того, Медведев осмотрел фотовыставку "Поколение победителей", состоящую из портретов ветеранов войны. "Хорошие портреты, кстати", — оценил работы премьер, поинтересовавшись, по какой технологии они были напечатаны		
На одной из недавних встреч с молодежью президент Российской Федерации Владимир Владимирович Путин затронул проблему студенческого спорта. Вопрос этот действительно очень важный, и от его правильного решения в стране зависит многое. Нет нужды говорить о пользе здорового образа жизни. Здесь важнее другое. Воспитание характера, умения концентрироваться в критических ситуациях, выработка настроя на победу — все эти чисто спортивные качества благополучно перекочевали в нашу повседневную жизнь. Она, кстати, по всем параметрам напоминает сегодня спортивный поединок, так что студенческий спорт — это еще и своеобразная школа жизни для молодежи.	путин студенческий спорт	На одной из недавних встреч с молодежью президент российской федерации владимир владимирович путин затронул проблему студенческого спорта ... она кстати по всем параметрам напоминает сегодня спортивный поединок так что студенческий спорт это еще и своеобразная школа жизни для ...
С 22 по 26 мая в ЦДХ на Крымском валу будет проходить XVIII международная выставка архитектуры и дизайна Арх Москва, главная и уникальная особенность которой — принцип избирательности: здесь представлена лучшая российская и зарубежная архитектура. На выставке будут представлены более 200 компаний-участников из 9 стран мира — Австрии, Белоруссии, Бельгии, Великобритании, Германии, Италии, Латвии, России и Франции. Разделы выставки включают в себя архитектуру, дизайн мебели, экстерьерные и интерьерные решения, свет и детали. Кроме того, в рамках выставки пройдут более 50 мероприятий (мастер-классы, лекции, семинары) с участием специалистов со всего мира, посвященные проблемам градостроительства, экологии, доступного жилья, развития инфраструктуры в регионах и охраны памятников архитектуры.	москва выставки	Крупные московские архитектурные проекты заявлены в новом формате — на выставке Новая Москва будут представлены наиболее значимые объекты города, как реализованные ... С 22 по 26 мая в ЦДХ на Крымском валу будет проходить XVIII международная выставка архитектуры и дизайна Арх Москва, главная и уникальная ... Выставка Зеленая Москва продемонстрирует эти территории — парки, скверы, бульвары и набережные ...
В целом, программа Арх Москвы 2013 объединена темой Next!, это взгляд в будущее, новые имена, тенденции, технологии; новые здания, города, новые отношения и взаимосвязи. Крупные московские архитектурные проекты заявлены в новом формате — на выставке Новая Москва будут представлены наиболее значимые объекты города, как реализованные, так и находящиеся в стадии проекта. Выставка продемонстрирует направления развития московского градостроения. Особое внимание уделено зеленым городским общественным пространствам. Выставка Зеленая Москва продемонстрирует эти территории — парки, скверы, бульвары и набережные. В 2013 году Арх Москва Next! обещает стать самым крупномасштабным шоу архитектурных талантов в мире. На выставке будет представлен международный конкурс Archiprix International: показ 300 лучших дипломных работ выпускников архитектурных школ во всем мире. Премия Авангард — конкурс для молодых российских архитекторов — в этом году посвящена архитектору Константину Мельникову. В конкурсе участвуют проекты 20 номинантов и 4 финалистов.		

На основе полученных результатов можно сделать следующие выводы:

- Алгоритм KGCDA генерирует хорошо читаемые аннотации. Данное качество обусловливается тем фактом, что фрагменты, из которых составлена аннотация, представляют собой начала предложений.
- Явно прослеживается наличие зависимости аннотации от запроса: фрагменты с ключевыми словами запроса всегда встречаются в аннотации.
- При достаточной длине аннотации фрагменты, из которых она состоит, в совокупности покрывают список тем исходного документа.

Принимая во внимание определенную долю субъективизма в оценке, тем не менее можно отметить, что полученные аннотации хорошо передают основное содержание документа. Как следствие, они достаточно эффективно могут быть использованы в качестве удобочитаемых подсказок для получения более релевантных результатов выдачи при поиске документов.

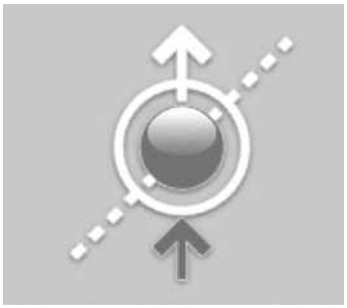
Заключение

В статье рассмотрена задача выбора и программной реализации эффективного алгоритма из числа существующих алгоритмов динамического аннотирования. После рассмотрения и анализа эффективности существующих алгоритмов, решающих поставленную задачу, в качестве базового был выбран алгоритм KGCDA (*Kenny-Goodman context-dependent annotation*). Данный алгоритм основан на построении многофакторной модели оценивания фрагментов текста документа и оптимизации ее параметров при помощи обучающей выборки документов. В качестве контекстно-зависимых критериев используются спектральные оценки лемм. Аргументами для выбора алгоритма KGCDA среди других были эффективность составления хорошо читаемых и передающих основное содержание документа аннотаций, а также простота реализации схемы его работы. Указанные преимущества описанного алгоритма доказаны его авторами на дорожке контекстно-зависимого аннотирования РОМИП-2009, где алгоритм занял первое место.

Результаты, полученные автором после тестирования работы алгоритма, демонстрируют, что алгоритм KGCDA обладает заявленной эффективностью и хорошо применим для динамического аннотирования небольших по длине документов. Проблема длинных документов заключается в их смысловой дифференцированности. Длинные документы могут содержать более одной ключевой мысли и темы. По этой причине при автоматическом аннотировании таких документов нужно, как минимум, учитывать их структуру (деление на параграфы, заголовки и пр.). Данные соображения задают направления для усовершенствования алгоритма KGCDA и, соответственно, его программной реализации.

Список литературы

1. **Marcu D.** The Rhetorical Parsing, Summarization, and Generation of Natural Language Texts. Department of Computer Science, University of Toronto, 1997.
2. **Kupiec J., Pedersen J., Chen F.** A Trainable Document Summarizer // SIGIR'95 Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval. N. Y.: ACM, 1995. P. 68—73.
3. **Tuefel S., Moens M.** Summarizing Scientific Articles: Experiments with Relevance and Rhetorical Status // Computational Linguistics — Summarization. Cambridge, MA, USA: MIT Press, 2002. P. 409—445.
4. **Goldberg A.** Advanced NLP: Automatic Summarization. 2007. URL: http://pages.cs.wisc.edu/~jerryzhu/cs838/summarization.pdf
5. **Page L., Brin S., Motwani R., Winograd T.** The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Libraries, 1998.
6. **Genest P., Lapalme G.** Fully Abstractive Approach to Guided Summarization // ACL'12. Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers. Stroudsburg, PA, USA: ACL, 2012. Vol. 2, P. 354—358.
7. **Яцко В. А.** Симметричное реферирование: теоретические основы и методика // Научно-техническая информация. Сер. 2. 2002. № 5. С. 18—28.
8. **Губин М. В., Меркулов А. И.** Эффективный алгоритм формирования контекстно-зависимых аннотаций // Международная конференция "Диалог'2005". Труды. М.: Наука, 2005. С. 116—120.
9. **Зябрев И. Н., Пожарков О. В.** Метод контекстно-зависимого аннотирования документов на основе спектральных оценок лексем KGCDA // Труды РОМИП'2009. СПб.: НУ ЦСИ, 2009. С. 167—174.
10. **Зябрев И. Н., Пожарков О. В.** Спектральное оценивание лексических единиц в задачах лингвистического моделирования. URL: http://www.altertrader.com/publications16.html
11. **Jones K. S., Galliers J.** Evaluating Natural Language Processing Systems: An Analysis and Review. Lecture Notes in Artificial Intelligence 1083. Berlin: Springer-Verlag, 1996.
12. **Lin Chin-Yew.** ROUGE: a Package for Automatic Evaluation of Summaries // Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004). Barcelona, Spain: ACL, 2004. P. 74—81.
13. **Mani I.** Summarization Evaluation: An Overview // In Proceedings of the NTCIR Workshop 2. Meeting on Evaluation of Chinese and Japanese Text Retrieval and Text Summarization. Tokyo, Japan: National Institute of Informatics, 2001. P. 77—85.



VIII Международная научная конференция

"Параллельные вычислительные технологии" (ПаВТ'2014)

31 марта—4 апреля 2014 г., Южный федеральный университет (г. Ростов-на-Дону)

ПаВТ — серия международных научных конференций, представляющих собой авторитетный и престижный форум в области применения параллельных вычислительных технологий в различных областях науки и техники.

Учредителями конференции являются Российская академия наук и Суперкомпьютерный консорциум университетов России.

В первый день работы конференции будет объявлена ***20-я редакция списка Top 50*** самых мощных компьютеров СНГ.

Тематика конференции охватывает следующие основные направления (но не ограничивается ими)

- Технологии параллельных и распределенных вычислений
- Грид и облачные вычисления
- Перспективные многопроцессорные архитектуры
- Параллельные и распределенные системы баз данных
- Администрирование, мониторинг и тестирование многопроцессорных систем
- Вычислительная математика
- Вычислительная физика
- Вычислительная химия
- Гидро-газодинамика и теплообмен
- Высоконелинейные и быстротекущие процессы в задачах механики
- Биоинформатика
- Нанотехнологии
- Климат и экология
- Криптография
- Визуализация
- Компьютерная алгебра
- Суперкомпьютерные научно-образовательные центры.

В рамках конференции будет работать ***индустриальная сессия***. На индустриальную сессию принимаются высококачественные презентации по инновационному коммерческому аппаратному и программному обеспечению, ориентированному на применение суперкомпьютерных и параллельных вычислительных технологий в различных областях науки и техники.

Официальный сайт конференции: http://ПаВТ.РФ

CONTENTS

ООО "Издательство "Новые технологии". 107076, Москва, Стромынский пер., 4

Дизайнер *Т. Н. Погорелова*. Технический редактор *Е. М. Патрушева*. Корректор *Е. В. Комиссарова*

Сдано в набор 09.09.2013 г. Подписано в печать 22.10.2013 г. Формат 60×88 1/8. Заказ PI1113
Цена свободная.

Оригинал-макет ООО "Адвансед солюшнз". Отпечатано в ООО "Адвансед солюшнз".
119071, г. Москва, Ленинский пр-т, д. 19, стр. 1.