ОПЫТ ПРИМЕНЕНИЯ ТЕХНОЛОГИЙ КОМПЬЮТЕРНОЙ ЛИНГВИСТИКИ В ПРОМЫШЛЕННОМ ПРОГРАММНОМ ОБЕСПЕЧЕНИИ

Д.С. Скатов, В.В. Окатьев, Т.Н. Ерехинская

Нижегородский госуниверситет им. Н.И. Лобачевского
ООО «Диктум», Нижний Новгород

Программное обеспечение, решающее задачи компьютерной лингвистики (морфологический, синтаксический анализ, исправление опечаток, извлечение фактов, эмоциональная оценка текста и т.д.) обладает техническими свойствами, обусловленными как самой его природой, так и контекстом применения. Авторский опыт разработки и внедрения такого ПО отражен в настоящем докладе.

Введение

Решение задач компьютерной лингвистики требуется в тех приложениях, где необходимо извлечь информацию из текстов на естественном языке. Извлечение понимается в широком смысле – это не только классические задачи information retrieval (информационный поиск, извлечение фактов), но и, например, обеспечение функционирования систем человеко-машинного взаимодействия (напр., поисковая подсказка, учитывающая опечатки).

В период с 2002 по 2011 г. участниками коллектива была разработана линейка программных модулей для решения ряда задач компьютерной лингвистики:

- синтаксический анализатор естественного языка DictaScope Syntax (русский, языки романо-германской группы) [1, 2, 4, 6, 8];
- модуль извлечения именованных сущностей DictaScope Tokenizer (извлечение персон, организаций, дат, адресов, чисел прописью, событий, названий географических объектов и т.д.) [7];
- модуль анализа иерархически структурированных текстов DictaScope Structure [3, 5];
- модуль бессловарной морфологии русского языка;
- система исправления опечаток в текстах на естественном языке [10]; модуль поисковой подсказки с поддержкой исправления опечаток DictaScope Query Processor;
- модуль разрешения анафор местоимений 3-го лица DictaScope Anaphora [9];
- модуль оценки мнений по шкале позитив-негатив DictaScope Opinion Mining.

Разработка моделей и методов ведется в сотрудничестве с факультетом вычислительной математики и кибернетики Нижегородского государственного университета им. Н.И. Лобачесвкого. Результаты совместных исследований отражены в статьях и научно-технических отчетах [1-10]. Первое промышленное внедрение синтаксического анализатора русского языка состоялось в 2006 году [11], а с 2008 года коллектив под руководством авторов создает на базе обозначенных выше модулей решения для различных программных B2B и B2C продуктов, а также для обеспечения внутренних нужд заказчика [14].

Расширяемость

Модуль для обработки текстов на естественном языке по своей природе является системой, управляемой данными. Наличие управляющих лингвистических данных по-

чти всегда является необходимым условием функционирования модуля. Под лингвистическими данными понимаются: а) статистическая информация, полученная по корпусам аннотированных текстов; б) управляющие данные, представленные обычно правилами на специально разработанном языке и табличными информационными базами. Можно оценить вклад данных и алгоритмов в конечный результат работы модуля либо как примерно одинаковый, либо же данные являются более важными. Например, коллектив из 3-х человек был занят в разработке программного ядра DictaScope Tokenizer в течение года, а описание управляющих данных заняло 6 человеко-лет и продолжается в настоящий момент.

Эволюция лингвистического модуля представима обычно следующим процессом. В первую очередь создается алгоритм, способный обрабатывать определенный класс текстов, и он целиком описывается программным кодом. Далее в результате достаточного количества экспериментов вырабатывается определенный способ описания управляющих данных – структуры таблиц и DSL (domain-specific language), после чего из кода выделяется алгоритмическое ядро. Дальнейшее развитие заключается преимущественно в наполнении управляющих данных (словарей, правил и т.д.) с обеспечением желаемой полноты и точности. Примером может служить текущее развитие модуля DictaScope Structure [5].

Другой пример представляет собой синтаксический анализатор DictaScope Syntax [4]. Вид управляющих данных в целом понятен из самой постановки задачи построения дерева подчинительных связей предложения, а качественно новый результат был получен улучшением самого алгоритма анализа – переход от переборного процесса к полиномиальному [1, 2] позволил использовать анализатор в промышленных приложениях, а дальнейшее улучшение алгоритма было связано с расширенной поддержкой пунктуации [8].

Требования к функциональности

Процесс работы над продуктом начинается с получения общей картины по имеющимся данным и желаемым выходным результатам. Составить детальную спецификацию для лингвистического продукта на данном этапе практически невозможно. Далее идет серия итераций, длиной от недели до месяца каждая, в результате (обычно в активном контакте с заказчиком) создается прототип желаемого изделия. Затем первые образцы попадают в тестовые версии системы (ранняя интеграция), последующее развитие идет на основе обратной связи от тестирующих со стороны заказчика.

Мультиязычность

Создание определенного программного модуля всегда связано с конкретным языком, обычно русским. После внедрения полученных результатов возникает необходимость в портировании технологии на другие языки. Получить изначально мультиязычный продукт затруднительно (как в силу требований по срокам, так и исходя из принципиальной разрешимости такой задачи), поэтому версия для поддержки второго языка оформляется как ветвь кода для первого. В процессе работы над второй версией начинаются работы по созданию мультиязычного модуля. Это, как правило, требует не только модификации управляющих данных, но и появления новых точек управления по данным внутри алгоритмов анализа.

Эффективность

Программное обеспечение, разрабатываемое коллективом, используется в приложениях с высокими требованиями к производительности. Например, система оценки мнений должна ежедневно обрабатывать сотни мегабайт нового текстового материала для обслуживания десятков клиентов. Модуль поисковой подсказки с исправлением

опечаток должен предоставлять производительность как минимум в десятки запросов в секунду на один процесс уже на стадии обкатки (с итоговой расчетной производительностью, достаточной для обработки тысяч запросов в секунду). В современных экономических условиях цену программного проекта, результат которого планируется в дальнейшем масштабировать (например, привлечением все большего числа клиентов в SaaS-продукт), нежелательно завышать стоимостью оборудования. Оперативная память является недорогим ресурсом, но не безграничным. Сложные программные системы включают множество компонентов, каждый из которых должен предоставлять разумные требования к объему необходимой памяти. У поставщиков такого рода продуктов есть склонность к использованию уже существующих вычислительных мощностей. Невысокая производительность лингвистического ПО в принципе допустима на стадиях обкатки, учитывая его изначально экспериментальную природу, но, в конечном счете, при успехе проекта должна быть возможность исправить ситуацию.

Кроме того, надо учитывать следующий факт. Экспоненциальная скорость роста информации (во многом представленной в текстовом виде) опережает скорость снижения стоимости вычислительных ресурсов. Мощности процессоров растут, но гораздо быстрее растет объем данных, подлежащих обработке. В этом контексте показателен пример компании Google, которая за счет рационально сконструированного ПО смогла использовать для поддержки функционирования всех своих проектов недорогое, типовое оборудование [13] на этапе активного их развития и закрепления рыночных позиций (первая половина 2000-х годов).

Удовлетворить требования производительности возможно несколькими путями.

Создание более эффективных алгоритмов и применение смешанных стратегий решения. Второе особенно важнó, т.к. позволяет получить желаемые результаты в срок, при этом даже улучшив эффективность модуля. Оптимизация уже существующего алгоритма – задача чаще всего глубоко исследовательская, с высокой вероятностью неудачи. Вместо этого при работе с лингвистическими данными полезен следующий прием: нужно изучить статистику реальных обрабатываемых данных и построить алгоритм, хорошо обрабатывающий наиболее частые образцы входов. Иногда удается построить критерий, указывающий, когда активировать специфичный алгоритм, а когда – общий. Специфичный алгоритм при этом получается более эффективным.

Улучшение емкостных характеристик структур данных. Алгоритмическая оптимизация в итоге приводит исследователя к следующему барьеру – кэшу процессора. Среди двух реализаций одной схемы обработки выиграет та, что более компактно способна обрабатывать данные, т.е. та, которая максимизирует порцию данных, способных быть загруженными в кэш процессора для обработки. Можно получить интересные результаты, не меняя алгоритма, но увеличив компактность данных (упаковка битов, альфа-кодирование, введение дополнительных уровней иерархии в данных и др. техники).

Кэширование. Процессор лингвистических данных традиционно представляет собой многослойный стек субобработчиков, так что получение выхода по заданному входу получается довольно трудоемким. Однако части входов при этом чаще всего повторяются в течение сеанса использования процессора. Эффективным является ограниченный кэш, поддерживающий в оперативной памяти результат обработки наиболее частых входов.

Техническая оптимизация. Сюда входит применение разнообразных инструментов профилирования и обнаружения «бутылочных горлышек». Они действительно позволяют получить приемлемые по эффективности результаты за несколько трудодней профилировки кода.

Кроссплатформенность

Специфика применения лингвистических модулей, разрабатываемых коллективом, состоит в том, что они, как правило, становятся частями серверного Интернет-приложения. В качестве серверной ОС используется FreeBSD и (реже) Linux, причем 64-битные версии. Использование платформы amd64 удобно тем, что не приходится решать вопросы выхода индексов за 4 Гб границу, это способствует более раннему запуску продуктов и большей эффективности их функционирования (больше данных можно загрузить в оперативную память без своппирования). При этом с помощью средств ОС обеспечивается отказоустойчивость модулей (перехват отказов и последующее возобновление работы, ограничение по времени функционирования, оформление модулей в виде fcgi-приложений и т.д.).

Критерии качества

Традиционной мерой качества в задачах information retrieval являются полнота (Recall) и точность (Precision) [12]. На начальных этапах конструирования продукта измерения ведутся «на глазок» по небольшому набору тестовых примеров. С определенной точки развития вводятся инструменты автоматизированного регрессионного тестирования с измерением показателей качества.

В задачах, где результаты работы предоставляются человеку, больший вес имеет точность, возможно, в ущерб полноте. Например, система может детектировать не все упоминания конкретной персоны, а лишь некоторые из них, но желательно, чтобы почти все выявленные системой упоминания были именно о данной персоне.

Обратный приоритет может иметь место в случаях, когда текст анализируется для получения данных, подлежащих дальнейшей ручной обработке – в таких случаях важно получить максимальное число упоминаний объекта, пусть и достаточно сильно разбавленных шумом.

В связи с объективным наличием приоритета полноты и точности интегральным показателем качества может служить мера Ван-Ризбергена $F = (\frac{\alpha}{P} + \frac{1-\alpha}{Q})^{-1}$ [12]. В ряде случаев удается создать алгоритмы, параметризованные подобным приоритетом α (напр., см. [9] – три различных режима).

Литература

1. Окатьев В.В., Гергель В.П., Алексеев В.Е., Таланов В.А., Баркалов К.А., Скатов Д.С., Ерехинская Т.Н., Котов А.Е., Титова А.С. Отчет о выполнении НИОКР по теме: «Разработка пилотной версии системы синтаксического анализа русского языка» (инвентарный номер ВНТИЦ 02200803750) // М.: ВНИТЦ, 2008.
2. Окатьев В.В., Гергель В.П., Скатов Д.С., Ерехинская Т.Н., Титова А.С., Ливерко С.В., Назаренко Д.А., Вдовина Н.А., Ратанова Т.Е., Стребков Д.Ю., Таланин Е.А., Щеглов А.В. Отчет о выполнении НИОКР по теме: «Разработка пилотной версии системы извлечения цитат из текстов на русском языке» (инвентарный номер ВНТИЦ 01200953184) // М.: ВНИТЦ, 2010.
3. Скатов Д.С., Окатьев В.В., Ерехинская Т.Н.. Анализ иерархически структурированных текстов в системе извлечения цитат Dictum // Технологии Microsoft в теории и практике программирования: Материалы конференции. / Под ред. проф. В.П. Гергеля – Н. Новгород: Изд-во Нижегородского госуниверситета, 2009. С. 420-425.
4. Окатьев В.В., Алексеев В.Е., Ерехинская Т.Н., Скатов Д.С. Синтаксический анализ естественного языка и библиотека синтаксического анализа DictaScope // Технологии Microsoft в теории и практике программирования: Материалы конференции. / Под ред. В.П. Гергеля – Н. Новгород: Изд-во Нижегородского госуниверситета, 2009. С. 319-325.

5. Скатов Д.С., Ерехинская Т.Н., Окатьев В.В. Модели и методы анализа иерархически структурированных текстов // Труды Международной конференции «Диалог'2009». – М.: Наука, 2009.
6. Окатьев В.В., Ерехинская Т.Н., Скатов Д.С. Модели и методы учета пунктуации при синтаксическом анализе предложения русского языка // Труды Международной конференции «Диалог'2009». – М.: Наука, 2009.
7. Скатов Д.С., Ливерко С.В., Вдовина Н.А., Окатьев В.В. Язык описания правил в системе лексического анализа ЕЯ текстов DICTASCOPE TOKENIZER // Труды Международной конференции «Диалог'2010». – М.: Наука, 2010.
8. Окатьев В.В., Ерехинская Т.Н., Ратанова Т.Е. Тайные знаки пунктуации // Труды Международной конференции «Диалог'2010». – М.: Наука, 2010.
9. Skatov D., Liverko S. Anaphora resolution of the third-person pronoun in texts from narrow subject domains with grammatical errors and mistypings // Труды Международной конференции «Диалог'2011». – М.: Наука, 2011.
10. Ерехинская Т.Н., Титова А.С., Окатьев В.В. Синтаксический анализ текста с орфографическими ошибками в системе Dictascope Syntax // Труды Международной конференции «Диалог'2011». – М.: Наука, 2011.
11. Горностай Т., Васильев А., Скадиньш Р., Скадиня И. Опыт латышско-русского машинного перевода // Труды Международной конференции «Диалог'2007». – М.: Наука, 2007.
12. Baeza-Yates R., Ribeiro-Neto B. Modern Information Retrieval – Addison-Wesley, 1999. – ISBN 0-201-39829-X.
13. http://news.cnet.com/8301-1001_3-10209580-92.html.
14. http://www.gosbook.ru/node/11386.