А. А. Аброскин

Поиск по корпусу: проблемы и методы их решения

а годы активного функционирования НКРЯ достаточно много было написано об этапах развития корпуса, о способах представления в нем текстов и о возможностях его использования. В то же время до сих пор в меньшей степени затрагивалась проблематика, связанная с ***поиском*** по корпусу. В частности, в сборнике [НКРЯ 2005] особенности поиска в НКРЯ в большей или меньшей степени затрагивались практически во всех статьях, но сборник не включает ни одной статьи, специально посвященной поисковым проблемам в корпусе. Несколько подробнее поисковые возможности корпуса описаны в [Инструкции 2007], однако инструкция как жанр ориентирована на интересы пользователя, в связи с чем структурные особенности поисковой машины НКРЯ в ней практически не отражены.

В этой статье я постараюсь рассказать о базовых принципах работы поиска по корпусу, проблемах, возникавших при его создании, и методах, использовавшихся при их решении.

Архитектура поиска

В настоящий момент поиск в корпусе реализован при помощи свободно распространяемой поисковой системы Яндекс.Сервер. Из размеченных текстов индексатор сервера строит инвертированный индекс, в котором каждому слову соответствуют все его характеристики. Текст предложений, уже без разметки, индексатор размещает отдельно, так что после осуществления поиска его можно получить и показать пользователю. При поиске слов или предложений по их характеристикам (морфологическим, семантическим и прочим) сервер открывает соответствующие запрошенным характеристикам индексы, после чего, в результате прохода по этим индексам, находит все нужные словопозиции.

Для упрощения работы пользователя создан специальная форма, в которой пользователь может в достаточно понятном виде задать запрос. Сформированный в этой форме запрос отправляется на поисковый сервер, где передаётся модулю формирования выдачи—специальной программе на языке C++, которая преобразует пользовательский запрос в формат запроса к поисковой системе, а затем по результатам поиска формирует выдачу.

Модуль формирования выдачи представляет результат в xml-формате, на который затем накладывается xslt-преобразование, в результате чего пользователь получает искомый материал в формате html. Такой подход позволяет отделить поиск от оформления результатов, тем самым упрощая построение, поддержание и модификацию системы.

Скорость поиска и проблема перемешивания характеристик слов

Представленная выше схема, реализованная без особых модификаций, работала вполне удовлетворительно, пока корпус был достаточно мал и включал в себя только тексты со снятой омонимией. Но со временем объём корпуса значительно увеличился, в том числе и за счёт большой коллекции автоматически размеченных текстов. Это, безусловно, привело к падению производительности поиска, но этим затруднения, возникшие в работе корпуса, не ограничились. Основная проблема заключалась в том, что разные грамматические разборы, приписанные одному и тому же слову вследствие морфологической омонимии, стали смешиваться. Так,

например, слово *берет*, разбираемое и как неодушевлённое существительное мужского рода, и как изъявительное наклонение глагола, можно было получить по запросу ‘глагол мужского рода’. Но если примеры такого рода вызывают лишь усмешку, то перемешивание более разумных характеристик (например, падежа и числа слова *книжки*—им,мн род,ед вин,мн) при поиске по корпусу с неснятой омонимией могло принести пользователю немало неприятных сюрпризов.

Как уже говорилось выше, при индексации строится инвертированный индекс для каждого поискового атрибута. Поэтому если мы, скажем, ищем существительное в винительном падеже, поиск должен открыть индекс для существительного и индекс для винительного падежа и затем построить их пересечение. При этом чем больше размер корпуса, тем тяжелее становится подобная операция. К сожалению, если мы хотим не только найти несколько примеров, но и посчитать общее количество найденных по запросу слов, мы не можем отказаться от полного поиска.

В то же время, если мы заведём отдельный индекс для существительных в винительном падеже, поиск таких слов получится настолько быстрым, насколько это возможно на выбранной архитектуре. Таким образом, положив в индекс все возможные сочетания характеристик для каждого слова, мы сможем не только решить проблему производительности, но и избавиться от перемешивания характеристик, пускай и за счёт многократного увеличения индекса. Так как поисковые запросы обычно достаточно детализированы (существительное в некотором падеже спрашивают чаще, чем просто существительное), это решение более чем на порядок повысило скорость поиска.

Компактное представление грамматической информации

Для удобства пользователя в корпусе предусмотрена возможность просмотра характеристик словоформ при выдаче материала. Однако первоначально механизм поиска в корпусе обладал двумя довольно существенными ограничениями: требовалось, чтобы каждое слово было представлено в тексте одной последовательностью литеральных символов и при этом чтобы каждое слово имело длину не более 50-ти символов. Первое ограничение не позволяло просто

записать разборы в скобках прямо за словом, а вследствие второго ограничения было практически нереально зашифровать все необходимые характеристики словоформы, так как слова с несколькими разборами, не говоря уже о семантической разметке, невозможно уместить в 50 символов.

Первоначально было принято решение для снятия этой проблемы использовать внешний словарь разборов, а в передаваемом на индексацию тексте оставлять только номера разборов, зашифрованные в виде букв. Однако, как и следовало ожидать, такой подход оказался не вполне жизнеспособным из-за слишком большого размера соответствующего словаря. Тем не менее размер словаря удалось значительно уменьшить, вынеся из него информацию о словоформе.

Это было осуществлено следующим образом. Для каждого разбора словоформы берётся тройка: длина общего префикса формы и леммы, часть леммы после общего префикса и некоторым образом нормализованное представление набора грамматических характеристик. Затем эти тройки определенным способом сортируются и склеиваются в единый ключ, который и помещается в словарь. Например, для слова мамы (мама сущ,жен,од,им,мн | сущ,жен,од,род,ед) ключ может быть следующим:

(3, а, ((сущ,жен,од,им,мн), (сущ,жен,од,род,ед)))

Полученный таким образом словарь для корпуса объемом 140 миллионов слов имеет 720 тысяч вхождений—против 5 миллионов, содержащихся в словаре всех разборов этого корпуса. В тексте при этом в зашифрованном виде хранится сама словоформа и номер соответствующей записи в словаре.

Надо заметить, что при используемой технологии, вообще говоря, способ нормализации разборов, метод сортировки и склейки ключей могут быть произвольными, так как эти процедуры используются только для того, чтобы по различным образом записанным эквивалентным разборам получить один и тот же ключ.

Расширение контекста

Одним из базовых ограничений архитектуры Яндекс-Сервера является то, что предложение в индексе не может быть длиннее 64 слов.

Поэтому все предложения большей длины при индексации разрезаются на несколько частей. Но что делать, если мы хотим при выдаче увидеть всё найденное предложение целиком и—более того—просмотреть его «окрестности»?

В случае, если мы можем управлять способом получения предложений из архива, решение этой проблемы не составляет сложности. Однако для этого мы должны иметь доступ к внутренним процедурам поискового сервера, а эта операция может оказаться слишком сложной или же вообще невозможной (например, если у вас есть уже готовая поисковая программа, но нет доступа к её исходному коду).

В то же время, если для каждого предложения, которое мы хотим расширить, направлять новый запрос серверу, это позволит в конечном итоге получить то, что нам требуется.

Для того, чтобы это стало возможным, следует каждому предложению присвоить его порядковый номер в документе. Документ при этом необходимо пометить его номером в коллекции. Таким образом, при поиске по корпусу каждому документу приписывается документный поисковый атрибут, соответствующий его номеру в коллекции, причём этот атрибут можно получить из архива стандартными средствами взаимодействия модуля формирования выдачи и поискового сервера. Для предложения, к сожалению, этот механизм уже не работает, так как ввиду ограничений используемой технологии мы не можем получить поисковые атрибуты предложения. Поэтому приходится первым словам частей предложения приписывать специальные пометки, в которых зашифрован номер предложения, и флаг, указывающий, является ли данное предложение завершённым или представляет собой лишь часть более крупного предложения.

Таким образом, при формировании выдачи, кроме самого текста предложения, мы получаем 1) его номер, 2) номер включающего его документа, 3) информацию о том, завершено ли это предложение или является лишь частью более длинного «составного» предложения. Для незавершённых предложений делается перезапрос, выдающий их полную версию. Аналогичным образом, если пользователю нужно, кроме самого предложения, получить ещё и его окрестности, следует—зная номер предложения (n)—запросить предложения

с номерами из интервала (n–k, n+k). Такой метод используется для расширения по запросу пользователя и для расширения слишком маленьких предложений, например, фраз вроде «ага», «ну да» и тому подобных в корпусе устной речи.

Заключение

Зачастую при реализации поиска по корпусам в качестве поискового механизма используются системы, построенные на основе реляционных систем управления базами данных, или же самостоятельно разработанные программы поиска. Первые, в случае больших корпусов, как правило, работают довольно медленно, создание же специальных поисковых программ требует чрезмерных усилий. В этой статье я постарался показать, как при помощи стандартных средств текстового поиска свободно распространяемой поисковой системы можно построить довольно эффективный поиск, рассказать о возникающих при этом проблемах и возможных способах их решения.

Литература

Инструкция 2007—Инструкция по пользованию Национальным корпусом русского языка *http://www.ruscorpora.ru/instruction-main.pdf*

НКРЯ 2005—Национальный корпус русского языка: 2003–2005. Результаты и перспективы.—М.: Индрик, 2005.

Яндекс.Сервер—*http://company.yandex.ru/technology/server*