Масштабируемая среда для многоуровневого анализа текстов

© А. Носков

Московский Государственный Университет им. М.В. Ломоносова
alexey.noskov@gmail.com

Аннотация

Характеризуется инструментальная среда, предназначенная для сборки из стандартных программных компонентов приложений для автоматической обработки текстовой информации, в первую очередь - обработки текстов на естественном языке. Описаны требования к среде, ее функциональность, используемая модель данных, а также модель сборки и модель взаимодействия задач при выполнении приложения.

1. Введение

В настоящее время в связи с быстрым ростом количества текстовой информации, в том числе в сети Интернет, возникает необходимость в быстрой разработке прикладных программных систем (приложений) для автоматической или автоматизированной обработки текстов на естественном языке (ЕЯ-текстов). Примерами такой обработки является сбор и фильтрация данных из различных источников, извлечение знаний [11], реферирование, аннотирование и т.п.

Разработка приложений для решения такого рода задач имеет ряд сложностей, в первую очередь - необходимость интеграции большого числа программных компонентов, реализующих алгоритмы обработки ЕЯ-текста, работающие на различных его уровнях (обработка слов, предложений, абзацев и т.п.).

Например, задача выделения сущностей и отношений [10] из ЕЯ-текстов требует графематического анализа (разбиения текста на слова и предложения), морфологического анализа, выделения определенных конструкций естественного языка на синтаксическом уровне, последующей семантической обработки результатов, а также преобразования текстовых данных между различными форматами и представлениями.

Другую сложность составляет организация внутреннего представления обрабатываемого ЕЯ-текста, поскольку оно должно поддерживать две его важные особенности: естественную иерархию составляющих текст элементов (абзацев, предложений, слов) и свойственную естественному языку множественность интерпретаций на каждом уровне (например, словоформа “мыла” на морфологическом уровне имеет 2 интерпретации: существительное или как глагол).

Налицо необходимость в инструментах, позволяющих упростить создание приложений по обработке текстов на естественном языке (ЕЯ-приложений) за счет повторного использования тех или иных программных компонентов и сборки приложения из них - это неоднократно обсуждается в различных работах, в том числе [7]). Эти же инструменты должны предоставлять гибкую модель данных, позволяющую учесть указанные свойства ЕЯ-текстов.

При ближайшем рассмотрении можно заметить, что могут быть повторно использованы:

- Данные, необходимые для работы прикладной системы - обучающие выборки, корпуса текстов, словари, онтологии;
- Концептуальные элементы системы - алгоритмы обработки текстовых данных, форматы их представления, а также шаблоны проектирования систем;
- Программный код, реализующий алгоритмы обработки данных и преобразование данных между различными форматами, а также связующий код, обеспечивающий взаимодействие компонентов и обмен данными между ними.

Программные средства могут поддерживать повторное использование программных компонентов обработки текстов различными путями, в соответствии с чем их можно разбить на две основных группы:

- Программные библиотеки, реализующие те или иные алгоритмы обработки данных. Они позволяют повторно использовать код программных компонентов, но часто ограничивают разработчика приложения в выборе модели представления данных.
- Программные среды, предназначенные для разработки некоторого класса приложений. В

Труды 13[й] Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» - RCDL’2011, Воронеж, Россия, 2011.

их случае возможно повторное использование как программных компонентов и моделей данных, так и кода, обеспечивающего взаимодействие различных компонентов. Однако использование программных сред накладывает ограничения на архитектуру приложения и модель данных, а также иногда на класс используемых алгоритмов.

Стоит отметить, что большинство существующих разработок относятся к классу программных библиотек, например, OpenNLP, LingPipe, Wraetlic [1]. Еще одним примером программной библиотеки является комплекс программных средств, разработанных в поддержку языка Lspl [13] и позволяющих выделять синтаксические конструкции из текстов на естественном языке по их формальному описанию. К другой группе относятся такие среды GATE, LT-NSL, Catalyst, Ellogon и т.п.

При сравнении библиотек и программных сред, можно заметить, что в последних больше объем повторно используемого кода, но более узкие рамки построения конкретных приложений. Целесообразно создание среды, сбалансированной с точки зрения возможностей повторного использования кода и ограничений, накладываемых на создаваемые приложения.

В данной работе предлагается архитектура инструментальной среды, позволяющей упростить разработку приложений автоматической обработки текстовой информации, в первую очередь - многоуровневой обработки текстов на естественном языке, за счет построения приложений из набора стандартных компонентов. Для этого были проанализированы архитектурные особенности известных инструментальных сред для построения ЕЯ-приложений. Предложенная архитектура охватывает модель представления данных, модель сборки приложения из компонентов и модель взаимодействия задач по обработке данных в процессе выполнения приложения.

2. Инструментальные среды построения приложений обработки ЕЯ-текстов

Инструментальные программные среды можно разделить на две группы, в которых используются различные модели представления данных - на основе разметки и на основе аннотаций [5]. В системах первой группы лингвистическая информация хранится непосредственно в обрабатываемом тексте в форме дополнительной разметки, например, на языке SGML или XML. В системах второй группы информация хранится отдельно от текста в виде аннотаций и содержит ссылки на аннотируемые участки обрабатываемого текста.

2.1 LT-NSL

Подход, основанный на использовании разметки, хорошо иллюстрируется на примере системы LT-NSL[8], которая использует язык SGML для представления обрабатываемой лингвистической информации и предназначена для построения приложений по обработке корпусов текстов.

В системе LT-NSL программные компоненты выполняются в отдельных процессах, а для интеграции компонентов используются каналы UNIX (pipes), которые соединяют стандартный ввод и вывод процессов. При этом текст вместе с разметкой рассматривается как поток данных, который не хранится в памяти полностью, что важно при обработке больших корпусов текстов. Каждый компонент работает как фильтр, т.е. принимает на вход поток данных и выдает модифицированный поток на выходе.

Однако использование разметки в тексте для представления лингвистической информации приводит к проблеме, связанной с представлением перекрывающихся элементов разметки (например, из-за необходимости представления различных морфологических интерпретаций слова). В LT-NSL (и в большинстве систем этой группы) эта проблема решается путем использования ссылок на разметку, находящуюся вне текста. Такой подход значительно усложняет разметку, что приводит к снижению эффективности ее использования.

2.2 GATE

Наиболее известной средой разработки ЕЯ-приложений является система GATE [5] и ее более поздняя версия GATE2 [3], основанные на использовании аннотаций и разработанные для создания приложений автоматического извлечения информации из текстов. Приложение в GATE строится на базе компонентов трех типов:

- Языковых, которые предоставляют доступ к лингвистическим ресурсам, таким как документы, корпуса и онтологии;
- Обрабатывающих компонентов, которые осуществляют различные задачи по анализу ЕЯ-текстов, такие как выделение лексем, морфологический анализ и т.п;
- Компонентов визуализации и редактирования обрабатываемой информации, а также редактирования процесса выполнения приложения.

Важной особенностью GATE является наличие языка правил Jape [4], позволяющего описывать правила обработки аннотаций в удобной форме.

Несмотря на свою популярность система GATE имеет ряд недостатков с точки зрения построения сложных приложений. Во-первых, GATE оперирует фиксированным представлением данных в форме аннотаций, что не позволяет представлять сложные структуры данных (в частности, деревья синтаксического разбора). Во-вторых, обработка данных в GATE ограничена линейной последовательностью запуска программных компонентов и не позволяет реализовать более сложные сценарии обработки (например, запускать

несколько компонентов одновременно или альтернативно).

2.3 Catalyst

Другой средой на базе аннотаций является система Catalyst [2], выделенная из вопросно-ответной системы Qanda и предназначенная для решения задач обработки текстов на естественном языке и извлечения информации.

Для соединения обрабатывающих компонентов в Catalyst используется модель на основе каналов данных, по которым передаются аннотации, упорядоченные в соответствии с их позициями в документе. Каждый компонент объявляет, какие типы аннотаций необходимы ему для обработки и какие аннотации он генерирует на выходе. Такая информация позволяет не передавать между компонентами аннотации, которые не будут ими использованы и тем самым значительно уменьшить объем информации, передаваемой по каналам, что важно при построении хорошо масштабируемых распределенных системы. Кроме того, указанная информация может быть использована для проверки корректности связей компонентов во время сборки приложения.

Такая модель взаимодействия компонентов имеет ряд преимуществ. Во-первых, многие ошибки, связанные с неправильной организацией приложения, могут быть выявлены на этапе сборки. Во-вторых, приложение может работать как на одной машине, так и на нескольких, т.е. распределенно. В-третьих, код компонентов упрощается, поскольку в них не требуется проверка данных на корректность (она выполняется на этапе сборки приложения).

Как недостаток систем GATE и Catalyst можно выделить отсутствие средств представления информации на различных уровнях текста, в них нет возможности отразить отношения иерархии между словами, предложениями и документом - все эти сущности представляются аннотациями на одном уровне. Кроме того, использование модели данных на основе аннотаций не позволяет представлять сложные структуры данных.

3. Описание среды

3.1 Состав и функциональность среды

Создаваемая иструментальная среда предназначена для разработки приложений автоматической обработки ЕЯ-текстов с учетом их характерных особенностей (многоуровневость и многоинтепретативность).

В состав среды входит библиотека стандартных компонентов, которые используются при разработке приложений. Она включает программные компоненты трех основных типов:

- Ввода данных - загрузки лингвистических ресурсов (онтологий, словарей, статистических моделей) из сети Интернет, файлов, или данных от пользователя;
- Обработки данных - преобразования и анализа текстов (графематического, морфологического и синтаксического и др.);
- Вывода данных - визуализации данных, сохранения в файл, публикации в сети Интернет.

Поскольку среда предназначена для анализа ЕЯ-текстов, то среда включает компоненты, выполняющие стандартные задачи, как модули такие морфологического анализа на базе библиотеки Snowball и анализатора MyStem и модули для графематического анализа, выделения сущностей и длугих задач на основе библиотек OpenNLP и LingPipe. Также в состав среды входит библиотека поддержки языка Lspl [13], позволяющего декларативно описывать выделяемые конструкции естественного языка.

Для разработки компонентов, их тестирования, а также сборки, отладки и развертывания приложений среда включает соответственные программные инструменты. Взаимодействие компонентов, составляющих приложения обеспечивается ядром среды.

Среди технологических требований к создаваемой среде ключевым является масштабируемость, которая обеспечивает возможность создания приложений, изменяющихся по функциональности и параметрам (например, по объему обрабатываемых данных, используемой аппаратной платформе и т.п.). При этом под масштабируемостью мы понимаем следующие виды:

- Вертикальную масштабируемость относительно вычислительных ресурсов - при увеличении производительности каждого из используемых вычислительных ресурсов (процессоров, серверов и т.п.) растет производительность приложения;
- Горизонтальную масштабируемость относительно вычислительных ресурсов - при увеличении количества используемых вычислительных ресурсов (например, количества используемых компьютеров) растет производительность приложения;
- Масштабируемость класса создаваемых приложений и решаемых задач - платформа позволяет разрабатывать приложения от небольших экспериментальных систем и встраиваемых библиотек до крупномасштабных систем;
- Масштабируемость процесса разработки и погружения в среду - разработка простейших приложений должна требовать минимального погружения в среду, чем глубже уровень погружения, тем более сложные приложения могут быть построены

Дополнительно, существенными являются требования возможности обработки больших объемов информации и минимальных ограничений на используемые алгоритмы и модели обработки (в

частности, для выполнения синтаксического анализа могут быть использованы как программные компоненты на основе грамматик составляющих или зависимостей, так и на основе статистических моделей, причем в рамках одного приложения). Это свойство обеспечивается за счет модели данных, которая накладывает минимальные ограничения на представляемые структуры данных и не требует хранения всех данных в памяти.

Горизонтальная масштабируемость обеспечивается предложенной моделью взаимодействия задач во время выполнения, а также возможностями запуска приложения как в одном процессе операционной системы, так и в нескольких, а также нескольких приложений в рамках одного процесса операционной системы.

Описание среды включает следующие основные модели: модель представления данных, которая поддерживает многоуровневость и многоинтерпретативность обрабатываемых данных, модель соединения компонентов во время разработки и модель их взаимодействия по время выполнения приложения.

3.2 Представление обрабатываемых данных

Для обработки больших объемов информации в приложениях модель данных должна предоставлять возможность не хранить ее всю в памяти, подобно тому, как это возможно в системах LT-NSL и Catalyst. В нашей модели предлагается передавать данные между обрабатывающими компонентами поэлементно, обрабатывая по мере получения. Такой подход в частности позволяет осуществлять выделение синтаксических конструкций в длинном тексте без его полной загрузки в память, по мере выполнения морфологического анализа слов текста.

Для отражения множественности интерпретаций последовательности элементов обрабатываемого текста предлагаемая модель использует представления различных интерпретаций в рамках единой структуры данных, называемой графом интерпретаций [12] - ациклического ориентированного мультиграфа, где параллельные варианты интерпретаций представляются параллельными ребрами. Такой граф позволяет представить в компактной форме большое число различных вариантов интерпретаций.

Модель данных предоставляет средства для представления иерархии ЕЯ-текста. Каждый элемент данных может иметь последовательность составляющих его элементов (например, корпус - тексты, текст - предложения, предложения - слова), которая, в свою очередь, может иметь несколько различных вариантов интерпретаций и подпоследовательностей составляющих элементов.

Таким образом, в общем случае обрабатываемые данные организованы в многоуровневую иерархию, где для каждого узла иерархии дочерние узлы организованы в граф интерпретаций, задающий порядок их передачи.

3.3 Спецификация сборки приложения

На этапе разработки приложения пользователь среды должен задать информацию об обрабатываемых данных, компонентах и связях между ними. Информация о данных и компонентах задается в виде спецификаций, отражающих их свойства и связи.

Между данными задаются связи, отражающие иерархию элементов и их составных частей. Например, данные “предложения” связаны с данными “тексты”, поскольку каждое предложение входит в подпоследовательность, принадлежащую некоторому тексту.

Компоненты соединяются с обрабатываемыми данными через входы и выходы, при этом один вход или выход может соединяться с различными данными, входящими в одну иерархию. Такое соединение позволяет видеть иерархию данных, передаваемых между компонентами в процессе последующей работы.

Например, на Рис. 1 компонент Tokenizer принимает на вход некоторую последовательность текстов, описываемую данными, и выдает иерархию данных “текст-слово” (b1 и b2). Компонент V ectorBuilder преобразует последовательность слов в вектор частотности слов в тексте. Компонент Clusterizer осуществляет кластеризацию текстов на основе характеризующих их векторов. Он принимает на вход иерархию “текст-вектор” (c1 и c2), а выдает иерархию “кластер-текст” (d1 и d2).

Рис. 1: Сборка приложения

Для каждого входа и выхода компонента имеется информация о спецификациях принимаемых и выдаваемых данных, что позволяет проверить корректность связей. В частности, возможна диагностика таких ошибок, как передача данных, которые не могут быть обработаны принимающим компонентом, или же недополучение компонентом данных, необходимых для его работы.

Проектирование приложения осуществляется или в графической форме в интегрированной пользовательской среде или путем спецификации компонентов и их связей на специальном языке. Ниже приведен пример спецификации на этом языке, состоящий из объявления используемых данных, компонентов для обработки текста и их связей.

```
def a = ds( "Sentences" )

def b1 = ds( "Sentences with words" )
def b2 = ds( b1, "Words" )

def c1 = ds( "Sentences with vectors" )
def c2 = ds( c1, "Vectors" )

def d1 = ds( "Clusters" )
def d2 = ds( d1, "Sentences in clusters" )

component( TokenizerTask.class ) {
    input << a // Связь входа с данными
    output >> b2 // Связь выхода с данными
}

component( VectorBuilderTask.class ) {
    input << b2
    output >> c2
}

component( ClusterizerTask.class ) {
    input << c2
    output >> d2
}
```

Конструкция ds используется для определения данных, при этом ее аргументами служат ссылка на спецификацию данных вышестоящего уровня (если они есть) и описание специфицируемых данных. Конструкция component позволяет специфицировать компонент, его свойства, и указать данные для его входов и выходов.

Предложенное представление на основе соединений входов и выходов компонентов через данные позволяет лучше (по сравнению с непосредственным их соединением) отобразить характер обрабатываемых данных и связей между ними, что особенно важно при многоуровневой обработке.

3.4 Взаимодействие задач во время выполнения приложения

Во время выполнения приложения для каждой конкретной последовательности элементов, которая должна быть обработана компонентом, выделяется отдельная задача, осуществляющая обработку и изолированная от остальных задач (то есть не пересекающаяся с ними по данным). Такое решение позволяет обрабатывать не зависящие друг от друга последовательности данных (например, слов различных предложений) одновременно. Кроме того, среда предусматривает возможность выполнения задач в других процессах или на других машинах, что обеспечивает горизонтальную масштабируемость создаваемого приложения.

Каждая задача, так же как и компонент, имеет набор входов, на которые поступают конкретные последовательности данных (возможно с последовательностями составных частей) и набор выходов, на которых выдаются новые последовательности данных.

Поскольку модель данных предусматривает множественность интерпретаций, во время выполнения в процессе обработки задачи могут происходить разветвления. В таких случаях на каждую интерпретацию в момент завершения общего участка интерпретаций создается копия задачи, обрабатываемая независимо от остальных. Это позволяет выполнять обработку различных интепретаций одновременно.

Задачи могут иметь внутреннее состояние процесса обработки, при этом в случае появления общего участка различных интепретаций задачи с одинаковым состоянием могут быть вновь объединены. Кроме того, состояние является сериализуемым, что позволяет сохранять задачи на диск или же переносить их на другие компьютеры. Это помогает обеспечить вертикальную и горизонтальную масштабируемость (например, при наличии малого объема памяти часть задач может быть временно сохранена на диск, а при подключении нового сервера часть задач может быть перенесена на него).

4. Заключение

В работе описана создаваемая инструментальная среда, позволяющая осуществлять построение приложений по обработке текстов на естественном языке из различных программных компонентов. Основу обработки в среде представляют собственные модель представления данных, модель сборки и модель взаимодействия задач.

Предложенная модель данных позволяет представлять информацию на различных уровнях текста и учитывает множественность интепретаций элементов ЕЯ-текста. Модель соединения компонентов позволяет строить сложные приложения за счет связей спецификаций компонентов и данных и проверять на этапе сборки корректность построения приложения. Модель взаимодействия задач позволяет эффективно обрабатывать данные в системе и допускает естественную горизонтальную масштабируемость приложения за счет переноса задач на другие компьютеры. По сравнению с GATE и Catalyst среда не ограничена использованием аннотаций и позволяет использовать любые данные, представляемые в виде последовательности элементов.

На текущий момент реализовано ядро системы и значительная часть базовых компонентов, происходит наполнение библиотеки компонентов для анализа текстов на естественном языке. При реализации использовалась технология Java, большая часть среды написана непосредственно на языке Java, в то время как часть модулей на языках Groovy[6] и Scala[9].

Текущими направлениями предполагаемой разработки являются: наполнение библиотеки

компонентов для анализа текстов на естественном языке, создание интегрированной пользовательской среды разработки для приложений и разработка средств интеграции с утилитами сборки и упаковки приложений.

Литература

[1] Enrique Alfonseca, Antonio Moreno-s, JosДе MarДа Guirao, и Maria Ruiz-casado. The wraetlic NLP suite. 2006.

[2] Pranav Anand, David Anderson, John Burger, John Griffith, Marc Light, Scott Mardis, и Alex Morgan. Qanda and the Catalyst Architecture. 2002.

[3] Kalina Bontcheva, Diana Maynard, Valentin Tablan, и Hamish Cunningham. GATE: A Unicode-based infrastructure supporting multilingual information extraction. In Proceedings Of Workshop On Information Extraction For Slavonic And Other Central And Eastern European Languages (Iesl'03), Borovets, 2003.

[4] Hamish Cunningham, Hamish Cunningham, Diana Maynard, Diana Maynard, Valentin Tablan, и Valentin Tablan. JAPE: a Java Annotation Patterns Engine. 1999.

[5] Hamish Cunningham, Kevin Humphreys, Robert Gaizauskas, и Yorick Wilks. Software Infrastructure for Natural Language Processing. 1997.

[6] G. Developers. Groovy an agile dynamic language for the Java platform. технический отчет, Technical report, http://groovy. codehaus. org/, accessed 2008, 2003.

[7] Jochen L Leidner. Current Issues in Software Engineering for Natural Language Processing. Proc. Of The Workshop On Software Engineering And Architecture Of Language Technology Systems (Sealts), The Joint Conf. For Human Language Technology And The Annual Meeting Of The Noth American Chapter Of The Association For Computational Linguistics (Hlt, 8:45—50, 2003.

[8] David McKelvie, Chris Brew, и Henry Thompson. Using SGML as a Basis for Data-Intensive NLP. In Proceedings Of The Fifth Conference On Applied Natural Language Processing (ANLP-97, 1997.

[9] Martin Odersky, StДephane Micheloud, Nikolay Mihaylov, Michel Schinz, Erik Stenman, Matthias Zenger, и et al.. An overview of the Scala programming language. 2004.

[10] Stuart C Shapiro и Shane Axtell. Natural Language Tools for Information Extraction for Soft Target Exploitation and Fusion. 2007.

[11] Хорошевский В.Ф.. Управление знаниями и обработка ЕЯ-текстов. В Девятая Национальная конференция по искусственному интеллекту с международным участием КИИ-2004: Труды конференции. В 3-х т., т. 2, страницы 565–572. М.: Физматлит, 2004.

[12] Большакова Е.И. и Носков А.А.. Анализ текста на основе лексико-синтаксических шаблонов с сокращением многовариантности. В Новые информационные технологии в автоматизированных системах: материалы тринадцатого научно-практического семинара., страница 309. М.: Моск. гос. ин-т. электроники и математики., 2010.

[13] Большакова Е.И., Баева Н.В., Бордаченкова Е.А., Васильева Н.Э., и Морозов С.С.. Лексико-синтаксические шаблоны в задачах автоматической обработки текстов. В Компьютерная лингвистика и интеллектуальные технологии: Труды Международной конференции Диалог'2007, страницы 70–75. М.: Издательский центр РГГУ, 2007.

Scalable Environment for Multi-level Analysis of Texts

© Alexey Noskov

Paper describes instrumental environment for assembling applications from standard program components targeting automatic text information processing, particularly natural language processing.

Requirements for environment, functionality, data model and interaction models are described.