ОПЫТ ЭКСПЕРИМЕНТАЛЬНОЙ РЕАЛИЗАЦИИ АЛГОРИТМОВ ПОВЕРХНОСТНО-СИНТАКСИЧЕСКОГО АНАЛИЗА[1]

THE EXPERIENCE OF A SAMPLE IMPLEMENTATION OF SURFACE-SYNTACTIC ANALYSIS ALGORITHMS

А.М. Баталина *(batalina_anna@rambler.ru)*,

М.Е. Епифанов *(xeme@rambler.ru)*,

Т.Ю. Кобзарева *(stamstam@mtu-net.ru)*,

Е.В. Кушнарёва *(likart@mail.ru)*,

Д.Г. Лахути *(delir1@yandex.ru)*

Российский государственный гуманитарный университет, Москва

Предметом доклада является применение инструментальной среды для экспериментов с алгоритмами поверхностно-синтаксического анализа (ПСА). Описывается опыт отладки и реализации некоторой совокупности алгоритмов ПСА в рамках данной среды в сжатые сроки.

Введение

В настоящей работе описывается опыт применения объектной среды для экспериментов с алгоритмами поверхностно-синтаксического анализа (ПСА) [1] к фрагменту системы ПСА русского предложения (S) [2, 3] [2]. И данная объектная среда, и указанная система ПСА разрабатываются в Институте лингвистики РГГУ.

Задачей системы ПСА, служащей объектом эксперимента, является построение полного графа синтагматических связей сегментов и слов внутри сегментов для русского S, где сегменты (в терминологии [4] – «окончательные (приведенные) сегменты») – эксплицитно заданные в тексте части S: простые главные предложения, придаточные и любые обособляемые обороты: деепричастные обороты, согласованные определения, вводные обороты и т.д.

Специфика объекта эксперимента – данной системы ПСА – в сравнении с обычно принятой (вслед за предложенной в [4]) структурой ПСА – состоит в делении процедур ПСА на модули, работающие в кардинально отличном от обычного порядке:

- предсинтаксис:
  - постморфология (морфосинтаксис),
  - снятие омонимии частей речи,
- предсегментация – построение необходимых для сегментации проективных отрезков атрибутивных именных, предложных групп, сложного сказуемого и т.д.,
- сегментация – построение сегментов,
- внутрисегментный анализ (ВА) – построение связей слов в сегментах,
- межсегментный анализ – построение связей сегментов.

Главные отличия состоят в том, что сегментация предшествует построению графа связей [3] и, благодаря использованию проективности и рекурсивности линейной структуры S в процедурах анализа, строятся сразу только синтаксически правильные связи с возможными неоднозначностями каждого уровня анализа.

На данном этапе эксперимента на входе – сегмент, равный простому S, на выходе – полный граф внутрисегментных связей. Модуль сегментации в настоящей версии из эксперимента исключен.

Собственно ПСА работает с выходом морфологического анализа – цепочкой грамматических образов слов и знаков препинания, содержащих информацию о морфологических характеристиках, грамматической – не содержащей лексико-семантической информации – модели управления (способность управлять падежами, инфинитивом, предлогом и подчинительным союзом) и о семантическом классе (предмет, одушевленность, имя собственное, параметр и т.д.) слова.

[1] Доклад подготовлен при частичной поддержке РФФИ (грант 06-06-80434).

[2] Синтаксический базис и собственно алгоритмы - Т.Ю.Кобзаревой.

Граф строится соответственно списку синтагм, где синтагма – вид связи без условий ее реализации. Различаем два типа отношений: связь направленная – подчинение (стрелка от хозяина к слуге) и ненаправленная – сочинение (условно – стрелка от левого сочиненного к ближайшему правому). Условия, определяющие специфику реализации синтагм, задаются в алгоритмах.

Под словом «алгоритм» понимается совокупность связанных между собой лингвистических правил ПСА. Правила содержат проверяемые на членах предложения условия и декларации распознавания соответствующих этим условиям лингвистических ситуаций. Правила «жестко» соединены между собой переходами, причем от одного правила можно перейти к рассмотрению не более чем двух других. При этом выбор следующего правила зависит от результата анализа предложения в текущем правиле.

Такая организация алгоритмов ПСА похожим на блок-схему образом удобна для их программной реализации. Однако описанный способ записи алгоритмов в виде диаграмм, сохраняемых в документах Word, затрудняет проверку их правильности и обозрение всей их совокупности.

Любая реализация ПСА основывается на программировании некоторого готового на данный момент варианта системы алгоритмов. При этом в программе фиксируется конкретное состояние системы правил, т.е. неизбежно и существующие в ней на этот момент неточности и неполнота. С целью эффективной обработки больших текстов ранее использовались компилируемые языки программирования, такие как C++, или Delphi Pascal, как в [5]. Такой подход ориентирован на окончательную эффективную реализацию, а не на усовершенствование системы правил – исправление ошибок, пополнение и совершенствование алгоритмов, оптимизация их структуры. При его использовании подобные модификации программы требуют существенных временных затрат.

Для обеспечения возможности отладки лингвистических алгоритмов на разных примерах, в различном порядке, по частям или целиком, была создана специальная инструментальная среда [1,6,7]. В ее основе лежит объектная модель, отображающая как структуру алгоритма, так и его вычисление на тестовом примере (тексте).

В ПСА мы постоянно сталкиваемся с разного рода неоднозначностями. Примером такой ситуации может служить неразрешенная на данный момент вычисления морфологическая омонимия. В этом случае на базе предложения со словом-омонимом возникает два или более его вариантов, в каждом из которых омонимия разрешена. Используемая модель поддерживает возникновение и обработку альтернатив в представлении и в процессе вычисления алгоритма, а также множественность связанных с этим вариантов анализа предложения.

Алгоритм в модели представляется как сеть. Его основными структурными элементами являются объект, представляющий алгоритм в целом (его общие свойства и поведение), узел алгоритма, правило. Правило содержит условия, применяемые к анализируемому предложению или некоторой его части и, возможно, действия, модифицирующие представление предложения, если соответствующие условия правила выполняются. Узел содержит (ссылается на) ровно одно правило. В каждой из альтернатив он ссылается не более чем на два дочерних узла. При вычислении переход на один из этих узлов определяется истинностью условий правила родительского узла.

В оригинальных алгоритмах используются обратные связи (для перехода на узлы, предшествующие данному), а также управляющие перечислительные конструкции вида «для слов предложения, удовлетворяющих <*условие*>, применить …». Эти ситуации в представлении алгоритма моделируются посредством специального объекта-перечислителя, а его обработка в процессе вычисления приводит к рекурсивным повторениям обхода соответствующей этому объекту подсети алгоритма.

Модель позволяет при вычислении одного алгоритма обращаться, передавая необходимые параметры, к вычислению другого (аналог вызова процедур в программировании).

Представление варианта предложения базируется на модели многофункциональных словарей, основанной на синтезе лингвистических единиц [8].

В [6, 7] была описана функциональность рассматриваемой инструментальной среды, обеспечивающая следующие возможности для проведения тестирования и отладки алгоритмов:

- установка прерывания,
- автоматическое выполнение части алгоритма,
- пошаговое выполнение алгоритмов,
- просмотр контекста выполнения (значений переменных, содержания узла и связанного с узлом правила, текущего состояния представления анализируемого предложения),
- пробное вычисление выражений элементов правил,
- сохранение промежуточных состояний модели анализируемого предложения,
- трассировка выполнения алгоритма.

К данному моменту времени в этой среде промоделированы и в значительной степени отлажены алгоритмы ПСА, представляющие собой блок внутрисегментного анализа и некоторые необходимые для него модули предсинтаксиса. В ходе этой работы была расширена функциональность: реализация так называемого пакетного тестирования существенно ускорила проверку правильности всей системы алгоритмов в процессе их модификации.

Общая характеристика реализованных алгоритмов ПСА

В разрабатываемой ПСА работе модуля внутрисегментного анализа (ВА) предшествуют алгоритмы модулей предсинтаксического анализа, предсегментации и сегментации. ВА строит графы синтагматических связей в уже построенных сегментах: простых-главных и придаточных предложениях, деепричастных и других обособляемых оборотах, т.е. работает в границах уже построенных сегментов с учетом проективности подчинительных и сочинительных связей, построенных к этому этапу анализа.

Важную роль в системе занимает универсальный модуль сегментации, так как сегментация значительно упрощает решение задач ВА. Лингвистические возможности модуля сегментации [3] демонстрирует его экспериментальная реализация в работе [5]. В представляемой программной версии он еще не реализован, и все эксперименты по отладке программ алгоритмов ВА проводились на простых (односегментных) предложениях.

В настоящей версии отчасти реализованы необходимые для ВА модули предсинтаксиса: стандартные универсальные подпрограммы проверки согласования, алгоритмы постморфологии, корректирующие и дополняющие результаты морфологического анализа, наиболее важные алгоритмы снятия омонимии частей речи и часть алгоритмов модуля предсегментации: построение атрибутивных именных групп (ИГ: существительное – хозяин прилагательного или его синтаксического аналога) и предложных групп (ПГ), конструкций с именами собственными, числами, сложных сказуемых, синтагм со слугами – обособленными приложениями, анализ сочинения и др.

Таким образом, к началу работы ВА уже построены необходимые для сегментации внутрисегментные связи.

Модуль ВА в настоящей версии системы состоит из четырех алгоритмов:

- поиск сказуемого и подлежащего,
- заполнение словарно заданных валентностей (управление инфинитивом и Род., Дат., Вин., Твор. падежами);
- поиск хозяина ПГ,
- поиск хозяев слабоуправляемых ИГ в Род.п. и наречий.

Как уже сказано выше, словам задается только грамматическое управление: в модели управления нет лексико-семантической информации, что отчасти компенсируется порядком работы модулей системы, отчасти – собственно структурой алгоритмов ВА.

Специфика организации алгоритмов ВА, обусловленная отсутствием лексико-семантической информации о заполнении валентностей, влечет необходимость задавать некоторые специфические условия построения синтагм списками возможных линейных конфигураций. С точки зрения организации лингвистического обеспечения алгоритмы ВА соединяют в себе некоторые особенности первого и второго поколений систем автоматического анализа. Каждый из алгоритмов состоит из основной – «грамматической» части анализа и подпрограммы «Частных случаев».

Грамматическая часть анализа устроена по принципу организации алгоритмов первого поколения: лингвистическая информация задается имплицитно циклической процедурой поиска универсальных признаков грамматически правильных манифестаций анализируемых синтагм.

Исключения из правил заданы в каждом алгоритме эксплицитно (т.е. по принципу организации систем второго поколения) – подпрограммой «Частных случаев», которая является словарем линейных конфигураций – «масок» для исключений из правил. При этом линейные конфигурации исключений могут быть заданы с любой степенью подробности, вплоть до конкретного словосочетания. В процессе работы алгоритма обращение к словарю «Частных случаев», как к любому списку исключений из правила, предваряет обращение к общеграмматической части алгоритма и обеспечивает возможность легкого уточнения и пополнения алгоритмов при изменении специфики анализируемых текстов.

Разберем на двух примерах работу модуля ВА (типы связи в системе задаются номерами синтагм Rn, которые при связях на картинке не проставляются, и мы их не даем, так как здесь нет возможности их комментировать).

Пример (1)

В предсегментации строится *будут* R *отстаивать, языке* R *родном*, *на* R *языке*.

На этапе ВА: при поиске сказуемого и подлежащего получаем *будут* R *они*; при заполнении валентностей – *отстаивать* R *образование*; хозяина ПГ *на родном языке* ищем по словарю «Частных случаев», и, не найдя в перечне заданных там конкретных линейных конфигураций нужной, переходим к Таблице списков нестандартных (неглагольных) хозяев для конкретных предлогов, управляющих определенными падежами:

на $_{тв}$	акцент, голосование, использование, образование, преподавание, рост, термины, ...

Пример (2)

В постморфологии строится грамматический образ аббревиатуры *ЕС*.

В предсегментации строятся группы *проблемой* R *основной, безгражданство* R *массовое, по* R *правам* и определяется хозяин *ЕС: комиссар* R *ЕС* (где ЕС в Род.п.).

На этапе ВА: при поиске подлежащего и сказуемого строится *назвал* R *комиссар*; при заполнении валентностей строится *назвал* R *безгражданство* и *назвал* R *проблемой*; при поиске хозяина группы *по правам человека* в словаре «Частных случаев» находим ситуацию:

> **16 [X … ПГ= *по*** R64 ***правам*** + ***человека*****]:** [между **X** и ***по*** м.б. только ИГ в Род.п., ПГ, D, частицы] & X = *библиотека, Госбюро, документы, законодательство, Евросуд, комитет,* ***комиссар****, Конвенция, линия, обзор, образование, просвещение, суд, уполномоченный, центр, …*,

Наш случай ей удовлетворяет, в результате строится *комиссар* R *по (правам человека)*; при поиске хозяев для слабоуправляемого Род.п. достраиваются связи *правам* R *человека* и *проблемой* R *Латвии*.

Процесс реализации и отладки

Как уже упоминалось в [7], процесс моделирования и отладки алгоритмов происходит при взаимодействии лингвиста (как специалиста предметной области) и специально обученного программиста, сочетающего, с одной стороны, понимание лингвистической проблематики, а с другой, владеющего технологией объектного моделирования в среде для экспериментов с алгоритмами ПСА.

Содержательная разработка ПСА – это прерогатива лингвиста. По его спецификации программист на входном языке инструментальной среды описывает модель алгоритмов. Синтаксис такого языка представляет собой Лисп-выражения, включающие как стандартные функции Common Лиспа, так и системно-определенные функции и формы объектной модели. При необходимости программист обеспечивает в среде новые функции. Вместе с возможностью использовать в кодах условий и действий правил произвольные выражения Common Лиспа, являющегося универсальным языком программирования, это делает систему открытой (в противоположность прикладным системам, функциональность которых ограничена их входным языком).

Написанные алгоритмы испытываются в различных сочетаниях на некотором множестве тестов. Программист имеет возможность вычислить алгоритм для данного предложения и, если нужно, произвести его пошаговую отладку, аналогично тому, как это делается в инструментальных средах для разработки программ.

С ростом количества моделируемых алгоритмов стало ясно, что необходима автоматизация тестирования всей совокупности алгоритмов или только их некоторой части за один «прогон». Дело в том, что при изменении части любого из алгоритмов нужно было просматривать результат его работы для каждого из проверенных ранее примеров, что при достаточно большом количестве примеров для каждого из алгоритмов приводит к существенным временным затратам.

Выходом из этой ситуации стал механизм, позволяющий разом протестировать выделенную совокупность алгоритмов, вычисляя каждый из них для актуального множества примеров – так называемое *пакетное тестирование*. Его применение существенно облегчило поддержку адекватности и корректности тестируемого комплекса алгоритмов после их изменений.

Пакетное тестирование и проект алгоритма

В основе пакетного тестирования лежит возможность автоматически применять составленные ранее тесты к объекту-результату вычисления алгоритма на некотором примере. Это обеспечивается тем, что в Лиспе можно интерпретировать построенные ранее Лисп-выражения.

Тестовые Лисп-выражения содержат перечень проверок, которые необходимо осуществить после применения какого-либо алгоритма к тестовому примеру. Каждая такая проверка представляет собой вызов стандартного (Common Lisp) или системно-определенного предиката. Аргументами таких предикатов являются вызовы системно-определенных функций доступа к различным данным совокупности объектов, представляющих результат ПСА предложения, и эталонные данные, с которыми сравниваются данные, полученные в результате анализа. Сведения о процессе тестирования выводятся в соответствующий протокол, причем глобальные флаги позволяют настроить такой вывод на подробную печать тестов и их результатов или только на вывод выражений

и результатов тестов, в которых проявились ошибки (failure-тестов). Можно также пропускать либо все тесты целиком, либо только до обнаружения первой ошибки.

Эти тесты могут быть получены автоматически по результатам вычисления алгоритма на примере – в случае, если полученный результат был расценен разработчиком как верный. Автоматически сгенерированный файл таких тестов охватывает проверки всех свойств слов предложения и связей между ними для всех получившихся альтернатив. Если же разработчику необходима только часть из них, остальные можно удалить из этого файла вручную.

Важную роль в организации всего комплекса тестов играют *проект алгоритма* и *проект тестирования*. Такие проекты аналогичны проектам в инструментальных системах разработки программ. Проекты поддерживают модульную организацию тестов и локальность связанных с разработкой ресурсов: описаний, кодов, данных.

Структура проекта алгоритма, точнее, конкретной версии данного алгоритма, основана на следующих естественных ассоциациях. С алгоритмом связаны:

- единственный файл, содержащий его код,
- единственный файл с оригинальным описанием,
- совокупность других алгоритмов и иных ресурсов, которые должны быть загружены для его работы,
- совокупность его тестовых примеров.

При этом учитывается, что один и тот же пример может использоваться для тестирования разных алгоритмов. При этом ясно, что, применяя разные алгоритмы к одному и тому же предложению, мы будем иметь разные результаты анализа, поэтому отдельный тестовый файл соответствует паре алгоритм-пример. Примеры хранятся в разных представлениях: как в виде текстового файла, так и в виде его сгенерированной объектной модели (как отдельный файл в соответствующем Лисп-синтаксисе).

В файле проекта такие ассоциации определяются путями к соответствующим файлам.

Проект тестирования (а их может быть несколько для данного алгоритма) лишь выделяет из списка всех тестовых примеров в проекте алгоритма актуальное подмножество для некоторого конкретного плана тестирования.

Собственно план тестирования определяется пакетным файлом тестирования, в котором специальная тестирующая функция применяется к одному или нескольким проектам тестирования (точнее, аргументами здесь являются пути к файлам соответствующих проектов).

Для каждого применения алгоритма к некоторому примеру может быть построен (выбором соответствующей опции) так называемый проект вычисления алгоритма, в котором сохраняются:

- файлы с промежуточными данными (например, входные и выходные файлы для «внешней» программы морфологического анализа), которые в частности могут быть использованы для дополнительной проверки,
- файлы с выходными данными для взаимодействия с другими приложениями (например, xml-файлы для визуализации синтаксических графов примеров[3], для интерактивной визуализации дерева обхода данного алгоритма с трассировкой его вычисления[4]).

Заключение

Итогом настоящей работы стала пробная реализация универсального модуля внутрисегментного анализа и некоторых необходимых для него модулей предсинтаксиса при помощи объектной среды для экспериментов с алгоритмами поверхностно-синтаксического анализа. При этом исполнителей интересовала возможность понять применимость описываемой среды не только для отладки лингвистических алгоритмов, но и для имплементации конкретного фрагмента синтаксического анализа текста, используемого в рамках реальной задачи, в которой его результат непосредственно передавался бы на вход подсистемы семантического анализа текста по нескольким фиксированным факторам.

Применение рассматриваемой инструментальной среды позволило выполнить эту работу в достаточно сжатые сроки ограниченными силами.

Совокупность промоделированных алгоритмов представляет собой замкнутый фрагмент поверхностно-синтаксического анализа, который покрывает любые сегменты (в смысле [3]), в частности, простые предложения, содержащие, однако, такие конструкции, как сочинение, именные и предложные группы, сложные сказуемые. Она отлаживалась на простых предложениях, но предполагается, что после реализации алгоритмов сегментации она будет работать в составе полной системы алгоритмов поверхностно-синтаксического анализа. Конечно, исполнители не гарантируют ее абсолютной применимости даже ко всем простым предложениям. Более того, эта задача вряд ли может быть решена окончательно. Однако в рамках рассматриваемого подхода реализация системы алгоритмов является открытой для пополнений и исправлений.

[3] Для визуализации изменений примера в результате работы алгоритма И.М.Ножовым было сделано средство графического отображения. Пример визуализации см. в данной статье в описании реализованных алгоритмов.

[4] Для наглядного отображения работы алгоритма в каждой сессии Г.Ю. Айрияном на основе разрабатываемой им технологии [9] было сделано приложение, показывающее проход алгоритма в виде дерева с соответствующей функциональностью сворачивания/разворачивания узлов.

Список литературы:

1. *Баталина А.М., Епифанов М.Е., Ивличева О.О., Кобзарева Т.Ю., Лахути Д.Г. Инструментальная среда для экспериментов с алгоритмами поверхностно-синтаксического анализа. // Компьютерная лингвистика и интеллектуальные технологии: Труды Международной конференции Диалог'2004 ("Верхневолжский", 2-7 июня 2004 г.), М.: Наука, с. 32-38*
2. *Кобзарева Т.Ю., Лахути Д.Г., Ножов И.М. Сегментация русского предложения. // Труды конференции. Седьмая национальная конференция по искусственному интеллекту с международным участием. КИИ' 2000 Москва. Издательство Физико-математической литературы. 2000.с. 879-880.*
3. *Кобзарева Т.Ю. Принципы сегментационного анализа русского предложения. // Московский лингвистический журнал, 2004, Т.8, №1, М.: РГГУ, с. 31-80.*
4. *Мельчук И.А. Автоматический синтаксический анализ.Т.1. Новосибирск, 1964.*
5. *Ножов И.М. Процессор синтаксической сегментации русского предложения.//НТИ, сер. 2, 2003, № 11, с. 26-37.*
6. *Баталина А.М. Объектное моделирование поверхностно-синтаксического анализа. / Девятая национальная конференция по искусственному интеллекту с международным участием КИИ-2004: Труды конференции. Т.2. - М.: Физматлит, 2004, с. 462-471*
7. *Баталина А.М., Айриян Г.Ю., Епифанов М.Е., Кобзарева Т.Ю., Лахути Д.Г. Автоматизация отладки алгоритмов поверхностно-синтаксического анализа. // Компьютерная лингвистика и интеллектуальные технологии: Труды Международной конференции Диалог'2005 (Звенигород, 1-6 июня 2005 г.), стр. 45 - 50.*
8. *Ивличева О.О., Епифанов М.Е., Лахути Д.Г. Объектная модель многофункциональных словарей, основанная на синтезе лингвистических единиц. // Компьютерная лингвистика и интеллектуальные технологии: Труды Международной конференции Диалог'2003 (Протвино, 11-16 июня 2003 г.), М.: Наука, с. 223-231*
9. *Айриян Г.Ю., Епифанов М.Е., Лахути Д.Г. Об интерактивной визуализации и представлении иерархических структур в гуманитарных приложениях.// Труды конференции. В 3-х т. Т.2. Девятая национальная конференция по искусственному интеллекту с международным участием. КИИ' 2004 Москва. Издательство Физико-математической литературы. 2004.с.443-451.*