

Санкт-Петербургский Государственный
Университет

Математико-механический факультет

Кафедра системного программирования

Автоматизированный анализ
информационных сообщений о
научных мероприятиях для
выявления актуальной тематики

Дипломная работа студента 545 группы

Белова Николая Валерьевича

Научный руководитель		к.ф.-м.н. А.Л. Тулупьев
	/подпись/	
Рецензент		д.т.н., проф. Б.В. Соколов
	/подпись/	
“Допустить к защите”		д.ф.-м.н., проф. А.Н. Терехов
заведующий кафедрой	/подпись/	

Санкт-Петербург
2009

Содержание

I	Введение	4
1	Цель работы	4
2	Задачи, которые позволяют достичь эту цель	5
3	Задачи автоматизации	5
II	Глава 1. Описание предметной области	6
4	Сравнение существующих методов автоматической рубрикации	6
4.1	Методы машинного обучения	7
4.2	Методы, основанные на знаниях	8
4.2.1	LexisNexis	8
4.2.2	Reuters	9
4.2.3	УИС РОССИЯ	10
5	Выбор принципа работы рубрикаторы	10
6	Лексико-грамматический анализ	11
III	Глава 2	13
7	Что и почему выбрал	13
7.1	Java Development Kit	13
7.2	GlassFish	14
7.3	NetBeans	15

IV	Глава 3	16
8	RuWordParser	17
9	AutomaticDefinitionRubricator	26
10	Принципы функционирования	26
11	Описание наиболее важных частей	26
V	Глава 4	27
VI	Заключение	28

Часть I

Введение

Задача рубрикации(классификации) текстовых сообщений является весьма актуальной в связи с ростом объема текстовой информации. При большом числе информационных сообщений имеет смысл говорить только об автоматической рубрикации. Предложено много методов для решения данной задачи посредством автоматических процедур. Данные методы можно разделить на два класса /citeAgeevMS: методы машинного обучения и методы основанные на знаниях (также иногда именуемые "Инженерный подход"). Однако данные методы разрабатывались для больших текстовых документов, и не подходят для рубрикации сверхкоротких текстов. Для сверхкоротких текстов, в частности для аннотаций конференций, нужны особые методы рубрицирования. Так как учитывая особенность строения сверхкоротких текстов, то есть их информативность, их лаконичность, можно добиться более быстрой работы рубрикатора, хорошей точности рубрицирования, при более простой структуре рубрикатора. К тому же практически реализованные рубрикаторы либо недоступны для изучения, ибо являются коммерческой тайной, либо не учитывают специфику русского языка.

1 Цель работы

Цель данной работы — разработать рубрикатор для сверхкоротких сообщений (аннотаций к конференциям). Который будет достаточно прост в настройке и прост по структуре.

**2 Задачи, которые позволяют достичь эту
 цель**

3 Задачи автоматизации

Часть II

Глава 1. Описание предметной области

Более подробное обоснование цели работы и постановка задачи автоматизации

Хорошо зарекомендовали себя следующие методы автоматизированной рубрикации:

- метод рубрицирования по дефинициям
- метод рубрикации на основе семантических образов рубрик

4 Сравнение существующих методов автоматической рубрикации

Предложено достаточно много методов автоматической рубрикации текстовых документов. Существующие методы можно разделить на два принципиально различных класса: методы машинного обучения и методы, основанные на знаниях.

При применении методом машинного обучения для построения классификатора используется коллекция документов, предварительно отрубрицированная человеком. Алгоритм машинного обучения строит процедуру классификации документов на основе автоматического анализа заданного множества отрубрицированных текстов. При использовании методов, основанных на знаниях правила отнесения документа к той или

иной рубрике задаются экспертами на основе анализа рубрикатора и, возможно, части текстов, подлежащих рубрицированию. Принципиальная разница между двумя группами методов состоит в том, что методы машинного обучения используют математические методы для извлечения знаний из обучающей коллекции текстов, в то время как методы основанные на знаниях используют знания эксперта о свойствах текстов, принадлежащих рубрикам. Знания эксперта основываются, в первую очередь, на предыдущем опыте, в частности на большой коллекции прочитанных ранее текстов, и во вторую очередь, на части текстов, подлежащих рубрицированию.

4.1 Методы машинного обучения

При классификации текстов методы машинного обучения работают с абстрактной векторной моделью документа и не учитывают особенностей задачи тематической классификации текстов и структуры рубрикатора. Качество рубрикации для систем, основанных на машинном обучении, является довольно высоким для небольших рубрикаторов, и сильно падает с увеличением количества рубрик и усложнением структуры рубрикатора [3]. Однако, во многих случаях, даже при наличии заранее отрубрицированной коллекции документов, методы машинного обучения неприменимы.

Для построения абстрактной векторной модели документа, большинство систем автоматической классификации текстов используют вероятностный подход, который плохо применим для сверх-коротких текстов. Собственно почти все методы рубрикации текстов рассчитаны на текст достаточно большого объема.

4.2 Методы, основанные на знаниях

В основе идеологии данных методов лежит убежденность в том, что рубрикатор создается осмысленно. То есть за каждой рубрикой лежит обычно некий раздел области деятельности, который может быть представлен небольшим вербальным описанием. Задача эксперта состоит в том, чтобы составить описание рубрики не некотором формальном языке. Полученное описание затем используется для автоматической рубрикации документов. К сожалению, для методов этого класса существует очень мало публикаций в научной литературе, а имеющейся информации не всегда достаточно для подробного анализа методов.

Недостаток публикаций связан в первую очередь с тем, что применение методов, основанных на знаниях, требует значительного вклада труда экспертов на описание каждой рубрики, и поэтому применяется в основном лишь в коммерческих системах классификации. Кроме того, в отличие от методов машинного обучения, классификация с использованием знаний экспертов специфична для каждой предметной области.

Рассмотрим три системы классификации, основанные на знаниях:

1. технология классификации компании LexisNexis;
2. технология классификации, разработанная для агентства новостей Reuters;
3. технология классификации, разработанная в рамках проекта УИС РОССИЯ;

4.2.1 LexisNexis

В статье [4] описывается технология автоматического рубрицирования, применяемая в большой электронной библиотеке LexisNexis

(<http://www.lexisnexis.com>). В описываемой технологии в основном используется ручное описание рубрик. Гибкий механизм описания правил приписывания рубрик поддерживает следующие поисковые возможности:

- поиск слов и словосочетаний, без учета морфологического словоизменения (для английского языка)
- учет частотности слов с ручным заданием пороговых значений
- учет местоположения слов в документе: заголовок, аннотация, текст в начале документа, основной текст
- поддерживается задание круга источников получения информации для данной рубрики

Описание одной рубрики требует 4-8 часов ручной работы без учета времени тестирования. Для большинства рубрик качество классификации находится на уровне более 90% полноты и точности.

4.2.2 Reuters

В статье [5] описывается система классификации документов CONSTRUE/TIS (Categorization of News Stories, Rapidly, Uniformly and Extensibly/Topic Identification System). Система CONSTRUE/TIS использовалась агентством Reuters для классификации потоков новостных сообщений. Для отнесения документов к той или иной рубрике эксперт описывает правила в виде булевских формул. В качестве элементов выступают слова.

4.2.3 УИС РОССИЯ

В статье [6] описывается технология классификации документов УИС РОССИЯ (<http://www.cir.ru>). Для рубрикации документов по большим классификаторам в УИС РОССИЯ применяется комплекс из нескольких компонентов:

- тезаурус по общественно-политической тематики;
- специальное программное обеспечение АЛОТ (Автоматической Лингвистической Обработки Текста), позволяющее строить модель тематического содержания текста;
- специальная технология описания смысла рубрики посредством понятий тезауруса.

Алгоритм классификации УИС РОССИЯ показал высокую эффективность при создании систем рубрикации для различных текстовых коллекций.

5 Выбор принципа работы рубрикаторы

Для сверхкоротких текстов — таких, например, как аннотации конференций — методика распознавания, основанная на подсчете весов неприменима [7]. Даже если встречается в тексте термин из словаря системы, у них практически нет шансов набрать для присвоения рубрики нужный вес. То есть статистические методы здесь плохо применимы. В частности, большинство методов машинного обучения используют статистические методы для построения абстрактной модели текста, что делает их плохо применимыми в данном случае.

Но у сверхкоротких текстов есть важная особенность — в них практически исключено случайное попадание тематически маркированных слов и особенно словосочетаний. Так, например, если в тексте статьи содержится словосочетание "база данных то совсем не обязательно, что статья посвящена исследованию в области баз данных. Встреча же в тексте аннотации случайного словосочетания практически исключена.

Направления научных исследований достаточно чётко разделяются на разделы, к тому же эти разделы чётко обозначены. Поэтому разумнее будет использовать рубрикатор основанный на знаниях. Построение текста аннотации довольно специфично: в нём либо просто указан список ключевых словосочетаний, либо те же ключевые словосочетания включены в небольшой связный текст. Исходя из данной специфики текстов, достаточно будет задать ключевые словосочетания для каждой из рубрик. Но так как слова в русском языке морфологически меняются по очень многим параметрам (род, число, падеж, ...), то требуется нормализация словосочетаний при их поиске. Для данной задачи не требуется частотный анализ слов, а также синтаксический анализ). Требуется сделать простой в конфигурации рубрикатор с простой настройкой.

6 Лексико-грамматический анализ

Задача лексико-грамматического анализа состоит в распознавании, какой части речи принадлежит слово в тексте, а так же найти его грамматическую информацию (род, число, падеж, категория одушевлённости). Для русского языка, благодаря его развитой морфологии, задачу по определению части речи можно решить практически со стопроцентной точностью [2].

Метод рубрицирования по дефинициям заключается в том, что

Часть III

Глава 2

Чужие достижения, которые использовались. Описание программных продуктов и математика. Компротивистика (что решил использовать и почему)

7 Что и почему выбрал

При постановке задачи, одним из условий было написание продукта на языке Java. В качестве платформы разработки было выбрано решение от компании Sun Microsystems, как наиболее полное и доступное для использования. Конкретнее, были выбраны:

- Интегрированная среда разработка (IDE) NetBeans IDE 6.5
- Сервер приложений GlassFish
- Java Development Kit (JDK)

Так же, для создания пользовательского web-интерфейса был выбран AJAX Java фреймворк Google Web Toolkit (GWT). Рассмотрим каждый компонент в отдельности.

7.1 Java Development Kit

Java Development Kit, сокращенно JDK[9] — бесплатно распространяемый фирмой Sun комплект разработчика приложений на языке Java, включающий в себя:

- Компилятор Java (javac)
- Стандартные библиотеки классов Java
- Примеры
- Документация
- Различные утилиты
- Исполнительная система Java (JRE)

В состав JDK не входит интегрированная среда разработки на Java (IDE). По состоянию на апрель 2009 года последняя версия JDK 6 Update 13.

Так же, помимо JDK от Sun, существуют JDK других производителей, в частности от IBM и Apple. Но для разработки проекта был выбран JDK от Sun.

7.2 GlassFish

GlassFish [10] — это сервер приложений с открытым кодом, разрабатываемый Sun Microsystems. На данный момент существует три версии GlassFish: v1, v2, v3. В качестве основы для работы приложения был выбран GlassFish v2. По сравнению с GlassFish v1 производительность и быстродействие GlassFish v2 было улучшено сразу во многих областях [12], например JSF, JSP и Grizzly в области веб-приложений, JAXB и JAX-WS со стороны веб-сервисов, CORBA и EJB. Результатом этих усилий явился мировой рекорд, одержанный GlassFish v2 по условиям теста SPECjAppServer 2004. В июле July 2007 Sun анонсировал 1й результат на сервере T2000 с показателем 883.66 JOPS@Standard. Это на 60% быстрее, чем GlassFish V1/SJSAS 9.0, на 10% быстрее, чем BEA WebLogic

и на 30% быстрее, чем IBM WebSphere 6.1! По результатам других эталонных тестов (813.73 JOPS@Standard при использовании базы данных PostgreSQL), соотношение цены и быстродействия у GlassFish v2 в 3 раза превосходит эти результаты для Oracle на машинах HP.

Примечание: SPEC и название теста SPECjAppServer 2004 – зарегистрированные торговые марки Standard Performance Evaluation Corporation. Sun Fire T2000 (1 процессор, 8 ядер) 1.4ghz 883.66 SPECjAppServer2004 JOPS@Standard. Сравнительные результаты тестов, приведенные выше, отображают данные, опубликованные 07/10/06 на www.spec.org.

Важным достижением стало увеличение скорости запуска сервера. В версии v2 была существенно упрощена архитектура запуска и остановки сервера. Были проанализированы все зависимости, чтобы сократить число запускаемых служб. В результате было получено серьезное уменьшение времени старта и остановки как простых, так и кластерных конфигураций сервера.

GlassFish v3 не текущий момент находится в состоянии разработки [11]. Поэтому в качестве основной платформы был выбран GlassFish v2 как наиболее устойчивая и быстрая из версий.

7.3 NetBeans

Часть IV

Глава 3

В результате работы были достигнуты следующие результаты:

- Написана библиотека RuWordParser для определения грамматических характеристик слов, а так же для изменения слов;
- Разработана библиотека AutomaticRubricator для определения рубрик текста;
- Сделан web-интерфейс для настройки рубрикатора.

Библиотека RuWordParser, основываясь на наборе правил изменения слов определяет возможные характеристики слова. Так же она позволяет изменять слово по данным характеристикам, например изменять число, падеж для существительных, или род для прилагательных.

Библиотека AutomaticRubricator по заданной иерархии рубрик и тексту, используя библиотеку RuWordParser выделяет из текста словосочетания, нормализует их и соотносит их с рубриками. Для каждой рубрики пользователь указывает ключевые словосочетания.

Для настройки рубрикатора:

- Задания правил изменения слов;
- Задания правил нормализации;
- Построения иерархии рубрик;
- Добавления исключений
- Добавления морфем

Был сделан web-интерфейс. Для облегчения работы пользователя, интерфейс был сделан на AJAX технологии.

8 RuWordParser

Библиотека RuWordParser (RWP) реализует следующие функциональности:

- Разбирает слово по составу
- Определяет атрибуты слова
- Изменяет слово по атрибутам

Основа данной библиотека - это классы для работы с морфемами, представлено на рисунке 1.

Морфемы реализуются четырьмя классами: Root, Prefix, Suffix и Ending. Данные классы расширяют абстрактный класс Morpheme и реализуют интерфейс IMorpheme. Классы морфем хранят значение морфемы и умеют проверять себя на соответствие типу (корень, суффикс, приставка или окончание). Морфемы объединяются или в тройки (MorphemeTrinity), либо в списки MorphemeList. На основе данных классов строятся слова.

Для работы со словами реализованы классы на рисунке 2. Основной для работы со словами служит абстрактный шаблонный класс Word. Он реализует интерфейс IWord.

Как уже упоминалось ранее, основой для определения атрибутов слова является тройка морфем (первая приставка, последний суффикс и окончание). У большинства слов при изменении по атрибутам, например изменение имени существительного по падежам, меняются только



Но у некоторых слов русского языка меняются не только тройка морфем. Это может быть как чередование классных в корне, так и разные словоформы, например шел - иду. Для работы с ними реализован класс `ExceptionWord`, расширяющий класс `Word`. В данном классе слова хранят-

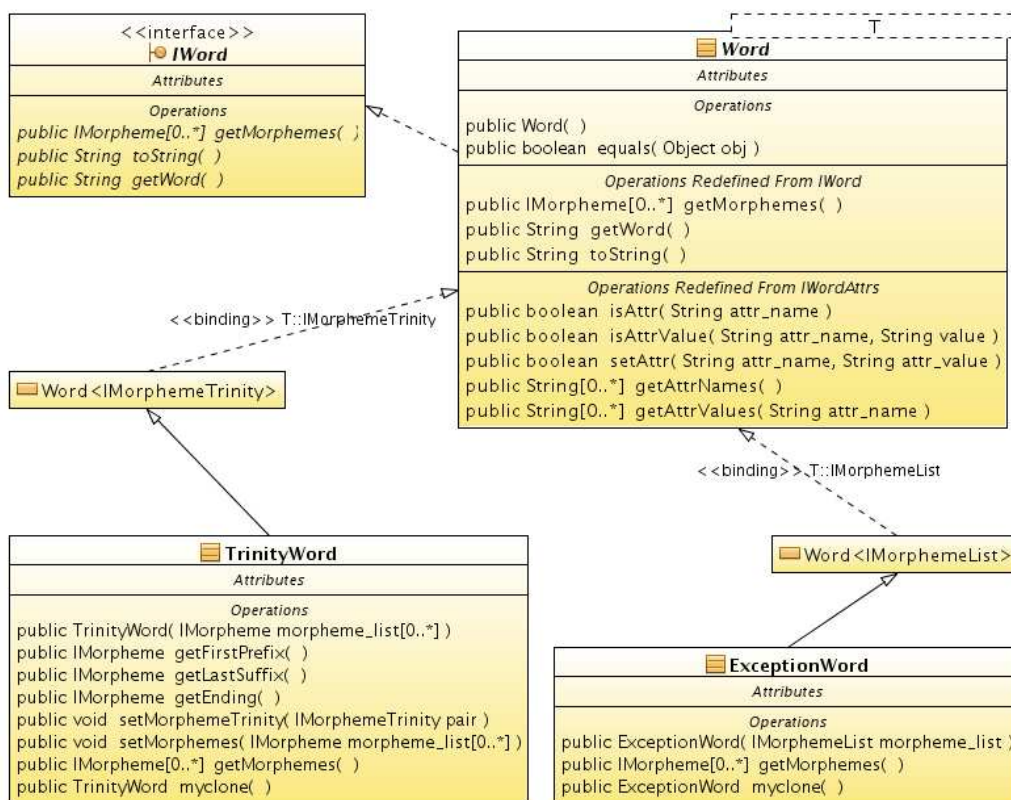


Рис. 2: Классы для представления слов

ся с помощью MorphemeList. При изменении слова по атрибуту просто происходит замена MorphemeList на правильный.

Рассмотрим теперь работу с атрибутами.

На рисунке 3 представлены классы для хранения атрибутов. Атрибут представляет из себя название атрибута и список значений. Например, название атрибута "Падеж", а значения:

- "Именительный"
- "Родительный"
- "Дательный"

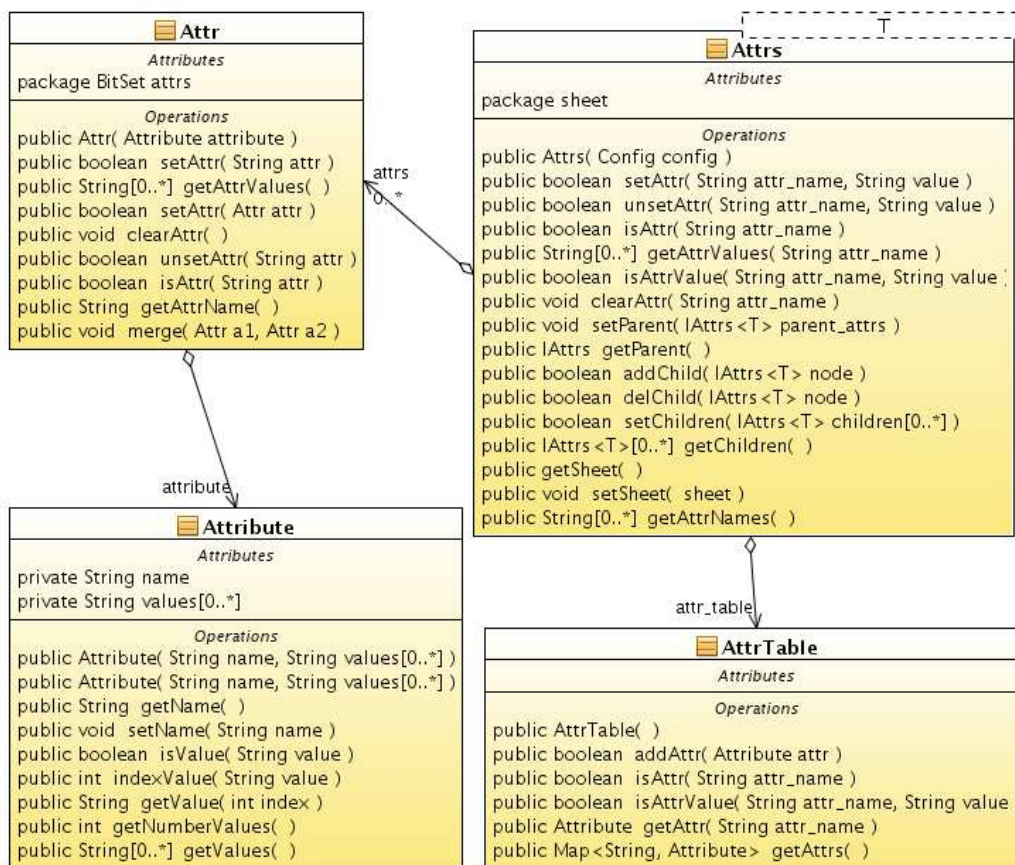


Рис. 3: Классы для представления атрибутов

- "Винительный"
- "Творительный"
- "Предложный"

Для удобства настройки программы и улучшения читаемости кода, было принято решение хранить название атрибута и его значения как строки. Для хранения атрибута используется класс Attribute. Помимо функции хранения он так же нумерует значения атрибута и предоставляет к ним доступ по индексу.

Все возможные атрибуты хранятся в классе `AttrTable` в единственном экземпляре.

С `Attribute` связан класс `Attr`. В нем устанавливаются значения атрибута для каждой конкретной ситуации. Для уменьшения объема занимаемой памяти, хранение установленных значений атрибута осуществляется с помощью битового массива. То есть для работы с атрибутом ”Падеж“ необходимо только 6 бит памяти для битового массива. К тому же это позволяет устанавливать сразу несколько значений атрибута.

Для работы с набором атрибутов реализован класс `Attrs`. `Attrs` использует `AttrTable` и динамически формирует список `Attr` при установке значений атрибутов.

Теперь рассмотрим связь `Attrs` с остальными классами библиотеки (рисунок 4).

`Attrs` реализует шаблонный интерфейс `IAttrs`.

Как уже упоминалось ранее — определение атрибутов слова происходит по трем морфемам (приставка, суффикс, окончание), или по двум (суффикс и окончание). Число морфем для определения атрибутов зависит от части речи слова. Так, например для имени существительного довольно только последнего суффикса и окончания, а для глагола требуется так же и первая приставка (ехал - приехал).

Атрибуты (`IAttrs`) образуют иерархию. Рассмотрим данную иерархию на примере имени существительного 5.

На примере все узлы дерева (обведены прямоугольниками) реализуются с помощью класса `Attrs`. Соответственно листья дерева (обведены овалом) — реализуются с помощью класса `MorphemeTrinity`. Таким вот образом задаются правила разбора. При изменении слова по какому-либо атрибуту, надо просто подняться по дереву вверх, и спуститься к

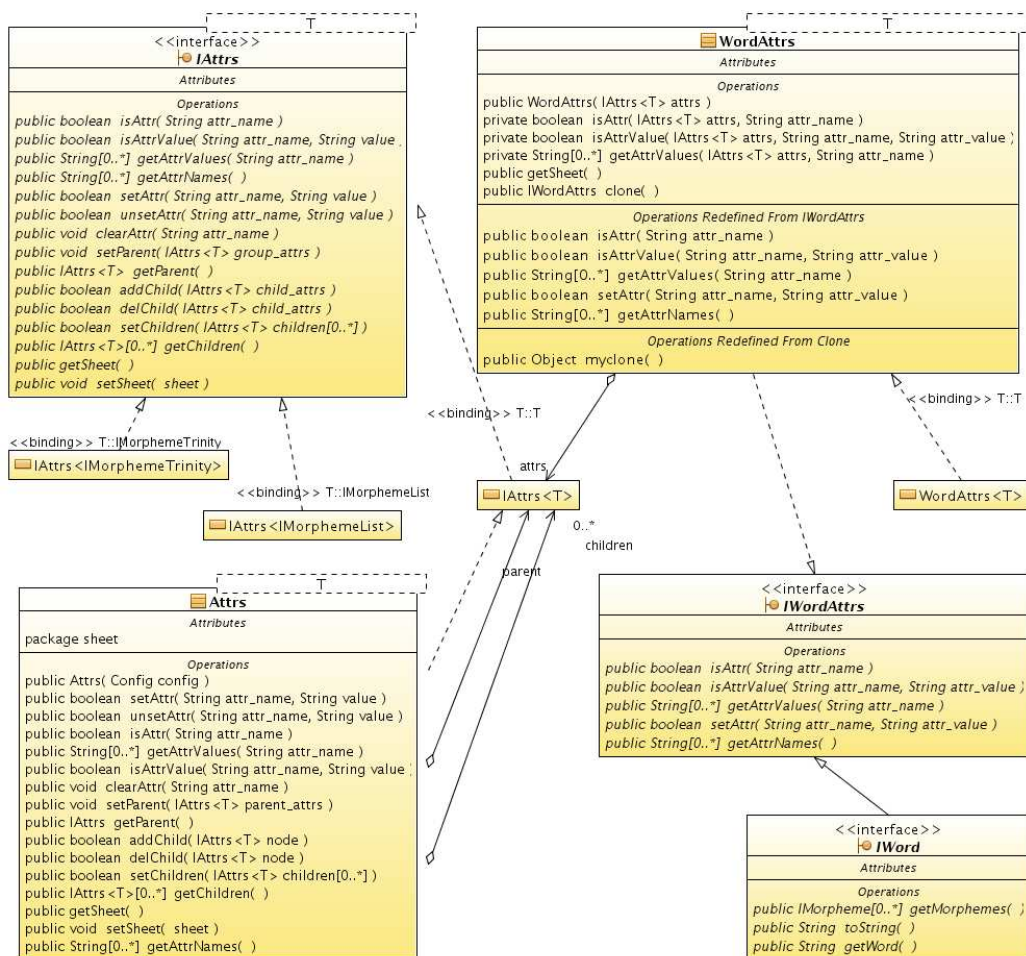


Рис. 4: Классы для работы с атрибутами

правильному листу.

Для слов-исключений дерева атрибутов практически аналогичные. Единственное отличие в том, что листья это не тройки морфем (MorphemeTrinity), а списки морфем (MorphemeList).

Для других частей речи дерева разбора более сложные, и не симметричные.

Для работы со словами сделан специальный интерфейс IWordAttrs, и его реализация WordAttrs. По сути — это всего лишь прослойка для ра-

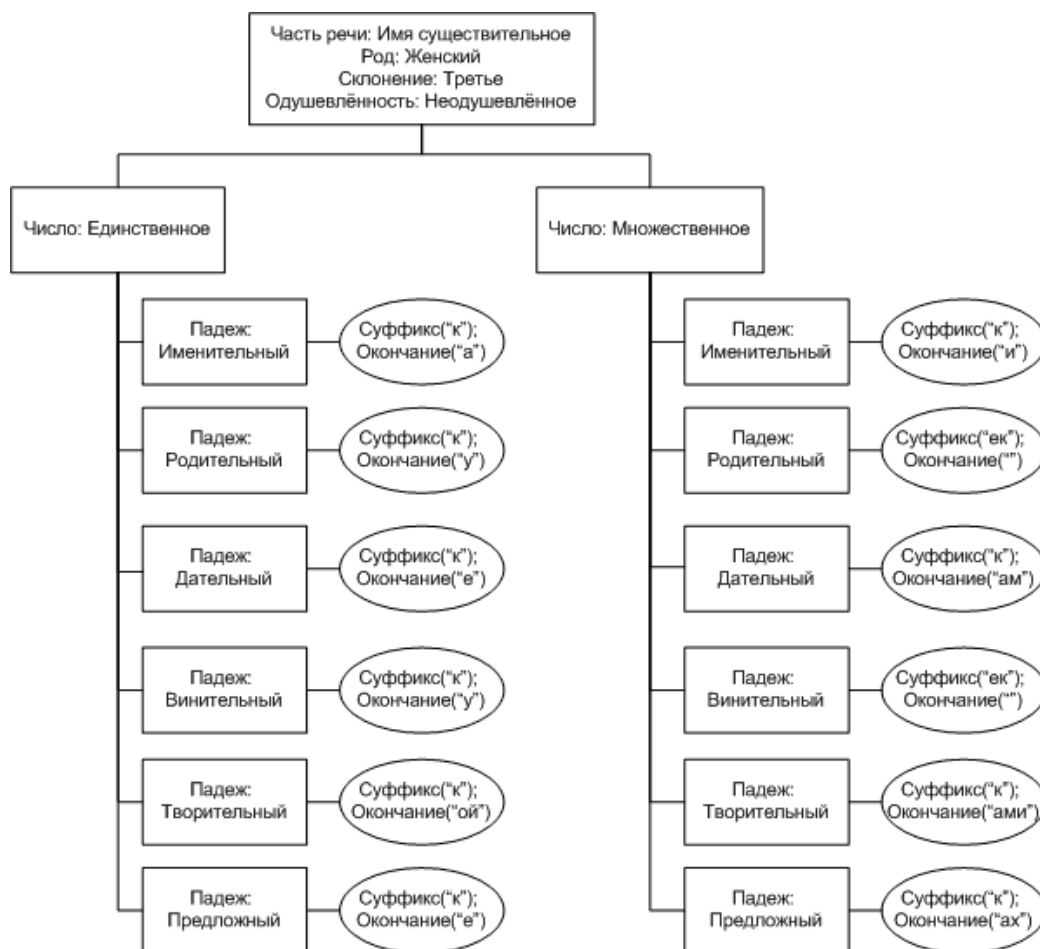


Рис. 5: Пример дерева атрибутов для имени существительного

боты с IAttrs. Основной смысл создания IWordAttrs в том, чтобы иметь доступ не ко всем методам IAttrs. Все слова (IWord) реализуют интерфейс IWordAttrs.

Рассмотрим хранения (рисунок 6).

Для более рационального использования памяти все морфемы хранятся в единственном экземпляре в экземпляре шаблонного класса IMorphemeTable. Каждый тип морфем хранится в отдельной таблице (MorphemeTable<Prefix>, MorphemeTable<Root>,

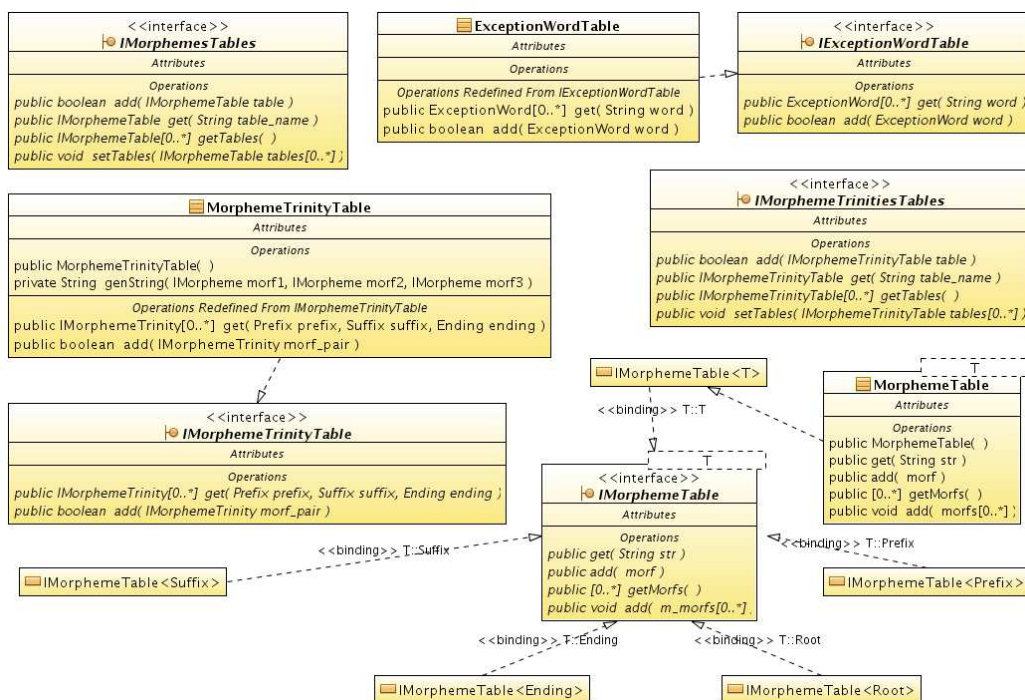


Рис. 6: Классы для хранения морфем, атрибутов и т.п.

MorphemeTable<Suffix>, MorphemeTable<Ending>). Для удобства использования данные таблицы хранятся в экземпляре класса MorphemesTables.

Аналогично хранятся тройки морфем и списки морфем в экземплярах классов MorphemeTrinityTable и ExceptionWordTable соответственно.

Все данные таблицы и все настройки хранятся в экземпляре класса Config.

Рассмотрим механизм идентификации слова (рисунок 7).

Определение атрибутов слова сделано по конвейерному принципу. Сначала слово разбирается по составу, а после по тройке морфем определяются атрибуты. В зависимости от настроек рубрикатора при разборе корень слова либо выбираем только те разборы по составу, у которых

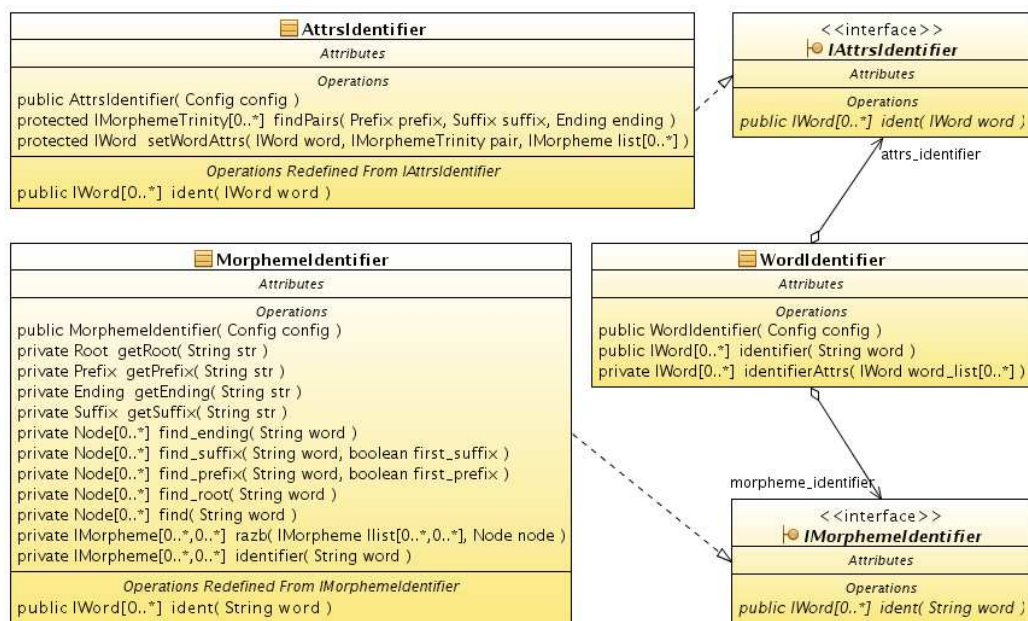


Рис. 7: Классы для разбора слова по составу и идентификации

корень есть в базе, либо считаем корнем всё, что осталось после разбора слова. Перед разбором по составу слово проверяется на исключение. Исключения бывают двух видов — жесткие (нет альтернатив и определение атрибутов по тройке морфем не происходит) и мягкие (так же проводится определение атрибутов по тройке морфем).

За разбор по составу отвечает класс `MorphemeIdentifier`, за определение атрибутов по тройке морфем — класс `AttrsIdentifier`. А за обеспечение их совместной работы и проверку слова на исключение отвечает класс `WordIdentifier`.

Работа библиотеки `RuWordParser` рассмотрена. Теперь рассмотрим работу библиотеки по рубрикации текста.

9 AutomaticDefinitionRubricator

Библиотека AutomaticDefinitionRubricator (ADR) реализует следующую функциональность:

10 Принципы функционирования

11 Описание наиболее важных частей

Часть V

Глава 4

Руководство пользователя. Как устанавливать. Сценарии работы. Примеры использования. и т.п. Можно в отдельную главу выделить результаты эксплуатации.

Часть VI

Заключение

Список литературы

- [1] Попов А.А., Морфологический анализ научного текста несловарного типа. <http://prof9.narod.ru/library/lib005/doc050.html>
- [2] Михайлян А., Некоторые методы автоматического анализа естественного языка, используемые в промышленных продуктах, <http://www.citforum.ru/programming/digest/avtestlang.shtml>
- [3] Агеев Михаил Сергеевич. Методы автоматической рубрикации текстов, основанные на машинном обучении и знаниях экспертов. Диссертация на соискание ученой степени кандидата физико-математических наук. Московский Государственный Университет им. М.В. Ломоносова. Москва. 2004 год.
- [4] Wasson M. Classification Technology at LexisNexis. // SIGIR 2001 Workshop on Operational Text Classification.
- [5] Hayes P.J., Weinstein S.P. Construe: A System for Content-Based Indexing of a Database of News Stories // Proceedings of the Second Annual Conference on Innovative Applications of Intelligence, 1990.
- [6] Добров Б.В., Лукашевич Н.В., Автоматическая рубрикация полнотекстовых документов по классификаторам сложной структуры // Восьмая национальная конференция по искусственному интеллекту – Коломна, 2002.
- [7] Белов А.А., Волович М.М., Автоматическое распознавание тематики сверхкоротких текстов. "Ашманов и Партнеры" "Поисковые технологии Москва. <http://www.dialog-21.ru/dialog2007/materials/html/05.htm>

- [8] Помыкина И.В., научный руководитель — Кисурин А.А., к.т.н., доцент, Воронежский государственный технический университет, Автоматический рубрикатор теста на основе морфологического анализа символьной информации.
- [9] http://ru.wikipedia.org/wiki/Java_Development_Kit
- [10] <http://ru.wikipedia.org/wiki/GlassFish>
- [11] <http://wiki.glassfish.java.net/Wiki.jsp?page=GlassFishUserFAQRu>
- [12] <http://developers.sun.ru/content/view/151/113/>