Министерство образования и науки Российской Федерации
Государственное образовательное учреждение
Высшего профессионального образования
Санкт-Петербургский государственный электротехнический университет «ЛЭТИ»
им. В.И. Ульянова (Ленина)
СПбГЭТУ

Факультет компьютерных технологий и информатики
Кафедра автоматизированных систем обработки информации и управления

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к выпускной работе бакалавра на тему:

Реализация утилиты для работы со словарем АОТ

г. Санкт-Петербург
2012 г.

Оглавление

Введение

Системы морфологического анализа и синтеза развиваются уже не одно десятилетие, и серьёзная обработка текста уже, пожалуй, немыслима без их помощи. Как в России, так и за рубежом на рынке существуют много коммерческих программ, которые могут успешно справляться с этими задачами, но, к сожалению, они не могут быть использованы для научных экспериментов из-за их крайней высокой цены и отсутствия исходного кода. С другой стороны, существуют бесплатные модули, которые, впрочем, часто неприемлемы из-за низкой скорости обработки слов и неполноты словарных баз.

Морфологические словари с сайта aot.ru, которые я использовал, призваны решить указанную выше проблему, обеспечив научные коллективы и вообще любых возможных энтузиастов-экспериментаторов системой морфологического анализа и синтеза.

Глава 1

Морфологический словарь morphs.mrd

В данном дипломном проекте используется два морфологических словаря: русский и английский. Русский базируется на грамматическом словаре А.А. Зализняка 1987 г.[1] На данный момент он включает в себя 162519 лемм и 2553 наборов окончаний. Английский словарь создан на основе WordNet[2] и включает в себя 104657 лемм и 442 наборов окончаний. Словари имеют одинаковую структуру — они состоят из пяти разделов: наборы окончаний, наборы ударений, история изменений словаря, приставки, леммы. Каждая часть словаря начинается со строки, в которой указано количество строк в данном разделе, что дает возможность итератору вовремя остановиться. Изменяя файл, нужно следить за этими строками.

Первая часть словаря — это строки с наборами окончаний (флексии) вида:

«%ЫЙ*йа%ОГО*йб»,

где «ЫЙ», «ОГО» — окончания, а «йа», «йб» — анкоды, обозначающие граммемы леммы (интерпретацию нужно смотреть в rgramtab.tab или egramtab.tab).

Анкодом («аношкинским кодом») называется уникальный двухбуквенный идентификатор, который соответствует некоторой комбинации значений селективных признаков и граммем. Конечное множество аношкинских кодов исчисляет все встречающиеся в данном языке комбинации морфологических характеристик. Всего в морфологическом анализаторе русского языка системы Диалинг насчитывается 870 таких кодов.

Окончание также может быть и нулевое, например: «%*яд».

Каждая строка детерминирует отдельную парадигму, поэтому словооснова ссылается на номер строки, соответствующую нужной парадигме.

Следующим разделом идут наборы ударений, которые не используются в моей программе.

Затем идёт блок информации об истории внесения изменений создателями словаря, который также не используется в моей программе.

Далее идут приставки (префиксы), которые подставляются перед словоосновой.

Последним разделом является набор лемм вида:

«ЯХТСМЕН 51 43 1 Фб -»,

где «ЯХТСМЕН» — словооснова;

«51» — ссылка на набор окончаний (номер строки в разделе окончаний);

«43» — ссылка на набор ударений;

«1» — ссылка на набор приставок;

[1] Основополагающий труд по морфологии, где впервые был предложен системный подход к описанию грамматических парадигм, включающих не только изменение буквенного состава слов, но и ударения.

[2] Семантическая сеть для английского языка, разработанная в Принстонском университете, и выпущенная вместе с сопутствующим программным обеспечением под некопилефтной свободной лицензией.

«Фб» — ссылка на общие граммемы данной леммы (поле Ancode) (может быть «-»);

«-» — не реализовано на данный момент.

Общие граммемы данной леммы, это те граммемы, которые должны быть приписаны всем словоформам данной леммы, например, граммема «фам» (фамилия), или граммема «лок» (локативность). Это часто уже семантизированные граммемы. Набор приставок леммы — это те приставки, с которыми лемма образует полное слова языка. В набор приставок может входить пустая приставка, что означает, что лемма может быть использована сама по себе (без приставок).

Глава 2
Структура файлов словарей rgramtab.tab и egramtab.tab

Данный словарь служит для определения по анкоду найденного слова части речи, рода, склонения, падежа и т.п. Состоит он из пустых строк, комментариев (строк, начинающихся с «//») и строк вида: «аа А С мр,ед,им», где

«аа» — анкод;

«А» — ???;

«С» — код части речи (Все обозначения можно узнать в таблице 2.1);

«мр,ед,им» — набор граммем.

Таблица 2.1: Полный перечень русских частей речи.

Часть речи в системе Диалинг	Расшифровка	Пример
С	существительное	мама
П	прилагательное	красный
МС	местоимение-существительное	он
Г	глагол в личной форме	идет
ПРИЧАСТИЕ	причастие	идущий
ДЕЕПРИЧАСТИЕ	деепричастие	идя
ИНФИНИТИВ	инфинитив	идти
МС-ПРЕДК	местоимение-предикатив	нечего
МС-П	местоименное прилагательное	всякий
ЧИСЛ	числительное (количественное)	восемь
ЧИСЛ-П	порядковое числительное	восьмой
Н	наречие	круто
ПРЕДК	предикат	интересно
ПРЕДЛ	предлог	под
СОЮЗ	союз	и
МЕЖД	междометие	ой
ЧАСТ	частица	же, бы
ВВОДН	вводное слово	конечно
КР_ПРИЛ	краткое прилагательное	красива
КР_ПРИЧАСТИЕ	краткое причастие	построена

Граммема — это элементарный морфологический описатель, относящий словоформу к какому-то морфологическому классу, например, словоформе стол с леммой СТОЛ будут приписаны следующие наборы граммем: «мр, ед, им, но» и «мр, ед, вн, но». Таким образом, морфологический анализ выдает два варианта анализа словоформы стол с леммой СТОЛ внутри одной морфологической интерпретации: с винительным «вн» и именительным падежами «им».

Ниже перечислены все используемые граммемы:

мр, жр, ср — мужской, женский, средний род;

од, но — одушевленность, неодушевленность;

ед, мн — единственное, множественное число;

им, рд, дт, вн, тв, пр, зв — падежи, соответственно: именительный, родительный, дательный, винительный, творительный, предложный, звательный;

2 — обозначает второй родительный или второй предложный падежи;

св, нс — совершенный, несовершенный вид;

пе, нп — переходный, непереходный глагол;

дст, стр — действительный, страдательный залог;

нст, прш, буд — настоящее, прошедшее, будущее время;

пвл — повелительная форма глагола;

1л, 2л, 3л — первое, второе, третье лицо;

0 — неизменяемое;

кр — краткость (для прилагательных и причастий);

сравн — сравнительная форма (для прилагательных);

имя, фам, отч — имя, фамилия, отчество;

лок, орг — локативность, организация;

кач — качественное прилагательное;

вопр,относ — вопросительность и относительность (для наречий);

дфст — слово обычно не имеет множественного числа;

опч — частая опечатка или ошибка;

жарг, арх, проф — жаргонизм, архаизм, профессионализм;

аббр — аббревиатура;

безл — безличный глагол.

Как уже было сказано, одной словоформе может соответствовать много морфологических интерпретаций. Например, у словоформы СТАТЬ две интерпретации:

СТАТЬ, С, «но», («жр,ед,рд»,«жр,ед,дт», «жр,мн,им», «жр,мн,вн») ;

СТАТЬ, Г, «нп,св»,(«мн,дст,прш»).

Глава 3
Работа со словарем

Глава 4

Код

```
package com.mycompany.dictionaryreader.analyzer;

import com.mycompany.dictionaryreader.dictionary.DictionaryReader;
import com.mycompany.dictionaryreader.dictionary.FlexiaModel;
import com.mycompany.dictionaryreader.dictionary.GrammaReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

/**
 *
 * @author pavlin
 */
public class WordAnalyzer {
    private static GrammaReader engGrammaInfo;
    private static DictionaryReader engDictionaryReader;
    private static GrammaReader rusGrammaInfo;
    private static DictionaryReader rusDictionaryReader;

    public static void main(String[] args) throws IOException {
        engGrammaInfo = new GrammaReader("src/main/resources/dicts/eng/egramtab.tab");
        engDictionaryReader = new DictionaryReader("src/main/resources/dicts/eng/morphs.
        engDictionaryReader.proccess();
        rusGrammaInfo = new GrammaReader("src/main/resources/dicts/rus/rgramtab.tab");
        rusDictionaryReader = new DictionaryReader("src/main/resources/dicts/rus/morphs.
        rusDictionaryReader.proccess();
        WordListReader wordListReader = new WordListReader();
        Iterator listIterator = wordListReader.getWordList().listIterator();
        String word;
        while (listIterator.hasNext()) {
           word = listIterator.next().toString();
           System.out.println(word + ": " + analyze(word));
        }
    }
```

```
    private static String analyze(String word) {
        String result;
        if (word.codePointAt(0) >= 65 & word.codePointAt(0) <= 90
                | word.codePointAt(0) >= 97 & word.codePointAt(0) <= 122) {
            result = "eng" + searchWord(engDictionaryReader, engGrammaInfo, word);
        } else {
            result = "rus" + searchWord(rusDictionaryReader, rusGrammaInfo, word);
        }
        return result;
    }

    private static String searchWord(DictionaryReader dictionaryReader, GrammaReader gra
        String base;
        String result = null;
        List<FlexiaModel> formsOfWord = new ArrayList();
        for (int i = word.length(); i >= 0; i--){
            base = word.toLowerCase().substring(0, i);
            if (dictionaryReader.getWords().containsKey(base)) {
                Iterator<FlexiaModel> wordForms = dictionaryReader.getWords().get(base).
                while (wordForms.hasNext()) {
                    FlexiaModel flexiaModel = wordForms.next();
                    if (flexiaModel.create(base).equalsIgnoreCase(word.toLowerCase()))
                        formsOfWord.add(flexiaModel);
                }
                if (!formsOfWord.isEmpty()){
                    result = " " + base;
                    break;
                }
            }
        }
        Iterator<FlexiaModel> it = formsOfWord.listIterator();
        while (it.hasNext()){
            result += ", " + grammaInfo.getGrammInversIndex().get(it.next().getCode());
        }
        return result;
    }
}
```