Морфологические модули на сайте www.aot.ru

А. В. Сокирко
Берлинская и Бранденбургская Академия наук
sokirko@yandex.ru, sokirko@dwds.de

Описываются модули для морфологического анализа и синтеза, помещенные на сайт www.aot.ru. Модули предназначены для русского, английского и немецкого языков. В сообщении дается отчет по следующим пунктам:

1. структура одного морфологического словаря: возможности и ограничения;
2. оболочка редактирования словаря: пополнение и редактирование;
3. бинарное представление словаря, построенное на конечном автомате: генерация конечного автомата и поиск в конечном автомате;
4. морфологическое предсказание;
5. пополнение словаря добровольцами через Web-интерфейс (проект).

Актуальность сообщения заключается в следующем:

6. Русский словарь используется в проекте Большого корпуса русского языка. Немецкий – в проекте словаря немецкого языка 20 века.
7. Бинарное представление словаря дает теоретически максимальную скорость обработки одного слова при лемматизации и морфоанализе.
8. Все описываемые словари и модули являются Open Source.

Обоснование

Системы морфологического анализа и синтеза развиваются уже не одно десятилетие, и серьезная 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 модули сайте www.aot.ru призваны решить указанную выше проблему, обеспечив научные коллективы и вообще любых возможных энтузиастов-экспериментаторов системой морфологического анализа и синтеза, которая:

9. уже обладает словарями достаточно большого объема, пополняется добровольцами, поэтому не должна в будущем устаревать;
10. при поиске в словаре использует конечный автомат, что позволяет находить слово за линейное от его длины время (очень быстро);
11. написана на C++, компилируется под Linux и под Windows;
12. обладает развитой системой добавления новых слов;
13. имеет в распоряжении русский, немецкий и английский лексиконы;
14. распространяется бесплатно под лицензией LGPL в исходных кодах.

Все указанные выше свойства по отдельности можно встретить в существующих модулях морфологического анализа, однако именно данное сочетание свойств составляет новизну и актуальность представленной системы.

Структура морфологического словаря

Морфологический словарь, или лексикон, содержит все словоформы одного языка, в нашем случае: английского, немецкого или русского. Структуру словаря проще всего представить в виде реляционной схемы:

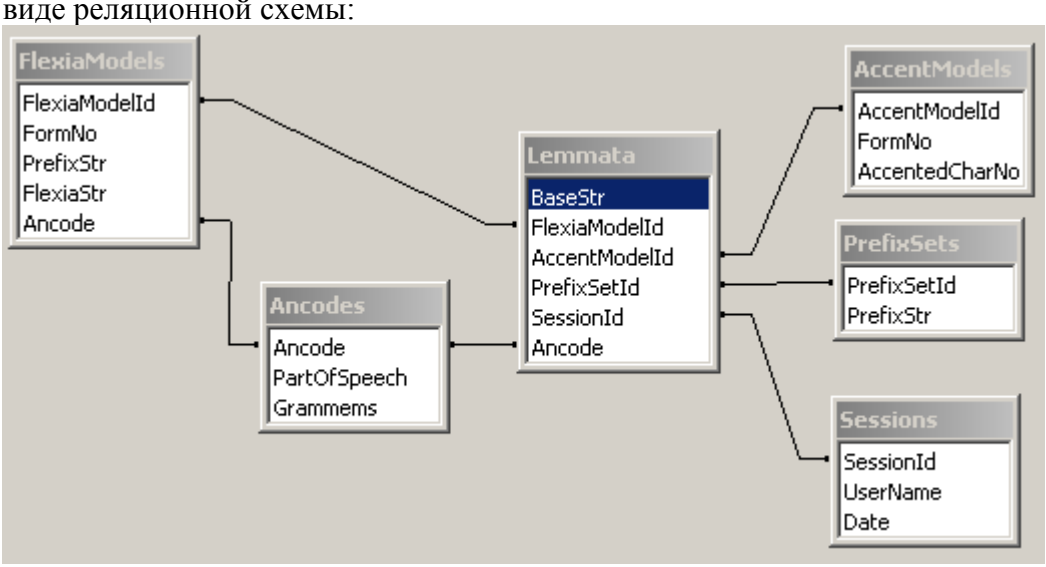

Таблица Lemmata содержит перечень всех лемм данного словаря, для каждой леммы даны ее свойства:

15. псевдооснова слова (общая для всех словоформ данного слова подстрока), (поле BaseStr);
16. ссылка на набор окончаний (поле FlexiaModelId) ;
17. ссылка на набор ударений (поле AccentModelId) ;
18. ссылка на набор приставок (поле PrefixSetId) ;
19. ссылка на пользовательскую сессию, при которой была внесено последнее изменение этой записи (поле SessionId) ;
20. ссылка на общие граммемы данной леммы (поле Ancode) (может быть пустым).

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).

Таблица FlexiaModels содержит перечень возможных окончаний всех лемм. Уникальным ключом здесь являются поля FlexiaModelId и FormNo. Поле FormNo содержит порядковый номер окончания в данном наборе окончаний, соответственно, FormNo не превосходит максимальное кол-во словоформ в одно парадигме. Далее:

21. Поле PrefixStr содержит префикс данной словоформы (возможно, пустой)
22. Поле FlexiaStr содержит окончание данной словоформы (возможно, пустое)
23. Поле Ancode содержит морфологическую интерпретацию данной словоформы.

Пусть у нас есть запись Q из таблицы Lemmata. Пусть P один из ее возможных префиксов, взятых по полю Q.PrefixSetId. Для того, чтобы получить i-ю словоформу данной леммы, надо найти в таблице FlexiaModels запись R, такую, что Q.FlexiaModelId=R.FlexiaModelId и R.FormNo=i, тогда i-я словоформа будет равна:

P+R.PrefixStr+Q.BaseStr+R.FlexiaStr.

Таблица AccentModels содержит перечень возможных номеров ударных гласных для словоформ. Уникальным ключом являются поля AccentModelId и FormNo. Поле FormNo выполняет такую же роль, что и в таблице FlexiaModels. Поле AccentedCharNo содержит номер ударной гласной с конца слова. Для каждой словоформы в словаре должно быть указано ударение, если ударения нет, тогда используется специальная константа (255).

Таблица Ancodes содержит все возможные морфологические интерпретации. Ключом является поле Ancode («аношкинский код»). Поле PartOfSpeech содержит часть речи (С,Г,П,...), а поле Grammems набор граммем, типа «мр,но,ед,им».

Вышеописанная схема показывает принципиальные возможности и ограничения структуры одного словаря. Видно, что словарь может хранить информацию о словах, возможных окончаниях, возможных приставках, которые могут присоединяться либо к отдельным словоформам, либо ко всем словоформам данной парадигмы. Словарь хранит еще информацию об ударениях. Однако очевидно, что данная схема не предназначена для хранения полного морфологического разбора слова, устройства компаундов и т.д.

Необходимо еще отметить, что в C++ реализации не используется реляционной базы данных, однако на этапе редактирования словаря C++ структуры фактически полностью повторяют вышеописанную схему.

В конце этого раздела мы дадим основные характеристики словарей текущей версии:

Язык	Кол-во лемм	Кол-во наборов окончаний
Русский[1]	162519	2553
Немецкий[2]	212560	1171
Английский[3]	104657	442

Оболочка редактирования словаря

Морфологический словарь в текущей версии может существовать в двух вариантах:

24. Вариант, предназначенный для редактирования, который следует реляционной схеме, указанной выше
25. «Бинарный» вариант, предназначенный для обработки текста, построенный на конечном автомате.

Оболочка для редактирования (MorphWizard) использует первый вариант словаря. Основными функциями оболочки являются:

26. Поиск в словаре по лемме, словоформе, морфологической интерпретации.
27. Редактирование одной парадигмы слова в т.н. slf-формате.
28. Добавление нового слова, используя предсказание по «лемме»; удаление слова.
29. Сравнение двух наборов окончаний, приписывание набора окончаний целому множеству лемм.
30. Экспорт в текстовый файл и импорт из текстового файла(в slf-формате).

Поиск по лемме, словоформе и морфологической интерпретации осуществляется с использованием таблицы Lemmata и MorphModels (см. выше). Здесь, кроме простого поиска, пользователю предоставлена возможность использования регулярных выражений, например, поиск по словоформе /^при.*ять$/ найдет все слова, в которых есть словоформы, которые начинаются с приставки «при» и заканчиваются на «ять».

---

[1] В основе русского словаря лежит морфологический словарь Зализняка.

[2] В основе немецкого словаря лежит словарь Morphy[3].

[3] В основе английского словаря лежит словарь Wordnet.

Редактирование одной парадигмы осуществляется в окне текстового редактора. Парадигма представлена в т.н. slf-формате, т.е. следующим образом. На каждой строке сначала стоит словоформа, а справа от словоформы стоят морфологические характеристики. Например,

ма'ма С жр,ед,им,од,
ма'мы С жр,ед,рд,од,

Словоформа в первой строке парадигмы объявляется леммой слова. Если у словоформы есть приставка, она должна быть отделена специальным символом "|", например:

ра'нний П мр,ед,им,од,но,
по|ра'ньше П од,но,сравн,

Ударение ставится с помощью апострофа. Основа парадигмы – это неизменяемая левая часть всех словоформ, если отбросить возможные приставки словоформ.

Добавление нового слова может быть осуществлено по крайней мере тремя способами:

31. Написание с нуля в окне редактирования;
32. Выбор для новой леммы набора окончаний, по уже существующей лемме, указанной пользователем;
33. Использование предсказание по «лемме».

Первый способ применяется, когда надо ввести абсолютно новую парадигму слова. Второй - если пользователь уверен, что новая лемма, склоняется так же, как другая уже существующая ему знакомая лемма. Третий, когда пользователь хочет выбрать подходящий вариант из возможных наиболее частотных наборов окончаний. Тогда пользователь вводит лемму и получает по окончанию введенной леммы возможные наборы окончаний, представленные существующими леммами. Результаты можно отсортировать по частоте или морфологической интерпретации.

Часть парадигм или весь словарь можно вывести в текстовый файл, где для каждой леммы даются вся информация и сама парадигма в slf-формате. Возможен так же импорт из текстового файла.

Данная оболочка написана на C++ под Windows и много раз менялась. Программисты (во временной последовательности): К. Серебряный(2000), С. Григорьев(2001), А. Сокирко(2002), Н. Кецарис(2003).

Бинарное представление словаря

Словарь в бинарном формате предоставляет следующие функции:

34. Морфологический анализ: получение по словоформе леммы, ее свойств, уникального ID леммы, морфологических характеристик входной словоформы.
35. Морфологический синтез: получение по уникальному ID леммы всей парадигмы слова со всеми словоформами и их морф. характеристиками.

Важно, что бинарное представление словаря оптимизировано прежде всего для проведения морфологического анализа. Основу этого представления составляет конечный автомат (аксептор) (см., например [1]). Автомат детерминирован и не имеет циклов, что позволяет минимизировать его в процессе построения, как это предложено в [2].

Основной цикл построения автомата выглядит так:

For all word forms W
be
AddStringToAutomat (W + '|'+Annot (W));
End

Символ '|'(annotation char) – специальный разделительный символ, которого нет в алфавите словаря, т.е. он не может встречаться в словоформе W . Функция Annot(W) выдает строку

аннотации словоформы W в некотором текстовом виде, например так: С жр,ед,им,од. Т.е, например, для словоформы «мама» в автомат может быть добавлена строка
мама|С жр,ед,им,од.
Функция AddStringToAutomat добавляет входную строку в автомат, сохраняя свойство минимальности и детерминированности автомата (см. [2]).
Поиск словоформы в таком автомате происходит за линейное от длины входной словоформы время: достаточно просто пройти все состояния автомата, которые соответствуют символам входной словоформы, далее пройти разделительный символ и получить обходом по графу все аннотации словоформы.
Основная проблема заключается в содержании аннотации. Проще всего было бы положить в аннотацию уникальный номер(ID) леммы и номер словоформы в парадигме слова. Этой информации достаточно, чтобы вычислить всю остальную необходимую информацию за константное время. Но тогда в автомате сильно вырастет кол-во состояний и связей, однако не столь фатально, чтобы не делать этого, если скорость обработки очень важна. Если, например, автомат должен выдавать только лемму, то надо включить в аннотацию длину окончания входной словоформы и окончание леммы, которое надо добавить справа к основе. В таком автомате число состояний будет невелико, и лемму он будет выдавать максимально быстро. В любом случае, содержание аннотации может зависеть от поставленной задачи и заданных параметров. В текущей версии в аннотации хранятся три числа: номер набора окончаний, номер словоформы в парадигме, номер префикса леммы. Эта аннотация позволяет за константное время вычислить лемму и морфологические свойства входной словоформы, но, например, для получения информации об ударении, нужен дополнительный бинарный поиск в числовом векторе.
Ниже приведены скоростные характеристики программы, порождающей бинарное представление[4]:

Порождение автомата					
Язык	Кол-во состояний автомата	Кол-во переходов автомата	Время порождения	Размер автомата	Размер автомата и всей остальной морф. инф.
Русский	392443	815071	62 сек.	4,7 Мб	9 Мб
Немецкий	335069	598395	26 сек	3,6 Мб	9 Мб
Английский	79102	179394	5 сек	1 Мб	3 Мб

Размер автомата и время его порождения прежде всего зависит от содержания аннотации. Например, автомат для русского языка, который может распознавать только лемму, будет содержать вдвое меньше число состояний и в два раза быстрее порождаться.
Ниже приведены скоростные характеристики самого морфологического анализа:

Скорость автомата		
Язык	Выдача леммы и морф. интерпретации словоформы	Выдача всей морф. информации
Русский	360 тыс. слов в сек.	202 тыс. слов в сек.
Немецкий	335 тыс. слов в сек.	193 тыс. слов в сек.

Первый столбец дает скорость, когда вся необходимая информация читается из аннотаций, записанных в конечном автомате, второй столбец – это, когда мы должны еще использовать

[4] Все расчеты выполнены на P4 2,6 GHz, 512 MB ОЗУ, Windows 2000.

аннотацию для получения дополнительной информации, например, ударений. Как уже сказано выше, в принципе, аннотацию можно построить так, чтобы вся информация искалась за константное время.
Материал для тестирования был взят из библиотеки Мошкова(русский язык) и из немецкого корпуса DWDS[4].

Предсказание ненайденных слов

Морфологическое предсказание работает в том случае, если слово не было найдено в словаре. Первым шагом предсказания является попытка найти существующую словоформу языка, которая максимально совпадала бы справа со входным словом. Если размер левой (неузнанной) части слова не превышает определенного предела (в текущей версии это 5 символов), а размер остатка (совпавший с какой-то словоформой) не меньше 4 символов, тогда слово предсказывается по найденному правому остатку. Это должно работать для слов, к которым были добавлены продуктивных префиксы, типа квази, мета и т.д. Поиск осуществляется последовательным отсечением символов слева и подачей «урезанного» слова в морфологический анализ.
Если слово нельзя найти таким способом, вступает в действие предсказание по окончанию. Для этого был специально создан другой конечный автомат, построенный на строках вида:

ReverseSuffix(X)|Annot(X),

где X какая-та словоформа словаря, Annot(X) – аннотация словоформы X, функция ReverseSuffix(X) возвращает перевернутое слева направо окончание словоформы X некоторой заданной длины (в текущей версии – 5). Кроме этого, в этот автомат попадают только те строки, для которых частота встречаемости ReverseSuffix(X) в словаре превосходит некоторый предел (в текущей версии – 3). Числовые параметры конечного автомата предсказания могут быть заданы в командной строке программы генерации словаря.
Есть еще одно важное ограничение автомата предсказания: т.н. факторизация по части речи. Для каждого языка указаны те части речи, которые могут продуктивными. Для русского – существительное, глагол, наречие и прилагательное. Если встречается окончание, для которого возможны разные интерпретации внутри одной продуктивной части речи, тогда в автомат добавляется только та, что содержит набор окончаний, который наиболее частотен в словаре. Таким образом, в автомате предсказания для каждого окончания и для каждой продуктивной части речи содержится только одна морфологическая интерпретации, причем наиболее продуктивная.
Поиск в таком автомате осуществляется следующим образом. Идем с конца слова по автомату до тех пор, пока существует состояние, в которое можно перейти, используя текущую букву слова. Далее обходом графа собираем все достижимые аннотации.
Если слово совпало не полностью с одним из окончаний, то возможно, что список аннотаций содержит несколько интерпретаций внутри одной части речи, тогда приходится снова выбирать наиболее продуктивную аннотацию, используя частотность набора окончаний.
Если слово не было предсказано как существительное, тогда в список возможных интерпретации добавляется вариант интерпретации как неизменяемого существительного во всех родах и числах (поскольку ненайденные слова чаще всего существительные). Таким образом, в конце получается набор из аннотаций, число которых не больше числа продуктивных частей речи и который обязательно содержит вариант интерпретации существительным.
Общая скорость предсказания(обе процедуры) в два раза ниже скорости основного поиска словоформ в словаре, но это не столь существенно, так как число ненайденных слов в нормальных текстах редко превышает 5 процентов. Нужно сказать еще, что скорость

основного автомата, которая была приведена в таблице выше, была замерена с включенным предсказанием.
Качество предсказания было подсчитано только для русского языка. Это было сделано следующим образом. Взяты новостные тексты, наугад выбраны 150 неповторяющихся предсказанных слов. Эти слова не должны быть аббревиатурами (все буквы в верхнем регистре). Все слова оказались либо существительными, либо прилагательными. Для 131 слова в результатах предсказания был хотя бы один правильный результат (одновременно лемма, часть речи, род, число и падеж). Т.е. точность предсказания – 87%. Этот результат вполне сравним с результатами других исследователей, например, для английского языка - 85 % (см. [5]) , или для французского – 88% (см [6]).

Литература

1. Гладкий А.В. Формальные грамматики и языки. М.: Наука, 1973. 368 с.
2. Jan Daciuk, Bruce Watson, and Richard Watson, *Incremental Construction of Minimal Acyclic Finite State Automata and Transducers*, proceedings of Finite State Methods in Natural Language Processing, pp. 48-56, Bilkent University, Ankara, Turkey, June 29 - July 1, 1998.
3. Lezius, Wolfgang (2000) Morphy - German Morphology, Part-of-Speech Tagging and Applications in Ulrich Heid; Stefan Evert; Egbert Lehmann and Christian Rohrer, editors, Proceedings of the 9th EURALEX International Congress pp. 619-623 Stuttgart, Germany. (http://www-psycho.uni-paderborn.de/lezius/)
4. Damir Ćavar, Alexander Geyken, Gerald Neumann(2000) **Digital Dictionary of the 20th Century German Language** in Language Technologies Conference 17 - 18 October 2000 Slovenia (см. сайт **www.dwds.de**)
5. Jan Daciuk, *Treatment of Unknown Words*, proceedings of Workshop on Implementing Automata WIA'99, Potsdam, Germany, 1999, (C) Springer Verlag LNCS Series Volume 2214, pp. 71-80, 2001.
6. Andrei Mikheev,A Automatic Rule Induction for Unknown Word Guessing, In Computational Linguistics vol 23(3), ACL 1997. pp. 405-423