АЛГОРИТМ ДЛЯ АВТОМАТИЧЕСКОГО ПОСТРОЕНИЯ СЛОВООБРАЗОВАТЕЛЬНОЙ БАЗЫ РУССКОГО ЯЗЫКА

Водолазский Д. И. (daniil.vodolazsky@mail.ru)
Новосибирский государственный университет (НГУ), Новосибирск, Россия

Аннотация. В работе предлагается основанный на правилах подход к автоматическому описанию наиболее продуктивных словообразовательных типов русского языка. На примере образований наречий показывается возможность покрыть лексику с полнотой более 92%. В статье описывается система простых правил, позволяющая представлять словообразовательные модели на формальном языке.

Ключевые слова: деривационная морфология, словообразовательные семьи, обработка естественного языка, лексикография.

AN ALGORITHM OF AN AUTOMATIC BUIDLING OF A DERIVATIONAL MORPHOLOGY RESOURSE FOR RUSSIAN

Vodolazsky D. I. (daniil.vodolazsky@mail.ru)
Novosibirsk State University (NSU), Novosibirsk, Russia

Abstract. The study is to show rule-based approach to the description of the most productive derivational types in the Russian language. As an example, the formation of adverbs shows the possibility to cover the vocabulary with the recall more than 92%. The article describes the system of simple rules that allow writing derivational models with a formal language.

Keywords: derivational morphology, derivational families, natural language processing, lexicography.

1. **Введение**

В задачах обработки естественного языка для заданного слова часто необходимо найти родственные или близкие по смыслу. В частности, это применимо в задачах перефразирования и поиска плагиата, а также в построении и тестировании моделей композициональной дистрибутивой семантики. Чтобы иметь возможность заниматься такого рода задачами, необходимо составить большую словообразовательную базу. В настоящее время вместо нее используются словари, тезаурусы и другие ресурсы, составленные преимущественно вручную. Проблема состоит в том, что в языке постоянно появляются новые слова и словари не успевают их фиксировать. Для решения этой проблемы мы разрабатываем аналог DErivBase [1] (для немецкого языка) и DerivBase.Hr [2] (для хорватского языка) под названием DerivBase.Ru. Наш алгоритм позволит получать словообразовательные цепочки для произвольных наборов слов. В данной статье мы опишем работу нашего алгоритма и продемонстрируем результаты на примере наречий.

2. **Обзор предыдущих исследований**

Работ, посвященных словообразованию в русском языке, не так много, а работ, в которых были бы представлены качественные и практичные алгоритмы, еще меньше. Нам известно о нескольких методах, позволяющих проводить морфемный разбор слова. В учебном пособии «Автоматическая обработка текстов на естественном языке и анализ данных» [3] приводятся следующие методы: метод Харриса, метод Дежона, метод Бернхард и Morfessor. Они основаны на статистике или машинном обучении и могут обучаться на неразмеченных корпусах. Однако такие подходы имеют ряд недостатков, например чрезмерное членение на морфы или недообучение для редких словообразовательных типов; также они требуют наличия размеченной обучающей выборки достаточного размера, поэтому мы не рассматриваем их в данной работе. Для русского языка существуют

словообразовательные словари (пожалуй, самые известные — словари А. Н. Тихонова [4, 5]). В начале 2000-х создавалась система «Ариадна» [6], однако она также была основана на словаре. Остается нерешенным вопрос, как создавать и легко пополнять такие системы. Достоинство нашей разработки в том, что правила, в отличие от лексики, подвергается изменениям значительно медленнее, и, соответственно, DerivBase.Ru может сочетать в себе как общеупотребительные слова, так и специальные термины и неологизмы.

3. **Описание алгоритма**

В основе нашего алгоритма лежат правила, как и в DerivBase для немецкого и хорватского языков. Описание правил и примеры их применения мы брали из «Русской грамматики» [7]. Для простоты мы решили работать только с продуктивными типами, в число которых входят типы, характерные для публицистической, научно-технической и художественной литературы и разговорной речи. Нам было необходимо формализовать правила, описанные в «Русской грамматике», и реализовать их удобное представление в программном коде, чтобы иметь возможность легко вносить правки и дополнения. В отличие от оригинальных DerivBase, исходный код (https://github.com/nsu-ai/DerivBaseRu) для DerivBase.Ru написан на Python, а не на Haskell.

В данной работе мы будем использовать следующую систему формальных операций:

- replsfx(s1, s2) — заменить суффикс s1 на s2;
- addsfx(s) — добавить суффикс s;
- delsfx(s) — удалить суффикс s;
- excsfx(s) — исключить слово, если у него префикс s;
- onlysfx(s) — пропустить слово, если у него префикс s;
- replpfx(s1, s2) — заменить префикс s1 на s2;
- addpfx(s) — добавить префикс s;
- delpfx(s) — удалить префикс s;
- excpfx(s) — исключить слово, если у него префикс s;
- onlypfx(s) — пропустить слово, если у него префикс s;
- delvowel() — удалить (беглую) гласную о, е, если это предпоследняя буква в слове, а последняя — согласная;
- addvowel() — добавить (беглую) гласную;
- plt() — заменить букву, обозначающую заднеязычный звук, на букву, обозначающую шипящий звук;
- pltinv() — инверсия предыдущего правила;
- soft() — добавить мягкий знак в конце слова;
- hard() — удалить мягкий знак в конце слова.

В дальнейшем возможно пополнение этого списка (например, специальными чередованиями).

Обратим внивание, термины «суффикс» и «префикс» используются в значениях, принятых в computer science: суффикс (префикс) — это подстрока, которой оканчивается (начинается) заданная строка.

Также в программе функциям можно передавать не один суффикс (или префикс), а целое множество.

Пример 1. Правило из «Русской грамматики» и его формальная запись.

Наречия с преф. по- и суф. -и имеют то же знач., что и в предыдущем типе; мотивирующие – прилагательные с суф. -ск- и -ий/-|j|-; по-дружески, по- дурацки, по-хозяйски, по-английски, по-московски, по-флотски, по-августовски, по-бабьи, по-охотничьи, по-собачьи; разг.: по-каковски, по-свойски. Перед суффиксом – |к'| и |j|. Этот тип обнаруживает высокую продуктивность...

onlysfx({'ский', 'цкий', 'ской', 'цкой', 'ий'}) **&** addpfx({'по-'}) **&** ((onlysfx({'ский', 'цкий', 'ской', 'цкой'}) **&** delsfx({'ий', 'ой'}) | excsfx({'ский', 'цкий', 'ской', 'цкой'}) **&** delsfx({'ий'}) **&** opt soft()) **&** addsfx({'и'})

В ряде случаев нам приходилось корректировать правила самостоятельно в силу того, что в «Русской грамматике» они описаны недостаточно формально.

Поскольку часто выполнение операций невозможно, мы используем три режима для каждой операции. В режиме «do» (по умолчанию), если возможно, функция выполняет операцию и возвращает полученное множество вариантов слов, а иначе возвращает пустой список. В режиме «try», если возможно выполнить операцию, то она выполняется, а иначе возвращается слово без изменений. В режиме «opt», если возможно, то операция выполняется или не выполняется, а иначе не выполняется и возвращается исходное слово.

Выполнить операцию f для слова w	**do**	**try**	**opt**
возможно	f(w)	f(w)	f(w), w
невозможно		w	w

Таблица 1. Режимы и результат работы для произвольной операции f и слова w

Правило в программном коде записывается в виде списка операций. Операции в списке выполняются последовательно (можно это интерпретировать как логическое «И»). Если возможно несколько случаев (например, конечная гласная у существительных), то операции исполняются параллельно (это соответствует логическому «ИЛИ»). Результатом является множество возможных слов.

Чтобы теперь получить словообразовательные цепочки, необходимо взять интересующий нас список слов, интерпретировать каждое слово как вершину графа и запустить из каждой вершины наш алгоритм. Если из вершины u в вершину v в графе есть ребро, значит, v могло быть образовано от u (но это не дает гарантий, поскольку существует много омонимичных форм, которые не являются однокоренными). Построение такого графа и оценка качества на нем — одно из дальнейших направлений нашей работы. Временная сложность этого алгоритма составляет O(mVlogV), где V — размер словаря, m — общее количество правил. Однако на практике такая оценка слишком завышенная, поскольку, если известна часть речи слова, не нужно рассматривать правила для других частей речи.

Приведем теперь несколько примеров сгенерированных множеств.

Пример 2. конь (noun) → кнем, кнём, конем, конём, конью, наконь, вконь.

Пример 3. нога (noun) → ногою, ногой, наногу, вногу.

Пример 4. редко (adv) → редковато, реденько, редкенько, редконько, редёшенько, редёхонько, редкёшенько, редкёхонько, нередко.

Пример 5. красный (adj) → красно, по-красному, вкрасную, докрасна, искрасна, накрасно.

Пример 6. ходить (verb) → ходьмя, находу, входу, находь, наход, входь, вход.

Пример 7. московский (adj) → московски, по-московскому, по-московскьему, помосковскему, по-московски, вмосковскую.

Как и следовало ожидать, в этих примерах есть некорректные с точки зрения орфографии слова, однако их не так много, и главное, что существующие слова наш алгоритм действительно генерирует. Многие некорректные варианты можно «запретить» в правилах (в нашем проекте это частично сделано при помощи «фильтров» вроде операций onlysfx).

4. **Оценка качества**

Для оценки работы нашего алгоритма мы использовали наречия из Wiktionary, предварительно убрав оттуда нецензурные, просторечные, устаревшие, немотивированные слова, а также большинство слов неясного происхождения. Это было сделано для того, чтобы упростить процесс оценки. Для оставшихся 6747 слов мы восстановили их «предков» (мы сделали это автоматически, используя инвертированные упрощенные правила без чередований, а потом провели ручную постобработку). Далее, из каждого такого «предка» мы получали множество его производных и проверяли, есть ли целевое слово в этом множестве. В качестве бейслайна мы использовали морфемноорфографический словарь А. Н. Тихонова и проверяли наличие в этом словаре целевого слова. Метрикой мы брали полноту.

Результаты приведены в Таблице 2.

	TruePos	**FalseNeg**	**Recall**	**Основные источники ошибок**
DerivBase.Ru	6217	530	0.9214	Малопродуктивные типы, неполное описание правил
Тихонов	5716	1031	0.8472	Редкие слова, неологизмы

Таблица 2. Сравнение нашего алгоритмы и словаря А. Н. Тихонова

Как видно из таблицы, наш алгоритм при поиске словообразовательных пар дает существенно более высокий результат, чем словарь А. Н. Тихонова. При этом мы ограничились только наиболее продуктивными типами. Полноту можно повысить еще на несколько процентов, добавив малопродуктивные словообразовательные типы.

5. **Анализ ошибок**

Среди 530 «нераспознанных» слов можно выделить несколько основных групп.

- Наречия, образованные от числительных (40: полвторого, впятером, в-третьих);
- Наречия с префиксом впол- (17: вполоборота, вполсилы);
- Наречия разного происхождения с префиксами в-/вз- (133: вчерне, втридорога, всмятку, вскорости, впрожелть, впридачу, вперегонку);
- Наречия с префиксами с-/со-/сыз- (51: снова, сызмальства, сверху);

- Наречия на -ами/-ями (12: временами, урывками, верхами);
- Наречия с префиксами на по- (47: подряд, подчистую, поблизости, позавчера);
- Наречия с префиксами за- (36: задолго, замертво, занужду, зачастую, зараз);
- Наречия с префиксами до-, из-/ис-, к-, на-;
- Наречия, омонимичные формам существительных в единственном числе в творительном падеже (временем, рядком, тишком);
- Исключения, единичные случаи и другие непродуктивные типы (броско, веско, ливмя, ходуном, рядышком).

6. **Дальнейшая работа**

В ближайшем будущем планируется доработать правила для прилагательных, глаголов и существительных, построить полноценную базу DerivBase.Ru, выложить ее в открытый доступ, добавить непродуктивные словообразовательные типы, применить полученные результаты к различным задачам компьютерной лингвистики. Также планируется рассмотреть модели словообразования, отличные от аффиксальных.

7. **Заключение**

В данной работе мы на примере наречий показали, что подход, основанный на правилах, является мощным и действенным для порождения словообразовательных цепочек русского языка. Хотя процесс задания большого числа правил представляется трудоемким, в результате ожидается получить уникальную для русского языка систему, позволяющую в автоматическом режиме находить словообразовательные гнезда.

Литература

[1] Zeller, Britta and Šnajder, Jan and Padó, Sebastian. DErivBase: Inducing and Evaluating a Derivational Morphology Resource for German, Proceedings of ACL 2013, pages 1201–1211.

[2] Šnajder, J.: DerivBase.hr: A High-Coverage Derivational Morphology Resource for Croatian. In:
Calzolari, N., et al. (eds.) Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2014). ELRA, Reykjavik (2014)

[3] Bol'shakova E.I., Voroncov K.V., Efremova N.Je., Klyshinskij Je.S., Lukashevich N.V., Sapin A.S. Avtomaticheskaja obrabotka tekstov na estestvennom jazyke i analiz dannyh: ucheb. posobie.
M.:Izd-vo NER HSE, 2017.

[4] Tikhonov A. N. Morfemno-orfograficheskij slovar' (2002).

[5] Tikhonov A. N. [Bol'shoj] slovoobrazovatel'nyj slovar' (1990).

[6] Packin A. I. Opyt postroenija polnoj morfemno-orientirovannoj semanticheskoj seti dlja russkogo jazyka (2004). Komp'juternaja lingvistika i intellektual'nye tehnologii

[7] Russkaja grammatika. T. 1 / N. Ju. Shvedova (gl. red.). — M., 1980.