Инженерный вестник Дона, №4 (2018)
ivdon.ru/ru/magazine/archive/n4y2018/5528

Объектно-ориентированная модель морфологического анализатора русскоязычного текста

Е.В. Ерискина, Д.С. Курушин, Д.В. Яруллин, Д.Ю. Корюкин

Пермский национальный исследовательский политехнический университет, Пермь

Аннотация: В статье описан процесс создания стеммера для морфологического анализатора Pymystem. Приведено теоретическое обоснование выбора морфологического анализа как приоритетного направления для лингвистического анализа текста. Рассмотрены основные анализаторы и выявлены их достоинства и недостатки. Описан основной алгоритм разделения вложенных структур на структурированное дерево классов. Приведена функция поиска нужных характеристик частей речи с использованием регулярных выражений в языке программирования Python. Рассмотрены и описаны основные шаги алгоритма построения необходимой иерархии. Проведен анализ результатов работы и сделаны необходимые выводы. Описаны дальнейшие перспективы развития представленной разработки.

Ключевые слова: стеммер, морфологический анализатор, дерево классов, регулярное выражение, анализ текста, компьютерная лингвистика, лемма, токен, словообразование, иерархия классов.

В процессе решения задач лингвистического анализа текста возникает проблема построения отношений между понятиями[1]. Для того, что бы связь была логически верной и ориентированной прибегают к различным видам анализа текста. Приоритетным среди них является морфологический анализ. Именно он лежит в основе любого лингвистического разбора[2].

Морфологический анализ основывается на извлечении множества словоформ из тела текста. Для каждой словоформы существует лемма — начальная форма и лексема, которая несет в себе множество всех словоформ. Слово обладает словообразовательными и формообразовательными параметрами. Отличие их состоит в том, что словообразовательные параметры не изменяются при изменении слова по формам. Процесс изменения привязан к набору определённых грамматических параметров, таких как часть речи, род, число, падеж, залог и т.д.[3].

Существует два вида морфологических анализаторов. Самый большой недостаток всего многообразия таких инструментов — привязка к одному

большому словарю, что автоматически указывает на его «конечность». Более продвинутые анализаторы могут работать на основе вероятностных методов. Анализатор, который в составе синтаксического метода имеет грамматический, называют парсером и применяют, в основном, для разбора сайтов и интернет-ресурсов. Другой вид таких программ имеет свой собственный словарь и предназначен для нахождения основы слова для заданного исходного слова. Его общепринятое название — стеммер. Существует ряд инструментов, которые наиболее распространены в современной компьютерной лингвистике. Их сравнительный анализ приведен в таблице 1.

Таблица №1

Основные морфологические анализаторы[4]

Название	Специализация	Словарь	Выходные данные
Rumor	Русский язык	125 тыс. слов	Слово и его словоформы
MyStem	Русский язык	Без словаря	Слово, его морфологические характеристики
Morphology	Русский язык Английский язык Немецкий язык	115 тыс. слов	Слово, его словоформы, корни и начальные формы
Ispell	Русский язык Немецкий язык Английский язык и др.	137,2 тыс. слов	Ориентированность на проверку орфографии; слово и его словоформы

Очевидно, что для решаемой в рамках разработки задачи наиболее подходящим анализатором является Mystem. Это единственный анализатор, который работает без встроенного словаря на основе эвристических методов. На этапе верификации модели предметной области [5] и формирования конечных отношений появилась сложность приведения полученных данных к виду структурированной иерархии классов [6]. Проблема возникает по

тому, что используемая библиотека для Python Pymystem выводит необходимые для морфологического алгоритма характеристики слов в неструктурированном виде.

Для того что бы организовать корректный поиск хотя бы одной характеристики в токенах необходимо для каждой характеристики использовать отдельное регулярное выражение. Это существенно увеличивает код, снижает читаемость алгоритма и увеличивает время его работы. Для того, чтобы выводить токены в более удобной для алгоритма морфологического поиска форме, решено создать стеммер для Pymystem, которая результатом морфологического анализа будет выводить иерархию классов. Всего стеммер содержит два исполняемых файла lingv.py и class.py.

Файл lingv.py представляет собой дерево классов. Здесь описаны все части речи русского языка и их характеристики в виде графа (рис.1).

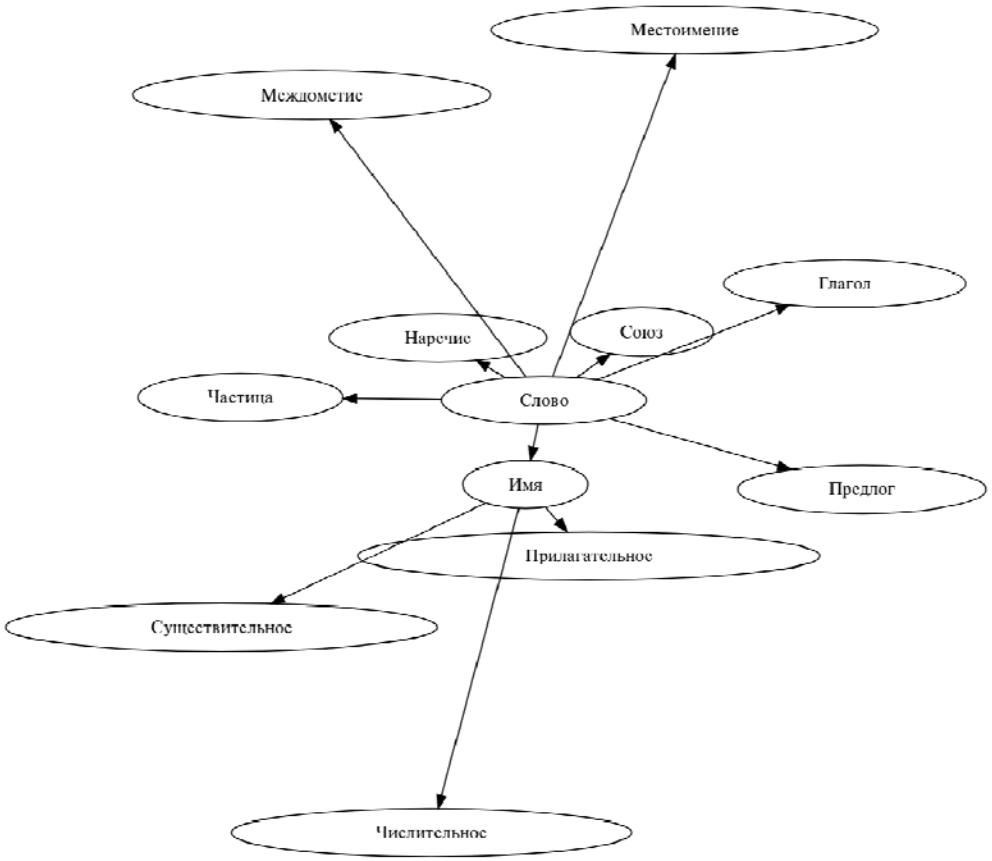

Рис. 1 – Дерево классов обёртки для Mystem

Ниже приведен код основной функции def_init_(листинг 1). Здесь происходит разбор каждого выводимого программой Mystem токена на отдельные смысловые части. Затем происходит отсечение «не нужных» частей, в данном случае, не содержащих наименования и характеристики частей речи. В завершении, с помощью регулярных выражений задаётся формат вывода нужных значений на экран. Таким образом, получается упорядоченная структура, из которой путем простых манипуляций можно извлечь значение любой характеристики. Далее в файле задаётся иерархия классов, представленная на рисунке 1. Определение принадлежности к экземпляру класса реализовано также с помощью регулярных выражений [7].

```
def __init__(self, analysis):
        try:
            self.lemma = analysis[0]['lex']
            self.analysis = analysis[0]['gr']
            self.pos = set(re.findall(r"^([A-Z]+).+", self.analysis))
        except (IndexError, KeyError):
            print(analysis)
            raise KeyError
```

Листинг 1 – Код конструктора класса

Файл class.py предназначен для конфигурации вывода результата обработки текста. Главная особенность здесь заключается в том, что если программно определить принадлежность к какому-либо классу не получается, на экране не будет выводиться значение None. Это позволит избежать избыточности (листинг 2).

```
hb = HierarchyBuilder()
for a in my.analyze(string):
    try:
        x = hb.build(a['analysis'])
        if x is not None:
            print(x, x.pos)
            if isinstance(x, Существительное):
            elif isinstance(x, Глагол):
            elif isinstance(x, Прилагательное):
            elif isinstance(x, Числительное):
           elif isinstance(x, Наречие):
           elif isinstance(x, Местоимение):
            elif isinstance(x, Союз):
            elif isinstance(x, Междометие):
            elif isinstance(x, Предлог):
            elif isinstance(x, Частица):
    except KeyError:
        pass
```

Листинг 2 – Псевдокод функции вывода дерева классов

Для сравнения результатов работы алгоритма приведен пример работы Pymystem без использования обёртки и с её подключением. Для этого использован отрывок одного и того же текста. Для упрощения восприятия результаты занесены в таблицу 2.

Таблица № 2

Сравнение результатов работы алгоритма

Исходный текст	Вывод Pymystem без обёртки	Вывод Pymystem с оберткой
Карл у Клары украл кораллы.	{'analysis':[{'lex':'**карл**', 'gr': 'S,имя,муж,од=им,ед'}], 'text': 'Карл'}, {'text': ' '}, {'analysis':[{'lex': '**у**', 'gr': 'PR='}], 'text': 'у'}, {'text': ' '}, {'analysis': [{'lex': '**клара**', 'gr': 'S,имя,жен,од=(род,ед|им,мн)'}], 'text': 'Клары'}, {'text': ' '}, {'analysis': [{'lex': '**украсть**', 'gr': 'V,сов,пе=прош,ед,изъяв,муж'}], 'text': 'украл'}, {'text': ' '}, {'analysis': [{'lex': '**коралл**', 'gr': 'S,муж,неод=(вин,мн|им,мн)'}], 'text': 'кораллы'}	**карл** {'S'} {'им,'} {'муж'} {'ед'} {'од'} **у** {'PR'} **клара** {'S'} {'им,'} {'жен'} {'мн'} {'од'} **украсть** {'V'} {'прош'} {'изъяв'} {'сов'} {'муж'} **коралл** {'S'} {'им,'} {'муж'} {'мн'} {'од'}

Заметим, что в таблице представлено строковое представление данных, выводимых в таком виде для удобства пользователя. В памяти же эта структура хранится в виде экземпляров классов.

Работа, представленная в статье, проделана в рамках исследования способов автоматического построения модели предметной области[8]. На этапе включения в алгоритм разбора текста морфологических характеристик, стало понятно, что их извлечение из результатов работы стандартного Pymystem представляется довольно сложным процессом. С помощью созданного дерева классов этот процесс укладывается в одну строку

программного кода. В дальнейшем, обёртка будет применяться для включения алгоритма поиска морфологических характеристик в текстовый анализатор, работа которого основывается на построении эталонного денотатного графа[9,10].

Литература

1. Cristina Cachero, Santiago Meliá, Jesús M. Hermida Impact of model notations on the productivity of domain modelling: an empirical study // Information and Software Technology. 2018. pp. 26-33.

2. Боярский К.К. Введение в компьютерную лингвистику. СПб.: ИТМО, 2013. С. 30-34.

3. Большакова Е.И., Воронцов К.В., Ефремова Н.Э., Клышинский Э.С., Лукашевич Н.В., Сапин А.С. Автоматическая обработка текстов на естественном языке и анализ данных. М.: НИУ ВШЭ, 2017. С. 48-56.

4. Сравнение и создание морфологических анализаторов в NLTK // Habr URL: habr.com/post/340404/ (дата обращения: 24.11.2018).

5. Бурякова Н.А., Чернов А.В. Классификация частично формализованных и формальных моделей и методов верификации программного обеспечения // Инженерный вестник Дона, 2010, №4. URL: ivdon.ru/magazine/archive/n4y2010/259

6. Смирнов И.В., Шелманов А.О. Семантико-синтаксический анализ естественных языков Часть I. Обзор методов синтаксического и семантического анализа текстов // Искусственный интеллект и принятие решений. 2013. С. 41-51.

7. Репозиторий relation_extractor // GitHub URL: github.com/eriskina/relation_extractor (дата обращения: 02.12.2018)

8. Elena Dolgova, E. V. Eriskina, Rustam Faizrakhmanov, E. A. Kasyanova, D. S. Kurushin, N. M. Nesterova, and O. V. Soboleva Automatic Test Generation

on the Basis of a Semantic Network // Digital Science. Budva: Springer, 2018. pp. 159-165.

9. Герте Н.А., Курушин Д.С., Нестерова Н.М., Соболева О.В. Экспериментальные исследования денотативной модели понимания в приложениях автоматического реферирования текста // Инженерный вестник Дона, 2015, №4. URL: ivdon.ru/ru/magazine/archive/n4y2015/3311

10. Новиков А.И., Нестерова Н.М. Реферативный перевод научно-технических текстов. М.: Академия наук СССР, Институт языкознания, 1991. С. 35-42.

References

1. Cristina Cachero, Santiago Meliá, Jesús M. Hermida Impact of model notations on the productivity of domain modelling: an empirical study Information and Software Technology. 2018. pp. 26-33.

2. Boyarskij K.K. Vvedenie v komp`yuternuyu lingvistiku. [Introduction to computational linguistics] SPb. ITMO, 2013. pp. 30-34.

3. Bol`shakova E.I., Voronczov K.V., Efremova N.E`, Kly`shinskij E`.S., Lukashevich N.V., Sapin A.S. Avtomaticheskaya obrabotka tekstov na estestvennom yazy`ke i analiz danny`x. [Automatic processing of natural language texts and data analysis]. M. NIUVShE`, 2017. pp. 48-56.

4. Sravnenie i sozdanie morfologicheskix analizatorov v NLTK. [Comparison and creation of morphological analyzers in the NLTK]. Habr URL: habr.com/post/340404/

5. Buryakova N.A., Chernov A.V. Inzenernyj vestnik Dona (Rus), 2010, №4. URL: ivdon.ru/magazine/archive/n4y2010/259

6. Smirnov I.V., Shelmanov A.O. Iskusstvenny`j intellekt i prinyatie reshenij. 2013. pp. 41-51.

7. Repozitiry relation_extractor GitHub URL: github.com /eriskina /relation_ extractor.

8. Elena Dolgova, E. V. Eriskina, Rustam Faizrakhmanov, E. A. Kasyanova, D. S. Kurushin, N. M. Nesterova, and O. V. Soboleva. Digital Science. Budva: Springer, 2018. pp. 159-165.

9. Gerte N.A., Kurushin D.S., Nesterova N.M., Soboleva O.V. Inzenernyj vestnik Dona (Rus), 2015. №4, URL: ivdon.ru/ru/magazine/archive/n4y2015/3311

10. Novikov A.I., Nesterova N.M. Referativny`j perevod nauchno-texnicheskix tekstov. [Abstract translation of scientific and technical texts] M. Akademiya nauk SSSR, Institut yazy`koznaniya, 1991. pp. 35-42.