Санкт-Петербургский государственный университет
Филологический факультет
Кафедра математической лингвистики

Москвина Анна Денисовна

ПРЕДСТАВЛЕНИЕ СИНТАКСИЧЕСКИХ СВЯЗЕЙ В ЛИНГВИСТИЧЕСКОМ ПРОЦЕССОРЕ NLTK4RUSSIAN

Магистерская диссертация

Направление «Лингвистика»,
Образовательная программа «Прикладная и экспериментальная лингвистика»,
профиль «Компьютерная лингвистика и интеллектуальные технологии»

Научный руководитель:
доц., к.ф.н. Митрофанова О.А.

Санкт-Петербург

2017

Оглавление

ВВЕДЕНИЕ

Задачи, связанные с автоматической обработкой данных, становятся все более и более востребованными в современном мире, и значительную часть этих данных представляют тексты, написанные на естественном языке, что является предметом изучения лингвистики. Одним из наиболее сложных этапов обработки текста, предшествующих семантическому анализу, является анализ синтаксический. **Целью** данного исследования является разработка синтаксического анализатора (парсера) для русского языка на основе ресурсов платформы NLTK и изучение его возможностей.

В соответствии с поставленной целью работы сформулированы следующие **задачи** исследования:

- проанализировать существующие теоретические подходы и инструменты для автоматического синтаксического анализа;
- разработать систему правил для выделения синтаксических групп;
- записать результаты в виде порождающей грамматики, работающей с NLTK;
- подключить к парсеру морфологический анализатор;
- провести оценку работы парсера и проанализировать встретившиеся ошибки и сложные случаи.

Предметом данного исследования являются синтаксические отношения в русском языке и способы их отражения в парсере, работающем на основе порождающей грамматики и групп составляющих. **Материалом** данного исследования стали представительные выборки русскоязычных предложений, клауз, словосочетаний, в которых реализуются основные синтаксические отношения, выбранные для тестирования разработанной грамматики и основанного на ней парсера.

Актуальность данного исследования связана с востребованностью синтаксического анализа во многих задачах современной компьютерной

лингвистики, таких как извлечение фактов, ключевых слов и словосочетаний, автоматического реферирования текстов и т.д. Несмотря на наличие множества разработок в данной области, существует лишь узкий круг инструментов, открытых для использования и пригодных без доработки для синтаксического анализа русскоязычных текстов. Платформа NLTK (Natural Language Toolkit) была выбрана как наиболее гибкая, содержащая основные алгоритмы для автоматической обработки текста, не требующая существенной перестройки для работы с русским языком, обладающая хорошей совместимостью и широким кругом инструментов для исследования и работы с естественным языком. В данной работе мы предлагаем формальную грамматику для русского языка и парсер на основе этой грамматики, созданный на открытой платформе, допускающий расширение и применимый в большом числе задач.

Теоретическая значимость данного исследования заключается в разработке набора правил, основывающихся на теоретических описаниях синтаксиса русского языка и на существующей практике автоматического синтаксического анализа. Правила, выработанные на основе теории, были переведены на язык формальной порождающей грамматики в формате, применяющимся в NLTK. Данные правила легли в основу функционирующего парсера, что обусловливает **практическую значимость** работы. Данная система правил, а именно выделение групп составляющих, была использована при оптимизации алгоритма извлечении ключевых выражений RAKE для работы с русским языком. Получившийся инструмент был протестирован на четырех русскоязычных корпусах текстов.

Апробация исследования: основное содержание диссертационного проекта представлено в двух публикациях [Москвина и др. 2016, 2017] и обсуждалось в докладах на конференциях.

ГЛАВА 1. АВТОМАТИЧЕСКЙ СИНТАКСИЧЕСКИЙ АНАЛИЗ

1.1 Синтаксический анализ в задачах автоматической обработки текстов

Словоформы, взаимодействуя в языке, всегда находятся в неких синтаксических отношениях, «организующих предложение в целостную единицу сообщения» [Шведова 1980]. Синтаксический анализ – это процедура, соотносящая предложения на естественном языке с некоторой системой правил их построения, или грамматикой, и позволяющая определить их структуру и взаимоотношения между единицами более низких уровней. Анализ синтаксических структур в компьютерной лингвистике в той или иной форме является неотъемлемой составляющей извлечения практически любой информации из текстов на естественном языке; синтаксическая информация может и должна применяться в задачах автоматического извлечения фактов, ключевых слов, построения вопросно-ответных систем, семантического анализа текста, машинного перевода и так далее. Синтаксические анализаторы, или парсеры, позволяют проводить данную процедуру автоматически. В зависимости от конкретной задачи может выполняться как глубокий, так и поверхностный синтаксический анализ. Как и в других задачах автоматической обработки текста, существует два основных подхода к синтаксическому анализу: подход, основанный на правилах и статистический, основанный на машинном обучении.

Исследования в данной области имеют уже достаточно долгую историю и традицию. В первую очередь, синтаксические анализаторы, или парсеры, работающие с естественным языком, разрабатывались для английского языка. Среди них одним из самых значимых является Стэндфордский парсер (Standford parser) [de Marneffe et al. 2006]. Некоторые из них, первоначально разработанные для английского языка, были адаптированы для работы с русским, как, например, MaltParser [Nivre et al. 2007]. Конечно, существуют и оригинальные отечественные разработки. Лингвистический процессор ЭТАП-3 [Iomdin 2012], основывается на модели

«Смысл <=> Текст». Одной из самых известных разработок, находящихся в открытом доступе, является модуль синтаксического анализа проекта АОТ (Диалинг). Link Grammar Parser, основанный на теории грамматики связей [Sleator, Temperley 1991] и разработанный С.В. Протасовым [Протасов 2005], представляет собой совершенно другой подход к формальному описанию синтаксических связей. Несмотря на наличие разнообразных наработок в данной области, в исследованиях долго не существовало общего подхода и выработанной традиции синтаксического описания. В 2012 году в рамках конференции «Диалог» было проведено соревнование синтаксических парсеров [Толдова и др. 2012], в котором приняло участие 8 команд [Antonova, Misyurev 2012]. Помимо освещения деятельности исследователей и разработчиков в данной среде, соревнование было призвано сделать шаг в направлении к выработке общего унифицированного стандарта синтаксического описания для русского языка. Благодаря соревнованию был создан так называемый «золотой стандарт» синтаксической разметки, представляющий собой 800 предложений с описанием допустимых расхождений.

Сегодня многие компании и коммерческие проекты, работающие в сфере компьютерной лингвистики, такие как, например, ABBYY [Anisimovich 2012], Dictum, RCO, разрабатывают свои собственные синтаксические анализаторы и пользуются ими для своих задач [Москвина и др. 2016]. Это делает особо острым вопрос о создании открытого, некоммерческого и совместимого с разнообразными лингвистическими инструментами парсера для русского языка.

1.2 Формальное представление синтаксической информации

Обычно синтаксическая структура легко представляется в виде дерева, узлами которого являются словоформы, а дуги выражают связь или зависимость между этими словами. Существует два основных подхода к

формальному описанию синтаксических структур – грамматика зависимостей и грамматика непосредственных составляющих. Грамматика непосредственных составляющих предлагает объединять слова в группы, а эти группы в более крупные группы следующего уровня, и так до предложения. Грамматика зависимостей предполагает определение для каждого слова его вершины, т.е. того, от какого слова оно зависит, и тип данной связи. Рассмотрим несколько подробнее преимущества и недостатки данных подходов при автоматической обработке текста.

1.2.1 Грамматика зависимостей

Формализм грамматики зависимостей предполагает, что синтаксическая структура состоит целиком и полностью из отношений зависимости. Зависимости понимаются как это ассиметричные бинарные отношения между двумя лексическими единицами. Каждое отношение зависимости может иметь свое название. Отношение зависимости между вершиной (H) и зависимым (D) в конструкции C существует при следующих условиях [Nivre 2006]:

- вершина H определяет синтаксическую категорию конструкции C;
- вершина H определяет семантическую категорию конструкции C, D – спецификацию;
- наличие вершины H является необходимым условием, наличие зависимого – необязательно;
- вершина определяет D и указывает, обязательно ли его наличие;
- форма D зависит от его вершины (согласованность);
- линейная позиция D зависит от его вершины.

Существуют разные типы зависимостей. В современных теориях часто выделяются следующие виды отношений зависимости [Nivre 2006].

- Отношение head-argument (экзоцентрическое): *собирать грибы*. Аргументы могут быть обязательны, но не более одного. Главное слово не всегда может заменить всю конструкцию.

- Отношение head-modifier (эндоцентрическое) в сочетаниях типа *вкусное яблоко.* Наличие определителей опционально, их может быть несколько. Вершина может заменять всю конструкцию.
- Отношение head-specifier (преобразование Теньера). Это отношение между такими функциональными словами, как предлоги, определители и их аргументами. Здесь синтаксическая вершина не совпадает с семантической. Так, в сочетании *the rat* стрелка зависимости будет идти от *the* к *rat*.
- Отношение координации. Такие отношения возникают в случае, когда остается неясным, какое слово является вершиной, например, в сочетании земля и небо.

Мельчук говорил о наличии трех видов зависимостей между словоформами в предложении: морфологической, синтаксической и семантической [Мельчук 1999].

Таким образом, граф представляет собой дерево. Деревья зависимостей не обязательно отражают порядок слов в предложении. Дерево зависимостей является проективным, если в нем нет пересекающихся связей (см. рис.1). Пример дерева зависимостей из [Nivre 2007]:

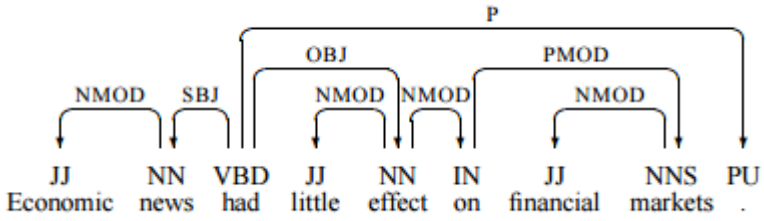

Рис. 1. Дерево зависимостей в PennTreebank

1.2.2 Грамматика составляющих

Альтернативным способом представления синтаксических структур является грамматика составляющих. Структура составляющих представляет собой в размеченном виде информацию об иерархической организации предложения. В отличие от структуры зависимостей, где описываются только синтаксические отношения зависимостей, в размеченной структуре

составляющих обозначаются все непосредственно синтаксические единицы – от отдельной словоформы до предложения, включая промежуточные группы. Правила применяются не к словам и поддеревьям их зависимостей, а к терминальным и фразовым категориям. Использование фразовых категорий отражает такое свойство синтаксических структур как рекурсивность.

Применение составляющих впервые было предложено представителями американской дескриптивной лингвистики и было разработано Л. Блумфилдом [Bloomfield 1933]. Критерии выделения опирались на фонологические, морфологические и синтаксические свойства.

Предложение в теории непосредственно составляющих рассматривается как некоторая линейная структура, в которой более крупные единицы, начиная с самого предложения, могут быть последовательно разложены на группы меньшего размера, и так до минимальной категории – то есть словоформы. Анализ может вестись в обе стороны – как от разложения более крупных единиц на более маленькие, то есть сверху вниз, так и снизу вверх, то есть путем объединения словоформ в между собой в группы, этих групп в группы следующего уровня и так до группы размером с предложение. На рисунке (рис. 2) можно увидеть структуру составляющих предложения, представленную в виде дерева. Пример взят из [Nivre 2007].

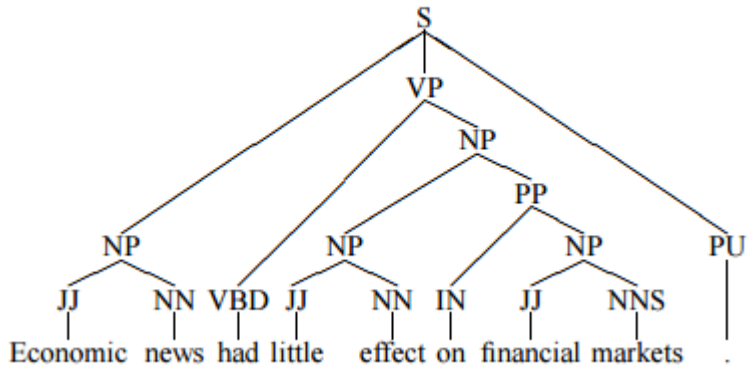

Рис 2. Структура составляющих из Penn Treebank

1.2.3 Сравнение грамматики зависимостей и грамматики составляющих

Как метод структуры зависимостей, так и метод непосредственных составляющих имеют свои преимущества и недостатки. В западной науке более традиционным долгое время оставался метод непосредственных составляющих, хорошо работающий для языков с бедной морфологией. Однако, на сегодняшний день более перспективным, в особенности для русского языка - как языка со свободным порядком слов - представляется описание синтаксических структур через зависимости.

В связи с развитием метода зависимостей, и, с другой стороны, наличием инструментов, работающих только со структурами составляющих, а также банков деревьев в виде структур составляющих, естественным образом возникает вопрос о трансформации одного представления в другое. Теоретические основания данного перехода были предложены в исследованиях А.В.Гладкого [Гладкий 1985], С.Я.Фитиалов [Фитиалов 1984], В.Д.Буторова [Буторов 1996] и других. Практическая работа была проведена командой, разрабатывающей Стэндфордский парсер. В их публикации [de Marneffe et al. 2006] описывается система, извлекающая типизированные зависимости из разборов предложений, выполненных в виде структур составляющих. Это возможно, поскольку существуют формальный изоморфизм между определенными структурами, как между грамматикой зависимостей и структурой составляющих на нижнем уровне. В работе использовалось 48 грамматических отношений, имеющих некую иерархическую структуру. Эти зависимости, наделенные типами, применялись с помощью применения некоторого набора правил (моделей) к деревьям структуры зависимостей. Процедура состояла из двух этапов. Сначала предложение разбирается парсером, работающим со структурами составляющих. Использовался Стэндфордский парсер – статистический парсер, основанный на структурах составляющих и обученный на Penn Wall Street Journal Treebank. Затем в каждой составляющей определялась вершина. Авторы предпочли использовать в качестве вершин содержательные слова,

так что извлекаемая вершина являлась в большей степени семантической, чем синтаксической. Так, в приводимом примере flat structure из Penn Treebank: (*NP the new phone book and tour guide*) выделяется две семантические вершины – *book* и *guide*, и получаются правильные зависимости:

- *nn* (book, phone)
- *nn* (guide, tour)
- *CC_and* (book, guide)
- *amod* (book, new)
- *det* (book, the)

На втором шаге каждая из определенных зависимостей получает наиболее точный тип (label). Для каждого грамматического отношения авторы определили один или более соответствующий шаблон поддерева структуры составляющих. Таким образом, каждая словоформа ставится в зависимость чему-то одному, и количество типизированных зависимостей в виде структуры соответствует количеству слов в предложении. Сложности в оценке при сравнении с результатами других парсеров, работающих с зависимостями были обусловлены, во-первых, различиями в структуре зависимости (то есть в том, конкретно между какими словами возникает отношение зависимости), во-вторых, в выборе типа зависимости (выборе грамматического отношения для конкретной зависимости). Полученная в результате оценки точность – 80.3%, однако, этот результат может рассматриваться как грубая предварительная оценка, поскольку оценка была проведена на маленькой выборке.

1.2.4 Формальные грамматики в синтаксическом анализе

Большую роль в прикладной лингвистике, и, в частности, в синтаксическом анализе, сыграло появление формальных грамматик. Формальная грамматика задает и ограничивает язык, определяя регулярное поведение его единиц. Теория сформулирована, например, в [Aho, Ullman

1972]. Утверждается, что формальный язык L есть множество конечных цепочек (или предложений), построенных путем объединения элементов из алфавита (конечного набора символов, словаря) по правилам грамматики G, заданной для данного языка. Формальная (порождающая) грамматика G задается четверкой {Vt, Vn, R, S}, где

Vt – алфавит терминальных символов,

Vn – алфавит нетерминальных символов,

$Vt \cap Vn = \varnothing$ (алфавиты Vt и Vn не пересекаются) и $Vt \cup Vn = V$,

R – множество правил; элемент (α, β) множества R есть правило вывода и записывается в виде $\alpha \rightarrow \beta$. Общий вид правил таков: $\gamma\alpha\omega \rightarrow \gamma\beta\omega$, где α, β, γ, ω – цепочки на алфавите V, которые могут быть пустыми (#) за исключением α.

S – начальный символ (цель) грамматики ($S \in Vn$).

Формальная грамматика задает потенциально бесконечный набор всех синтаксически верных сочетаний и предложений. Грамматики подразделяются на контекстно-свободные, вероятностные, лексикализованные и контекстно-зависимыми. Имея набор синтаксических категорий и ограниченный набор правил вывода, контекстно-свободная грамматика указывает как фраза категории A может быть представлена в виде последовательности более маленьких частей $\alpha 1 \ldots \alpha n$ [Bird et al, 2009].

Контекстно-свободные грамматики описывают грамматическую структуру большинства языков программирования, однако они активно применялись в задачах анализа многих западноевропейских языков, и хорошо зарекомендовали себя в описании, например, английского языка.

Автоматическая реализация разбора предложения на основе заданной контекстно-свободной грамматики реализуется при помощи алгоритма Эрли или алгоритма Кока — Янгера — Касами.

1.3 Обзор существующих инструментов и ресурсов для синтаксического анализа

В первом разделе, говоря о современном положении дел в синтаксическом анализе, мы упомянули несколько проектов. Рассмотрим некоторые из существующих разработок подробнее.

В 1990е годы начали разрабатывать парсеры, работающие на статистической информации. Статистические парсеры извлекают данные о языке из вручную размеченных коллекций текстов, на основании которых они создают наиболее вероятную структуру разбора того или иного предложения. Статистический подход в лингвистике работает достаточно хорошо, однако никогда не может достигнуть идеальной точности. К статистическим парсерам относится, например, Стэндфордский парсер, разработанный командой The Stanford Natural Language Processing Group и продолжающий получать регулярные обновления. Парсер был разработан для английского языка, однако был позже адаптирован для немецкого и арабского, и был также адаптирован и использован для некоторых других языков.

MaltParser – парсер, разработанный Johan Hall, Jens Nilsson и Joakim Nivre. Это система, основанная на индуктивном машинном обучении и работающая с деревьями зависимостей [Nivre, 2006]. Используется размеченный корпус для построения модели, а затем, с использованием этой модели, парсер может строить деревья для новых данных. Система может работать с самыми разными языками, однако требует оптимизации для конкретного языка. Существует версия для русского языка, обученная на корпусе синтаксических структур СинТагРус. MaltParser релизован на Java и доступен для свободного использования.

Link Grammar Parser – синтаксический анализатор для русского языка, базирующийся на теории грамматики связей. Разработан С.В. Протасовым Парсер предлагает необычный подход к описанию синтаксических отношений, представляя их ни в виде структур составляющих, ни в виде

деревьев зависимостей. На слова, представляющие собой терминальные элементы грамматики, накладываются ограничения или «требования по связям» [Протасов 2005]. Этими связями должны соединяться все слова в предложении, причем должны соблюдаться условия непроективности и связности, а требования соблюдены для всех слов. Каждое слово обладает так называемым коннектором, и может соединяться только с подходящим ему другим коннектором. Разбор фразы *особенно боятся смерти от огня,* произведенный Link Grammar Parser представлен на рис.3.

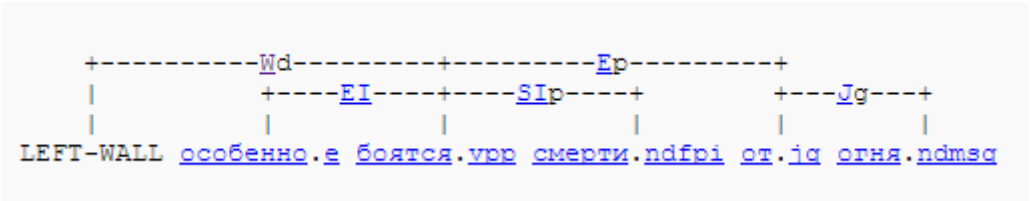

Рис. 3. Разбор предложения Link Grammar Parser

Синтаксический анализатор в проекте ДИАЛИНГ (АОТ) [http://aot.ru/] строит синтаксические группы для одной на клаузы на одном варианте морфологического разбора [http://aot.ru/docs/synan.html]. Группы объединяются на основе правил в предложении, разбитом на слова (юниты). Каждое правило содержит в себе следующие параметры:

- последовательность слов (групп), которые объединяются в новую,
- то, при каком условии это выполняется,
- какая группа является главной, её тип,
- тип получившейся синтаксической группы,
- граммемы этой группы.

Например, правило для построения группы однородных наречий, соединяет два наречия при условии наличия между ними союза или знака препинания. Пример разбора на из демо-версии анализатора на сайте (см. рис.4)

Рис. 4. Разбор синтаксического анализатора АОТ

Лингвистические данные АОТ находятся в свободном доступе и могут использоваться для любых проектов.

Одной из важнейших разработок в изучаемой области является синтаксический анализатор ЭТАП-3. Парсер был разработан группой ученых Института проблем передачи информации им. А.А. Харевича (ИППИ РАН). Он опирается на лингвистическую теорию «Смысл <-> Текст» и используется в системе разметки синтаксически аннотированного корпуса русских текстов СинТагРус (объём более 860 тыс. словоупотреблений). Синтаксический анализ в системе ЭТАП происходит следующим образом. С помощью специальных синтаксических правил (или синтагм) строится синтаксическая структура фразы в виде дерева зависимостей [Дружкин, Цинман 2008]. Все правила являются бинарными. Сначала строится граф гипотетических синтаксических связей (гипотез), затем используются различные средства для фильтрации лишних связей и остается одно дерево синтаксической структуры предложения. Примерами синтаксических групп, являются, например, предложная группа: *пошел → (В → лес)*, или элективная группа: пришел → *(ОДИН → из → мальчиков)*. Также используется больше шестидесяти видов синтаксических отношений (СинтО): сочинительные, атрибутивные, актантные, служебные и другие. В процессе разработки внимание акцентировалось на двух противоположных задачах. С одной стороны, авторов интересовало создание корпуса синтагм, которое бы наиболее полно отражали синтаксические явления естественного языка. С другой стороны, необходимо было также сформулировать и воплотить

развитую системы фильтровых средств, для выделения из графа гипотетических связей необходимую правильную структуру.

Стоит выделить также синтаксический анализатор Treevial, входящий в систему морфосинтаксического анализа Treeton. Принципы его работы описываются в статье [Мальковский, Старостин 2007]. В парсере используется математическая модель, позволяющая создавать синтаксические правила и формировать штрафные функции. На используемом языке правила могут выражаться как в терминах грамматики зависимостей, так и в терминах составляющих. Синтаксическая структура понимается как направленное размеченное дерево. С каждым узлом дерева соотносится пара атрибут-значение, например, так (POS=N,CAS=nom). Некоторые узлы соответствуют синтаксическим группам, некоторые морфологическим интерпретациям словоформ. Из первых исходит так называемая w-связь, из вторых нет. Особенностью, отличающей этот парсер от остальных, авторы называют его мультиагентную архитектуру: «разнофункциональные агенты независимо друг от друга работают над частями входного предложения, имея при этом возможность видеть и использовать результаты работы друг друга» [Мальковский, Старостин 2007]. Благодаря наличию шрафных функций, струтуры в процессе анализа получают оценку, сравниваются друг с другом и ранжируются.

Очень важным ресурсом являются синтаксически аннотированные корпусы текстов. Для английского языка важнейшим является банк синтаксических структур Penn treebank. Хотя он был создан уже давно, огромное количество данных, представление которых реализовано в этом проекте, является незаменимым источником для компьютерной лингвистики. [Taylor 2003].

Для русского языка первым синтаксически размеченным корпусом стал СинТагРус [http://www.ruscorpora.ru/instruction-syntax.html]. Объем корпуса составляет около 900 тыс. словоупотреблений. Корпус разнороден по составу, в него входят тексты из современной русской прозы, научно-

популярные и общественно-политические статьи последних нескольких десятилетий, новостные тексты из сети Интернет. Процесс создания корпуса описывается в статье [Апресян и др. 2005]. СинТагРус строился в полуавтоматическом режиме, проходя этапы морфологического и синтаксического анализа. В его создании применялись компоненты системы ЭТАП-3, такие как автоматический морфологический словарь русского языка, насчитывающий более 100 тысяч лексических единиц, комбинаторный словарь, содержащий приблизительно 900 000 лексических единиц, и грамматика русского языка, представленная в виде синтагм. После автоматического этапа проводилась экспертная проверка и корректирование неверных разборов редакторами-лингвистами. Деревья синтаксических разборов в СинТагРусе представлены в виде структур зависимостей, в узлах (в прямоугольниках) находятся лексемы, а стрелки помечены типами отношений (см. рис. 5).

Рис. 5. Разбор предложение из СинТагРуса

1.5 Входные данные для синтаксического анализа. Предобработка текста

Говоря о синтаксическом разборе, мы обычно имеем в виду некоторое формализованное представление структуры предложения или словосочетания. Обсуждаемые сложности, их устранение, и применяемые формализмы обычно касаются вопросов, решаемых в рамках уже выделенного заранее предложения (клаузы, словосочетания). Однако в реальных прикладных задачах обычно входным материалом является

некоторый текст, или их коллекция, представляющая собой множество формально ещё не разграниченных предложений.

Таким образом, самой первой задачей автоматического синтаксического анализа любого языка является определение границ предложений, то есть задача сегментации.

1.6 Токенизация

Даже на уровне разбиения текста на предложения могут возникать сложности. Знаки препинания, которые могут обозначать конец предложения, не могут трактоваться однозначно. Точка может обозначать некоторую аббревиатуру или быть частью дробного числа. Или предложения, завершаемые точной, вопросительным или восклицательным знаком могут, в свою очередь, являться частью прямой речи и входить в состав более крупного, но единого предложения.

С другой стороны, синтаксические отношения всегда связаны с отношениями между отдельными словами и их сочетаниями, и в этом смысле минимальной единицей, с которой работает синтаксис, является слово. Таким образом, предложение нужно разбить на слова. Этап выделения слов из текста называется токенизацией. На уровне токенизации тоже могут возникать препятствия. Самый простой способ использует в качестве разграничителей пробелы, знаки переноса строки и табуляции и считает все, находящееся между ними, токенами. Для эффективности последующего анализа уже на уровне токенизации токенам может приписываться некоторый тип – обычное слово, аббревиатура, число, имя собственное и т.п. Для точности будущего синтаксического анализа так же полезным является выделение некоторых более крупных конструкций как именованные сущности, multi-word expressions. Для решения этой задачи могут использоваться правила на основе регулярных выражения, словари имен собственных и т.д.

1.7 Морфологический анализ и снятие морфологической неоднозначности

Для того, чтобы установить наличие отношения между словами, необходимо обладать некоторой информацией о форме этих слов. Таким образом, следующим шагом, предваряющим синтаксический анализ, является анализ морфологический. Каждой словоформе приписывается часть речи и значения её морфологических категорий. Важной задачей, решающейся на этом этапе, является снятие морфологической неоднозначности, в первую очередь связанной с определением части речи рассматриваемой словоформы. Существует два основных подхода к определению части речи: основанный на правилах и вероятностный.

Теггеры, работающие с вероятностными методами, определяют вероятности для частеречных тегов на основе информации, полученной из обучающего корпуса. Например, могут использоваться метод скрытых марковских моделей или метод максимальной энтропии (Standford POS tagger).

Есть также гибридный метод, разрешающий морфологическую информацию на основе правил, выводимых из текста при помощи управляемого или неуправляемого обучения, разработанный Э. Бриллом [Brill 1995].

Высокая точность в снятии морфологической неоднозначности обеспечивается только при учете контекста словоформы и ее синтаксических отношений. Таким образом, этап морфологического анализа является не только частью предобработки текста, но и должен производиться параллельно с синтаксическим анализом [Kakkonen 2007].

Морфологический анализ, выходящий за рамки определения части речи, выполняется с помощью морфологических анализаторов, которые, в свою очередь, основываются на морфологических словарях. В свою очередь большинство компьютерных морфологических словарей для русского языка опирается на Грамматический словарь А.А. Зализняка. Сегодня существует ряд русскоязычных морфологических анализаторов, среди которых можно

выделить AOT (http://aot.ru), Mystem (http://company.yandex.ru/technologies/mystem), PyMorphy (http://pymorphy2.readthedocs.org/en/latest) и другие.

Обычно морфологические анализаторы, применяемые для синтаксических парсеров, находят все возможные формы слова и уже на уровне синтаксического анализа из них выбирается подходящая.

Таким образом, начиная работать именно над синтаксическим анализом, мы должны обладать набором определенных данных о тех единицах, которые собираемся анализировать: границы предложений, границы словоформ, их возможные морфологические интерпретации.

1.6 Модуль синтаксического анализа в NLTK

NLTK (Natural Language Toolkit) – это специализированная платформа для разработки программ на языке Python для работы и исследования текстов на естественном языке. В этой среде существует возможность работать с десятками готовых корпусов и такими ресурсами как WordNet. Так же внутри среды существует набор готовых библиотек для решения задач классификации, токенизации, кластеризации, машинного обучения. Инструменты NLTK позволяют осуществлять полный цикл автоматической обработки текста: от графематического анализа и токенизации до синтаксического анализа и логической семантики. В NLTK включены не только готовые инструменты для анализа текстов, но и корпусы и алгоритмы, позволяющие решать задачи автоматической обработки текста на основе машинного обучения.

1.6.1 Проект NLTK4RUSSIAN

В задачи проекта NLTK4RUSSIAN входит использование современных ресурсов и инструментов для автоматической обработки текста. В рамках проекта был разработан гибридный морфологический анализатор для русского языка на основе NLTK и морфологического анализатора PyMorphy

[Паничева и др. 2015]. Следующей задачей стала разработка синтаксического анализатора с использованием тех же инструментов.

1.6.2 Синтаксический анализ в NLTK

NLTK позволяет проводить синтаксический анализ на основе формальных грамматик [Bird et al. 2009]. Исследователям предоставляется возможность самим создавать такие грамматики и, пользуясь встроенными инструментами, получать синтаксический разбор.

Среди ресурсов, предоставляемых NLTK, присутствуют не только инструменты для обработки текста, но и лингвистические данные в виде корпусов, которые можно использовать, в том числе, для машинного обучения. Так, в NLTK представлено 10% из банка синтаксических деревьев The Penn Treebank Project – универсального корпуса для тестирования инструментов, работающих с синтаксисом английского языка. Помимо этого, в NLTK есть корпусы для английского, китайского и арабского языков Ontonotes 5.0 и некоторые другие материалы.

Как уже было описано выше, с помощью конечного набора правил (грамматики) можно описать многочисленные и потенциально бесконечные конструкции некоторого естественного языка. Теория генеративных грамматик рассматривает язык как очень большой набор всех грамматически правильных предложений, а грамматику как формальное описание того, каким образом эти предложения могут быть построены. Грамматики используют рекурсивные правила (или продукции). Модуль синтаксического анализа в NLTK позволяет работать с несколькими типами грамматик: контекствно-свободными (CFG), вероятностно контекстно-свободными (PCFG), лексикализованными и контекстно-зависимыми. Контекстно-свободная грамматика работает с правилами, состоящими из нетерминальных элементов, таких как именная (NP) или глагольная группа (VP), предтерминальными элементами, как глагол (V) или существительное

(N), и терминальными элементами – лексикой естественного языка ("man", "the", "cat").

Пример простейшей контекстно-свободной грамматики:

```
S -> NP VP
PP -> P NP
NP -> Det N
VP -> V PP
Det -> 'the' | 'my'
N -> 'legs' | 'stairs'
V -> 'run'
P -> 'up'
```

Определив такую грамматику, мы можем воспользоваться парсером в NLTK (Chart parser) для предложения *my legs run up the stairs* и получить следующий разбор:

```
S
  (NP (Det my) (N legs))
  (VP (V run) (PP (P up) (NP (Det the) (N stairs)))))
```

Эту структуру можно представить так же в виде дерева.

Вероятностная контекстно-свободная грамматика позволяет добавлять правилам вес, и таким образом оценивать вероятность каждого получившегося разбора. Это выглядит так:

```
VP -> V NP [0.4]
VP  -> V NP PP [0.3]
```

Категориальная грамматика (feature-based grammar) позволяет вводить для каждой категории систему признаков и работать с ними.

Рассмотренные грамматики хранятся в файлах определенного формата, в зависимости от своего типа, и к ним можно обращаться, работая в NLTK.

NLTK предлагает разработчикам самостоятельно создавать подобные грамматики для любых естественных языков и применять их в разных задачах автоматической обработки текстов.

Для того, чтобы анализировать структуры естественного языка с помощью таких грамматик в NLTK встроено несколько парсеров,

реализующих разные подходы. Во-первых, это парсер, использующий метод рекурсивного спуска (Recursive-Descent Parser). Этот парсер использует разбирает предложение сверху вниз («top-down»). Правила из грамматики рассматриваются по порядку для данной левой части, и им находятся в соответствия правила с подходящей правой частью. Если получается дойти до терминальных символов, разбор продолжается дальше. Если прийти к терминальным элементам не удалось, сработавшие правила возвращаются назад и процесс повторяется дальше. При таком подходе много времени уходит на неправильные разборы, которые не пригодятся в будущем.

Парсер, называющийся Shift-Reduce Parser работает в обратном направлении. Он обрабатывает по очереди слова в предложении и находит их соответствия в правых частях правил. После нахождения подходящего разбора, продолжается движение дальше по тексту. Такой парсер предоставляет не более одного разбора и может не получить («потерять») правильный.

Метод, называющийся chart parser, находит и сохраняет частичные разборы для фрагментов предложения, затем соотносит их друг с другом и соединяет. Одной из реализаций такого подхода в NLTK является парсер, основанный на алгоритме Витерби.

ГЛАВА 2. АРХИТЕКТУРА РАЗРАБАТЫВАЕМОГО СИНТАКСИЧЕСКОГО АНАЛИЗАТОРА И ОЦЕНКА ЕГО РАБОТЫ

2.1 Особенности используемой категориальной грамматики

В рамках данной работы, из вариантов формальных грамматик, с которыми работает NLTK, мы выбрали такой вид грамматики как грамматика, основанная на признаках категорий (feature-based grammar). Данный тип грамматики позволяет использовать в правилах, помимо основных типов категорий, значения их признаков. То есть для каждой категории сначала выделяется ряд признаков. Так, для именной группы это могут быть признаки рода, числа и падежа. Управляя значениями этих признаков, мы можем контролировать в процесс объединения в группу словоформ с различными морфологическими характеристиками, а также явно указывать, какие признаки унаследует получившаяся в результате объединения группа. В некотором смысле это соотносится с понятием вершины в терминах грамматики зависимостей. Данная грамматика показалась нам наиболее подходящий для описания синтаксического строения русского языка с его богатой морфологией при помощи структур составляющих. С категориальными грамматиками, основанными на признаках категорий, в NLTK работает алгоритм Эрли (Early-chart parser), который находит фрагменты дерева предложения, а затем объединяет их в группы.

Рассмотрим строение типичного правила на примере объединения двух именных групп в генитивную именную группу:

NP[+gent, C=?c, G=?g, NUM=?n] -> NP[C=?c, G=?g, NUM=?n] NP[C=gent]

В левой части правила (или, иначе, грамматической продукции) указывается получающаяся в итоге объединения группа, в правой части – её составляющие, следующие друг за другом в линейном порядке. В данном случае мы работаем с объединением двух именных групп (NP – сокращенно

от noun phrase) в другую именную группу более высокого уровня. C, G, и NUM – признаки категории NP, падеж, род и число соответственно. С помощью перечисления категорий с вопросительным знаком в левой и правой части правила мы задаем наследование этих признаков из правой части правила в левую. С помощью записи C=gent мы фиксируем значение признака (в данном случае падежа). То есть правило объединяет некоторую именную группу и следующую за ней именную группу в родительном падеже и только в нем в группу, которую принято называть генитивной. В структуре правила мы также можем использовать некоторый бинарный параметр, который может иметь значения + или – и позволять нам хранить информацию о подкатегории. В данном случае, с помощью записи +gent мы сохранили информацию о присоединении генитива, хотя получившаяся в результате продукции категория является именной группой NP и может участвовать в этом статусе в других правилах, таким образом обеспечивается необходимая рекурсия. Рассмотренное правило соберет в группы такие словосочетания как *ручка двери, осенние листья деревьев, улицы старого города* и тп.

Введение такого параметра позволяет также накладывать ограничение на применение правила, а именно предупреждать его лишнее рекурсивное срабатывание. Например, если к глагольной группе присоединяется прямое дополнение – объект, то она, в рамках категориальной системы, очевидно, остается глагольной группой, и, следовательно, может снова присоединить к себе объект по тому же самому правилу. Однако, мы понимаем, что глагол может присоединять только одно прямое дополнение. Тогда мы можем запретить второе срабатывание правила на той же глагольной группе следующим образом:

VP[+objt] -> VP[-objt] NP[CASE=accs]

Тогда при разборе, например, предложения *я рисую клюв орла*, группа *рисую клюв*, объединившись, получит значение параметра objt «+», или истина, и уже не сможет присоединить к себе именную группу *орла,* следующую за

ней линейно по тому же правилу, поскольку в правой части правила параметр objt имеет значение «-», или ложь. Таким образом, мы получим только правильный вариант разбора, при котором сначала будет собираться группа *клюв орла*, а уже потом она присоединится к глаголу. Мы использовали аналогичный ход, например, в правиле присоединения существительного к компаративу или сравнительной степени прилагательного. Оно выглядит следующим образом:

COMP[+comp, +noun,] -> COMP[-comp] NP[CASE=gent]

Возможность присоединить только одну именную группу в генитиве позволяет исключить неправильное группирование составляющих в таких случаях как, например, *быстрее скорости света*.

Таким образом, задание параметра с бинарным значением в нашей грамматике выполняет две функции: сохранение информации о подкатегории, которая позволяет минимизировать количество используемых категорий, и при этом отличать, к примеру, именные группы с прилагательным от именных групп с генитивом, и наложение ограничений на срабатывание правила, что позволяет отбросить часть разборов как неверные.

Важность первой функции заключается в том числе в том, чтобы наглядно видеть, в каком порядке применялись правила при разборе. Сравним две структуры для фразы *полная тарелка супа*:

```
(XP[]
  (NP[C='nomn', G='femn', NUM='sing', +gent]
    (NP[C='nomn', G='femn', NUM='sing', +adjf]
      (AdjP[C='nomn', G='femn', NUM='sing']
        (ADJF[C='nomn',      G='femn',      NF='полный',
NUM='sing', PER=3]
          полная))
      (NP[C='nomn', G='femn', NUM='sing', PER=3]
        (NOUN[C='nomn',     G='femn',     NF='тарелка',
NUM='sing', PER=3]
          тарелка)))
    (NP[C='gent', G='masc', NUM='sing', PER=3]
      (NOUN[C='gent',  G='masc',  NF='суп',  NUM='sing',
PER=3] супа))))
```

В данном случае, мы наглядно видим, благодаря параметру +gent, что присоединение генитива происходит на более верхнем уровне, то есть сначала складывается именная группа *полная тарелка*, а уже потом к ней присоединяется генитив. Или наборот:

```
(XP[]
  (NP[C='nomn', G='femn', NUM='sing', +adjf]
    (AdjP[C='nomn', G='femn', NUM='sing']
      (ADJF[C='nomn',       G='femn',       NF='полный',
NUM='sing', PER=3]
        полная))
    (NP[C='nomn', G='femn', NUM='sing', +gent]
      (NP[C='nomn', G='femn', NUM='sing', PER=3]
        (NOUN[C='nomn',     G='femn',     NF='тарелка',
NUM='sing', PER=3]
          тарелка))
      (NP[C='gent', G='masc', NUM='sing', PER=3]
        (NOUN[C='gent', G='masc', NF='суп', NUM='sing',
PER=3] супа)))))
```

1. 2.2 Правила выделения синтаксических групп

2.2.1 Принципы организации системы правил

При разработке грамматики мы старались минимизировать количество используемых категорий. Правила можно разделить на несколько блоков. Самый первый блок правил является скорее техническим и описывает то, какие фрагменты парсер может считать конечными, то есть какие типы групп он может предоставлять в виде результата. Они выглядят, например, так:

% start XP

XP -> NP

XP -> VP

XP -> AdjP

XP -> AdvP

XP -> COMP

XP -> PP

Так представлена информация о том, что текст, подаваемый парсера для разбора, может быть именной группой, глагольной группой, группой

прилагательного, причастия, компаративом, предложной группой, предложением и так далее.

Ещё один блок правил представляет собой некоторые тривиальные утверждения, важные при работе с категориальной грамматикой, и констатирующие то, что отдельное существительное представляет собой простейшую именную группу, глагол – глагольную, прилагательное – группу прилагательного и так далее. Они выглядят так:

NP[C=?c, G=?g, NUM=?n, PER=?p] -> NOUN[C=?c, G=?g, NUM=?n, PER=?p]

VP[TR=?tr, TENSE=?t, G=?g, NUM=?n, PER=?p] -> VERB[TR=?tr, TENSE=?t, G=?g, NUM=?n, PER=?p]

AdjP[C=?c, G=?g, NUM=?n] -> ADJF[C=?c, G=?g, NUM=?n]

PrtfP[C=?c, G=?g, NUM=?n] -> PRTF[C=?c, G=?g, NUM=?n]

Правила верхнего уровня собирают свои составляющие в клаузы или предложения. Среди правила более низкого уровня наиболее крупными блоками являются правила объединения в именную группу и правила объединения в глагольную группу. Что касается выделения словосочетаний, в Академической грамматике [Шведова 1980] выделяется несколько способов формального выражения синтаксической зависимости:

- Уподобление зависимым словом формы главного. Уподобление носит характер анафорической отсылки.
- Постановка зависимого в определенную падежную или предложно-падежную форму.
- Контактность синтаксической позиции (характерно для неизменяемых форм, стоят рядом).

Именно эти свойства находят отражения в правилах категориальной грамматики. При разработке правил мы старались использовать опыт других систем и теоретический описания русского синтаксиса. Рассмотрим эти правила подробнее.

2.2.2 Правила уровня клаузы и предложения

Предложение всегда строится по отвлеченному грамматическому образцу. Эти образцы мы попытались описать в возможностях категориальной грамматики. В теоретическом синтаксисе русского языка выделяется ряд структурных схем, внутри которых слова являются грамматически связанными и структурно представляют собой единую единицу.

Первые три правила представляют собой процесс объединения согласованных именной группы и группы глагола в простое предложение или клаузу. В связи с техническими особенностями представления морфологии в нашем парсере такая ситуации описывается двумя правилами.

Правило для выделения простой клаузы без инверсии с глаголом прошедшего времени, согласование с подлежащим по числу и роду, падеж именной группы подлежащего – номинатив, признак лица глагольной группы задан нулем как отсутствующий:

S[-inv] -> NP[C=nomn, NUM=?n, PER=?p, G=?g] VP[NUM=?n, PER=0, G=?g]

Правило для выделения простой клаузы без инверсии с именной группой представленной группой существительного или местоимения и глагольной группой в настоящем или будущем времени, согласование по лицу и числу, признак рода глагольной группы задан нулем, падеж подлежащего – номинатив:

S[-inv] -> NP[C=nomn, NUM=?n, PER=?p, G=?g] VP[NUM=?n, PER=?p, G=0]

Примеры предложений: *Я не мог оторвать глаз от тебя; Стихи попадают в печать.* В том числе, правило будет действовать и для клауз с составным именным сказуемым, например: *они были очень красивые*.

Аналогичные правила для случаев, где в роли подлежащего выступает группа прилагательного:

S[-inv] -> AdjP[C=nomn, NUM=?n, PER=?p, G=?g] VP[NUM=?n, PER=0, G=?g]

S[-inv] -> AdjP[C=nomn, NUM=?n, PER=?p, G=?g] VP[NUM=?n, PER=?p, G=0]

Правило для выделения клаузы со сказуемым, представленным группой прилагательного:

S[+adj] -> NP[C=nomn, NUM=?n, PER=?p, G=?g] AdjP[C=nomn, NUM=?n, PER=?p, G=?g]

Например, предложения типа: *они очень красивые*, *следующее задание совсем простое*. Сказуемое уподобляется в числе. При изменении времени в таких предложениях может меняться падеж прилагательного (*он был красивым*), однако поскольку наша грамматика описывает скорее только поверхностный синтаксис, в таком случае сработает другое правило, описанное выше, с глагольной группой. Предложения с кратким прилагательным также подходят под это правило.

Правило для соединения клаузы, где в роли подлежащего выступает группа с количественным числительным:

S[-inv] -> NUMRNP[C=nomn] VP

Пример: *два друга вошли в комнату*

Поскольку организация структуры правил предполагает линейный порядок используемых групп, для случаев с инверсией нужно прописывать отдельные правила, как:

S[+inv] -> VP[NUM=?n, PER=0, G=?g] NP[C=nomn, NUM=?n, PER=?p, G=?g]

S[+inv] -> VP[NUM=?n, PER=?p, G=0] NP[C=nomn, NUM=?n, PER=?p, G=?g]

Правило для предложений с группой предикатива типа *мне надо идти:*

S[+predp] -> NP[C=datv] PREDP

Правило для предложений с компаративом, типа *мне тяжелее*

S[+comp] -> NP[C=datv] COMP

Правило для предложения типа *Москва – столица*. Согласование по числу.

S -> NP[C=nomn, NUM=?n] '-' NP[C=nomn, NUM=?n]

Следующая группа правил представляет собой правила для распространения простого предложения. Например, такое правило описывает предложения типа *обычно он вставал рано*, когда можно считать, что обстоятельственное наречие относится ко всему предложению.

S[+advp] -> AdvP S

Правило, присоединяющее союз к клаузе, например, *а потом я пошла домой*:

S[+conj]-> CONJ S

Следующая группа правил пока не получилась подробной проработки, но предполагает соединение клауз, простых предложений, обозначенных символом S, в сложные предложения. Рассмотрим несколько простых случаев. Такое правило описывает сложносочиненное предложение с сочинительным союзом.

S[+S] -> S ',' CONJ S

Например, такое правило в нашей грамматике работает для предложения *Я иду пешком, ты едешь на автобусе.*

Аналогичная структура, но без союза:

S[+S] -> S ',' S

Пример: *поезд едет, самолет летит*.

2.2.3 Правила объединения в именную группу (NP)

В теории русского синтаксиса существительное обладает тремя типами присловных связей – согласование, управление и примыкание [Шведова 1980].

Процесс объединения именной группы с группой прилагательного представляет собой связь типа согласование, в простом случае прилагательного с существительным. Он описывается двумя правилами – для единственного и множественного числа. Для единственного числа согласование происходит по падежу и роду в сочетаниях типа *красивый конь, бумажный самолетик.*

NP[+adjf, C=?c, G=?g, NUM=sing] -> AdjP[C=?c, G=?g, NUM=sing] NP[C=?c, G=?g, NUM=sing]

Для множественного числа согласование по падежу. Сочетания типа *красные занавески, разноцветные глаза*.

NP[+adjf, C=?c, NUM=plur] -> AdjP[C=?c, NUM=plur] NP[C=?c, NUM=plur]

Оба правила описывают собой случаи полного согласования, то есть уподобления по всем возможным признакам.

Подобным образом выглядят правила, объединяющие причастие с именной группой.

NP[+prtf, C=?c, G=?g, NUM=sing] -> PrtfP[C=?c, G=?g, NUM=sing] NP[C=?c, G=?g, NUM=sing]

NP[+prtf, C=?c, NUM=plur] -> PrtfP[C=?c, NUM=plur] NP[C=?c, NUM=plur]

Сюда будут относиться такие сочетания как *сломанный стул*, *падающий хлопьями снег, написанное на полях слово*.

Следующее правило обозначает процесс объединения двух именных групп в генитивную группу:

NP[+gent, C=?c, G=?g, NUM=?n] -> NP[C=?c, G=?g, NUM=?n] NP[C=gent]

Формально правило описывает все случаи присоединения именной группы (в тривиальном случае - существительного) в родительном падеже к линейно предшествующей именной группе, причем последняя берет на себя роль вершины и её морфологические характеристики становятся морфологическими характеристиками всей группы. Сюда относятся различные случаи беспредложного сильного управления с родительным падежом (типа *доля секуды, стая воробьев, командир полка*), а также случаи беспредложного падежного примыкания с определительным значением родительного падежа: субъектно-определительные (*суд глупца, грохот трамвая*), определение по обладателю (*пережитки прошлого, лапка кролика*), определение по назначению (*год литературы*), по наличию (*страна озер*) и тд.

Следующее правило описывает объединение именно группой с линейно следующей за ней предложно-падежной группой. В левой части продукции мы фиксируем информацию о таком присоединении. Это правило будет работать для таких последовательностей, как *человек с тростью*, *следы на дороге* и тп:

NP[+pp, C=?c, G=?g, NUM=?n] -> NP[C=?c, G=?g, NUM=?n] PP

С точки зрения теоретического описания русского синтаксиса сюда будут относиться случаи предложного сильного управления со значением восполнения информации (*мастерица на выдумки*, *главный по тарелочкам*), существительные со значением процесса или абстрактного чувства (*движение против войны*, *любовь к халве*), случаи, когда предлоги могут чередоваться и замещать собой косвенные падежи (*цензура на печать / печати, вино гостям / для гостей*), а также случаи варьирования косвенных падежей с инфинитивом (*причина молчать / к молчанию*). С другой стороны, это правило применяется и для случаев примыкания падежей с предлогами, часто несущими определительное значение (*мастерская по ремонту*) или обстоятельственное (*дом у озера*).

Следующие правила являются тривиальными и обеспечивают рекурсивность грамматики. Они относятся к нижнему уровню разбора и имеют отношение уже к непосредственно терминальным элементам. Так, первое правило говорит, что мы считаем отдельное существительное именной группой, а второе говорит то же самое о личных местоимениях (точнее, местоимениях-существительных).

NP[C=?c, G=?g, NUM=?n, PER=?p] -> NOUN[C=?c, G=?g, NUM=?n, PER=?p]

NP[C=?c, NUM=?n, PER=?p, G=?g] -> NPRO[C=?c, NUM=?n, PER=?p, G=?g]

Таким образом, к личным местоимениям, обладающим теми же признаками, что и существительные, могут применяться те же правила, что и к именным группам, и не возникает необходимости писать для них отдельные правила, выражающие те же отношения.

Данное правило описывает объединение двух именных групп при помощи союза.

NP[C=?c, +conj] -> NP[C=?c] CONJ NP[C=?c]

Например, такие сочетания как *дым и зеркала, он и она*. Согласование задается только по падежу, так как вступающие в такие отношения именные группы могут отличаться по числу и роду.

2.2.2 Правила объединения в глагольную группу (VP)

Глагол в русском языке обладает связями управления и примыкания [Шведова 1980]. Как и до этого, нас интересует в первую очередь формальное выражение этих связей. То есть, в первую очередь нас волнует разница между предложной и беспредложной связью, а также вариативность линейной позиции присоединяемой группы. При связи типа управление к глаголу беспредложно может присоединяться именные группы в разных нескольких косвенных падежах (аккузатив, генетив, датив, инструктив). Переходные глаголы управляют винительным падежом, и обязательно присоединяют к себе объект. В нашей грамматике это выражается следующим правилом:

VP[+objt, NUM=?n, PER=?p, G=?g, TR=tran] -> VP[-objt, NUM=?n, PER=?p, G=?g, TR=tran] NP[C=accs]

Значение признака переходности зафиксировано как **tran,** то есть данное правило будет действовать только глаголов, которые морфологический анализатор считает переходными. Например, правило сработает для такого словосочетания, как *разбить бокал*, но не сработает для сочетания *летать попугая.* Иногда винительный падеж прямого объектного дополнения при переходном глаголе может заменяться родительным:

VP[+objt, NUM=?n, PER=?p, G=?g, TR=tran] -> VP[-objt, NUM=?n, PER=?p, G=?g, TR=tran] NP[C=gent]

Например, это такие сочетания как *выпить воды.*

Для случаев с инверсией, когда объект стоит перед глагольной группы, создано отдельное правило:

VP[+objt, NUM=?n, PER=?p, G=?g, TR=tran] -> NP[C=accs] VP[-objt, NUM=?n, PER=?p, G=?g, TR=tran]

Пример: *стол накрыли вечером*

Следующее правило описывает управление глагола творительным падежом. Падеж фиксируется при помощи указания значения категории

падежа в правой части правила (C=ablt]), глагольная группа наследует значения признаков глагола, такие как время, число, лицо и род.

VP[+instr, TENSE=?t, G=?g, NUM=?n, PER=?p] -> VP[TENSE=?t, G=?g, NUM=?n, PER=?p] NP[C=ablt]

Именная группа в таких случаях выражает объект действия (*править страной*), эмоционального состояния или процесса (*любоваться пейзажем*). С глаголами, называющими телодвижение, объектное значение частично утрачивается (*шевелить ушами, трясти головой*).

Аналогично задается управление глагола предложным падежом:

VP[+datv, TENSE=?t, G=?g, NUM=?n, PER=?p] ->VP[TENSE=?t, G=?g, NUM=?n, PER=?p] NP[C=datv]

С точки зрения теоретического синтаксиса, управление глаголами именными группами в дательном падеже обозначает процесс восприятия объекта или обращение состояния к какому-то предмету, как *радоваться снегу, махать другу*. Также именная группа в дательном падеже, управляемая глаголом, может получать субъектное значение, не будучи субъектом предложения, в случаях, тогда обычно группа в дативе предшествует сказуемому, как в случае *мне хочется пойти туда.* Для таких случаев работает правило:

VP[+datv, TENSE=?t, G=?g, NUM=?n, PER=?p] -> NP[C=datv] VP[TENSE=?t, G=?g, NUM=?n, PER=?p]

Что касается предложного управления предлогами, нет смысла дифференцировать его в пределах нашей грамматики, важно различать только случае, когда группа, которыми управляет глагол, идет после него, и когда она стоит перед ним. Отсюда два правила:

VP[+pp, TENSE=?t, G=?g, NUM=?n, PER=?p] -> VP[TENSE=?t, G=?g, NUM=?n, PER=?p] PP

Пример: *идти по дороге*, *синее небо над головами*

VP[+pp, TENSE=?t, G=?g, NUM=?n, PER=?p] -> PP VP[TENSE=?t, G=?g, NUM=?n, PER=?p]

Пример: *по пояс закопался, из земли торчал*

Случаи двойного управления глагола (*одолжить книгу другу, учить людей добру*) обеспечиваются рекурсивным применением правил.

VP[+advb, TR=?tr, TENSE=?t, G=?g, NUM=?n, PER=?p] -> AdvP VP[TENSE=?t, TR=?tr, G=?g, NUM=?n, PER=?p]

Помимо управления, глагольным формам в русском языке свойственно примыкание. Примыкают к глаголам формы наречия, деепричастия, компаратива и инфинитива. В нашей грамматики эти связи выражаются следующими правилами:

Во-первых, примыкание наречия (или наречной группы):

VP[+advb, TENSE=?t, G=?g, NUM=?n, PER=?p] -> VP[TENSE=?t, G=?g, NUM=?n, PER=?p] AdvP

Пример: *действовать быстро, бежать быстрее*

Во-вторых, примыкание инфинитива (или группы инфинитива):

VP[+infn, TR=?tr, NUM=?n, PER=?p, G=?g] -> VP[NUM=?n, PER=?p, G=?g] VP[TR=?tr, NUM=0, PER=0, G=0]

Напомним, что в нашей грамматики форма инфинитива задается как глагольная групп с нулевыми признаками числа, лица и рода.

Пример: *пойти побегать, ложиться спать, собираться начать готовить обед*

В третьих, примыкание компаратива:

VP[+comp, TENSE=?t, G=?g, NUM=?n, PER=?p] -> VP[TENSE=?t, G=?g, NUM=?n, PER=?p] COMP

Пример: *спать крепко, знаю получше*

Правило для примыкания группы числительного:

VP[+numr, TENSE=?t, G=?g, NUM=?n, PER=?p] -> VP[TENSE=?t, G=?g, NUM=?n, PER=?p] NUMRNP

Пример: *видеть четырех людей*

Правило, объединения глагола с отрицательной частицей *не* для формирования отрицательной формы глагола.

VP[+neg, TENSE=?t, G=?g, NUM=?n, PER=?p] -> 'не' VP[TENSE=?t, G=?g, NUM=?n, PER=?p]

Пример: *не вижу, не обманешь*

2.2.4 Другие правила

Рассмотрим теперь и другие правила, не образующие столь крупные блоки.

Правила, описывающие объединение в предложную группу, представляют собой процесс объединения предлога с существительным в падеже, отличном от именительного. Поскольку на данный момент в реализации парсера отсутствует возможность рациональным образом связать конкретный предлог с падежом или падежами, которыми он управляет, часто получаются лишние разборы из-за морфологической неоднозначности слова после предлога.

PP-> PREP NP[C=datv, G=?g, NUM=?n, PER=?p]

Пример: *к машине, к телу*

PP-> PREP NP[C=loct, G=?g, NUM=?n, PER=?p]

Пример: *о жизни, в огне*

PP-> PREP NP[C=accs, G=?g, NUM=?n, PER=?p]

Пример: (*переехать) в новый дом*

PP-> PREP NP[C=gent, G=?g, NUM=?n, PER=?p]

Пример: *(гуляю) у реки*

PP-> PREP NP[C=ablt, G=?g, NUM=?n, PER=?p]

Пример: *(борьба) с коррпуцией*

PP-> PREP NP[C=datv, G=?g, NUM=?n, PER=?p]

Пример: *к машине, к телу*

PP-> PREP NP[C=loct, G=?g, NUM=?n, PER=?p]

Пример: *о жизни, в огне*

PP-> PREP NP[C=accs, G=?g, NUM=?n, PER=?p]

Пример: (*переехать) в новый дом*

PP-> PREP NP[C=gent, G=?g, NUM=?n, PER=?p]

Пример: *(гуляю) у реки*

PP-> PREP NP[C=ablt, G=?g, NUM=?n, PER=?p]

Пример: *(борьба) с коррпуцией*

Так же в нашем парсере используются правила объединения в группу прилагельного (AdjP), группу компаратива (COMP), группу причастия (PrtfP), некоторые правила союзного объединения однородных членов типа:

INFN[+conj] -> INFN CONJ INFN

Помимо этого, выделяются группы наречия и некоторые предикативные группы. Полный список правил можно найти в Приложении Б.

2.3 Программная реализация инструмента

2.3.1 Используемое программное обеспечение

Программа написана на языке программирования Python (версия 3.4.3) и работает с инструментами NLTK. Парсер, разрабатываемый нами на основе описанной выше грамматики, работает опирается на морфологические характеристик словоформ. Для получения морфологической информации было принято использовать морфоанализатор PyMorphy2. Соответственно, работа разрабатываемого инструмента предполагает предустановку Python (версии 3+), NLTK и PyMorphy2.

2.3.2 Морфологический компонент

PyMorphy2 был также создан на Python и обладает тегсетом, позволяющим нам использовать удобный формат морфологической аннотации. PyMorphy, к тому же, обладает высокой точностью по сравнению с другими аналогичными инструментами для русского языка. Морфологический анализатор умеет восстанавливать нормальную форму слова, приводить слово к указанной форме, то есть, например, менять число, лицо или падеж, и, наконец, возвращать грамматическую информацию о словоформе в виде набора тегов. В построении синтаксического анализатора нам потребуется именно последняя функция. PyMorphy2 работает со словарями OpenCorpora и использует принятые в OpenCorpora граммемы. Используется 17 граммем для обозначения частей речи, а также дополнительные граммемы для числа, падежей и тд. Есть также несколько

собственных, не используемых в OpenCorpora, граммем, включая NUMB для чисел, PNCT для знаков препинания и LATN для слов, состоящих из латинских букв. PyMorphy предоставляет все возможные варианты разбора (интерпретаций) словоформы, однако каждому разбору присваивается оценка score, отражающая его вероятность - P(tag|word). Вероятность вычисляется на основе встречаемости в корпусе OpenCorpora. Первый вариант разбора – самый вероятный, и далее по убывающей. Для неизвестных слов работает предсказатель. Так выглядит морфологический разбор слова *стекло*, предоставляемый морфоанализатором:

m.parse("стекло")

[Parse(word='стекло', tag=OpencorporaTag('NOUN,inan,neut sing,nomn'), normal_form='стекло', score=0.75, methods_stack=((<DictionaryAnalyzer>, 'стекло', 545, 0),)), Parse(word='стекло', tag=OpencorporaTag('NOUN,inan,neut sing,accs'), normal_form='стекло', score=0.1875, methods_stack=((<DictionaryAnalyzer>, 'стекло', 545, 3),)), Parse(word='стекло', tag=OpencorporaTag('VERB,perf,intr neut,sing,past,indc'), normal_form='стечь', score=0.0625, methods_stack=((<DictionaryAnalyzer>, 'стекло', 968, 3),))]

Для каждого из вариантов разбора мы можем посмотреть набор тегов. Так, для слова *стекло* предлагается три варианта разбора:

- OpencorporaTag('NOUN,inan,neut sing,nomn') – существительное в именительном падеже
- OpencorporaTag('NOUN,inan,neut sing,accs') – существительное в винительном падеже
- OpencorporaTag('VERB,perf,intr neut,sing,past,indc') – глагол среднего рода в прошедшем времени

Одной из задач в данной работе, стало, соответственно, подключение данного морфоанализатора к нашему парсеру. Сначала для каждой части речи (категории) мы отобрали значимые параметры, то есть те признаки категорий, которые пригодятся нам в правилах выделения синтаксических

групп. Так, для существительных этими признаками стали род, число и падеж, для глаголов – род, лицо, время, переходность. Для неизменяемых частей речи использовался только собственно признак части речи.

Поскольку NLTK работает с правилами в виде формальных грамматик, мы создали специальную функцию, переводящую грамматические теги для каждой словоформы в вид правил категориальной грамматики с терминальными элементами. Эти правила называются лексические продукции и записываются в файл с грамматикой, туда же, куда и синтаксические продукции, то есть правила объединения в синтаксические группы. Лексические продукции имеют следующий вид:

NOUN[C=gent, G=neut, NUM=sing, PER=3, NF=u'крыло'] -> 'крыла'

NOUN[C=nomn, G=neut, NUM=plur, PER=3, NF=u'крыло'] -> 'крыла'

NOUN[C=accs, G=neut, NUM=plur, PER=3, NF=u'крыло'] -> 'крыла'

То есть, при работе со словоформой *крыла* данные строки запишутся в грамматику.

2.3.3 Алгоритм работы программы

В общем виде, алгоритм программа работает следующим образом:

1. Запуска программы, запись в файл синтаксической части грамматики
2. Ввод пользователем предложения для разбора
3. Запись в файл информации о словоформах в виде правил категориальной грамматики
4. Разбор предложения парсером NLTK
5. Вывод результата в виде одного или нескольких разборов (или их отсутствие) в командную строку или в отдельный файл

Допустим, пользователь хочет разобрать предложение *Один молодой человек вышел из своей каморки.* Ещё до ввода предложения, после запуска программы, в папке располагающейся там, где хранятся данные NLTK, а именно грамматики, будет создан файл формата .fcfg. В него запишется та часть грамматики, которая представляет собой синтаксические правила

объединения составляющих в группы. Затем пользователю предлагается ввести предложение или сочетание для разбора. Это предложение будет разбито на слова, после чего для каждого варианта морфологической интерпретации словоформы предоставляемых PyMorphy2 в грамматику будет записано по строчке. В зависимости от части речи, из всех тегов, предоставляемых морфоанализатором, будут записаны значения некоторых отобранных нами ранее релевантных признаков. Так, для предложения *Один молодой человек вышел из своей каморки* в файл с грамматикой попадут следующие лексические продукции:

```
ADJF[C=nomn, G=masc, NUM=sing, PER=3, NF=u'один'] -> 'один'
ADJF[C=accs, G=masc, NUM=sing, PER=3, NF=u'один'] -> 'один'
NOUN[C=gent, G=femn, NUM=sing, PER=3, NF=u'молодая'] -> 'молодой'
NOUN[C=datv, G=femn, NUM=sing, PER=3, NF=u'молодая'] -> 'молодой'
NOUN[C=ablt, G=femn, NUM=sing, PER=3, NF=u'молодая'] -> 'молодой'
NOUN[C=loct, G=femn, NUM=sing, PER=3, NF=u'молодая'] -> 'молодой'
ADJF[C=nomn, G=masc, NUM=sing, PER=3, NF=u'молодой'] -> 'молодой'
ADJF[C=accs, G=masc, NUM=sing, PER=3, NF=u'молодой'] -> 'молодой'
ADJF[C=gent, G=femn, NUM=sing, PER=3, NF=u'молодой'] -> 'молодой'
ADJF[C=datv, G=femn, NUM=sing, PER=3, NF=u'молодой'] -> 'молодой'
```

```
ADJF[C=ablt, G=femn, NUM=sing, PER=3, NF=u'молодой'] ->
'молодой'
ADJF[C=loct, G=femn, NUM=sing, PER=3, NF=u'молодой'] ->
'молодой'
NOUN[C=nomn, G=masc, NUM=sing, PER=3, NF=u'человек'] ->
'человек'
NOUN[C=gent, G=masc, NUM=plur, PER=3, NF=u'человек'] ->
'человек'
VERB[TR=intr, TENSE=past, G=masc, NUM=sing, PER=0,
NF=u'выйти'] -> 'вышел'
PREP[NF=u'из'] -> 'из'
ADJF[C=gent, G=femn, NUM=sing, PER=3, NF=u'свой'] ->
'своей'
ADJF[C=loct, G=femn, NUM=sing, PER=3, NF=u'свой'] ->
'своей'
ADJF[C=ablt, G=femn, NUM=sing, PER=3, NF=u'свой'] ->
'своей'
ADJF[C=datv, G=femn, NUM=sing, PER=3, NF=u'свой'] ->
'своей'
NOUN[C=gent, G=femn, NUM=sing, PER=3, NF=u'каморка'] ->
'каморки'
NOUN[C=nomn, G=femn, NUM=plur, PER=3, NF=u'каморка'] ->
'каморки'
NOUN[C=accs, G=femn, NUM=plur, PER=3, NF=u'каморка'] ->
'каморки'
```

2.3.4 Выходные данные

Так выглядит разбор предложения *Один молодой человек вышел из своей каморки*

```
(XP[]
  (S[-inv]
    (NP[C='nomn', G='masc', NUM='sing', +adjf]
      (AdjP[C='nomn', G='masc', NUM='sing']
        (ADJF[C='nomn',       G='masc',       NF='один',
NUM='sing', PER=3] один))
      (NP[C='nomn', G='masc', NUM='sing', +adjf]
        (AdjP[C='nomn', G='masc', NUM='sing']
          (ADJF[C='nomn',    G='masc',    NF='молодой',
NUM='sing', PER=3]
            молодой))
        (NP[C='nomn', G='masc', NUM='sing', PER=3]
          (NOUN[C='nomn',    G='masc',    NF='человек',
NUM='sing', PER=3]
            человек))))
    (VP[G='masc', NUM='sing', PER=0, TENSE='past', +pp]
      (VP[G='masc',  NUM='sing',  PER=0,  TENSE='past',
TR='intr']
        (VERB[G='masc',  NF='выйти',  NUM='sing',  PER=0,
TENSE='past', TR='intr']
          вышел))
      (PP[C='gent', G='femn', NUM='sing', PER=?p]
        (PREP[NF='из'] из)
        (NP[C='gent', G='femn', NUM='sing', +adjf]
          (AdjP[C='gent', G='femn', NUM='sing']
            (ADJF[C='gent',     G='femn',     NF='свой',
NUM='sing', PER=3]
              своей))
          (NP[C='gent', G='femn', NUM='sing', PER=3]
```

```
            (NOUN[C='gent',    G='femn',    NF='каморка',
NUM='sing', PER=3]
              каморки))))))
```

Здесь мы видим, что верхняя выделенная группа S[-inv], то есть простое предложение без инверсии, раскладывается на две составляющие NP[+adjf], это именная группа с определением в виде группы прилагательного (*один молодой человек*), и VP[+pp], то есть глагольная группа с присоединенной к ней предложно-падежной группой (*вышел из своей каморки*). В свою очередь NP[+adjf] раскладывается на группу прилагательного AdjP (*один*) и именную группу с прилагательным NP[+adjf] (*молодой человек*); а VP[+pp] на группу глагола VP (*вышел*) и предложно-падежную группы PP (*из своей каморки*). Далее группа *молодой человек* раскладывается на группу прилагательного и группу существительного, *из своей каморки* на предлог и именную группу (*своей каморки*) и тд. Последними правилами перед переходом к терминальным элементам являются тривиальные правила перехода существительного в именную группу, глагола в глагольную, прилагательного – в группу прилагательного. В данном случае предложение разбирается однозначно, и мы видим только один разбор. В случае неоднозначности синтаксического разбора в нашей системе правил, а так же в случае настоящей синтаксической неоднозначности парсер предложит на выходе несколько вариантов разбора, например для сочетания *в ручке ящика тумбочки* мы увидим четыре разбора:

```
(XP[]
  (PP[C='loct', G='femn', NUM='sing', PER=?p]
    (PREP[NF='в'] в)
    (NP[C='loct', G='femn', NUM='sing', +gent]
      (NP[C='loct', G='femn', NUM='sing', +gent]
        (NP[C='loct', G='femn', NUM='sing', PER=3]
          (NOUN[C='loct',     G='femn',     NF='ручка',
NUM='sing', PER=3]
            ручке))
        (NP[C='gent', G='masc', NUM='sing', PER=3]
```

```
          (NOUN[C='gent',      G='masc',      NF='ящик',
NUM='sing', PER=3]
            ящика)))
      (NP[C='gent', G='femn', NUM='sing', PER=3]
        (NOUN[C='gent',     G='femn',     NF='тумбочка',
NUM='sing', PER=3]
          тумбочки)))))
(XP[]
  (PP[C='loct', G='femn', NUM='sing', PER=?p]
    (PREP[NF='в'] в)
    (NP[C='loct', G='femn', NUM='sing', +gent]
      (NP[C='loct', G='femn', NUM='sing', PER=3]
        (NOUN[C='loct',      G='femn',      NF='ручка',
NUM='sing', PER=3]
          ручке))
      (NP[C='gent', G='masc', NUM='sing', +gent]
        (NP[C='gent', G='masc', NUM='sing', PER=3]
          (NOUN[C='gent',      G='masc',      NF='ящик',
NUM='sing', PER=3]
            ящика))
        (NP[C='gent', G='femn', NUM='sing', PER=3]
          (NOUN[C='gent',    G='femn',    NF='тумбочка',
NUM='sing', PER=3]
            тумбочки))))))
(XP[]
  (PP[C='datv', G='femn', NUM='sing', PER=?p]
    (PREP[NF='в'] в)
    (NP[C='datv', G='femn', NUM='sing', +gent]
      (NP[C='datv', G='femn', NUM='sing', +gent]
        (NP[C='datv', G='femn', NUM='sing', PER=3]
          (NOUN[C='datv',     G='femn',     NF='ручка',
NUM='sing', PER=3]
            ручке))
        (NP[C='gent', G='masc', NUM='sing', PER=3]
          (NOUN[C='gent',      G='masc',      NF='ящик',
NUM='sing', PER=3]
            ящика)))
      (NP[C='gent', G='femn', NUM='sing', PER=3]
        (NOUN[C='gent',     G='femn',     NF='тумбочка',
NUM='sing', PER=3]
          тумбочки)))))
(XP[]
  (PP[C='datv', G='femn', NUM='sing', PER=?p]
    (PREP[NF='в'] в)
    (NP[C='datv', G='femn', NUM='sing', +gent]
```

```
      (NP[C='datv', G='femn', NUM='sing', PER=3]
        (NOUN[C='datv',      G='femn',      NF='ручка',
NUM='sing', PER=3]
          ручке))
      (NP[C='gent', G='masc', NUM='sing', +gent]
        (NP[C='gent', G='masc', NUM='sing', PER=3]
          (NOUN[C='gent',      G='masc',      NF='ящик',
NUM='sing', PER=3]
            ящика))
        (NP[C='gent', G='femn', NUM='sing', PER=3]
          (NOUN[C='gent',    G='femn',    NF='тумбочка',
NUM='sing', PER=3]
            тумбочки)))))
```

Первое расхождение связано с тем, что парсер напрямую не учитывает информацию о том, какими падежами могут управлять предлоги, и, помимо правильного предложного, приписывает именной группе *ручке ящика двери* дательный падеж, возможных для этой группы, но невозможный после предлога *в.* С другой стороны, парсер предлагает два пути составления именной группы *ручка ящика тумбочки*, соединяя сначала *ящика тумбочки* или *ручка ящика.* В рассмотренном случае, на самом деле, не существует настоящей синтаксической неоднозначности и только один разбор является правильным. Выбор его из числа полученных является следующей задачей в продолжении исследования.

2.4 Оценка результатов

2.4.1 Метод оценки

Наш парсер представляет синтаксические деревья разбора в виде структур составляющих, что усложняет выбор метода оценки результатов. Поскольку синтаксический анализатор, разработанный в выбранных нами рамках даже в идеальной реализации не будет давать однозначного разбора, а с увеличением числа правил и охвата возможных синтаксических отношений количество формальных интерпретаций одного предложения будет только увеличиваться, мы решили на данном этапе не ставить перед собой задачу стремиться получить один разбор для каждого предложения. То, на что претендует работающий таким образом парсер, это то, что среди

предоставляемых им структур присутствует один, или, в случае синтаксической неоднозначности, несколько, правильных разборов. Именно наличие такого правильного разбора и есть тот параметр, по которому мы хотим провести оценку. Для удобства мы решили взять корпус достаточно коротких предложений, тем не менее охватывающий широкий спектр синтаксических отношений. Предложения были взяты из корпуса, предоставленного на сайте проекта Link Grammar Протасова [http://slashzone.ru/parser/]. Корпус состоит из 2.5 миллионов предложений, причем он разбит на отдельные подкорпуса, исходя из длины и сложности предложения. Поскольку сравнение проводилось вручную, было принято решение ограничиться предложениями и словосочетаниями, состоящими из четырех слов. Мы отобрали по пять групп по 20 предложений из подкорпусов разной сложности. Сложность была посчитана на основе частотного рейтинга встречающихся в предложениях слов. В первый подкопус вошли предложения с частотным рейтингом 2980 – 3640, во второй – 2440 – 2980, в третий 148 – 181, в четвертый с рейтингом 66-81, и, наконец, в последний, самый сложный, с рейтингом 36-40. Мы не стали брать предложения длиннее по нескольким причинам. Во-первых, из-за эффекта комбинаторного взрыва, вызывающего сложность ручной оценки выбора правильного разбора из сотен альтернативных. Во-вторых, даже в коротких предложениях реализуется достаточное количество разнообразных синтаксических отношений, которые затем рекурсивно повторяются при соединении групп в более крупных предложениях, и, таким образом, их будет достаточно для первичной оценки работоспособности грамматики. Мы намеренно не брали также слишком сложные предложения, так как на данном этапе наш парсер не претендует на абсолютную полноту грамматики, а также не работает в режиме частичного разбора предложения, то есть, если у него не получается собрать все предложение в некоторую группу на основе правил описанной грамматики, то он не предложит ни одного разбора. За образец брался разбор, предоставляемый синтаксическим анализатором

проекта АОТ [http://aot.ru/onlinedemo.html]. Синтаксический анализатор проекта АОТ так же работает с группами составляющих, которые подразделяются на так называемые синтаксические фрагменты и синтаксические группы. Прежде чем проводить сравнение, мы сопоставили типы используемых групп в АОТе и в нашем парсере. Результат представлен в таблице 1:

АОТ	NLTK4RUSSIAN	пример	комментарий
ГЛ_ЛИЧН	S, VP	Я иду домой	
ПРЕДК	S[+predp], PredP	мне нужно	
КР_ПРЧ	S[+adj]	он унижен	
КР_ПРИЛ	S[+adj]	ты злой	
ПРЧ	NP[+particip]	Зверь, лежащий на ковре	
ИНФ	S, VP	чтобы лучше жить	
ВВОД	-	На самом деле	
ТИРЕ	S	Мой друг - инженер	
СРАВН	COMP	глупее кота	
КОЛИЧ	NUMR, NumrNP	восемнадцать	
ФИО	-		
НАР_ПРИЛ	AdjP	Очень высокий	
ОДНОР_ПРИЛ	AdjP [+conj]	Красивый и умный	
ОДНОР_НАР	AdpP[+conj]	Тяжело и трудно	
ОДНОР_ИНФ	VP, INFN[+conj]	Терять и находить	
ОДНОР_ПРИЛ	COMP[+conj]	Быстрее и выше	Сравнительная степень
ДАТА	-	Июль 1993	
СРАВН-СТЕПЕНЬ	COMP[+advb]	Гораздо унизительнее	Аналитическая форма

			сравнительной степни
НАРЕЧ-ГЛАГОЛ	VP[+advb]	Мерзко хихикать	
ПРИЛ-СУЩ	NP	Теплая белая ночь	
НАР-ЧИСЛ-СУЩ	NP[+advb]	Мало очень хороших идей	
ЧИСЛ-СУЩ	NUMRNP	Тридцать четыре ореха	
ГЕНИТ_ИГ	NP[+gent]	Огни города	Именна группа с генитивом
ПГ	PP	На берегу озера	Предложная группа
ОДНОР_ИГ	NP[+conj]	Ива и ольха	
ОТР_ФОРМА	VP[+neg]	Не бояться	Отрицание глагольной формы
ПРЯМ_ДОП	VP[+objt]	Чесать пятку	
ГЛАГ_ИНФ	VP[+infn]	Лечь спать	

Табл. 1

Рассмотрим пример того, как проходило сравнение.

карлик нехотя вернул автоматы

```
 (XP[]
  (S[-inv]
    (NP[C='nomn', G='masc', NUM='sing', PER=3]
      (NOUN[C='nomn',       G='masc',       NF='карлик',
NUM='sing', PER=3]
        карлик))
    (VP[G='masc', NUM='sing', PER=0, TR='tran', +objt]
      (VP[G='masc',  NUM='sing',  PER=0,  TENSE='past',
TR='tran', +advb]
        (AdvP[] (ADVB[NF='нехотя'] нехотя))
```

```
        (VP[G='masc', NUM='sing', PER=0, TENSE='past',
TR='tran', +objt]
          (VERB[G='masc',   NF='вернуть',   NUM='sing',
PER=0, TENSE='past', TR='tran']
            вернул)))
      (NP[C='accs', G='masc', NUM='plur', PER=3]
        (NOUN[C='accs',     G='masc',     NF='автомат',
NUM='plur', PER=3]
          автоматы)))))
```

АОТ:

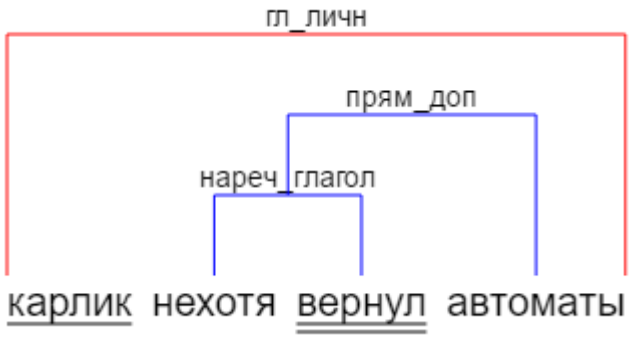

Рисунок 4. Разбор системы АОТ

В примере мы видим, что разборы парсера NLTK4RUSSIAN и АОТа совпадаю: сначала происходит объединение наречия и глагола в глагольную группу с наречием (*нехотя вернул*), затем к ней присоединяется объект – прямое дополнение (*автоматы*), затем происходит объединение именной и глагольной группы в простое предложение без инверсии в разборе NLTK, или в аналогичную, но более общую константу, называющуюся фрагмент с личной формой глагола у системы АОТ.

Поскольку в большинстве случаев наш парсер предоставлял неоднозначный разбор, из всех предложенных им вариантов выбиралась наиболее правильная интерпретация, а затем разбор сравнивался с предлагаемым АОТом. Здесь стоит отметить несколько наблюдений:

Во-первых, в некоторых случаях разбор системы АОТ явно не являлся эталонным. Например, в предложении *Цветок придал девушке уверенности*, система объединяет последовательность словоформ *девушке уверенности* в

генитивную именную группу, а уже потом присоединяет её к остальному предложению. Поскольку наш парсер не пытается выбрать более правдоподобный вариант из всех возможных, в таких случаях мы искали более верный вариант разбора, опираясь на личную экспертную оценку, и, при наличии такового, парсер все равно получал максимальную оценку за разбор данного предложения. В случае с приведенным выше примером это был разбор, последовательно присоединяющий к глаголу придал дополнение *девушке,* а затем к получившейся глагольной группе дополнение *уверенности.*

Во-вторых, в разборах, предоставляемых системой АОТ не всегда отображаются некоторые промежуточные процессы объединения составляющих. Например, если на разборе, представленном на выше рисунке 1, мы видим подробное выделение всех возможных групп, то в следующем примере (рисунок 2) мы явно не видим, по какому принципу, и к чему именно присоединяется именная группа *высокими голосами.*

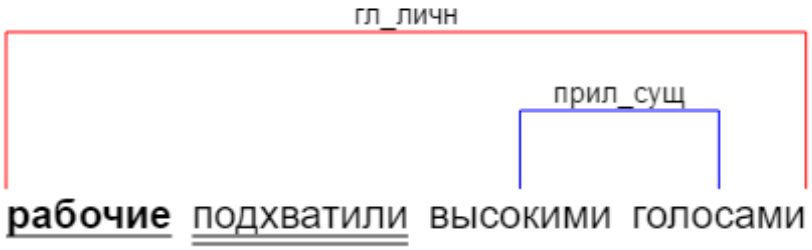

Рисунок 5. Разбор системой АОТ

2.4.1 Анализ результатов и ошибок

Сравнение проводилось по трехбальной шкале: разборы совпадают по структуре полностью (2), частично (1), различаются (0). Результаты представлены в таблице 2, числа обозначают количество предложений из каждого корпуса, получивших соответствующую оценку.

	Оценка 0	Оценка 1	Оценка 2
Корпус 1	2	1	17
Корпус 2	2	0	18
Корпус 3	2	1	17
Корпус 4	3	3	14
Корпус 5	7	8	5

Табл. 2

В целом, результаты можно назвать положительными. Оценку 0 получили, по большей части, предложения, с которыми парсер не справился в силу несовершенства грамматики, однако в подавляющем большинстве случаев, если предложение было разобрано, то среди разборов присутствовал и верный, а именно это мы и хотели получить на данном этапе исследования. Несмотря на то, что формулировка задачи не предполагала попытки избавиться от неправдоподобных разборов, мы все-таки можем сформулировать некоторые слабые места работы парсера, исправив которые, можно уменьшить количество неверных интерпретаций.

Первая группа причин связана с морфологическим модулем. Во-первых, часто среди разборов, предлагаемых PyMorphy2 есть маловероятные и даже невозможные. Например, словоформу *рис* морфоанализатор приводит к нормальной форме *рисунок*, и на основании этого предположения парсер строит несколько заведомо неверных разборов. Избыточность морфологической информации проявляется и в менее абсурдных случаях. Например, в предложении *потом уже будет поздно*, несколько разборов строится на гипотезе, что наречие *уже* является сравнительной степенью прилагательного *узкий.* А предложение *отовсюду слышались веселые песни* получает два разбора только потому, что слово *песни* может быть формой множественного числа как слова *песня,* так и слова *песнь*. С другой стороны, некоторые ошибки, невозможности разбора и неправильная интерпретация

связана с ограниченностью работы морфоанализатора, который работает только на уровне отдельных словоформ (точнее, слов в смысле «от пробела до пробела»). Поэтому парсер пытается соединить в одно целое предлоги или наречия, состоящие из нескольких слов (*в следствие*), на синтаксическом уровне, хотя они представляют собой единое целое на уровне лексическом.

Были и случаи, с которыми плохо справлялись как АОТ, так и наш синтаксический анализатор на основе NLTK. Например, предложение *только недавно ходить перестал.*

Одной из самых частотных ошибок является разрыв именной группы с прилагательным, которое должно в неё входить, как в таком случае:

потянулись долгие мгновения ожидания

```
 (XP[]
  (VP[G=None, NUM='plur', PER=0, TR='tran', +objt]
    (VP[G=None, NUM='plur', PER=0, TENSE='past', +adj]
      (VP[G=None, NUM='plur', PER=0, TENSE='past', TR='intr']
        (VERB[G=None, NF='потянуться', NUM='plur', PER=0, TENSE='past',
TR='intr']
          потянулись))
      (AdjP[C='nomn', G=None, NUM='plur']
        (ADJF[C='nomn', G=None, NF='долгий', NUM='plur', PER=3]
          долгие)))
    (NP[C='gent', G='neut', NUM='sing', +gent]
      (NP[C='gent', G='neut', NUM='sing', PER=3]
        (NOUN[C='gent', G='neut', NF='мгновение', NUM='sing', PER=3]
          мгновения))
      (NP[C='gent', G='neut', NUM='sing', PER=3]
        (NOUN[C='gent', G='neut', NF='ожидание', NUM='sing', PER=3]
          ожидания)))))
```

Здесь происходит объединение в группу глагола с прилагательным *потянулись долгие*, и присоединение этой глагольной группы в качестве сказуемого к подлежащему, выраженному именной группой, вместо того чтобы *долгие мгновения ожидания* выделялось как единая именная группа и уже потом происходило последующее объединение.

ГЛАВА 3. ПРИМЕНЕНИЕ. ЭКСПЕРИМЕНТЫ ПО ИЗВЛЕЧЕНИЮ КЛЮЧЕВЫХ СЛОВОСОЧЕТАНИЙ

Несмотря на то, что грамматика зависимостей полнее и точнее описывает синтаксическую структуру предложений русского языка, в некоторых задачах требуется и вполне достаточно именно автоматического определения типа и границ синтаксических групп в тексте. Это может рассматриваться как поверхностный синтаксический анализ. Мы использовали некоторые группы составляющих, с которыми работает наш парсер в задаче извлечения ключевых словосочетаний.

Извлечение ключевых слов и словосочетаний – одна из важных задач современной компьютерной лингвистики. Один из методов основан на алгоритме RAKE (Rapid automatic keyword extraction), запатентованный авторами в 2009 году [Rose et al. 2009]. В отличие от некоторых других способов, RAKE не требует фонового корпуса для расчета частоты встречаемости словоформ, а также рассматривает не только отдельные лексические единицы, но и словосочетания. Ключевые фразы, или словосочетания, состоят из нескольких слов, но не могут включать в себя служебные слова, знаки пунктуации, слова широкого значения и тд.

Первый этап предполагает определение в тексте слов и словосочетаний – кандидатов в ключевые слова. Для этого текст разбивается на сегменты по знакам препинания и стоп-словарю, содержащему стоп-слова, которые не могут входить в будущие ключевые слова (словосочетания), то есть артикли, местоимения, вводные слова, предлоги и тд. Для каждой цепочки слов, представляющей собой кандидата в ключевую, рассчитывается вес. Сначала вес рассчитывается для каждого составляющего её слова на основе общей его частоты встречаемости и средней длины фразы, а потом рассчитывается вес всей цепочке путем сложения весов входящих в нее слов. Алгоритм RAKE для задач обработки естественного языка был реализован на языке Python. Позже он был адаптирован для работы с библиотеками NLTK (URL:

http://www.nltk.org/) [Bird et al. 2009]. Однако, и в таком виде он был приспособлен лишь для работы с английским языком. Если применить его в неадаптированном виде, на выходе мы получали слишком длинные цепочки, так как в кандидаты попадали целые предложения, а не правдоподобные словосочетания. Мы создали стоп-словарь, аналогичный использующемуся в оригинальной реализации, однако для эффективного выделения словосочетаний-кандидатов нам показалось разумным использовать информацию о границах синтаксических групп и их типов. Предобработка текста включает в себя разбиение текста на условные слова (по пробелам) и проставление границ условных синтаксических групп (используется знак-разделитель “|”). Было принято решение использовать результаты работы над грамматикой для NLTK, а именно выделить в тексте некоторые типы использующихся в ней групп и расставить между ними границы. Наречия, краткие формы прилагательных и причастий, компаратив, глаголы и различные глагольные формы выделяются в отдельные самостоятельные группы, т.е. встретив в тексте такое слово, алгоритм «окружает» его границами с обеих сторон. Генитивная группа и именная группа с прилагательным целиком выделяются как кандидат в ключевое словосочетание вместе со своими составляющими. Существительные, идущие линейно подряд, но не образующие синтаксическую группу, разделялись знаком-разделителем. Правила, как и формальная грамматика, сохраняют рекурсию и позволяют сохранить некоторые относительно длинные вложенные конструкции, как, например, *производство пилотируемых кораблей*, или *трансформация непилотируемых космических аппаратов*. Здесь, в отличие от задач, связанных с применением глубокого синтаксического анализа, нас не интересует внутренняя структура выделенного фрагмента и даже тип. Пример предобработанного предложения:

Наиболее | совершенными герметизирующими материалами, обеспечивающими | надежную | и | устойчивую герметичность соединений, | являются | самовулканизирующиеся пасты.

В некоторых случаях, однако, мы получили слишком длинные фрагменты, как генитивная группа *собственную программу ускоренного научно-технического развития.* Такие длинные конструкции не разумно считать кандидатами в ключевые словосочетания, однако и просто отбросить их, наложив ограничение на длину, не будет правильным решением. Правила составлялись на основе эмпирических данных, преимуществом является их наглядный вид и легкость в изменении, подключении или отключении любого из них.

Обработанный таким образом текст принимался уже оригинальным инструментом, реализующим алгоритм RAKE в NLTK. Ключевые выражения в упорядоченном по весу порядке сохраняются в отдельный файл с указанием веса. Так выглядит фрагмент выдачи:

```
моделирование конструкций 4.12469911041
радиоэлектронной техники 4.12463100295
разгонных двигателей 4.12435745354
теория эксперимента 4.12429959977
эксплуатационной пригодности 4.12380952381
оптимального уровня 4.1234375
собственных колебаний 4.12312312312
…
микропроцессорную излучения 4.11538461538
планетного излучения 4.11538461538
лазерного излучения 4.11538461538
импульсного излучения 4.11538461538
солнечного излучения 4.11538461538
```

Мы провели также предварительные эксперименты по применению модифицированного нами алгоритма RAKE в NLTK для русского языка на материале четырех корпусов русскоязычных текстов. Каждый корпус представлял какой-то функциональный стиль: научный, публицистический, официально-деловой и художественный. Тематика всех корпусов - ракетостроение и аэрокосмические исследования. Объем каждого составляет примерно 500 тыс. с/у.

Уже опираясь на один внешних вид выдачи ключевых слов и словосочетаний можно сделать выводы об целесообразности применения поверхностного синтаксического анализа и некоторых особенностях работы используемого инструмента. Во-первых, в самую верхнюю часть попало достаточно много слишком длинных фраз. Например, верхушка выдачи ключевых выражений по корпусу, охватывающему тексты научного функционального стиля выглядит так:

```
самых мощных циклотронах современных лабораторий 22.5
ставший крупнейшим конструктором ракетно-космических
систем 21.9230769231
основным видом штатного технического обслуживания
19.6547619048
свечение специальных «холодных» электрических ламп
19.4090909091
совместное комплексное проектирование технологических
процессов 18.6527777778
```

Эти конструкции представляют собой единые синтаксические группы, выделяемые по соответствующему правилу согласованных именных групп с определением в виде прилагательного, однако содержат слишком много слов, чтобы считаться полноправными ключевыми выражениями. Появление таких длинных групп именно в верху списка обусловлено тем, что вес для каждого ключевого выражения рассчитывается как сумма весов составляющих её лексических единиц. Решением этой проблемы может стать, во-первых,

расширение стоп-словаря, в который должны были бы войти такие слова как *самых* и *ставший* из приведенного выше примера, и, во-вторых, введение ограничения на объем выделяемой синтаксической группы и разработку правил по отделению наиболее значимой ее части.

Оценка эффективности была произведена исходя из преположения, что ключевые выражения, составляющие семантическое ядро корпуса, могут быть сопоставимы с его тематической моделью. При построении тематических моделей корпусов использовался алгоритм LDA (Latent Dirichlet Allocation) в пакете GenSim для Python [Митрофанова 2015]. Для каждой тематической модели отбирались 200 статистически значимых лемм (по 10 из 20 тем), а затем проверялось, присутствует ли данная лемма в списках ключевых выражений. Практически все леммы были обнаружены в верхней трети списка ключевых выражений, которая считается наиболее информативной [Rose et al. 2009]. Полученные данные дают основания считать результаты работы алгоритма RAKE приемлемыми, а сам алгоритм RAKE в русскоязычной модификации пригодным для использования в лингвистических исследованиях. В дальнейшем планируется сравнение RAKE с другими алгоритмами и экспертиза результатов с участием информантов.

ЗАКЛЮЧЕНИЕ

В диссертационном исследовании мы осуществили описание основных явлений синтаксиса русского языка при помощи категориальной грамматики в терминах структур составляющих.

В ходе работы были выполнены все поставленные задачи:

- представлена теоретическая база исследования и обзор современного состояния синтаксического анализа в компьютерной лингвистике;
- разработана система правил, покрывающих основные синтаксические отношения, эта система правил представлена в виде формальной грамматики, работающей с инструментами NLTK и структурами составляющих;
- к парсеру подключен морфологический анализатор PyMorphy2, с помощью автоматического кодирования морфологических параметров словоформ в виде терминальных элементов в категориальной грамматике NLTK;
- проведена оценка работы созданного на основе данной грамматики инструмента, выделены основные преимущества и недостатки;
- предложено возможное применение подобного инструмента, представлены данные экспериментов по извлечению ключевых словосочетаний при помощи поверхностного синтаксического анализа и выделения групп составляющих в NLTK.

Возможные направления для развития исследования включают в себя ряд шагов для повышения точности синтаксического анализа: расширение и усовершенствование грамматики, подключение дополнительных инструментов для получения более подробной информации о словоформах на уроне токенизации, а также применение парсера для разнообразных задач автоматической обработки текста и его интеграция с другими инструментами NLTK.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. АОТ: Синтаксический анализ. Построение дерева зависимостей всего предложения [Электронный ресурс] // Aot.ru: информ.-справочный портал. URL: http://www.aot.ru/docs/synan.html (дата обращения: 17.05.2017).
2. Апресян, Ю.Д. Синтаксически и семантически аннотированный корпус русского языка: современное состояние и перспективы / Ю.Д. Апресян, И.М. Богуславский, Б.Л. Иомдин // Национальный корпус русского языка 2003–2005 г. Результаты и перспективы. М.: Индрик, 2005. — С. 193–214.
3. Буторов, В. Д. Моделирование синтаксиса естественного языка / В. Д. Буторов; В. В. Богданов; Г. Я. Мартыненко; А. С. Штерн; И. В. Азарова. Прикладное языкознание / отв. ред. А. С. Герд - СПб. : Изд-во СПбГУ, 1996. - с. 142-161
4. Гладкий, А.В. Синтаксические структуры естественного языка в автоматизированных системах общения. / А.В. Гладкий. М.: Наука, 1985. — 144 с.
5. Дружкин, К.Ю. Синтаксический анализатор лингвистического процессора. Этап 3: эксперименты по ранжированию синтаксических гипотез. / К.Ю. Дружкин, Л.Л. Цинман; ред. А.Е. Кибрик. // В кн.: Компьютерная лингвистика и интеллектуальные технологии. По материалам ежегодной международной конференции «Диалог» (2008). — М.: Изд-во РГГУ, 2008. — Вып. 7 (14). — С. 147–153.
6. Иорданская, Л.Н. Автоматический синтаксический анализ. Том 2. Межсегментный синтаксический анализ. / Л.Н. Иорданская; ред. А.А. Ляпунова, О.С. Кулагина. Новосибирск: Наука, 1967.
7. Каневский, Е.А., Семантико-синтаксический анализатор SEMSIN. / Е.А. Каневский, К.К. Боярский; ред. А.Е. Кибрик. // Научно-

технический вестник информационных технологий, механики и оптики. — СПб: Университет ИТМО, 2015. — Т. 15 — № 5 — С. 869-876.

8. Мельчук И.А. Автоматический синтаксический анализ. Т. 1. Общие принципы. Внутрисегментный синтаксический анализ. / И.А. Мельчук; ред. А.А. Ляпунова, О.С. Кулагина. Новосибирск: Наука, 1967.
9. Мельчук И.А. Опыт теории лингвистических моделей «Смысл <=> Текст». / И.А. Мельчук. М.: Школа «Языки русской культуры», 1999 — 346 с.
10. Митрофанова, О.А. Вероятностное моделирование тематики русскоязычных корпусов текстов с использованием компьютерного инструмента GenSim // Труды международной конференции «Корпусная лингвистика–2015». СПб., 2015.
11. Москвина А.Д. Разработка ядра синтаксического анализатора для русского языка на основе библиотек NLTK. / А.Д. Москвина, Д. Орлова, П.В. Паничева, О.А. Митрофанова. // Компьютерная лингвистика и вычислительные онтологии. Труды XIX Международной объединенной научной конференции «Интернет и современное общество». СПб: Университет ИТМО, 2016. — С.44-54.
12. Паничева П.В. Разработка лингвистического комплекса для морфологического анализа русскоязычных корпусов текстов на основе Pymorphy и NLTK. / Е.В. Протопопова, О.А. Митрофанова, А.Р. Мирзагитова. // Труды международной конференции “Корпусная лингвистика – 2015”. СПб: СпбГУ, 2015.
13. Протасов, С.В. Преимущества грамматики связей для русского языка // Труды международной конференции «Диалог 2005». М., 2005.
14. Русская грамматика. Т. 2: Синтаксис / гл. ред. Н. Ю. Шведова. М.: Наука, 1980.
15. Старостин, А.С. Алгоритм синтаксического анализа, используемый в системе морфо-синтаксического анализа «TREETON» / А.С.

Старостин, М.Г. Мальковский; ред. Л.Л. Иомдин, Н.И. Лауфер, А.С. Нариньяни и др. // В кн.: Компьютерная лингвистика и интеллектуальные технологии: Труды международной конференции «Диалог 2007». — М.: Изд-во РГГУ, 2007. — С. 516–524.

16. Старостин, А.С., Арефьев Н.В., Мальковский М.Г. Синтаксический анализатор «Treevial». Принцип динамического ранжирования гипотез. / А.С. Старостин, Н.В. Арефьев, М.Г. Мальковский; ред. А.Е. Кибрик. // В кн.: Компьютерная лингвистика и интеллектуальные технологии. По материалам ежегодной международной конференции «Диалог» (2010). — Вып. 9 (16). М.: Изд-во РГГУ, 2010. — С. 477-490.
17. Тестелец, Я.Г. Введение в общий синтаксис. / Я.Г. Тестелец. М.: Изд-во РГГУ, 2001.
18. Толдова, С.Ю. Оценка методов автоматического анализа текста 2011–2012: синтаксические парсеры русского языка. / Е.Г. Соколова, И. Астафьева, А. Гарейшина; ред. А.Е. Кибрик// В кн.: Компьютерная лингвистика и интеллектуальные технологии. По материалам ежегодной международной конференции «Диалог» (2012). — Вып. 11. — Т.2. — М.: Изд-во РГГУ, 2012. — С. 78-92.
19. Фитиалов, С.Я. Формальные грамматики. / С.Я. Фитиалов. Л., 1984.
20. Aho, A. V. and Ullman J. D. Theory of Parsing, Translation and Compiling, Vol. 1. Englewood Cliffs, NJ: Prentice Hall, 1972
21. Anisimovich K. V. Syntactic and semantic parser based on ABBYY Compreno linguistic technologies. / K. V. Anisimovich, K. Ju. Druzhkin, F. R. Minlos et al.; ред. А.Е. Кибрик. // В кн.: Компьютерная лингвистика и интеллектуальные технологии. По материалам ежегодной международной конференции «Диалог» (2012). — Вып. 11. — Т.2. — М.: Изд-во РГГУ, 2012. — С. 91-106.
22. Antonova A. A., Misyurev A. V. Russian dependency parser SyntAutom at the DIALOGUE-2012 parser evaluation task. / A. A. Antonova, A. V. Misyurev; ред. А.Е. Кибрик. // В кн.: Компьютерная лингвистика и

интеллектуальные технологии. По материалам ежегодной международной конференции «Диалог» (2012). — Вып. 11. — Т.2. — М.: Изд-во РГГУ, 2012. — С. 104-119.

23. Bird, S. Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. / S. Bird, E. Klein, E. Loper. Beijing, 2009.
24. Bloomfield, L. Language. New York, 1933.
25. Gildea, D. Synchronous Context-Free Grammars and Optimal Parsing Strategies. / D. Gildea, G. Satta // Computational Linguistics. 2016. – Vol. 42, No. 2: 207–243.
26. Iomdin L. ETAP parser: state of the art. / L. Iomdin, V. Petrochenkov, V. Sizov et al.; ред. А.Е. Кибрик. // В кн.: Компьютерная лингвистика и интеллектуальные технологии. По материалам ежегодной международной конференции «Диалог» (2012). — Вып. 11. — Т.2. — М.: Изд-во РГГУ, 2012. — С. 119-136.
27. Kahane, K. Why to choose dependency rather than constituency for syntax: a formal point of view. In J. Apresjan, M.-C. L'Homme, M.-C. Iomdin, J. Milicevi ´ c, A. Polguère, and L. Wanner, editors, Meanings, Texts, and other exciting things: A Festschrift to Commemorate the 80th Anniversary of Professor Igor A. Mel'ˇcuk, pages 257–272. Languages of Slavic Culture, Moscow, 2012.
28. Kakkonen T., Sutinen E. (2006) Towards a Framework for Evaluating Syntactic Parsers. / T. Kakkonen, E. Sutinen; eds.: T. Salakoski, F. Ginter, S. Pyysalo et al. // Advances in Natural Language Processing. Lecture Notes in Computer Science, vol 4139. Springer, Berlin, Heidelberg.
29. Kakkonen, T. Framework and Resources for Natural Language Parser Evaluation. // Computer Science Dissertation 19. Joensuu: University of Joensuu, 2007.
30. Korobov, M. Morphological Analyzer and Generator for Russian and Ukrainian Languages. Analysis of Images, Social Networks and Texts: 4th

International Conference, AIST 2015. Communications in Computer and Information Science, Springer. Yekaterinburg, 2015.

31. Levine, R.D. Head-Driven Phrase Structure Grammar Linguistic Approach, Formal Foundations, and Computational Realization. / R.D. Levine, W. D. Meurers. The Ohio State University.
32. Marneffe, de M.-C. Generating Typed Dependency Parses from Phrase Structure Parses. / M.-C. de Marneffe, B. MacCartney, C. D. Manning. 2006.
33. Mirroshandel, S.A. Integrating Selectional Constraints and Subcategorization Frames in a Dependency Parser Free Access. S.A. Mirroshandel, A. Nasr // Computational Linguistics. Vol. 42, No. 1: 55–90. 2016.
34. Nivre, J. Dependency Grammar and Dependency Parsing. 2007.
35. Nivre, J. Maltparser: A language-independent system for data-driven dependency parsing. / J. Nivre, J. Hall, J. Nilsson et al. //Natural Language Engineering. 2007. – 13:95–135
36. Pollard, C. Head-Driven Phrase Structure Grammar. / C. Pollard, I. A. Sag. Chicago: University of Chicago Press and Stanford, 1994.
37. Rose S.J., Cowley W.E., Crow V.L., Cramer N.O. (2009), Rapid Automatic Keyword Extraction for Information Retrieval and Analysis. / Rose S.J., Cowley W.E., V.L. Crow et al. // URL: http://www.google.co.ve/patents/US8131735
38. Sharoff S. The proper place of men and machines in language technology: Processing Russian without any linguistic knowledge. / S. Sharoff, J. Nivre; ред. А.Е. Кибрик. // В кн.: Компьютерная лингвистика и интеллектуальные технологии. По материалам ежегодной международной конференции «Диалог» (2011). — Вып. 10. — Т.2. — М.: Изд-во РГГУ, 2011. — С. 591-604.

39. Sleator, D. Parsing English with a Link Grammar. / D. Sleator, D. Temperley. // Carnegie Mellon University Computer Science technical report CMU-CS-91-196, October 1991.
40. Taylor, A. (2003). The Penn treebank: An overview. / A. Taylor, M. Marcus, B. Santorini; ed. A. Abeille. // Building and Using Parsed Corpora, volume 20 of Text, Speech and Language Technology. Springer, 2003.
41. Zhang, X. Transition-Based Parsing for Deep Dependency Structures. / X. Zhang, Y. Du, W. Sun et al. // Computational Linguistics. 2016 – Vol. 42, No. 3: 353–389.

```
XP -> NP
XP -> VP
XP -> AdjP
XP -> AdvP
XP -> COMP
XP -> PP
XP -> NUMRNP
XP -> NUMR
XP -> S
XP -> CONJP
XP -> PREDP
XP -> PrtfP
S[-inv] -> NP[C=nomn, NUM=?n, PER=?p, G=?g] VP[NUM=?n,
PER=0, G=?g]
S[-inv] -> NP[C=nomn, NUM=?n, PER=?p, G=?g] VP[NUM=?n,
PER=?p, G=0]
S[-inv]   ->   AdjP[C=nomn,   NUM=?n,   PER=?p,   G=?g]
VP[NUM=?n, PER=0, G=?g]
S[-inv]   ->   AdjP[C=nomn,   NUM=?n,   PER=?p,   G=?g]
VP[NUM=?n, PER=?p, G=0]
S[+adj]   ->   NP[C=nomn,   NUM=?n,   PER=?p,   G=?g]
AdjP[C=nomn, NUM=?n, PER=?p, G=?g]
S[-inv] -> NUMRNP[C=nomn] VP
S[+inv] -> VP[NUM=?n, PER=0, G=?g] NP[C=nomn, NUM=?n,
PER=?p, G=?g]
S[+inv] -> VP[NUM=?n, PER=?p, G=0] NP[C=nomn, NUM=?n,
PER=?p, G=?g]
S[+advp] -> AdvP S[+advp]
S[+advp] -> NP[C=gent, NUM=?n, PER=?p, G=?g] AdvP
S[+predp] -> NP[C=datv] PREDP
S[+comp] -> NP[C=datv] COMP
S -> CONJ S
S -> NP[C=nomn, NUM=?n] '-' NP[C=nomn, NUM=?n]
S[+S] -> S ',' CONJ S
S[+S] -> S ',' S
S[+S] -> S ',' VP
PREDP -> PREDP  VP[TR=?tr, NUM=0, PER=0, G=0]
PREDP -> AdvP PREDP
PREDP -> PRED
NUMRNP[C=?c] -> NUMR[C=?c] NP[C=?c, NUM=plur]
NUMRNP[C=accs] -> NUMR[C=accs] NP[C=gent]
NUMRNP[+nomn, C=nomn] -> NUMR[C=nomn] NP[C=gent]
NUMR[C=?c] -> NUMR[C=?c] NUMR[C=?c]
```

```
PP[C=?c, G=?g, NUM=?n, PER=?p] -> PREP NP[C=datv, G=?g,
NUM=?n, PER=?p]
PP[C=?c, G=?g, NUM=?n, PER=?p] -> PREP NP[C=loct, G=?g,
NUM=?n, PER=?p]
PP[C=?c, G=?g, NUM=?n, PER=?p] -> PREP NP[C=accs, G=?g,
NUM=?n, PER=?p]
PP[C=?c, G=?g, NUM=?n, PER=?p] -> PREP NP[C=gent, G=?g,
NUM=?n, PER=?p]
PP[C=?c, G=?g, NUM=?n, PER=?p] -> PREP NP[C=ablt, G=?g,
NUM=?n, PER=?p]
PP -> PREP NUMRNP[-nomn]
PP[+advb] -> AdvP PP
PP -> PP AdvP
AdvP[+conj] -> AdvP CONJ AdvP
AdvP[+numr] -> AdvP NUMRNP
AdvP -> ADVB AdvP
AdvP -> ADVB
PrtfP[C=?c, G=?g, NUM=?n] -> PRTF[C=?c, G=?g, NUM=?n]
PrtfP[+pp, C=?c, G=?g, NUM=?n] -> PrtfP[C=?c, G=?g,
NUM=?n] PP
PrtfP[+instr, C=?c, G=?g, NUM=?n] -> PrtfP[C=?c, G=?g,
NUM=?n] NP[C=ablt]
PrtfP[+instr, C=?c, G=?g, NUM=?n] -> PrtfP[C=?c, G=?g,
NUM=?n] AdvP
NP[+adjf, C=?c, G=?g, NUM=sing] -> AdjP[C=?c, G=?g,
NUM=sing] NP[C=?c, G=?g, NUM=sing]
NP[+adjf, C=?c, NUM=plur] -> AdjP[C=?c, NUM=plur]
NP[C=?c, NUM=plur]
NP[+gent, C=?c, G=?g, NUM=?n] -> NP[C=?c, G=?g, NUM=?n]
NP[C=gent]
NP[+prtf, C=?c, G=?g, NUM=sing] -> PrtfP[C=?c, G=?g,
NUM=sing] NP[C=?c, G=?g, NUM=sing]
NP[+prtf, C=?c, NUM=plur] -> PrtfP[C=?c, NUM=plur]
NP[C=?c, NUM=plur]
NP[C=?c, +conj] -> NP[C=?c] CONJ NP[C=?c]
NP[C=?c, G=?g, NUM=?n, PER=?p] -> NOUN[C=?c, G=?g,
NUM=?n, PER=?p]
NP[C=?c, NUM=?n, PER=?p, G=?g] -> NPRO[C=?c, NUM=?n,
PER=?p, G=?g]
NP[+pp, C=?c, G=?g, NUM=?n] -> NP[C=?c, G=?g, NUM=?n]
PP
NP[+prcl, C=?c, G=?g, NUM=?n] ->PRCL NP[C=?c, G=?g,
NUM=?n]
NP[+particip, C=?c, NUM=plur] -> NP[C=?c, NUM=?n] ','
PrtfP[C=?c, NUM=?n]
```

```
VP[+advb, TR=?tr, TENSE=?t, G=?g, NUM=?n, PER=?p] ->
AdvP VP[TENSE=?t, TR=?tr, G=?g, NUM=?n, PER=?p]
VP[+advb, TENSE=?t, G=?g, NUM=?n, PER=?p] ->
VP[TENSE=?t, G=?g, NUM=?n, PER=?p] AdvP
VP[+infn, TR=?tr, NUM=?n, PER=?p, G=?g] -> VP[NUM=?n,
PER=?p, G=?g] VP[TR=?tr, NUM=0, PER=0, G=0]
VP[+objt, NUM=?n, PER=?p, G=?g] -> VP[-objt, NUM=?n,
PER=?p, G=?g, TR=tran] NP[C=accs]
VP[+objt, NUM=?n, PER=?p, G=?g] -> VP[-objt, NUM=?n,
PER=?p, G=?g, TR=tran] NP[C=gent]
VP[+objt, NUM=?n, PER=?p, G=?g, TR=tran] -> NP[C=accs]
VP[-objt, NUM=?n, PER=?p, G=?g, TR=tran]
VP[TENSE=?t, G=?g, NUM=?n, PER=?p, TR=?tr] ->
INFN[TENSE=?t, G=?g, NUM=?n, PER=?p, TR=?tr]
VP[+instr, TENSE=?t, G=?g, NUM=?n, PER=?p] ->
VP[TENSE=?t, G=?g, NUM=?n, PER=?p] NP[C=ablt]
VP[+pp, TENSE=?t, G=?g, NUM=?n, PER=?p] -> VP[TENSE=?t,
G=?g, NUM=?n, PER=?p] PP
VP[+pp, TENSE=?t, G=?g, NUM=?n, PER=?p] -> PP
VP[TENSE=?t, G=?g, NUM=?n, PER=?p]
VP[+datv, TENSE=?t, G=?g, NUM=?n, PER=?p] -
>VP[TENSE=?t, G=?g, NUM=?n, PER=?p] NP[C=datv]
VP[+comp, TENSE=?t, G=?g, NUM=?n, PER=?p] ->
VP[TENSE=?t, G=?g, NUM=?n, PER=?p] COMP
VP[+numr, TENSE=?t, G=?g, NUM=?n, PER=?p] ->
VP[TENSE=?t, G=?g, NUM=?n, PER=?p] NUMRNP
VP[+neg, TENSE=?t, G=?g, NUM=?n, PER=?p] -> 'не'
VP[TENSE=?t, G=?g, NUM=?n, PER=?p]
VP[TR=?tr, TENSE=?t, G=?g, NUM=?n, PER=?p] ->
VERB[TR=?tr, TENSE=?t, G=?g, NUM=?n, PER=?p]
AdjP[+advb, C=?c, G=?g, NUM=?n] -> AdvP AdjP[C=?c,
G=?g, NUM=?n]
AdjP[+advb, C=?c, G=?g, NUM=?n] -> AdjP[C=?c, G=?g,
NUM=?n] PP
AdjP[C=?c, G=?g, NUM=?n, +conj] -> AdjP[C=?c, G=?g,
NUM=?n] CONJ AdjP[C=?c, G=?g, NUM=?n]
AdjP[C=?c, G=?g, NUM=?n] -> ADJF[C=?c, G=?g, NUM=?n]
AdjP[+adjs, G=?g, NUM=?n] -> ADJS[G=?g, NUM=?n]
COMP[+conj] -> COMP CONJ COMP
COMP[+advb] -> AdvP COMP
COMP[+noun, +comp] -> COMP[-comp] NP[C=gent]
COMP[+vp] -> COMP VP
INFN[+conj] -> INFN CONJ INFN
CONJP[+vp] -> CONJ VP
```

ПРИЛОЖЕНИЕ Б. Корпус тестовых предложений

rating 2980 – 3640:

крышка люка беззвучно откинулась

волшебник постарался унять беспокойство

синяка криво дернул плечом

форма приняла человеческий облик

монахи приняли позу приветствия

собравшихся охватило легкое беспокойство

генри изучал карту местности

прозрачный купол беззвучно растаял

золотая маска тускло сверкала

древняя раса перестала существовать

маги обменялись быстрыми взглядами

жертва помогает поймать преступника

оттого нынче дорог мед

серая пена бегущих собак

вверху плыли редкие облачка

раб неохотно прекратил атаку

чувствовалось глубокое знание предмета

рис постарался улыбнуться детям

остаются металлические полосы юбки

кот ненадолго прервал ужин

rating 2440 – 2980:

неторопливо допил прохладный сок .

герцог удивленно приподнял бровь .

вездеход плавно свернул направо .

робот работает круглые сутки .

четыреста шестьдесят фунтов золота .

девушке отвели небольшую каюту .

тишину разорвал пронзительный вопль .
стихло рычание многочисленных глоток .
сергей выдержал небольшую паузу .
мох покрывал валун целиком .
цветок придал девушке уверенности .
таков уровень развития науки .
отовсюду слышались веселые песни .
холл наполнился грохотом выстрелов .
имеются многочисленные человеческие жертвы
шульгин удивленно приподнял бровь
члены совета удивленно переглянулись
калинов почесал кончик носа
потянулись долгие мгновения ожидания
тонкий свист заполнил кабину
rating 148 – 181:
твоя жизнь вне опасности
внутрь вела маленькая дверь
хромой снял свою маску
рис услышал свои слова
вдруг раздался громкий писк
решение пришло само собой
другой вдруг изумленно смеялся
об остальном узнаешь позже
тяжелый меч потащил назад
вокруг вновь раздался хохот
ноги едва держали тело
лена протянула вперед руку
воздух вокруг казался живым
артем непроизвольно отступил назад
оставалось самое важное дело

пошел дорогой своей судьбы
нужно обсудить одну проблему
девочке лучше покинуть планету
лучше заняться своим делом
ночь пролетела удивительно быстро
постепенно воздух стал теплее
rating 66-81:
тебя там ждет смерть
вас там четыреста человек
там целая гора ящиков
наши люди остались там
время перестало иметь значение
поль ждал этого вопроса
нас осталось совсем немного
твоя жизнь будет прекрасна
тут больше нечего делать
вот такой длинный круг
кубок исчез со стола
сейчас будет действительно трудно
следующий трон будет мой
нас осталось совсем мало
такой человек очень опасен
глаза мальчика снова закрылись
нас осталось очень мало
однако такого шанса нет
кот совсем закрыл глаза
синие глаза стали холодными
rating 36-40:
девушка уже мирно спала
раздался еще один голос

птица подошла еще ближе
валентин только руками развел
иначе только половину увижу
потом будет уже поздно
затем еще раз , еще
детей спасти еще можно
рано еще делать выводы
впереди еще долгий путь
еще один трудный день
просто еще одна дорога
еще более долгая пауза
только недавно ходить перестал
нам остается только ждать
только нам туда нельзя
прошло еще немного времени
отряд уже уходил вперед
хозяина вот только надо
атаковать только ближайшие цели