Глава 2. Компьютерный синтаксис

Автор: Алексей Владимирович Добров, выпуск 2008 года, ассистент кафедры математической лингвистики СПбГУ

1. Разные подходы к анализу синтаксических структур

1.1. Что такое парсинг

При решении многих задач, относящихся к машинной обработке текстовых данных, требуется точная и подробная информации о структуре предложений. Необходимость в этой информации возникает при создании систем машинного перевода, интеллектуального (семантического) информационного поиска, извлечения фактических данных (data mining) и мнений (opinion mining), автоматической рубрикации и реферирования текстов — систем, так или иначе связанных с **автоматическим пониманием текстов** (АПТ/NLU). Обычно в них используются методы, основанные на «правилах» (rule-based methods) — математические формализации языковой грамматики и семантики. Вместе с тем, и статистические методы (statistical methods), в том числе — некоторые методы машинного обучения (machine-learning) — часто основываются на синтаксической разметке текстовых коллекций.

Парсинг (синтаксический анализ) в широком смысле — это анализ структуры любых текстовых данных. В более узком понимании термин «парсинг» означает процедуру машинного анализа структуры естественноязыкового текста, в том числе — структуры предложения.

Простейший анализ структуры предложения входит в школьную программу. Например, в предложении *Белка нашла большой орех* требуется подчеркнуть подлежащее и сказуемое, определить, к чему относится прилагательное *большой*, подписать вопросы над стрелками: *кто? что? какой?* Это лишь один из возможных способов синтаксического анализа. В некоторых западных школах ученики расставляют в предложении квадратные скобки. Казалось бы, разница не столь велика, но на самом деле за расстановкой скобок стоит иное понимание устройства предложения.

Три основных подхода к описанию синтаксиса — это **грамматики зависимостей** (ГЗ), **грамматики непосредственных составляющих** (НС-грамматики) и комбинированные теории, например, теория **синтаксических групп** (ТСГ).

1.2. Грамматики зависимостей

В отечественной лингвистике и в некоторых западных научных школах принят подход, основанный на ГЗ, который представляет структуру предложения в виде дерева (графа) зависимостей. Его основоположником принято считать французского лингвиста Луи Теньера ([Теньер 1988]). С точки зрения ГЗ, строение предложения подобно строению молекулы: предложение состоит из слов и связей между ними. Вместе с тем, синтаксическую связь нельзя назвать двусторонней; в большинстве случаев синтаксическая связь является подчинительной. Синтаксические связи «устанавливают между словами отношения зависимости»: из двух слов одно является *главным*, а другое — *зависимым*. Например, в словосочетании *Большая Советская Энциклопедия* имеется две связи, изображённые на рис. 1. Эти связи образуют отношение зависимости: в обеих связях главной является словоформа *Энциклопедия*; словоформы *Большая* и *Советская*, соответственно, оказываются зависимыми.

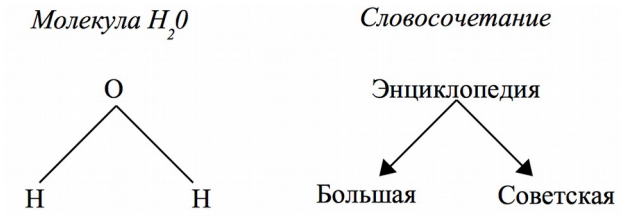


Рис. 1. Дерево зависимостей подобно строению молекулы, но связи обладают направленностью

Количество и состав возможных зависимых слов определяется способностью главного слова вступать с ними во взаимодействие. Эта способность называется ***валентностью***. Главное слово может само стать зависимым по отношению к какому-либо другому слову, но только к одному (именно поэтому синтаксическая структура оказывается древовидной). Получается, что любое связное сочетание слов (словосочетание, фраза) обладает одним центральным (корневым) элементом, т.е. одно и только одно слово является главным во всей фразе, а все остальные подчинены ему или другим словам.

В словосочетаниях типа *умный студент* носителям русского языка свойственно интуитивно считать слово *студент* главным, а слово *умный* — зависимым (так учили в школе — задавать вопрос *какой* «от существительного к прилагательному», хотя иностранные учащиеся часто удивляются, почему не задать вопрос *кто* «в обратном направлении»). Как передать эту интуицию компьютеру? Дело осложняется еще тем, что в лингвистике не существует формально строгой процедуры обнаружения синтаксической зависимости. Основные критерии для «человеческого» выявления синтаксических зависимостей описаны Я. Г. Тестельцом в первой главе его книги «Введение в общий синтаксис [Тестелец 2001], однако при компьютерном моделировании правила выявления зависимостей вынужденно постулируются, причём по-разному в разных системах.

Как научить компьютер выстраивать структуры зависимостей? Прямой путь — это взять предложение, в котором методами морфологического анализа (описанными в главе 1) для каждой словоформы уже определена возможная часть речи и грамматические признаки. В этом предложении нужно найти основной глагол (если он есть) и понять, сколько зависимых слов должно быть у этого глагола. Затем необходимо найти эти зависимые слова — здесь помогут их формы. Например, глаголу *видеть* требуется дополнение — скорее всего, это существительное в винительном падеже. После этого можно проанализировать оставшиеся слова и найти для них возможные вершины. Для этого нужно заранее предоставить компьютеру множество правил, которые будут тесно связаны со словарём.

Существующие алгоритмы для построения деревьев зависимостей обычно основаны на правилах продукций — условных переходах вида «*если* …, *то* …». Эти правила позволяют реализовать произвольный механизм логического вывода и потому широко применяются при представлении знаний в экспертных системах. Совокупность правил продукций представляет собой продукционную модель, которая теоретически «производит» все возможные корректные выводы о заданном наборе объектов. В применении к синтаксису правила продукций часто формулируются в виде замен одних символьных последовательностей другими (заменяемая строка является условием, а замена — выполняемым действием). Например, правило

продукции может выглядеть как «Vt S4 → Vt ->[Acc] S4», где «Vt» — переходный глагол, «S4» — имя существительное в винительном падеже, «->[Acc]» — подчинительная связь между ними. При этом правила обычно нумеруются, что позволяет исключить работу одних правил, если ещё не сработали другие. Основная сложность при создании таких моделей состоит в том, что правила в какой-то момент начинают противоречить друг другу. Эта проблема часто решается с помощью дополнительных управляющих структур, искусственно разрешающих возникающие противоречия.

Метод фулькрумов ([Беляева 1996]) основан на поиске «фулькрумов» — корней синтаксических деревьев или некоторых опорных ключевых слов, которые далее связываются с соседними словами в несколько «прогонов» в соответствии с правилами продукций. **Метод фильтров** ([Лесерф 1963]) предполагает одновременное применение всех правил продукций, после которого получившаяся структура (ориентированный граф) подвергается фильтрации.

Анализ на основе грамматики зависимостей вызывает ряд проблем. Кроме подчинительных связей в языке есть и сочинительные, которые сам Теньер рассматривает как «особые функции», они не получают трактовки в виде подчинительных связей, и возникает противоречие: ведь если синтаксические связи устанавливают между словами именно отношение зависимости, то почему некоторые связи оказываются «равнее»?

В компьютерных реализациях ГЗ преобладают подходы, в которых сочинительная связь раскладывается на подчинительные. Сочинительная связь всегда обозначается союзом или знаком препинания, который можно искусственно включить в дерево зависимостей, наравне со словами (Теньер же, напротив, старался исключить из синтаксической структуры любые служебные слова, и, тем более, знаки препинания). Вместе с тем, если союз *и* в словосочетании *Вася и Петя* является зависимым, то главным получается *Вася* или *Петя*, но обе словоформы используются в единственном числе, что противоречит множественному числу глагольной формы в предложении *Вася и Петя играли*. Если же главным является союз, то тогда словосочетание *Вася и Петя* вообще не обладает свойствами существительного.

Придаточные предложения, пожалуй, представляют собой самое «слабое» место ГЗ. В русском языке ГЗ хуже всего «справляется» с определительными придаточными предложениями со словом *который*. В дереве зависимостей предложения *Это был человек, достижениями которого гордилась вся страна* форма *которого* оказывается подчинена форме *достижениями*, что объясняет родительный падеж, но абсолютно не объясняет мужской род, напрямую зависящий от определяемого существительного *человек*. Столкнувшись с этой проблемой, Теньер вводит в синтаксическую структуру ещё один тип связи — анафорическую связь. Устанавливая между формами *человек* и *которого* анафорическую связь, Теньер приравнивает в данном случае согласование в категории рода к совпадению рода личных местоимений (***анафоров***) и предшествующих им имён существительных (***антецедентов***).

Такая трактовка выглядела бы логичной, если бы анафорическая связь вообще была грамматической (синтаксической), что вызывает некоторые сомнения. Действительно, очевидная анафорическая связь между словоформами *молодёжи* и *они* в грамматически правильном предложении *Ленин сказал молодёжи, что они должны учиться* была бы невозможной: ведь у *молодёжи* число — единственное, а у *они* — множественное. Тем не менее, предложение *Ленин сказал молодёжи, что она должна учиться* не выглядит более правильным: для смысла не важно, что собирательное существительное *молодёжь* имеет единственное число, ведь оно обозначает совокупность людей, которая вполне может быть обозначена

словом *они*. В то же время в придаточных предложениях с *который* такая вариативность недопустима (ср. *молодёжь, которая должна учиться* и **молодежь, которые должны учиться*). Следовательно, связь между словом *который* и определяемым существительным является не только смысловой, но и грамматической, но если это так, то нарушается постулат ГЗ о том, что каждое слово может быть грамматически зависимым не более чем от одного слова.

Факты, напрямую свидетельствующие о недостаточности ГЗ при работе с естественным языком, были обнаружены А.В. Гладким. Предложение «*По графику мы работаем в среду*» [Гладкий 1985, с. 119] синтаксически неоднозначно. Первая возможная трактовка состоит в том, что компонент *По графику* относится непосредственно к глаголу *работаем*. В этом случае одно из возможных значений предложения можно было бы сформулировать как ‘В среду мы будем работать в соответствии с графиком, а в остальные дни — возможно и не в соответствии’. Вместе с тем, у предложения есть и вторая возможная трактовка — ‘В соответствии с графиком, мы должны работать именно в среду’. Эта трактовка может быть объяснена только тем, что компонент *По графику* относится ко всему предложению *мы работаем в среду* в целом. Если не учитывать второе возможное толкование этого предложения, то не будет правильно разобрано, например, предложение *По прогнозам предсказателей завтра будет конец света* — будет выстроена смехотворная трактовка ‘Завтра конец света будет в соответствии с прогнозами предсказателей, а в остальные дни — возможно и не в соответствии’.

В рамках ГЗ обе трактовки выглядят совершенно одинаково: словоформа *по* подчинена словоформе *работаем* и не может быть подчинена целому предложению — ведь отношения зависимости выстраиваются между словами. Чтобы учесть обе возможные трактовки в ГЗ, необходимо предположить неоднозначность самих синтаксических связей и, в частности, связи между глаголом и предлогом. В этом случае, однако, такую неоднозначность придётся предполагать всегда, в том числе — в обычных предложениях типа *Мы едем в Москву*. Для таких предложений пришлось бы выстраивать противоестественные трактовки вроде ‘Едем мы в направлении Москвы, а все остальные перемещения, возможно, осуществляем в другом направлении’.

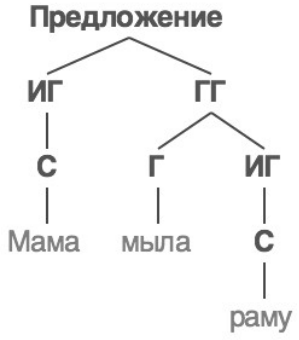
Эти примеры показывают, что синтаксические связи выстраиваются не только между словами, но и между группами слов, что плохо согласуется с ГЗ, зато находит подтверждение в НС-грамматиках, где связи между словами напрямую не выстраиваются, а выводятся из структуры непосредственных составляющих.

1.3. Грамматики непосредственных составляющих

Непосредственные составляющие (НС, фразы) — это части предложения или словосочетания, которые входят в него непосредственно, т.е. не являются компонентами более крупных его частей.

На рисунке 2 приведён пример структуры предложения *Мама мыла раму*. У этого предложения имеются две НС: *Мама* и *мыла раму*. В свою очередь, *мыла раму* имеет НС *мыла* и *раму*, т.е. эти НС не входят в предложение непосредственно, а являются компонентами более крупной его части. Вообще, ни одна словоформа не является НС целого предложения, но каждая словоформа является НС по отношению к тем или иным его составляющим.

НС-грамматика объясняет явления синтаксиса исключительно в терминах НС, оперируя теми или иными классами НС, такими как Предложение, Именная Группа, Глагольная Группа, и т.д. Названия этих классов соответствуют, на самом деле, представлениям о том, что словоформа той или части речи является главной во всей фразе, например, в глагольной группе главным является глагол, однако эти названия могут быть и иными, никак не связанными с



частями речи.

Рис. 2. Структура непосредственных составляющих

В отличие от стандартной ГЗ, НС-грамматика не только отражает строение предложения, но и объясняет способ его порождения (деривацию). В грамматике задаётся ряд правил «переписывания», позволяющих преобразовывать абстрактные обозначения (Именная группа, Глагол) в более мелкие линейные цепочки, которые также подвергаются преобразованиям до тех пор, пока они не будут состоять из «конечных» (терминальных) обозначений — конкретных словоформ или морфем. Поскольку деривация высказывания отражает способ его порождения, процедуру синтаксического разбора можно свести к поиску всех возможных деривaций анализируемого высказывания. К числу таких алгоритмов относятся **нисходящие** (Top-Down), **восходящие** (Botom-Up) и **комбинированные** (Left-Corner) алгоритмы.

Нисходящие алгоритмы строят синтаксическое дерево сверху вниз, т.е. начиная с самой верхней НС. Эти алгоритмы фактически воспроизводят правила переписывания, но применяют их сначала к первым обозначениям в имеющейся последовательности. Эти алгоритмы сразу строят некую структуру, однако для того, чтобы привести её в соответствие с фактическими входными данными, им требуется осуществить непредсказуемый по длительности перебор возможных деривaций, порождающих любые другие цепочки (фактически, перебираются все высказывания, порождаемые грамматикой, до тех пор, пока начальные компоненты порождённого и анализируемого высказываний не совпадут; затем процедура повторяется до тех пор, пока не найдётся деривация всего высказывания).

Восходящие алгоритмы работают по противоположной схеме: движение происходит снизу вверх. Нижестоящие НС заменяются на вышестоящие до тех пор, пока для такой замены не потребуется соединить несколько НС. Движение вверх в последнем случае возобновится только после того, как все нижестоящие позиции будут заполнены. Эти алгоритмы обладают высокой производительностью и не порождают ничего лишнего, но они либо неприменимы в случаях, когда возможны пропуски составляющих (эллипсис), либо допускают эллипсис, но неприменимы к так называемым рекурсивным грамматикам.

Комбинированные алгоритмы предполагают совмещение этих двух стратегий. Эти алгоритмы обладают более высокой производительностью, чем нисходящие, и применимы к

рекурсивным грамматикам с эллиптическими НС.

Порядок построения НС тремя видами алгоритмов в предложении *Мама мыла раму* изображён на рисунке 3 (здесь изображена работа более простого в реализации поиска «в глубину», при поиске «в ширину» порядок меняется). Следует отметить, что НС «ГГ» при реализации восходящего или комбинированного алгоритма встраивается в НС «Предложение» только после шага 10, т.е. когда все её дочерние позиции заполнены.

Рис. 3 Порядок работы нисходящих, восходящих и комбинированных алгоритмов

Рассмотрим пример простейшей НС-грамматики и одной из возможных дериваций.

Грамматика:

1) Предложение (П) → Именная группа (ИГ) + Глагольная группа (ГГ)

2) ИГ → форма имени существительного (С)

3) С → *Мама*

4) С → *раму*

5) ГГ → ГГ + ИГ

6) ГГ → форма глагола (Г)

7) Г → *мыла*.

Деривация (в скобках между стрелками указаны номера правил грамматики):

П →(1)→ ***ИГ*** + *ГГ* →(2)→ ***С*** + *ГГ* →(3)→ *Мама* + ***ГГ*** →(5)→ *Мама* + ***(ГГ*** + *ИГ)* →(6)→ *Мама* + ***(Г*** + *ИГ)* →(7)→ *Мама* + *(мыла* + ***ИГ)*** →(2)→ *Мама* + *(мыла* + ***С)***; →(4)→ *Мама* + *(мыла* + *раму)*.

В результате последовательного применения правил грамматики выстраивается последовательность словоформ, соответствующая базовому порядку слов в предложении *Мама мыла раму*.

Важный недостаток НС-грамматик связан с тем, что они предполагают соответствие между линейным порядком слов и фразовой структурой. То есть, они не учитывают, что в языках со свободным порядком слов составляющая может разрываться. Именно это происходит с составляющей *нерукотворный памятник* в строчке Пушкина *Я памятник себе воздвиг нерукотворный*. Разрывы происходят и в языках со строгим порядком слов, например, в вопросительных предложениях, ср. *Whom are you waiting for*? (*Кого вы ждёте? / Кого ты ждёшь?*)

Чтобы объяснить различные порядки слов, к правилам переписывания в процедуре деривации пришлось добавить перемещения. Например, для объяснения порядка слов в вопросительном предложении в деривацию добавляется правило вида «С Г Вопр → Вопр С Г», где «Вопр» — вопросительное слово, перемещаемое в начало предложения. Перемещения срабатывают уже после порождения цепочки конечных обозначений с базовым порядком слов, поэтому формулируются именно в терминах конечных обозначений (С, Г, но не ИГ или ГГ). Синтаксическая структура, возникающая до перемещений, называется глубинной (ГСС, deep structure); перемещения трансформируют ГСС в поверхностную структуру (ПСС, surface structure).

1.4. Комбинированные теории анализа предложения

Подход, объединяющий ГЗ и НС-грамматики, предполагает установление отношений зависимости не только между словами, но и между группами слов. На материале русского языка наиболее подробно была разработана в теории синтаксических групп А. В. Гладкого. Она допускает включение в структуру предложения тех групп, которые участвуют в отношении зависимости «целиком», а не посредством одной словоформы (ср. приведенное выше предложение *По графику мы работаем в среду*). Такой подход позволял легко обрабатывать разрывные составляющие.

В некоторых прикладных задачах комбинированный подход реализуется с помощью расширенных сетей переходов (Augmented Transition Networks, ATN) и основанных на них ATN-грамматиках, которые удачно адаптируются к русскому языку. В частности, именно на этой грамматике основан синтаксический анализ предложений в системе ПРОМТ. ATN-грамматики — это комбинация НС-грамматик и Марковских цепей (сетей переходов), в которых переходы производятся не только от слова к слову (что приводит к проблемам при анализе сложных фраз), но и между словами и НС или непосредственно между НС, однако при этом для каждого класса НС строится своя независимая сеть, позволяющая анализировать эту НС по-отдельности. Кроме того, в ATN-грамматиках используются специальные «регистры» для обеспечения грамматического согласования и управления. Расширенные сети переходов являются самостоятельными алгоритмическими моделями, по мощности эквивалентными машинам Тьюринга.

2. Неоднозначность и проблема комбинаторного взрыва

Главная проблема синтаксического анализа связана с неоднозначностью языковых единиц. Даже, казалось бы, однозначные слова, вроде слова *телевизор*, имеют так называемые «контекстуальные» значения, т.е. значения, проявляющиеся в особых условиях: например, телевизором, хоть и редко, могут назвать и монитор вычислительной машины, и даже, в

переносном смысле, телевидение — ср. *Телевизору нельзя верить*; *Как их вообще пустили в телевизор*.

Помимо очевидной многозначности слов (**лексической неоднозначности**), есть и менее очевидная многозначность грамматических форм (**морфологическая неоднозначность**): ср., например, предлог *при* и форму повелительного наклонения единственного числа глагола *переть*, или форму винительного падежа слова *семья* (*семью*) и форму творительного падежа числительного *семь*. Морфологическая неоднозначность может показаться редким явлением, но только потому, что она редко бывает очевидной.

Больше всего проблем вызывает, однако, не морфологическая, а **синтаксическая неоднозначность**. Из-за морфологической неоднозначности одна и та же фраза может иметь несколько разных пониманий, которые соответствуют разным структурам (*Он увидел их семью своими глазами*, *Эти типы стали есть в цехе*). Более того, одна и та же фраза может быть понята по-разному даже при однозначности всех её словоформ (ср. *Вася встретил Петю в коридоре* и *Вася встретил Петю в костюме*). В действительности, обе фразы имеют, по крайней мере, три возможные синтаксических структуры. Им соответствует три разные ситуации. Для первой фразы они таковы:

1) Вася встретил Петю, и это произошло в коридоре
2) Вася встретил Петю, одетого в коридор
3) Вася, одетый в коридор, встретил Петю

По наблюдению И.А. Мельчука, одно предложение в первой статье Конституции США допускает 16 различных синтаксических структур. В научно-технических текстах неоднозначность встречается в большинстве фраз.

Чтобы найти верный вариант разбора целого предложения, можно попытаться перебрать все комбинации вариантов разбора его частей. Проблема в том, что при компьютерном анализе это приводит к так называемому **комбинаторному взрыву**.

Действительно, если три словоформы в предложении имеют хотя бы два варианта разбора, то в сумме получается $2^3 = 8$ возможных комбинаций, которые необходимо рассмотреть. 7 неоднозначных словоформ в предложении приводят уже к $2^7 = 128$ комбинациям. Время работы машины напрямую зависит от количества перебираемых комбинаций, которое в данном случае растёт экспоненциально.

Несложно убедиться в том, что в реальных текстах число неоднозначных словоформ часто оказывается выше, чем семь, а число вариантов разбора часто превышает два даже в русскоязычном тексте. Если же речь идёт об арабском тексте, где чаще всего пропускаются обозначения гласных фонем, то семь — это нормальное, даже довольно низкое количество версий разбора для каждой словоформы ($7^7 = 823543$).

Таким образом, даже неоднозначности словоформ было бы достаточно для серьёзных проблем с производительностью парсеров. Между тем, неоднозначность словоформ — это лишь часть проблемы. Выше рассматривались предложения *Вася встретил Петю в коридоре* и *Вася встретил Петю в костюме*, различающиеся лишь одной словоформой, причём однозначной. Эти два предложения имеют различные синтаксические структуры, но установить этот факт машина может только на основании семантических, а не синтаксических знаний. Вместе с тем, в данном случае все словоформы в предложениях однозначны. Синтаксическая неоднозначность возникает из-за того, что одна и та же цепочка слов в одних и тех же грамматических формах может быть порождена разными, неравнозначными способами (дерива-

циями). В терминах теории синтаксических групп, группа *в костюме* может относиться как к словоформе *Вася*, так и к словоформе *Петя*, или даже ко всему предложению *Вася встретил Петю*, если допустить, что костюм был местом встречи.

Кроме того, в языке встречаются пропуски словоформ — эллипсис. Чтобы машина могла обнаруживать эллипсис там, где они действительно есть, ей приходится сначала предполагать подобные пропуски везде. Это выводит масштабы комбинаторного взрыва на уровень миллионов комбинаций.

В определённом смысле масштаб и сложность, да и сама суть проблемы, о которой идёт речь, напоминают известную задачу коммивояжёра. Эта задача обычно формулируется так. Имеется набор городов, соединённых дорогами, каждая из которых имеет длину (стоимость) и может быть односторонней. Нужно найти самый выгодный маршрут, который проходил бы через все города хотя бы по одному разу и возвращался бы в исходный город, и, самое главное, доказать, что более выгодного маршрута не существует. Человек (конечно, ничего не доказывая) может найти такой маршрут, как правило, мгновенно, но как именно он это делает, на сегодняшний день точно неизвестно. Машина же куда хуже справляется с этой задачей.

Найти оптимальное правильное синтаксическое дерево, которое бы связывало все словоформы в предложении, — ничуть не проще, чем решить задачу коммивояжёра. Вместе с абстрактными единицами грамматики словоформы конкретного предложения образуют сложную систему связей, в которой синтаксическое дерево — это маршрут. Решению задачи коммивояжёра посвящено множество исследований, равно как и проблеме комбинаторного взрыва при компьютерном синтаксическом анализе. В обоих случаях на практике вместо универсальных часто используются **эвристические алгоритмы** — частные решения, дающие верный или приблизительный результат в большей части случаев, но не пригодные для полноценного решения задачи. Например, в области компьютерного синтаксиса к числу таких решений относятся так называемые **одноцелевые парсеры** — синтаксические анализаторы, которые для каждого предложения строят лишь одну версию разбора. Одноцелевые парсеры часто используют вероятностные грамматики (см. ниже) в качестве эвристики для выбора первой версии, но также используются и эвристические правила (например, исключать версию именительного падежа в существительных после предлогов, привязывать предложные группы только к ближайшему соседу и т.д.). Как правило, для разных предложений, состоящих из одних и тех же грамматических форм, они показывают одну и ту же синтаксическую структуру. Поэтому, если, например, для предложения *Вася встретил Петю в костюме* такой анализатор выстроит одну из верных версий разбора (т.е. *в костюме* будет отнесено к *Вася или к Петю*, а не к *встретил*), то и для предложения *Вася встретил Петю в коридоре* синтаксическая структура будет выстроена точно так же, и при семантическом анализе кто-то из участников встречи непременно окажется «одетым» в коридор (таково одно из возможных значений предлога *в* при прямом подчинении именной группе, обозначающей живое существо).

Такой подход приводит и к более серьёзным проблемам. Синтаксическая неоднозначность может быть в принципе не разрешимой на уровне синтаксиса — например, в широко известном предложении *Он увидел их семью своими глазами* (те, кому неизвестен этот пример, редко сразу догадываются, что машина может предположить наличие семи глаз у действующего лица, но даже те, кто догадываются, редко осознают, что машина может ока-

заться права, например, если речь идёт о пауке, лишённом одного глаза). Верным результатом *синтаксического* анализа таких предложений (а значит, в общем случае и любых других), всё же, является не одно наиболее вероятное дерево, а совокупность всех возможных деревьев. В данном случае их, по меньшей мере, четыре (Толкования этих четырёх структур выглядят приблизительно так: ‘Он сам увидел их семью’, ‘Он увидел их при помощи семи своих глаз’, ‘Он увидел, что их семья может стать его глазами’ и ‘Он увидел, что они могут стать семью его глазами’). Поэтому в современных системах, как правило, всё же используются **многоцелевые парсеры** — анализаторы, которые для каждого предложения порождают набор версий синтаксического дерева.

Между тем, именно при разработке многоцелевых парсеров проблема комбинаторного взрыва оказывается особо важной.

3. Статистический парсинг

Весьма популярна идея использования при синтаксическом анализе статистических данных ([Jurafsky, Martin 2008]), позволяющая парсерам рассматривать в первую очередь наиболее **вероятные** варианты анализа. В основе современных статистических алгоритмов лежит формализм вероятностных контекстно-свободных грамматик (probability context-free grammars, PCFG). При таком подходе к каждому правилу переписывания добавляется вероятность применения этого правила. Эта вероятность оценивается статистически, на материале синтаксически размеченных текстовых коллекций, и на самом деле отражает то, как **часто** в реальных текстах реализуется то или иное синтаксическое правило.

Добавление вероятностной меры в формальную грамматику действительно позволяет строить алгоритмы, ранжирующие свои версии и в первую очередь обнаруживающие наиболее вероятные (частотные) синтаксические конструкции. К числу таких алгоритмов относятся алгоритмы CKY, Viterbi parser (реализован в пакете NLTK), тензорная декомпозиция ([Cohen, Satta, Collins 2013]), Coarse-to-fine algorithm и другие алгоритмы.

Основным недостатком PCFG и основанных на ней алгоритмов является то, что вероятности оцениваются, как правило, в отрыве от лексического контекста, то есть реальных слов или словоформ, в то время как, очевидно, фактическая вероятность употребления синтаксических конструкций не является безусловной и существенно зависит от их лексического наполнения (вспомним пример с костюмом и коридором). Для решения этой проблемы был разработан формализм лексических PCFG (LCFG), однако этот формализм и основанные на нём алгоритмы сталкиваются уже с другой проблемой — проблемой лексической неоднозначности.

Для оценки качества существующих решений, как и в прочих областях компьютерной лингвистики, в компьютерном синтаксисе принято использовать стандартную F-меру (меру Рисбергена), представляющую собой гармоническое среднее точности и полноты анализа, где точность (precision, p) — отношение количества корректных выданных результатов к общему количеству выданных результатов; полнота (recall, r) — отношение количества корректных выданных результатов к общему количеству возможных корректных результатов в коллекции; среднее гармоническое — удвоенное произведение точности и полноты, делённое на их сумму:

$$\frac{2pr}{p+r}$$

К сожалению, эти метрики позволяют корректно оценить эффективность лишь одноцелевых парсеров; в случае с многоцелевыми парсерами учитывается только первая версия анализа, что не вполне корректно в случае неоднозначности. При этом корректность самой расстановки вероятностей не учитывается вообще. Кроме того, как правило, расчёты F-меры производятся на той же коллекции текстов, что и расчёты вероятностей синтаксических правил. Это приводит к завышенным показателям (более 90% F-меры) и не позволяет с уверенностью говорить об их корректности.

Статистические модели позволяют лишь частично улучшить результаты анализа, но даже существующие метрики и основанные на них «дорожки» («соревнования» синтаксических парсеров, например, РОМИП 2012) показывают, что описанные методы уступают в качестве методам, основанным на взаимодействии синтаксиса и семантики (системы Compreno и ЭТАП-3 показали более 95% F-меры, в то время как F-мера для парсера АОТ составила 87%), в которых, выстроив любую синтаксическую связь, парсер может двигаться дальше, только получив от семантического компонента подтверждение: у этой связи действительно есть осмысленная непротиворечивая трактовка.

Это означает, что синтаксический и семантический анализ вряд ли целесообразно производить по-отдельности в современных системах. Ниже изучается, к чему приводит нарушение или соблюдение этого принципа в современных синтаксических анализаторах.

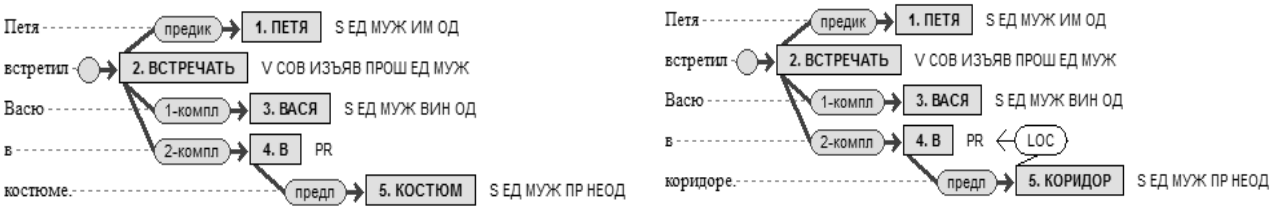
4. Современные синтаксические анализаторы: семь глаз и типы в цехе

4.1. ЭТАП

Лингвистический процессор ЭТАП возник на основе самого успешного компьютерного переводчика Советского союза. Он начал создаваться в 1972 году в необычном месте — в институте «Информэлектро», директор которого, С. Г. Малинин, брал на работу тех лингвистов, которых увольняли из других институтов за протесты против вторжения в Чехословакию и прочее вольнодумство. ЭТАП стал единственной российской системой тех лет, дожившей до нашего времени.

Сегодняшняя версия ЭТАП-3 — это полномасштабный лингвистический процессор. Его главная задача — анализировать текст и извлекать из него некоторое абстрактное представление, отражающее его смысл.

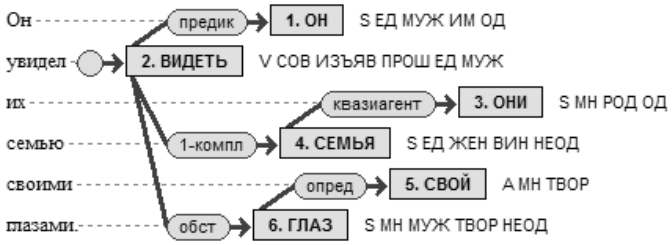
Синтаксический анализатор лингвистического процессора **ЭТАП-3** работает в связке с



толково-комбинаторным словарём, поэтому некоторые различия между *Петя встретил Васю в костюме* и *Петя встретил Васю в коридоре*, всё же, «замечает», хотя в первом случае *в костюме* относит всё же к глаголу, а не к существительному:

Рис. 4: ЭТАП-3: работа толково-комбинаторного словаря

Для предложения *Он увидел их семью своими глазами* строится лишь одна, самая очевидная версия:



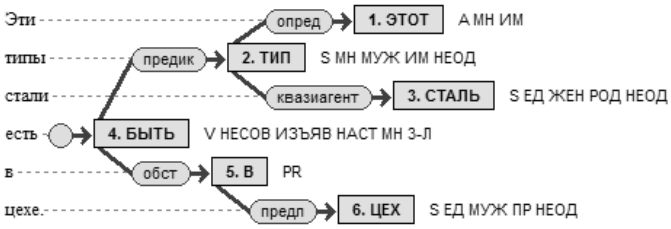


Рис. 5: ЭТАП-3: некорректный разбор неоднозначного предложения

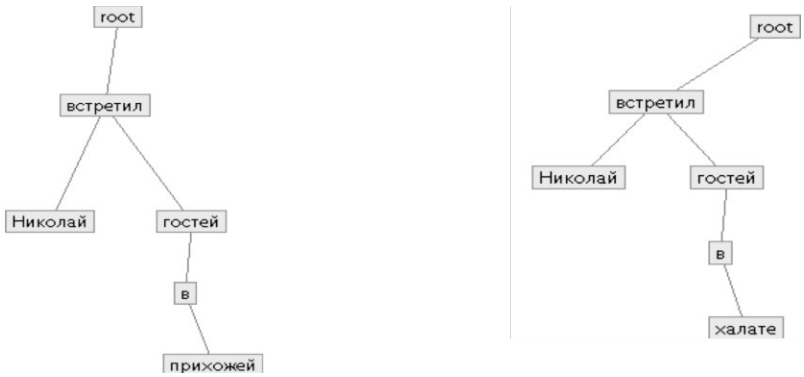
Одна версия строится и для предложения *Эти типы стали есть в цехе*:

Рис. 6: ЭТАП-3: некорректный разбор неоднозначного предложения

Из приведённых примеров видно, что система ЭТАП-3 удачно справляется с синтаксически неоднозначными предложениями благодаря совместной работе синтаксического и семантического компонентов, но при этом теряются версии разбора, которые могли бы быть верными в особом контексте.

4.2. DictaScope и AOT

Синтаксический анализатор **DictaScope** (разработка российской компании Dictum) работает независимо от какого-либо семантического анализатора. Он пытается самостоятельно выбирать версию синтаксического разбора, что приводит к ряду проблем. Приведённые ниже схемы показывают, что предложения *Николай встретил гостей в прихожей* и *Николай*



встретил гостей в халате интерпретируются неправильно (гости одеты в прихожую или в один и тот же халат).

Рис. 7: DictaScope: гости одеты в прихожую

Основная проблема с системами, подобными ЭТАП-3 и DictaScope, состоит даже не в том, что иногда они строят неправильные синтаксические деревья, а в том, что других вариантов они попросту не строят или, по крайней мере, не показывают их пользователю.

Сходная ситуация наблюдается и в системе **AOT**.

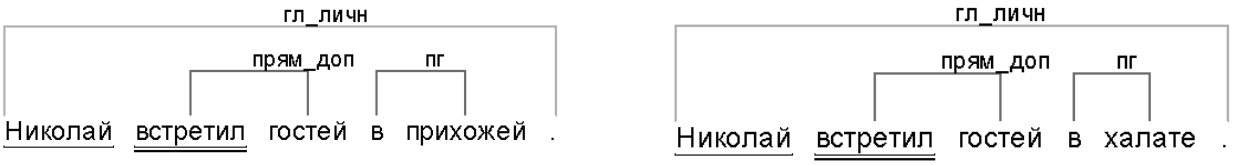
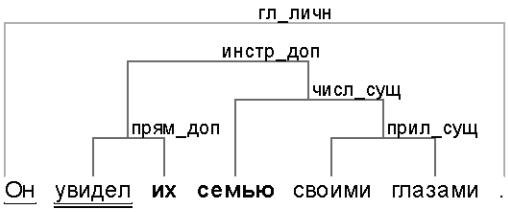


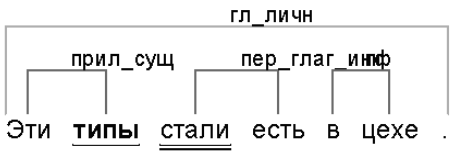
Рис. 7: АОТ: место встречи — халат

Для предложения *Он увидел их семью своими глазами* система АОТ выдаёт одну вер-



сию разбора, в которой у действующего лица — семь глаз.

Рис. 8: АОТ: семь глаз

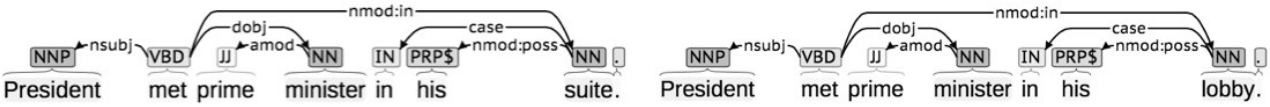


Аналогична участь предложения *Эти типы стали есть в цехе*, в котором предполагается наличие неких типов, которые стали употреблять пищу в цехе.

Рис. 9. АОТ: подозрительные типы употребляют пищу в цехе

4.3. Stanford NLP, RASP, OpenNLP

Сходная ситуация наблюдается и в зарубежных одноцелевых парсерах. Так, парсер системы Stanford NLP выдаёт одинаковые версии разбора для предложений *President met prime minister in his suite* (Президент встретил премьера в его/своём костюме) и *President met prime*



minister in his lobby (Президент встретил премьера в его/своём коридоре).

Рис. 10: Stanford NLP: костюм — место встречи

Аналогично поведение парсера RASP в составе системы Gate. К сожалению, результаты анализа можно получить лишь в текстовом виде.

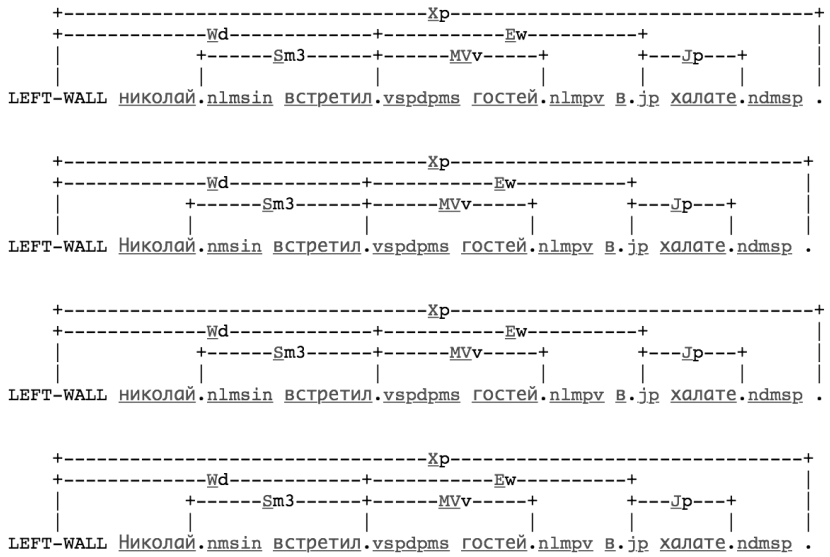
Листинг 1: RASP/Gate: костюм — место встречи

```
("T"
 ("S" ("NP" "President")
  ("VP" "meet+ed" ("N1" ("AP" "prime") "minister")
   ("PP" "in" ("NP" "his" "suite"))))
 ".")
("T"
 ("S" ("NP" "President")
  ("VP" "meet+ed" ("N1" ("AP" "prime") "minister")
   ("PP" "in" ("NP" "his" "lobby"))))
 ".")
```

К числу одноцелевых парсеров, не различающих эти две синтаксические структуры, относится и парсер системы OpenNLP.

4.4. **Link Grammar Parser**

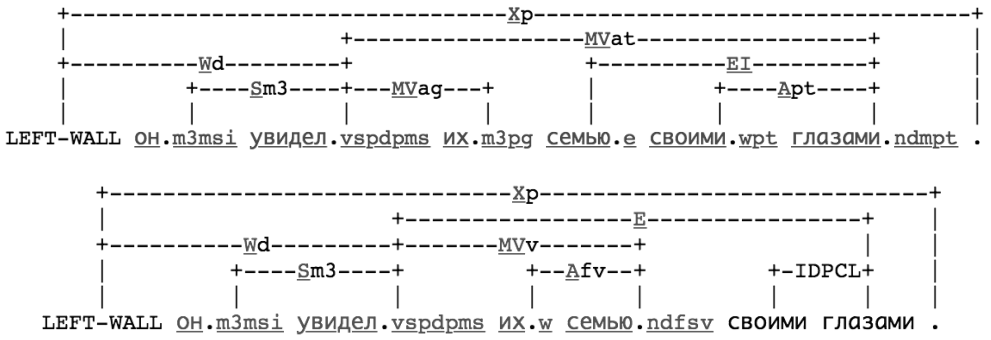
Синтаксический анализатор **Link Grammar Parser** для русского языка строит четыре версии разбора как для предложения *Николай встретил гостей в прихожей*, так и для предложения *Николай встретил гостей в халате*, но третья и четвёртая версии полностью повторяют первые две, а вторая отличается от первой лишь тем, что в первой версии слово



Николай почему-то разобрано как имя нарицательное. Во всех версиях разбора второго предложения *в халате* оказывается обстоятельством места, т.е. халат — это место встречи.

Рис. 11: Link Grammar Parser: неоднозначность обнаружена, но халат — место встречи

Вместе с тем, для предложения *Он увидел их семью своими глазами* Link Grammar

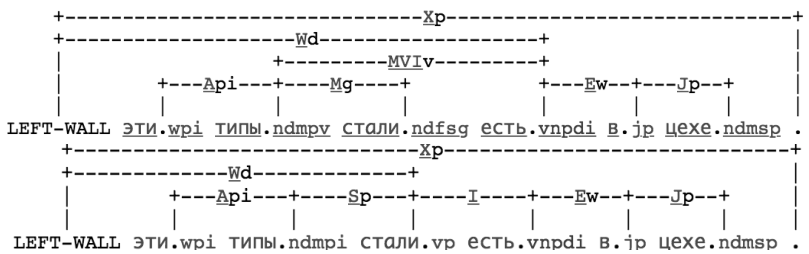


Parser строит целых 14 версий разбора, среди которых есть и те, что были указаны выше, и даже совсем не очевидные версии (например, версия, которой можно сопоставить только такую трактовку: ‘Он сам увидел, что они могут стать семью’).

Приведём лишь первую и последнюю версии (в первой версии речь идёт о семье, в последней — о семи глазах):

Рис. 12: Link Grammar Parser: обнаружены все версии разбора

Для предложения *Эти типы стали есть в цехе* строится 12 версий разбора. В версиях с подозрительными типами, которые выдаются в первую очередь, различаются возможные способы отнесения обстоятельства *в цехе*: в одной группе версий оно относится к глагольной форме *стали*, в другой — к *есть*, в третьей — к типам. Версии с типами стали (сплава) почти



не различаются; лишь в одной из них форма *есть* верно распознана как инфинитивная. Приведём здесь лишь первую и последнюю версии.

Рис. 12: Link Grammar Parser: обнаружены все версии разбора

Link Grammar Parser отличается от рассмотренных выше систем тем, что он в действительности является многоцелевым. К сожалению, эта система не позволяет встраивать в процедуру разбора семантические ограничения и тем самым хотя бы частично снимать неоднозначность, и у этой системы отсутствуют способы настройки правил грамматики, которыми пользуется система.

4.5. NLTK

Лингвистический пакет **NLTK**, как и Link Grammar Parser, строит несколько версий разбора; в отличие от всех вышеперечисленных систем, он позволяет пользователю самостоятельно записывать грамматику в виде правил переписывания (используется НС-грамматика). Ниже приводится пример короткой программы на языке Python, использующей NLTK для разбора предложения *Николай встретил гостей в коридоре* и выдача этой программы.

Листинг 2: пользовательская НС-грамматика в NLTK

```
#coding: utf-8

import nltk
from nltk.draw.tree import draw_trees
grammar = nltk.CFG.fromstring("""
Предложение  ->  ИГ ГГ
ПГ           ->  П ИГ
ИГ           ->  С | С ПГ
ГГ           ->  Г ИГ | ГГ ПГ
С            ->  'Николай' | 'гостей' | 'коридоре'
Г            ->  'встретил'
П            ->  'в'
""", encoding="utf-8")
sent = [u'Николай', u'встретил', u'гостей', u'в', u'коридоре']
parser = nltk.ChartParser(grammar)

draw_trees(*(tree for tree in parser.parse(sent)))
```

Эта программа выведет на экран графические представления построенных деревьев:

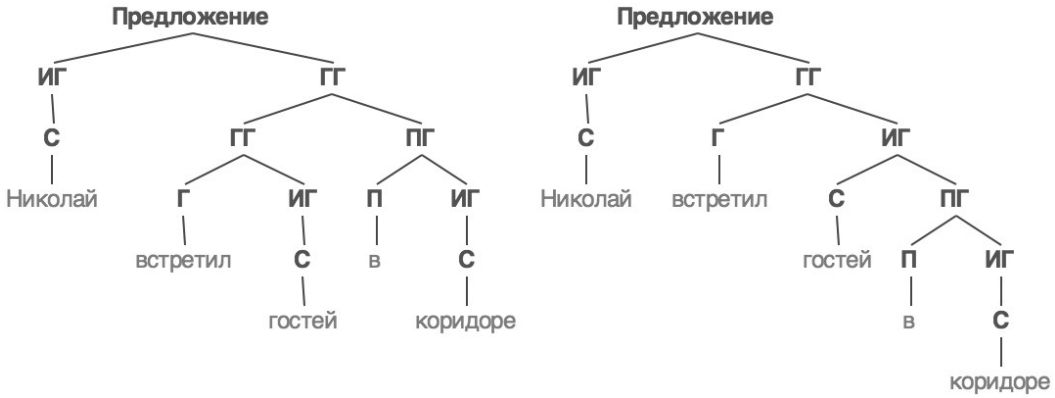


Рис. 13: NLTK: обнаружены все версии разбора

NLTK позволяет также строить версии разбора и в виде деревьев зависимостей, однако на сегодняшний день не выполняет разбор в терминах ТСГ (комбинированных структур НС и ГЗ). Кроме того, для разбора в терминах ГЗ требуется указание в грамматике всех возможных зависимостей между конкретными словоформами в явном виде, что в некотором роде делает эту процедуру бессмысленной.

Способов семантической фильтрации версий синтаксического разбора в nltk не предусмотрено, однако пользователь может указывать вероятности для каждого правила грамматики в квадратных скобках (реализуя тем самым модель PCFG, см. выше).

5. Дальнейшие задачи

Существующие проблемы компьютерного синтаксиса могут получить полноценное решение лишь в случае создания компьютерных систем, полностью моделирующих языковую картину мира, но существующие на сегодняшний день системы справляются с этой задачей лишь частично. Такие системы в современной литературе принято называть универсальными компьютерными онтологиями. Для построения такой онтологии необходимо создание так называемых банков деревьев (tree banks), или синтаксически размеченных корпусов текстов, в которых синтаксическая и лексическая неоднозначность снята вручную, и на основании которых могут быть объективно установлены существующие в языке семантические валентности и частотность тех или иных связей в текстах.

Во многом по этому принципу построен проект FrameNet, хотя степень формализации этой базы знаний на сегодняшний день не позволяет использовать её в компьютерном синтаксисе в прямом виде. Семантические валентности выявляются на основе синтаксически размеченных корпусов текстов, но для получения этих корпусов необходимы средства синтаксического анализа, снятия синтаксической неоднозначности и построения однозначной синтаксической разметки. Средства снятия неоднозначности основаны на валентностях, получается некоторый «порочный круг». Кроме того, для выявления семантических валентностей в разрешении нуждается не только синтаксическая, но и лексическая неоднозначность, что означает необходимость дополнительной семантической разметки синтаксически размеченных корпусов текстов.

Таким образом, универсальная компьютерная онтология не сможет быть построена и проверена до тех пор, пока не будет собран достаточно объёмный синтаксически и семантически размеченный корпус текстов. Между тем, чтобы построить такой корпус, необходимо сначала хотя бы частично решить проблему синтаксической неоднозначности — иначе разметка корпуса потребует неоправданных и весьма объёмных трудозатрат.

Одним из решений этой проблемы может стать использование метода последовательных приближений. По этому принципу построен проект AIIRE, одним из участников которого является автор данной главы.

Компьютерный синтаксис на сегодняшний день является одной из самых многообещающих, и в то же время вызывающих массу разночтений и затруднений областей компьютерной лингвистики. Ещё 15 лет назад эта область считалась мало перспективной, именно из-за этих разночтений и проблем, прежде всего — из-за проблемы производительности (комбинаторного взрыва). Появление компьютерных онтологий и совмещение их с синтаксическими анализаторами дало принципиально новые возможности; создание собственных синтаксических анализаторов вновь стало популярным занятием. Остаётся надеяться на то, что острый интерес к этой области не иссякнет и прорыв в области компьютерного синтаксиса будет совершён в ближайшие годы.

Литература

1. *Беляева, Л.Н.* Автоматический (машинный) перевод // Прикладное языкознание: Учебник — Санкт-Петербург, 1996 — С. 360-388
2. *Гладкий, А.В.* Синтаксические структуры естественного языка в автоматизированных системах общения — Москва: Наука, 1985 — 144 с.
3. *Леонтьева, Н.Н.* Автоматическое понимание текстов. Системы, модели, ресурсы — Москва: Academia, 2006 — 303 с.
4. *Лесерф, И.* Применение программы и модели конкретной ситуации к автоматическому синтаксическому анализу // НТИ. № 11 — Москва: ВИНИТИ, 1963 — С. 42-50
5. *Теньер, Л.* Основы структурного синтаксиса — Москва: Прогресс, 1988 — 656 с.
6. *Тестелец, Я.Г.* Введение в общий синтаксис — Москва: РГГУ, 2001 — 798 с.
7. *Cohen, S.B., Satta, G., Collins M.* Approximate PCFG Parsing Using Tensor Decomposition // Proc. of NAACL 2013
8. *Jurafsky, D., Martin, J.H.* Speech and Language Processing, 2nd Edition — Prentice Hall, 2008 — 1024 pp.

Полезные ссылки:

1. Система ЭТАП-3: http://proling.iitp.ru/ru/etap3
2. Парсер DictaScope: анализатор долгое время был доступен по адресу: http://www.dictum.ru/ru/syntax/blog, однако сейчас доступна только кнопка «анализировать»; по какой-то причине сервер синтаксического анализа (http://212.92.158.66:3025) не отвечает на запросы
3. Синтаксический анализатор AOT: http://aot.ru/demo/synt.html, но для его работы необходима установка Java Plugin и специальная настройка этой надстройки обозревателя на страх и риск пользователя, т.к. данная система не удовлетворяет требованиям безопасности
4. FrameNet: https://framenet.icsi.berkeley.edu
5. Stanford NLP: http://nlp.stanford.edu:8080/corenlp/process
6. Парсер RASP в составе системы Gate: последняя версия доступна по ссылке http://ilexir.co.uk/applications/rasp/download/
7. OpenNLP: (https://opennlp.apache.org)
8. Link Grammar Parser: http://slashzone.ru/parser/parse.pl
9. Лингвистический пакет NLTK**:** Пользователи современных UNIX-подобных операционных систем могут установить этот пакет при помощи команды «sudo pip install NLTK». Для пользователей Windows всё несколько сложнее, инструкции по

установке приведены по адресу http://www.nltk.org/install.html

10. AIIRE http://aiire.org, http://svn.aiire.org/repos/t