Синтаксический анализатор «Treevial». Принцип динамического ранжирования гипотез[1]

«Treevial» syntax parser. Paradigm of the dynamic hypothesis ranking

Старостин А. С. (starost@rinet.ru),
Арефьев Н. В. (nick.arefyev@gmail.com),
Мальковский М. Г. (malk@cs.msu.su)

Московский государственный университет им. М. В. Ломоносова

В статье подробно обсуждается главный принцип, в соответствии с которым работает синтаксический анализатор Treevial, — *принцип динамического ранжирования гипотез*. Описывается формальный аппарат, используемый в Treevial для описания грамматики. Отдельный раздел посвящен механизму штрафов, функционирующему вместе с базовым формализмом. Излагается схема работы анализатора. Обсуждаются достоинства и недостатки предлагаемого подхода. В конце статьи приводится описание инструментов штрафования, заложенных в анализатор.

Введение

Работа посвящена описанию синтаксического анализатора Treevial. Этот анализатор является базовой составляющей системы морфо-синтаксического анализа Treeton (см. [Мальковский, Старостин, 2006]), разрабатываемой на факультете ВМиК МГУ с 2005 г. Система была задумана как исследовательская платформа, базируясь на которой студенты и аспиранты факультета могли бы решать различные задачи компьютерной лингвистики. В данный момент система довольно динамично развивается. В рамках Treeton ведутся исследования в области автоматического морфологического и синтаксического анализа, а также в области автоматизации базовых процедур лингвостиховедческого анализа. Система является некоммерческой и открытой для всех, кто желает принять участие в ее развитии.

Главной темой данной статьи является принцип динамического ранжирования гипотез, заложенный в основу анализатора Treevial. Поясним, в чем он заключается. Пусть имеет место процесс восходящего синтаксического анализа, в рамках которого возникают различные конфигурации, связывающие группы слов друг с другом. Кроме того, допустим, что существует способ некоторым образом оценивать «качество» (степень правдоподобности) каждой возникающей конфигурации. Тогда принцип динамического ранжирования может быть сформулирован следующим образом: **предпочтение более правдоподобных конфигураций менее правдоподобным должно отдаваться не после, а в процессе синтаксического анализа**.

Авторы продемонстрируют, какие достоинства имеет синтаксический анализатор, построенный в соответствии с принципом динамического ранжирования, и какие ограничения приходится накладывать на компоненты анализатора. Статья состоит из четырех разделов. В первом разделе описывается базовый формализм, используемый в Treevial для описания грамматики (способ формулировки синтаксических правил). Во втором разделе обсуждается концепция штрафов. В третьем разделе приводится схема работы анализатора и обсуждается принцип динамического ранжирования. В четвертом разделе описываются инструменты штрафования, которые в данный момент поддерживает анализатор.

1. Формализм для описания грамматики

Язык Treevial позволяет задавать грамматику естественного языка в виде системы декларативных правил, описывающих возможные пути сборки синтаксических структур из исходных словоформ (вариантов морфологического анализа).

[1] Работа выполнена при поддержке РФФИ (проект 08-06-00192)

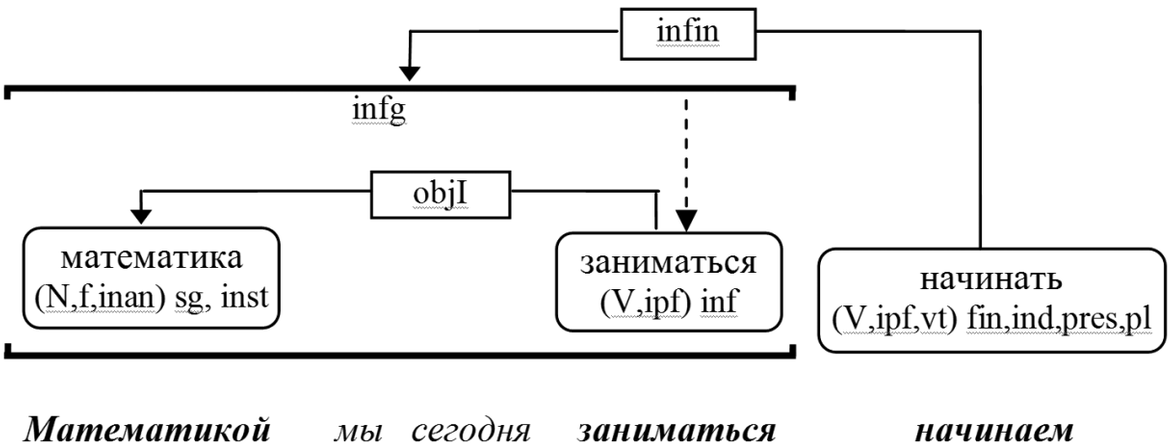

Рис. 1. Пример синтаксической интерпретации

Идеология системы подобна той, которая принимается при работе с категориальными грамматиками — перед анализатором ставится задача для заданной цепочки атрибутированных словоформ (категорий) найти возможность связать все словоформы в одну древовидную структуру (сократить все категории), т. е. доказать принадлежность данной цепочки описываемому языку. Следует отметить, что возможностей организовать одну и ту же цепочку форм в структуру обычно находится больше, чем одна[2]. Наиболее близким к описываемому формальным аппаратом из области категориальных грамматик является [Dekhtyar, Dikovsky, 2008] (в нем допускаются дистантные зависимости между элементами).

Базовым для языка Treevial является понятие синтаксической структуры или синтаксической интерпретации фрагмента предложения. Последний термин предпочтительнее, т. к. он подчеркивает «гипотетичность» всего, что делает анализатор, и имплицирует наличие альтернативных интерпретаций для одного и того же фрагмента. Синтаксическая интерпретация — это направленное дерево с размеченными дугами, узлами которого являются либо морфологические интерпретации слов предложения, либо виртуальные узлы, созданные в процессе анализа. Все элементы синтаксической структуры имеют линейные координаты, т. е. интерпретация всегда спроецирована на исходное предложение. Следует отметить, что такая проекция не обязательно представляет собой сплошной отрезок — допускаются разрывные интерпретации.

На рисунке 1 приводится пример синтаксической интерпретации фрагмента предложения «математикой мы сегодня заниматься начинаем», иллюстрирующий все описательные возможности этого формального объекта: фиксация связей между элементами, объединение элементов в группы (создание виртуальных узлов), потенциальная разрывность. Жирным выделены слова, образующие проекцию синтаксической интерпретации.

Из описанных в литературе способов описания синтаксических структур наиболее близким аналогом синтаксической интерпретации является размеченная система синтаксических групп (РССГ, [Гладкий, 1985]). Возможность создания виртуальных узлов позволяет описывать синтаксические явления в духе грамматик составляющих (ср. [Хомский, 1962]). Возможность использования связей, напротив, допускает взгляд на синтаксические структуры, близкий к грамматикам зависимостей (ср. [Tesnière, 1959; Mel'cuk, 1988]). Существенно, что имеется возможность одновременного использования преимуществ и той, и другой парадигмы.

Язык Treevial задает правила преобразования синтаксических интерпретаций. Каждое правило описывает ситуацию, в которой две интерпретации, обладающие определенными свойствами, и находящиеся в определенных отношениях (например, контактность или согласованность вершин по какой-то категории), могут образовать новую интерпретацию. Например, может быть проведена связь от некоторого элемента одной интерпретации к корню другой или обе интерпретации могут быть объединены в группу, которая получит некоторые новые свойства (и в том, и в другом случае будет образована новая интерпретация). Помимо использования бинарных правил (работающих с двумя интерпретациями), в языке Treevial есть возможность

[2] Заметим, что сходство с категориальными грамматиками на уровне идеологии и заканчивается. Дело в том, что, в категориальных грамматиках центральной идеей является сокращение категорий, т. е. превращение двух элементов (обычно соседних) в один. После того как сокращение было произведено, сокращенный элемент перестает принимать участие в деривации. В языке Treevial подход другой — если два элемента объединяются (образуют группу или связываются связью), то образуется новый составной объект, который в дальнейшем может участвовать в деривации, присоединяя что-либо к любой из своих частей.

задавать унарные правила. Эти правила применяются только к одной интерпретации, обладающей определенными свойствами. Результатом применения унарного правила всегда является создание группы (виртуального узла), обладающей определенными свойствами и включающей исходную интерпретацию.

Схематично синтаксическое правило может быть записано следующим образом:

```
rule MyRule() {
  спецификация аргументов
  ::
  ограничения
  →
  действия
  ::
  условия взимания штрафов
}
```

Действия, описанные в блоке действий, выполняются над теми интерпретациями, узлы которых удовлетворили спецификации аргументов и ограничениям, описанным в блоке ограничений. В результате применения правила создается новая синтаксическая интерпретация. Рассмотрим составные части правила по отдельности.

1.1. Спецификация аргументов

Спецификация аргументов состоит из одного или двух шаблонов (логических выражений, составленных из элементарных ограничений на значение атрибутов, скобок, знаков дизъюнкции и конъюнкции). С шаблонами сопоставляются узлы синтаксических интерпретаций (как корневые, так и промежуточные). Если шаблонов два, то между ними может стоять знак ~ . Это означает, что элементы, сопоставляемые с шаблонами могут быть расположены дистантно. Если между двумя шаблонами стоит знак +, это означает, что сопоставляемые с шаблонами элементы должны примыкать друг к другу. При этом каждый из шаблонов может быть окружен квадратными скобками, что означает, что примыкание требуется не от самого элемента, сопоставляемого с шаблоном, а от проекции всего поддерева данного элемента. По умолчанию порядок, в котором должны стоять сопоставляемые с шаблонами элементы, может быть произвольным. Однако, его можно зафиксировать, поставив после спецификации аргументов знак ^.

Следует отметить, что в блоках, описываемых ниже (ограничений, действий, условий взимания штрафов), узел, сопоставленный с левым шаблоном спецификации аргументов, всегда обозначается как *A*, а узел, сопоставленный с правым шаблоном, как *B*.

1.2. Ограничения

Ограничения задаются с помощью логического выражения, в котором могут участвовать атрибуты двух элементов, за счет чего можно учитывать такие явления как, например, согласование. После сопоставления узлов синтаксических интерпретаций и шаблонов из спецификации аргументов переменные *A* и *B* получают конкретные значения и становится возможным проверить, обращается ли логическое выражение в истину или нет.

Пример ограничения (согласование по падежу и числу):

$$A.CAS == B.CAS \;\&\&\; A.NMB == B.NMB$$

1.3. Действия

Блок действий представляет собой список элементарных действий, состоящий минимум из одного элемента. Действия выполняются в случае, если узлы синтаксических структур, сопоставленные с шаблонами, удовлетворяют ограничениям. Вторым необходимым условием для выполнения действия является его корректность по отношению к принципу древесности. Это означает, что ни одно правило не может создать интерпретацию, в которой будут циклы, или у какого-нибудь узла будет более одного хозяина.

Действия бывают четырех типов:

- Связывание

(имя_шаблона_1,имя_шаблона_2){тип_связи}

Проводится связь определенного типа от узла, сопоставленного с первым шаблоном, к узлу, сопоставленному со вторым шаблоном. Например, *(A,B){myRelType}*.

- Унарная агрегация

C[имя_шаблона]{последовательность присваиваний атрибутов}

Узел, сопоставленный с шаблоном, помещается внутрь нового элемента, атрибуты которого задаются в результате выполнения последовательности присваиваний (см. ниже). Например, *C[A]{NMB=A.NMB;GTYPE="Clause";CAS=null;}*.

- Бинарная агрегация

C[имя_шаблона_1,имя_шаблона_2]{последовательность присваиваний атрибутов}

Узлы, сопоставленные с шаблонами, помещаются внутрь нового элемента «на равных правах», не подчиняясь друг другу. Атрибуты нового узла задаются в результате выполнения последовательности присваиваний. Например, *C[A,B]{CAS=A.CAS;NMB=B.NMB;GTYPE="conj";}*.

- Включение

*имя_шаблона_1**[**имя_шаблона_2**]***

Узел, сопоставленный с шаблоном 2, помещается внутрь элемента, сопоставленного с шаблоном 1. Например, *B[A]*.

При описании бинарной и унарной агрегации упоминалась последовательность присваиваний атрибутов. Это список элементов вида: *название_атрибута = значение.*

В момент выполнения агрегации список проходится слева направо, вычисляются соответствующие значения и выполняются присваивания. Значения могут задаваться как явно (конкретными литералами), так и ссылкой на значение некоторого атрибута одного из узлов, сопоставленных с шаблонами из спецификации аргументов. В качестве значения может также выступать служебное слово *null*. В этом случае при выполнении присваивания соответствующий атрибут стирается.

1.4. Условия взимания штрафов

Условия взимания штрафов очень похожи на ограничения. Однако если при невыполнении ограничения применение правила запрещается, то при невыполнении условия взимания штрафов запрета не происходит — просто штраф получившейся интерпретации увеличивается на некоторую векторную величину (штрафы в системе представляются целочисленными векторами). Подробно штрафы обсуждаются в следующем разделе. Приведем пример условия взимания штрафов:

A.@start < B.@start : (0,0,100)

Это условие означает, что если A начинается левее, чем B, следует увеличить штраф на *(0,0,100)*.

До сих пор не было упомянуто еще одно важное ограничение на процесс применения правил: проекции интерпретаций, участвующих в правилах, не могут пересекаться. Напомним, что проекции синтаксических интерпретаций в общем случае могут быть разрывными. Сформулированное условие гарантирует отсутствие циклов после применения правил и не позволяет использовать в рамках одной структуры различные морфологические трактовки одного и того же слова.

Таким образом, можно резюмировать, что каждое синтаксическое правило применяется к одной или к двум синтаксическим интерпретациям. В каждой интерпретации выбираются узлы, с опорой на которые сначала проверяются ограничения, потом выполняются действия и, наконец, проверяются условия взимания штрафов. Заметим, что благодаря принципу древесности и тому факту, что правило всегда содержит хотя бы одно действие, в качестве опорного узла одной из интерпретаций всегда выбирается ее корень. Другими словами, как минимум одна синтаксическая интерпретация всегда оказывается подчиняемой. А вот для интерпретации, которая играет роль хозяина, опорный узел может выбираться различными способами. Это означает, что могут существовать различные способы применения одного и того же правила к одной и той же комбинации синтаксических интерпретаций. Например, одна и та же предложная группа может быть присоединена к любому из узлов какого-либо «большого» синтаксического дерева, если только этот узел имеет характеристики глагола.

Дополнительно следует сказать, что могут существовать варианты получения одной и той же синтаксической интерпретации различными путями (рис. 2). Это не приводит к размножению гипотез и не влияет существенным образом на эффективность анализатора — система умеет находить дубликаты. Может показаться, что описанный принцип «сборки» синтаксических структур явля-

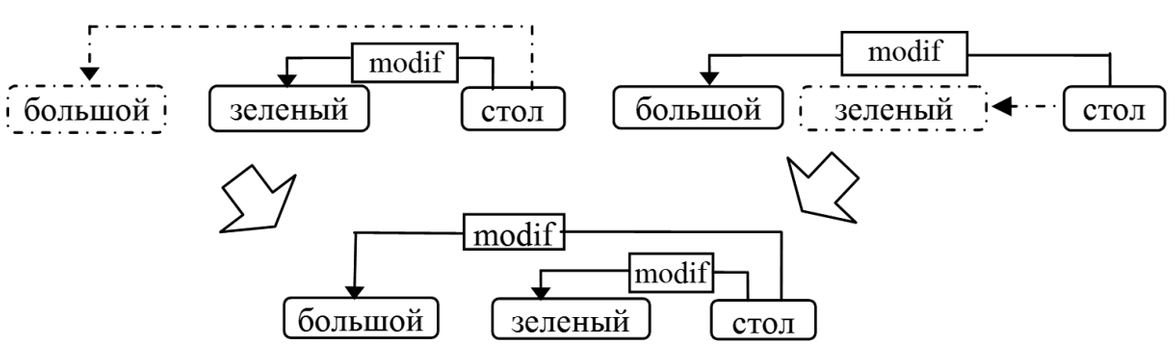

Рис. 2. Пример альтернативных вариантов получения одной структуры

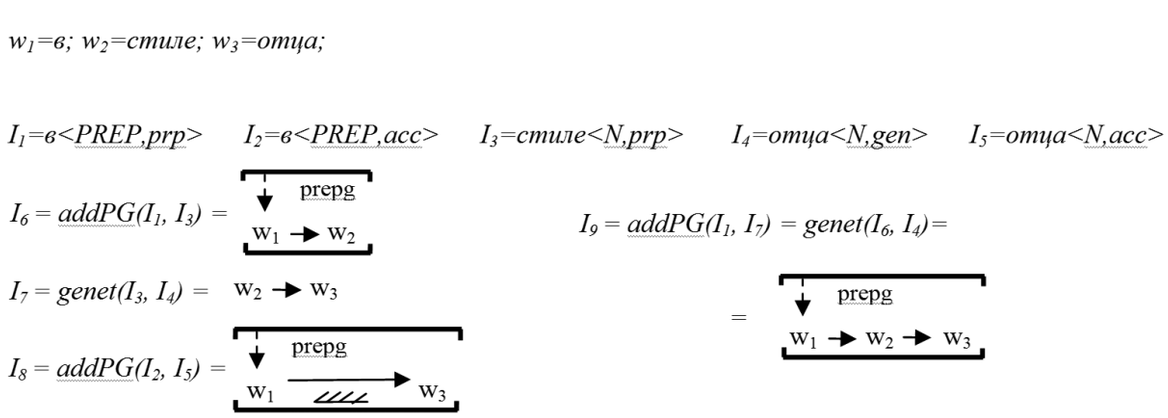

Рис. 3. Пример действия синтаксических правил для словосочетания «В стиле отца»

ется слишком свободным. Однако, любые попытки авторов ограничить деривацию с помощью дополнительного правила (аксиомы) на уровне формального аппарата приводили либо к невозможности описывать сложные языковые явления, либо к проблемам с вычислительной эффективностью[3]. Кроме того, оказалось, что такая «плазмоподобная» модель хорошо согласуется с идеологией штрафов, о которой пойдет речь в следующем разделе.

Еще один аспект предлагаемого аппарата, который следует прокомментировать, это локальность опорных шаблонов. Дело в том, что в рамках синтаксического правила есть возможность обращаться только к одному узлу каждой из интерпретаций (тому, который сопоставился с шаблоном). Это значит, что мы неявно исходим из предпосылки, что все свойства необходимые для формулировки правила будут сконцентрированы в одном узле. Однако, практика показывает, что в естественных языках встречаются так называемые дальние согласования (ср. [Тестелец, 2001]). В этих случаях для формулировки корректного ограничения нужно обращаться не к самому узлу, а к одному из его потомков (*видишь* ***таблицу****, один из столбцов* ***которой*** *заполнен нулями?,* жирным выделены согласующиеся слова). В языке Treevial существует специальная конструкция, позволяющая учитывать дальние согласования. Ее описание выходит за рамки данной статьи.

На рисунке 3 демонстрируется работа анализатора с системой из двух синтаксических правил. Первое правило создает предложную группу, второе устанавливает посессивное (или генитивное) отношение:

```
rule addPG {
  {POS == "PREP"}+[{POS == "N"}] ^
  ::
  A.GCAS == B.CAS
  →
  (A,B){preposit}
  C[A]{PHRASE="prepg";}
}

rule genet {
  {POS == "N"}~{POS == "N" && CAS ==
  "gen"}
  →
  (A,B){genet}
}
```

2. Аппарат штрафов

Прежде чем приступить к описанию аппарата штрафов, остановимся на нескольких моментах, обуславливающих необходимость его использования.

Во-первых, легко видеть, что несмотря на то, что в синтаксических правилах разрешено учитывать геометрически дистантные зависимости, правила, по сути своей, являются контекстно-свободными, т. к. отсутствует возможность учитывать какие-то факты, касающиеся частей уже собранного синтаксического дерева, не имеющих отношения к опорным узлам. Например, нельзя написать правило, выражающее следующий смысл: «такое-то слово можно связать с таким-то, если между ними (не) стоит некоторое слово». По нашему глубокому убеждению, описание синтаксиса естественного языка может быть построено без использования контекстно-зависимых правил. Более того, предлагаемый в этом разделе аппарат штрафов в определенном смысле исключает потребность в использовании таких правил.

Второе, о чем следует сказать, это изначально принятая авторами установка на малый объем лингвистических средств. Вслед за [Перцов, Старостин, 1999] мы придерживаемся идеи о необходимости создания синтаксического анализатора, способного опираться в основном на морфологические характеристики слов без использования богатой словарной информации о сочетаемости. Такая установка про-

[3] Проблемы возникали в первую очередь при анализе непроективных предложений.

диктована в первую очередь тем, что в открытом доступе не существует хорошо формализованных словарей сочетаемости единиц достаточно большого объема, в то время как грамматический словарь А. А. Зализняка доступен всем.

Оба упомянутых факта приводят к тому, что системы правил, которые можно написать на языке Treevial, помимо правильного разбора предложения всегда допускают большое количество альтернативных неправильных вариантов анализа. Многие из них даже могут быть осознаны носителем языка при некотором умственном усилии (ему приходится либо распознать какой-то сложный интонационный контур, допускающий странный порядок слов, либо как-то исказить смысл предложения, либо допустить нехарактерное управление и т. п.). Помимо очевидной проблемы выбора правильного варианта анализа, порождение большого количества результатов имеет еще один существенный недостаток — оно происходит медленно. Полный анализ предложения из 15 слов может занимать порядка 10 минут, из которых 99% времени будет потрачено на очень далекие от действительности варианты анализа.

Путь дальнейшего усложнения языка синтаксических правил казался тупиковым. Было принято решение «разделить переменные» и попытаться разработать обособленный механизм, позволяющий ограничивать функционирование базового механизма (аппарата синтаксических правил).

Таким образом возникла идея ранжирования спектра гипотез, порождаемых анализатором, в соответствии с некоторой эвристической функцией, называемой *штрафной функцией*. Штрафная функция соотносит с любой синтаксической интерпретацией неотрицательное действительное число. При вычислении этой функции могут использоваться различные характеристики структуры: уровень проективности, количество повторений неповторимых связей, наличие пропусков слов в структуре, корректность расстановки запятых, условия взимания штрафов в конкретных правилах, согласованность с моделями управления (при наличии последних) и др. В принципе, при ее вычислении могла бы учитываться и семантика (при наличии соответствующих формальных средств). Различные явления, влияющие на значения штрафной функции, описываются набором штрафных конструкций языка Treevial. Примером такой конструкции является упоминавшееся в предыдущем разделе условие взимания штрафов, формулируемое в теле синтаксического правила. В четвертом разделе этой статьи коротко описываются другие возможности языка Treevial, позволяющие управлять вычислением штрафной функции.

Идея сортировки всех вариантов анализа относительно некоторой шкалы качества хороша с логической точки зрения, но, к сожалению, сама по себе она никак не решает проблему производительности. Очевидно, что принцип «сначала получи все, потом вычисли значения штрафной функции и отсортируй» оказывается нежизнеспособен. Встал вопрос о том, можно ли построить алгоритм синтаксического анализа таким образом, чтобы менее штрафованные варианты обнаруживались раньше, чем более штрафованные. Понятно, что выполнение такого требования имеет высокую практическую ценность. Во-первых, результаты анализа можно использовать, не дожидаясь завершения работы алгоритма. Кроме того, появляется возможность отсекать гипотезы в соответствии с некоторым пороговым значением штрафа. Оказалось, что выполнение такого требования становится возможным, если наложить некоторые ограничения на штрафную функцию. Будем считать, что заданы система синтаксических правил R и штрафная функция $P(I)$. Тогда ограничение формулируется следующим образом:

Для любого правила $r \in R$ должно быть верно, что для любых I_i из того, что $I_r = r(I_0, I_1)$, следует, что $P(I_r) \geq P(I_i)$ $(i=0,1)$

Другими словами, штрафная функция должна быть неубывающей (если рассматривать множество интерпретаций как частично упорядоченное[4]). На бытовом языке это можно сформулировать следующим образом: «качество целого никогда не должно превосходить качество составляющих». Ниже будет показано, каким образом приведенное ограничение позволяет построить анализатор, удовлетворяющий требованию динамической сортировки результатов. Но сначала необходимо обсудить, как строится штрафная функция, которая, с одной стороны, удовлетворяет этому ограничению и, с другой стороны, имеет под собой лингвистические основания.

В ранних версиях анализатора Treevial авторы пытались строить требуемую функцию, исходя из свойств всего дерева синтаксической интерпретации. Подсчитывались различные суммы различных характеристик (количество пересечений связей, количество повторяемых связей, количество пропусков слов и т. п.). Существенной проблемой оказалась такая характеристика интерпретации, как количество пропущенных слов. Дело в том, что, с одной стороны, взимание штрафов за пропуски слов кажется вещью совершенно естественной и очень существенно ограничивает перебор, но, с другой стороны, очевидно, что если в двух инерпретациях есть пропуски слов, то в их объединении пропусков может уже не быть (они могут друг друга

[4] На интерпретациях можно ввести следующее отношение частичного порядка: $I > I_0$ тогда и только тогда, когда I может быть получено из I_0 в результате применения последовательности синтаксических правил. Рекурсивно это выражается так: **1)** если $I = r(I_0,I_1)$, то $I > I_0$ и $I > I_1$; **2)** если $I = r(I_m,I_n)$ и либо $I_m > I_0$, либо $I_n > I_0$, то $I > I_0$.

дополнить). Это означает, что с учетом этой характеристики свойство неубывания штрафа не выполняется. Не ограничивать возможности анализатора пропускать слова авторы не могли, т. к. это было критично с точки зрения производительности. Полный запрет на пропуск слов оказался слишком жестким ограничением (этот вопрос дополнительно обсуждается в четвертом разделе). Кроме того, существовала некоторая логическая нестыковка в самой постановке задачи. Требовалось построить функцию, свойства которой формулируются, исходя из предпосылки, что аргумент функции имеет рекурсивную природу, а при этом при построении функции рекурсивность объекта никак не учитывается.

Все это навело авторов на мысль о внесении в систему дополнительного условия (или закона), опирающегося на рекурсивную природу объекта, за счет которого свойство неубывания штрафа может быть строго доказано. Это условие носит название *условия инкрементальности*. Идея очень проста: по определению считается, что штраф результата применения некоторого правила к нескольким интерпретациям равен сумме штрафов этих интерпретаций и некоторой неотрицательной составляющей, величина которой зависит от соединяемых интерпретаций и от действий, выполняемых при применении правила. Штраф конкретной интерпретации при этом вычисляется как наименьший из штрафов, которые могут быть вычислены в соответствии с условием инкрементальности при различных способах получения данной интерпретации. Математически это выражается следующим образом:

$$P(I) = \min_{i,j,r} P(I_i) + P(I_j) + f(I_r),$$

$$\text{где } I = I_r = r(I_i, I_j),\ f \geq 0$$

Таким образом, мы приходим к модели, при которой каждая синтаксическая интерпретация ассоциируется с оптимальным способом ее сборки. Это довольно существенное идеологическое изменение. При таком подходе в ряде случаев становится важным не только то, как устроена синтаксическая интерпретация (какие в ней связи и группы), но и то, как именно она была создана, т. к. в ее штрафную оценку включаются штрафы ее предков. Практика показывает, что такое усложнение не вызывает существенных проблем при работе с анализатором. Оптимальный способ сборки интерпретации сохраняется в памяти анализатора. Поэтому всегда есть возможность его проанализировать и понять, почему величина штрафа интерпретации именно такая. То, как именно описанная идеология влияет на проблему пропуска слов, обсуждается в четвертом разделе данной статьи. Здесь скажем лишь, что при таком подходе предложения, которые не могут быть собраны без пропусков слов, попадают в особую штрафную зону.

Дополнительно следует отметить, что с точки зрения эффективности алгоритма анализа важным является требование того, чтобы и функция *f* была построена рекурсивно, т. е. для ее вычисления использовались бы какие-то интегральные характеристики соединяемых интерпретаций и то, какие именно действия к ним применяются. Обращение ко всему дереву интерпретации при вычислении этой функции нежелательно. В текущей реализации анализатора Treevial процедура вычисления штрафа для интерпретации построена именно таким образом. В частности, пример штрафов за непроективность, описанных в четвертом разделе, хорошо иллюстрирует сказанное.

До сих пор о штрафах говорилось как о действительных числах. Это позволило доступно изложить концепцию штрафования интерпретаций. Однако, в реальности анализатор устроен немного сложнее. На практике работать с одним действительным числом оказывается неудобно, т. к. возникает естественное желание не смешивать показатели, характеризующие различные свойства структуры. Например, уровень непроективности структур и количество повторений одинаковых связей, исходящих из одного узла, разумно наблюдать как независимые показатели. В связи с этим в Treevial все штрафы представляются векторами из действительных чисел. В управляющем файле анализатора есть возможность назначить каждому из учитываемых свойств свой разряд штрафного вектора. Все изложенное выше, с небольшими изменениями остается верным и для штрафных векторов, т. к. во всех случаях, когда требуется одно действительное число (для сравнения величин штрафов), для штрафного вектора вычисляется норма (равная сумме всех элементов).

Использование штрафных векторов позволяет не только «навести порядок» среди штрафных показателей, но и гибко настраивать анализатор при анализе конкретных текстов. Появляется возможность отсекать структуры, в которых наблюдаются определенные явления. Пусть, например, заранее известно, что в анализируемых текстах не встречаются атрибутивные инверсии (ср. *человек умный*). Тогда, если в системе правил заложено, что в определенном разряде подсчитываются штрафы на инверсии, то при работе анализатора можно отсечь все интерпретации, штрафные вектора которых имеют в этом разряде ненулевые значения.

3. Схема работы анализатора

Как уже говорилось, цель анализатора — найти все возможности «уложить» слова входного предложения в одну синтаксическую структуру или, другими словами, найти все синтаксические интерпретации входного предложения. Каждый вариант

морфологического анализа каждого слова во входном предложении трактуется системой как элементарная синтаксическая интерпретация (состоящая из одного узла). Начинается процесс применения правил. Возникают новые интерпретации, в свою очередь участвующие в правилах. Процесс замыкается. Заканчивается он в тот момент, когда не остается ни одного варианта применения ни одного правила ни к одной комбинации интерпретаций. То, какая именно комбинация интерпретаций должна быть обработана в конкретный момент, определяется по значениям штрафных векторов синтаксических интерпретаций, образующих комбинацию. Система всегда предпочитает комбинации интерпретаций, суммарный штраф которых на текущий момент минимален. В случае наличия нескольких комбинаций с одинаковым суммарным штрафом, предпочтение отдается комбинации, интерпретации которой покрывают наибольшее число слов входного предложения. Интерпретации, проекция которых покрывает все предложение, помещаются в очередь результатов[5]. Одинаковые интерпретации, возникающие в процессе анализа, «схлопываются» — из всех одинаковых интерпретаций остается одна, штраф которой минимален.

В случае, если выполнено свойство неубывания штрафной функции, построенный таким образом процесс гарантирует упорядоченность результатов относительно нее[6].

Для того, чтобы эффективно реализовать описанную переборную схему, потребовалось запрограммировать специальные объекты, называемые *комбинаторами*. Комбинатор представляет собой объект с n входами и одним выходом (в текущей реализации n никогда не превышает 2). Входы комбинатора являются динамически сортируемыми относительно штрафной функции множествами синтаксических интерпретаций. На выходе комбинатор генерирует различные комбинации из n синтаксических интерпретаций. При этом выполняются следующие условия:

1. Элемент комбинации с номером i принадлежит входному потоку с номером i.
2. Комбинации никогда не повторяются.
3. В каждый момент времени на выходе комбинатора находится комбинация с наименьшей из возможных на данный момент суммой штрафов.

Важным свойством комбинатора является то, что он способен динамически реагировать на изменения во входных множествах. Несколько упрощая, можно сказать, что если в каком-то входном множестве появляется «хороший» элемент, то комбинатор автоматически переключается и начинает генерировать комбинации с этим элементом.

Перед началом анализа система синтаксических правил преобразуется в систему комбинаторов. Это преобразование может осуществляться различными способами. Самый простой принцип следующий: с каждым правилом в соответствии с количеством аргументов соотносится набор множеств интерпретаций и один комбинатор[7].

Для того, чтобы на каждом шаге выбиралась оптимальная комбинация, требуется из набора текущих выходов комбинаторов выбирать лучший. Для этого уже сами комбинаторы организуются в список, который динамически поддерживается в упорядоченном состоянии.

Для того, чтобы возникающие синтаксические структуры эффективно распределялись по входным потокам комбинаторов, используется специальный модуль, динамически «индексирующий» синтаксические интерпретации.

На рисунке 4 приводится схема, иллюстрирующая процесс анализа. Эта схема реализует принцип динамического ранжирования гипотез. Остановимся на основных достоинствах и недостатках предлагаемого подхода.

Основным достоинством описанной схемы является то, что она позволяет использовать для записи синтаксических правил формальный аппарат, предоставляющий высокую степень свободы. Например, в нем есть возможность не требовать контактности от соединяемых элементов. С помощью этого формализма удается лаконично описывать синтаксис на некотором базовом уровне, но без штрафов его использование невозможно, т. к. количество порождаемых вариантов для любого предложения средней длины оказывается огромным. С помощью аппарата штрафов, в свою очередь, удается моделировать ограничения, накладываемые на базовый механизм. Схема динамического

---

[5] В действительности анализатор проверяет не только полноту покрытия, но и соответствие корня интерпретации некоторому шаблону, задаваемому в управляющем файле. Этот механизм описывается в четвертом разделе статьи.

[6] На самом деле мы сознательно упрощаем описание. Для того, чтобы упорядоченность результатов была действительно гарантирована, необходимо наличие промежуточного буфера, в который попадают результаты. На рисунке 4 этот буфер обозначен.

[7] В Treevial на сегодняшний день используется более сложная схема, связанная с использованием знаков препинания. Алгоритм анализа построен так, что предложение перед началом обработки делится на минимальные фрагменты, ограничиваемые знаками препинания. Анализ сначала происходит внутри этих фрагментов. Если интерпретация достигает границы фрагмента (соприкасается с делимитатором, ограничивающим фрагмент), она получает возможность комбинироваться с интерпретациями из соседнего фрагмента. В четвертом разделе описывается механизм штрафования, контролирующий «соприкосновения» интерпретаций с делимитаторами. Таким образом, при анализе предложения из n фрагментов в памяти анализатора возникает n элементарных систем комбинаторов (построенных по принципу «одно правило — один комбинатор»).

ранжирования позволяет совместить две модели так, чтобы они работали параллельно. За счет того, что оценка интерпретаций происходит в процессе, а не после анализа, наименее штрафованные варианты обнаруживаются очень быстро, т. к. благодаря неубыванию штрафов «хорошие» интерпретации по определению образуются от «хороших». В нормальной ситуации, если штрафной механизм адекватно настроен, то те варианты, которые анализатор порождает первыми, оказываются ближе всего к действительности. Поэтому, получив, к примеру, один или пять первых результатов (в зависимости от решаемой задачи), процесс можно останавливать. С помощью такого приема вычислительные ресурсы экономятся очень существенно.

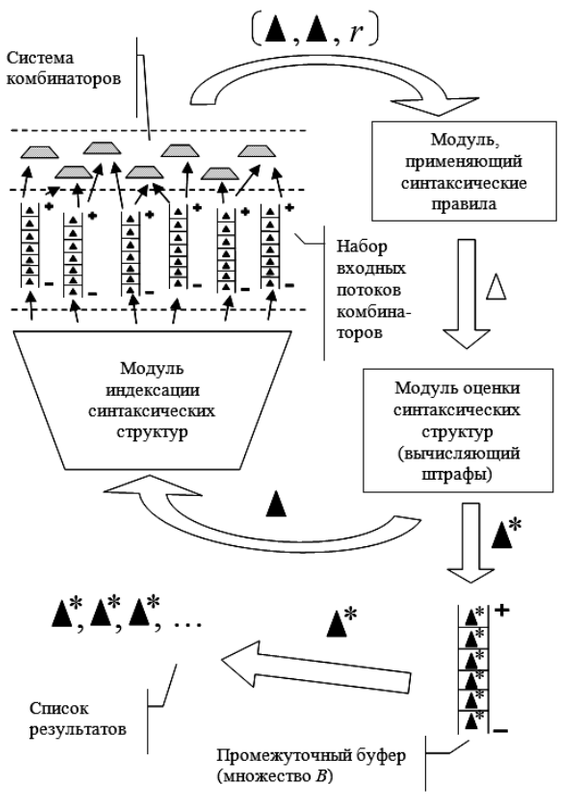

Рис. 4. Схема работы синтаксического анализатора. Треугольники обозначают синтаксические интерпретации. Прозрачные треугольники обозначают интерпретации, для которых еще не посчитан штраф. Треугольники со звездочкой обозначают интерпретации, покрывающие все входное предложение.

Предлагаемая парадигма оказывается очень удобной при управлении процессом синтаксического анализа. За счет того, что штрафы за каждое из учитываемых явлений накапливаются в определенных разрядах, появляется возможность влиять на процесс анализа, различными способами задавая допустимые значения для каждого из разрядов. Например, становится возможным осуществлять анализ в несколько этапов. При первом проходе анализатор можно запускать с большим количеством ограничений (запрещая разрывы, непроективность, повторимость и т. п.). Если при этом удается получить хороший результат, то анализ можно не продолжать. Если же результатов нет или они слишком штрафованы (по тем параметрам, которые не были ограничены), можно продолжить анализ, прямо «на лету» расширив зону допустимых штрафов. Такой подход дает существенный выигрыш в производительности, т. к. первый запуск оказывается молниеносным и в большом количестве случаев позволяет получить результат.

С точки зрения программной архитектуры, схема динамического ранжирования также имеет некоторые достоинства. Тот факт, что модуль оценки синтаксических структур отделен от остальных частей системы, позволяет легко добавлять в систему новые механизмы штрафования, в рамках которых могут использоваться произвольные алгоритмы. К примеру, все механизмы, описывающиеся в четвертом разделе, добавлялись в систему постепенно и каждое добавление не производило «революции» в программном коде. Хорошим примером, иллюстрирующим то, как отсутствие инкапсулированности оценочного механизма может вызывать проблемы, являются вероятностные контекстно-свободные грамматики (PCFG). Из [Johnson, 1998] хорошо видно, что жесткая привязка вероятностных оценок к продукциям контекстно-свободной грамматики (их аналогом являются правила Treevial) не позволяет в полной мере учитывать совместную встречаемость единиц. Следует отметить, что в Treevial на данный момент существует лишь эскизный вариант механизма штрафов на лексическую сочетаемость, который не может конкурировать с PCFG по количеству сил и времени, потраченных исследователями. Однако установка на учет контекстных зависимостей заложена в этот механизм изначально. Во многом за счет того, что правила и модуль оценки логически разделены, это оказалось легко сделать.

Отдельно следует сказать, что в рамках описанной выше концепции достаточно четкие очертания обретает схема взаимодействия со всей системой абстрактного модуля, моделирующего семантику. Причем, не имеет значения, как именно этот модуль будет устроен внутри. Важно, что он должен функционировать параллельно с описанной переборной схемой. В динамике для каждой синтаксической интерпретации могли бы строится семантические представления. Если при этом они оценивались бы аналогично синтаксическим, то семантические штрафы влияли бы на ход анализа наряду с синтаксическими. Существенным кажется то, что при таком подходе «обретать смысл» будут не только конечные структуры , но и все фрагментарные. Это в чем-то согласуется с человеческим механизмом

восприятия — мы начинаем осмыслять то, что слышим, сразу, не дожидаясь конца сообщения.

Авторы столкнулись с двумя недостатками схемы динамического ранжирования. Первый из них связан с требованием неубывания штрафной функции и с условием инкрементальности. Одним из существенных следствий этих постулатов является невозможность взимания штрафов за отсутствие каких либо элементов структуры. Так, например, если у слова есть какая-то обязательная (или очень желательная) валентность, невозможно взять штраф за то, что она не выражена в анализируемом предложении. Точнее сказать, такие штрафы невозможно взимать в динамике (после завершения анализа целую структуру всегда можно проанализировать и оценить каким угодно образом). Попытка взимать такие штрафы при анализе фрагментарных интерпретаций в сочетании с принципом инкрементальности неизбежно приведет к противоречивой ситуации. Так, например, если в предложении присутствует слово, заполняющее обязательную валентность, то интерпретации, в которых это слово еще не присоединено к хозяину, будут оштрафованы (за незаполненную валентность), а интерпретации, в которых валентность, наконец, заполнится, окажутся оштрафованными «незаслуженно» (т. к. вторые породятся из первых, а штрафы накапливаются). Существуют технические приемы, позволяющие обойти это ограничение, но все они имеют недостатки. Следует отметить, что на практике без использования штрафов за отсутствие удается обходиться. Для авторов вопрос о необходимости использования штрафов этого типа до сих пор остается открытым.

Вторым недостатком описанной схемы является необходимость соотносить друг с другом абсолютные величины штрафов, отражающих различные явления. На сегодняшний день это делается вручную при настройке анализатора на корпус текстов и представляет собой достаточно трудоемкий процесс. В системе Treeton существует графическая среда, позволяющая производить синтаксический анализ коллекции предложений, при работе с которой процесс балансировки штрафов несколько упрощается: система автоматически отслеживает динамику изменений штрафных векторов конкретных предложений при изменении системы штрафов и правил.

В ближайшее время авторы не планируют предпринимать шаги в направлении автоматизации процесса балансировки штрафов, т. к. первоочередной задачей на данный момент является расширение системы правил и штрафов для русского языка и тестирование анализатора на корпусе. Только после того, как это будет сделано, станет ясно, какое количество штрафных разрядов требуется для качественного анализа. Необходимость использования методов автоматической настройки существенным образом зависит от порядка этой величины.

4. Инструменты штрафования

В этом разделе описываются средства языка Treevial, позволяющие влиять на значения штрафной функции в процессе анализа. Мы сознательно приводим лишь концептуальное описание этих инструментов, т. к. полноценное описание, снабженное примерами и фрагментами формального синтаксиса, заняло бы место, сопоставимое по размеру со всей статьей.

4.1. Штрафы за непроективность

В анализаторе Treevial предусмотрены возможности для оценки уровня проективности структур. В момент применения каждого правила системой вычисляются проекции соединяемых узлов. Если между проекциями есть зазор (одно или более слов), взимается штраф за непроективность (его величина настраивается в управляющем файле). Помимо базового механизма в Treevial поддерживается возможность тонкой настройки политики взимания штрафов за непроективность, позволяющая учитывать то, какие синтаксические фрагменты являются статистически более плотными (неразрывными), а какие допускают разрывы. Примером первых могут служить отдельные конъюнкты, входящие в сочинительную группу, вынос слов из которых едва ли возможен (ср. **умный пришли Вася и глупая Даша*). Примером фрагмента, допускающего разрывы, может служить инфинитивный оборот (ср. *кожаную ты хочешь купить куртку?*). Тонкий контроль проективности в языке Treevial обеспечивается с помощью специальной таблицы из шаблонов и штрафных векторов.

4.2. Штрафы за повторение связей

Штрафы за повторение используются в тех случаях, когда требуется ограничить количество связей одного типа, выходящих из одного узла синтаксической структуры. В качестве примера можно привести связь между глаголом и существительным в именительном падеже. В русском языке эта связь больше одного раза не повторяется (ср. **Вася Петя играл в футбол*). В момент применения каждого правила проверяется, не была ли добавлена связь, тип которой объявлен неповторимым, к узлу, из которого связь такого типа уже исходит. Если была, то взимается штраф за повторение связей. Список неповторимых связей и величина штрафа задаются в управляющем файле.

4.3. Механизм работы со знаками пунктуации

В анализатор Treevial заложено понятие знака пунктуации (или делимитатора). В управляющем

файле имеется конструкция, позволяющая задать шаблон, используя который система может отличить знаки пунктуации от остальных символов. В предыдущем разделе упоминалось, что знаки пунктуации имеют значение для алгоритма анализа, т. к. по ним производится первоначальное разбиение предложения на фрагменты. Для того, чтобы в процессе синтаксического анализа появилась возможность комбинировать некоторую синтаксическую интерпретацию с интерпретацией из соседнего фрагмента, необходимо, чтобы знак пунктуации, стоящий на границе, оказался «синтаксически оправдан» в контексте одной из интерпретаций. Это вычисляется, исходя из специальной таблицы, задаваемой в управляющем файле. Если оправдать знак не удается, доступ все равно открывается, но при этом взимается особый штраф на «неоправданный знак препинания». Такой подход позволяет достаточно гибко работать с пунктуацией. В частности, система оказывается устойчива к текстам, в которых есть пунктуационные ошибки, и к текстам, с нестандартной — несовременной или индивидуально-авторской — пунктуацией.

4.4. Штрафы при применении правил

Штрафы при применении правил уже упоминались в первом разделе. Они используются в тех случаях, когда требуется наложить штраф на то или иное явление, связанное со спецификой конкретного синтаксического правила. Например, штраф на обратный порядок существительного и прилагательного при образовании атрибутивной связи между ними («*человек умный*» вместо «*умный человек*»).

4.5. Штрафы за пропуски

Как уже говорилось, формальный аппарат синтаксических правил допускает создание интерпретаций, проекции которых не являются непрерывными. Однако при анализе реальных предложений необходимость использования разрывных интерпретаций возникает совсем не часто. Заметим, что необходимость использования разрывных интерпретаций в процессе создания и непроективность являются связанными, но отнюдь не эквивалентными свойствами синтаксических интерпретаций. Так, например, предложение на рисунке 1 хоть и является непроективным, но может быть проанализировано без использования разрывных интерпретаций[8]. В общем случае можно утверждать следующее:

- Любая интерпретация, которую нельзя собрать без использования разрывных интерпретаций, является непроективной.
- Многие непроективные интерпретации можно собрать без использования разрывных интерпретаций.

Будем называть предложения, имеющие структуру, которая не может быть собрана без использования разрывных интерпретаций, *сверхнепроективными*. Примером сверхнепроективного предложения является фраза «*книгу я красную люблю*». Действительно, какая бы связь не была проведена первой, обязательно возникнет разрывная интерпретация. Обычно сверхнепроективные предложения встречаются в разговорной речи. Еще одним примером является классическая латинская поэзия, изобилующая такими предложениями [Ботвинник, Гладкий, 2009].

Однако существует широкий класс текстов, в которых встречаемость сверхнепроективных предложений очень низка. Кроме того, запрет на использование разрывных интерпретаций в процессе анализа значительно ускоряет работу анализатора, т. к. существенно уменьшает комбинаторику. Для того, чтобы контролировать появление разрывных интерпретаций, в Treevial были введены штрафы за пропуски. В управляющем файле с помощью специальной конструкции можно задавать величину штрафа, который взимается при возникновении всякой разрывной интерпретации. Важно понимать, что благодаря принципу инкрементальности штраф за разрыв остается «клеймом» на всех потомках разрывной интерпретации даже если в дальнейшем сам разрыв устраняется. Таким образом, все сверхнепроективные предложения всегда попадают в особую штрафную зону, что позволяет гибко настраивать анализатор при работе с корпусом.

4.6. Штрафы за некомпактность

В данном случае мы исходим из того, что в ситуации выбора из двух похожих по смыслу синтаксических структур более предпочтительной оказывается та, в которой зависимые в среднем расположены ближе к своим хозяевам. При проведении каждой новой связи взимается штраф, пропорциональный ее длине (расстоянию от хозяина до зависимого), умноженный на константу (число, которое задается с помощью конструкции языка Treevial). Понятно, что такая константа должна быть достаточно маленькой, т. к. фактор компактности не должен быть сильнее других. Хотя эта эвристика и не играет ключевой роли, она все равно кажется полезной. В ряде случаев с помощью нее удается, например, угадывать хозяев для предложных групп.

[8] Разрывная интерпретация на рисунке 1 была приведена исключительно в иллюстративных целях.

4.7. Механизм задания целевого шаблона

Для того, чтобы анализатор был способен обрабатывать не только полные предложения (в которых есть глагол в личной форме или предикатив), но и «кусочные» предложения (например, отдельные именные или предложные группы), синтаксис формального языка был расширен специальной конструкцией, позволяющей с помощью шаблонов и штрафных векторов описывать то, какие целевые структуры допустимы и какие из них предпочтительней (например: полное предложение на первом месте, далее именная группа, потом все остальные). В первую очередь, такая потребность была вызвана необходимостью анализировать заголовки стихотворений и литературных произведений.

4.8. Механизм учета лексической сочетаемости

Особняком стоит штрафной механизм, позволяющий учитывать при оценке синтаксических структур их конкретное лексическое наполнение, а именно сочетаемость лексических единиц, которые в процессе перебора были связаны некоторой синтаксической связью. Этот механизм был создан сравнительно недавно. Некоторые концептуальные вопросы, касающиеся его организации, до сих пор составляют предмет активных дискуссий авторов. Таким образом, можно сказать, что описание, приводимое ниже, носит эскизный характер.

Механизм учета лексической сочетаемости управляется не из основного управляющего файла, а обращается к лексической базе, в которой описаны различные ограничения на сочетаемость. За основу формального описания сочетаемости лексических единиц была взята модель, предложенная в [Перцов, Старостин, 1999]. В случае обнаружения несоответствия каких-то связей, проводимых в процессе анализа, модели сочетаемости, штраф соответствующих интерпретаций увеличивается.

В системе могут учитываться три типа ограничений на сочетаемость лексических единиц: морфо-синтаксические, лексические и семантические. К морфо-синтаксическим ограничениям на сочетаемость некоторого слова относятся ограничения на предложно-падежную форму зависимых слов (*агитация **[род]** [за+**вин**] [против+**род**]; блеснуть **[тв]** [перед+**тв**]; благодарный **[дат]** [за+**вин**]*).

Лексические ограничения на сочетаемость выражаются в том, что некоторые слова могут иметь в качестве зависимых только определенные лексемы, причем зачастую выделить общие признаки этих лексем оказывается невозможно. Например, в словосочетаниях *уменьшать / сбрасывать скорость, давление, громкость* глаголы *уменьшать / сбрасывать* имеют близкие значения, однако разную лексическую сочетаемость: можно уменьшить, но не сбросить длину, количество, сопротивление. Таким же образом описываются устойчивые выражения (глагол *обратить* сочетается с предлогом *на* только в выражении *обратить внимание на+**вин***).

Семантические ограничения отражены в лексической базе при помощи помет. Например, глагол *вернуться* имеет два актанта, отвечающих на вопросы *куда?* и *откуда?*, причем морфологически эти актанты могут быть выражены различными способами (*вернуться из Москвы в Петербург*, *вернуться издалека*, *вернуться домой*). В подобных случаях в лексической базе описываются не все возможные варианты выражения актантов, а указывается семантическая помета, которой актанты должны соответствовать (*вернуться **[НАПР]** **[ИСХ]***). Для некоторых лексических единиц также указано, как они трактуются в сочетании с различными наборами актантов (*издалека → **ИСХ***; *из **[род]** → **ИСХ***; *в **[пр]** → **ЛОК***; *в **[вин]** → **НАПР***). Для синтаксических интерпретаций, построенных на основе таких лексических единиц, строится их семантическая трактовка, которая затем при подчинении этих интерпретаций проверяется на соответствие семантическим ограничениям на сочетаемость подчиняющей интерпретации.

Рассмотрим два предложения: *Он познакомился с агитатором против истребления диких животных* и *Он выступал с друзьями против истребления диких животных*. Без использования механизма оценки сочетаемости анализатор не сможет отсеять неправильные гипотезы (в которых словосочетание *против истребления диких животных* зависит от глагола *познакомился* и существительного *друзьями* соответственно). При использовании описываемого механизма анализатор предпочтет правильные гипотезы, поскольку только существительное *агитатор* (соответственно, глагол *выступать*) сочетается с предложно-падежной группой *против+**род***).

Рассмотрим еще один пример: *Встретил беженцев из Палестины* и *Вез сигареты из Москвы*. В лексической базе указана следующая информация о сочетаемости: *беженцы **[ИСХ]***, *вез **[вин]** **[ИСХ]** **[НАПР]***, *встретил **[вин]***, *из **[род]** → **ИСХ***. Поскольку для предлога *из* в сочетании с родительным падежом указана семантическая трактовка *ИСХ*, для синтаксической интерпретации словосочетаний *из Москвы*, *из Палестины* будет построена семантическая трактовка *ИСХ*, которая в первом предложении сочетается только с существительным *беженцев*, а во втором — только с глаголом *вез*. Таким образом, приоритет получат правильные варианты анализа предложений (*беженцы из Палестины* и *вез из Москвы*).

Очевидно, что описания, которые можно заносить в лексическую базу, носят сильно упрощенный характер и ни в коей мере не претендуют на полноценное описание семантики лексических единиц. Однако, оказывается, что даже такие простые средства могут существенно улучшать качество синтак-

сического анализа. Важным свойством построенного механизма является то, что лексическая база может пополняться постепенно. При любой степени наполненности синтаксический анализ будет работать. Но чем больше информации будет внесено, тем точнее будет результат анализа. Следует отметить, что авторы далеки от мысли о ручном наполнении такой базы. В будущем планируется разработать методы автоматизированного извлечения сочетаемостных моделей из синтаксически-размеченных корпусов.

Заключение

В данной работе было описано то, как работает синтаксический анализатор Treevial, и изложены концепции, лежащие в его основе. В заключение следует отметить, что приведенная схема динамического ранжирования в принципе могла бы быть распространена на весь процесс анализа текста на естественном языке. «Поводы» для взимания штрафов имеют место на различных языковых уровнях. Приведем лишь некоторые из них:

- На нижних языковых уровнях
  - учет частотности употреблений словоформ
  - нечеткое сопоставление при распознавании звуков (символов) или текстов с ошибками
- На синтаксическом уровне
  - статистический учет сочетаемости слов
- На уровне семантики
  - использование семантических моделей, способных оценивать «осмысленность» синтаксической структуры

Авторы надеются, что впоследствии им удастся создать анализатор текстов, в рамках которого будут интегрированы все перечисленные функции. Архитектура анализатора Treevial уже сегодня ориентирована на расширение и подключение новых модулей.

Авторы горячо признательны Н. В. Перцову за многочисленные консультации и помощь в организации семинаров, посвященных анализатору Treevial. Мы благодарим всех участников семинара по вопросам автоматического лингвостиховедческого и морфосинтаксического анализа (ИМК МГУ / ЦТС ИРЯ РАН) за бурные дискуссии и обсуждения, а также С. А. Минора за ряд ценных советов. Особую благодарность авторы выражают И. А. Пильщикову — за помощь в подготовке окончательного текста данной статьи.

Литература

1. *[Ботвинник, Гладкий, 2009]* — Гладкий А. В., Ботвинник Н. М. «Переплетение слов» в русской и латинской поэзии // «Слово — чистое веселье...»: Сборник статей в честь Александра Борисовича Пеньковского. М.: Языки слав. культуры, 2009. С. 299–310.
2. *[Гладкий, 1985]* — Гладкий А. В. Синтаксические структуры естественного языка в автоматизированных системах общения // М.: Наука, 1985.
3. *[Гладкий, Мельчук, 1969]* — Гладкий А. В., Мельчук И. А. Элементы математической лингвистики // М.: Наука, 1969.
4. *[Мальковский, Старостин, 2006]* — Мальковский М. Г., Старостин А. С. Модель синтаксиса в системе морфосинтаксического анализа «TREETON» // Компьютерная лингвистика и интеллектуальные технологии: Труды международной конференции «Диалог 2006». М.: изд-во РГГУ, 2006. С. 481–492.
5. *[Мальковский, Старостин, 2007]* — Мальковский М. Г., Старостин А. С. Алгоритм синтаксического анализа, используемый в системе морфо-синтаксического анализа «TREETON» // Труды международной конференции Диалог 2007. М.: изд-во РГГУ, 2007. С. 516–524.
6. *[Перцов, Старостин, 1999]* — Перцов Н. В., Старостин С. А. О синтаксическом процессоре, работающем на ограниченном объеме лингвистических средств // Труды международной конференции Диалог 1999. Таруса: 1999. Т. 2. С. 224–230.
7. *[Тестелец, 2001]* — Тестелец Я. Г. Введение в общий синтаксис // М.: РГГУ, 2001.
8. *[Хомский, 1962]* — Хомский Н. Синтаксические структуры // Новое в линвистике. Вып. II. М., 1962. С. 412–526.
9. *[Cocke, Schwartz 1970]* — John Cocke and Jacob T. Schwartz Programming languages and their compilers: Preliminary notes // Technical report Courant Institute of Mathematical Sciences, New York University, 1970.
10. *[Dekhtyar, Dikovsky, 2008]* — Michael I. Dekhtyar, Alexander Ja. Dikovsky. Generalized Categorial Dependency Grammars // Pillars of Computer Science, 2008. P.230–255
11. *[Johnson, 1998]* — Johnson M. PCFG models of linguistic tree representations // Computational Linguistics, 1998. №24(4). P. 617–636.
12. *[Kasami, 1965]* — Kasami T. An efficient recognition and syntax-analysis algorithm for context-free languages // Scientific report Afcrl-65-758, Air Force Cambridge Research Lab, Bedford, MA., 1965
13. *[Mel'cuk , 1988]* — Mel'cuk I. A. Dependency Syntax: Theory and Practice // SUNY Series in Linguistics. Albany: State University of New York Press, 1988
14. *[Pollar, Sag, 1994]* — Carl Pollard, Ivan A. Sag. Head-Driven Phrase Structure Grammar // Chicago: University of Chicago Press, 1994.
15. *[Schneider, 1998]* — Garold Schneider. A Linguistic Comparison of Constituency, Dependency and Link Grammar // http://www.ifi.unizh.ch/cl/study/lizarbeiten/lizgerold.pdf, 2007
16. *[Sleator & Temperley, 1991]* — Sleator D., Temperley D. Parsing English with a Link Grammar // Carnegie Mellon University Computer Science technical report CMU-CS-91-196, 1991.
17. *[Tesnière 1959]* — Tesnière L. Éléments de syntaxe structurale // Paris: 1959.
18. *[Younger, 1967]* — Daniel H. Younger. Recognition and parsing of context-free languages in time n3 // Information and Control, 1967. № 10 (2). P. 189–208.